Claudio Marche, Luigi Atzori, Virginia Pilloni, Michele Nitti,
How to exploit the Social Internet of Things: Query Generation Model and Device Profiles' Dataset in *Computer Networks*, Volume 174 (2020), art. 107248.

The publisher's version is available at:
https://doi.org/ 10.1016/j.comnet.2020.107248

When citing, please refer to the published version.

How to exploit the Social Internet of Things: Query Generation Model and Device Profiles' Dataset

Claudio Marche[a,b], Luigi Atzori[a,b], Virginia Pilloni[a,b], Michele Nitti[a,b]

[a]*DIEE, University of Cagliari, UdR CNIT of Cagliari, Italy*
[b]*National Telecommunication Inter University Consortium (CNIT), Research Unit of Cagliari, Italy*

Abstract

The future Internet of Things (IoT) will be characterized by an increasing number of object-to-object interactions for the implementation of distributed applications running in smart environments. The Social IoT (SIoT) is one of the possible paradigms that is proposed to make the objects' interactions easier by facilitating the search of services and the management of objects' trustworthiness. In this scenario, we address the issue of modeling the queries that are generated by the objects when fulfilling applications' requests that could be provided by any of the peers in the SIoT. To this, the defined model takes into account the objects' major features in terms of typology and associated functionalities, and the characteristics of the applications. We have then generated a dataset, by extracting objects' information and positions from the city of Santander in Spain. We have classified all the available devices according to the FIWARE Data Models, so as to enable the portability of the dataset among different platforms. The dataset and the proposed query generation model are made available to the research community to study the navigability of the SIoT network, with an application also to other IoT networks. Experimental analyses have also been conducted, which give some key insights on the impact of the query model parameters on the average number of hops needed for each search.

Keywords:
Social Internet of Things; query generation model; dataset; test management algorithms

---

1. Introduction

The Internet of Things (IoT) has become a reality with billions of devices able to send key information about the physical world and implementing simple actions, which leads to the paradigm of the anytime and anyplace connectivity for anything [1]. The massive amount of data flowing through the IoT has pushed forward the development of new applications in several domains, such as the management of industrial production plants, the logistics and transport supply chain, the e-health, the smart building, just to cite a few.

However, IoT solutions have posed new challenges in the management of the amount of information produced. Indeed, searching for (reliable) time- and location- relevant information, services and resources for the deployment of running applications exploiting the IoT infrastructure is a crucial challenge: in addition to the size of the searching space, most of the data produced by the sensors produce rapid changes, making the system highly dynamic, as it happens for instance when tracking the position of moving objects. A further complication derives from the shift we are witnessing in the interaction model. From a paradigm where humans look for information provided by objects (human-object interaction), the IoT will surely move towards a model where things look for other things to provide composite services for the benefit of human beings (object-object interaction). With such an interaction model, it will be essential to understand how the information provided by each object can be processed automatically by any other peer in the system. This cannot clearly disregard the level of trustworthiness of the object providing information and services, which should take into account the profile and history of it. If not, attacks and malfunctions would outweigh any of the benefits of these technologies [2].

An approach with the potential to properly address the mentioned issues, which is recently gaining increasing popularity, is based on the exploitation of social networking notions into the IoT, as formalized by the Social IoT (SIoT) concept [3]. It introduces the vision of social relationships among different devices, independently from the fact that they belong to the same or different platforms owned and managed by different individuals or organizations. According to this vision, all the IoT objects are willing to collaborate with others and create relationships among them as humans do. This is expected to make the exchange of information and services among different devices easier and to perform the identification of malicious nodes by creating a society-based view about the trust level of each member of the community.

In the resulting social network, each application running in the devices (or in the cloud) will be looking for information and services by crawling the social network starting from a requesting node towards the potential service provider(s). The performance of such a process of service/information retrieval is clearly dependent on several aspects: i) the structure of the social network; ii) the types of service/information requests that will mostly characterize the interaction in the IoT/SIoT; iii) the rules that are used to navigate the network.

To analyze these aspects, there is a need for a proper model of the behavior of objects that generate queries of services and information when interacting with other peers in the SIoT. Such a model represents an alternative to the analysis of large sets of real interactions among devices; indeed, this data is not publicly available, so the model can represent a valid alternative for the analysis of the networks. Understanding the interactions among peers can help in discovering which devices are more likely to interact and then can assist in the design of search engines, in the management of the trustworthiness or the creation of clusters of nodes with frequent interactions. To this, the model should take into account the objects' major features in terms of typology and associated services, and the applications that may need to interact with the different objects. Another essential element to test management algorithms is represented by a suitable IoT dataset. It has to exhibit a realistic behavior based on real IoT objects and show information regarding the position and the profile of a large set of IoT devices, both public and private, both fixed and mobile, arranged with their respective owner. All these objects need to be categorized by their typology, brand and model, but also considering the set of services they are able to offer and the possible applications they can request. The major contributions of the paper are the following:

- Definition of the generic process of service search in the Social IoT and modeling of objects behavior when interacting with other peers in the network for the exchange of information and services.
- Definition of a query generation model, which is able to simulate the correlation between objects and applications and represents a fundamental tool to test the interaction among peers in the network. The proposed model is then used to evaluate the benefits of the social approach in terms of global navigability.
- Creation of a dataset, which not only include objects' information and positions as done in [4], but also the services and applications they offer and use. The collected data derives from the devices installed in the city of Santander in Spain and on the data about people's mobility. This is made available to the research community to test (S)IoT management algorithms (e.g., relationship management, service search, trustworthiness management), with particular attention to network navigability.

The paper is organized as follows. Section 2 introduces the relevant background regarding the social IoT and provides a short survey of the research related to service search in the IoT. In Section 3 we provide an introduction to the proposed solution by defining the scenario and introducing the used notations, whereas in Section 3.2 we propose a query generation model and test it based on real traces. Section 5 presents a dataset based on real objects and uses it to construct a social network of devices and Section 6 draws final remarks.

2. Background

2.1. The Social IoT

The SIoT represents the convergence of the technologies belonging to two domains: IoT and Social Networking. The result is the creation of social networks in which things are nodes that establish social links as humans do [3]. This concept is fast gaining ground thanks to the key benefits deriving from the potentials of the social networks within the IoT domain, such as: simplification in the navigability of a dynamic network of billions of objects [3]; robustness in the management of the trustworthiness of objects when providing information and services [5]; efficiency in the dynamic discovery, selection and composition of services (and of information segments) provided by distributed objects and networks [6]. According to the SIoT model, every node is an object that is capable of establishing social relationships with other things autonomously, according to rules set by the owner.

To this aim, as underlined in [7], there is a strong need to improve the degree of connectivity between users and things, where things should be socialized to allow humans to easily establish relationships with them. The resulting paradigm of the Social Internet of Things (SIoT) [3] includes these notions, so that people, through their IoT devices, can transparently (although according to clear policies they have set for inter-device interactions) improve the experience in the fruition of smart services and applications.

When it comes to the IoT paradigm, the idea is to exploit social awareness as a means to turn communicating objects into autonomous decision-making entities. The new social dimension shall, somehow, be able to mimic interactions among users and to motivate a drift from an egoistic behavior to altruism or reciprocity. The main principle is to enable objects to autonomously establish social links with each other (by adhering to rules set by their owners) so that "friend" objects exchange data in a distributed manner. Every network object will be capable of: (a) establishing social relationships with other objects autonomously with respect to the owner, but according to the preset rules for the owner; (b) interact with its friends when in need for some assistance, such as the provisioning of a piece of important information or a key service.

According to this model, a set of forms of socialization among objects is foreseen. The parental object relationship (POR) is defined among similar objects, built in the same period by the same manufacturer (the role of the family is played by the production batch). Moreover, objects can establish co-location object relationship (CLOR) and co-work object relationship (CWOR), like humans do when they share personal (e.g., cohabitation) or public (e.g., work) experiences. A further type of relationship is defined for objects owned by the same user (mobile phones, game consoles, etc.) that is named ownership object relationship (OOR). The last relationship is established when objects come into contact, sporadically or continuously, for reasons purely related to relations among their owners (e.g., devices/sensors belonging to friends); it is named social object relationship (SOR). These relationships are created and updated on the basis of the objects features (such as type, computational power, mobility capabilities, brand, etc.) and activities (frequency in meeting the other objects, mainly).

However, to fully exploit the benefits of a SIoT network, realistic networks and models of object-object interaction are still needed, which we investigate in this paper.

2.2. Query Generation models

A common problem in the IoT is how to efficiently retrieve information among the billions of devices composing it. Anytime there is the need to obtain data from a given system, two essential elements are needed [8]:

- The objects, which are the entities that provide information through the services they can offer.
- The query, which is the formal statement of the needed information (e.g., a search string).

The link between these two elements is represented by a search engine, which has the goal to match the query with the data provided by the objects.

The IoT provides several approaches to searching services, i.e., to the development of engines able to look for the required objects, such as [9] and [10]. These works proposed different mechanisms for both indexing and ranking of the objects that can be used to search and select the objects and their offered services. In the first work, the authors propose a context-aware sensor search, based on a ranking model, to improve the selection of relevant sensors in large sets; the context information related to each sensor can be used to search the sensors in accordance with the user's requirements. The second paper focuses on the requirements and challenges that need to be addressed to construct efficient search engines.

Another approach is described in [11]: a framework composed of a context module and a search engine is developed to interact with the IoT devices. The context module is responsible to assign the semantic characteristics to the devices while the search engine has to evaluate the users' queries, select and interact with the objects, based on a proposed indexing technique.

Moreover, a recent work [12] analyses the most important IoT search engines in the literature and introduces a classification. One of the major issues pointed out by the authors is the lack of open datasets that contain IoT data and of query models to test the proposed engines. These aspects are critical for the community since they simplify carrying out the simulations and making them repeatable.

Even if all these systems analyzed different features for the service search systems, there are almost no works about the modeling of the query generation process in the IoT, which, as said, represents a key component to test search engines. Among these few works, in [13] the authors propose four basic query models to search devices by their name, identifier, time and location of data. Another example is provided in [14] which proposes spatial range queries with location constraints to facilitate data indexing.

However, the problem of generating requests of information has been deeply investigated for the World Wide Web, where the first approaches can be dated back to the late '90s. Indeed, there is a strong similarity between searching the Web, which is composed of a widely accessible, large and distributed source of text data, i.e. documents, and the service search in IoT, whereas documents can be seen as IoT applications and the words describing a document as the services needed by an application. Accordingly, query generation methods used to search the Web might be adapted to be used in the IoT.

Table 1 describes all the examined query generation methods. Among them, the simplest approach is based on uniform distribution, where all the terms from a given vocabulary have the same probability to occur in a query [23].

Table 1: Query Generation Methods.

Method	Description
Uniform [15]	Select terms with uniform probability from current vocabulary
Term-frequency [15]	Select most frequent words from a cluster of documents
Probabilistic term-frequency [16]	Select terms with probability proportional to their term frequency
Odds-ratio [17]	Select terms in according to maximum odds-ratio score
Probabilistic odds-ratio [18]	Select terms with probability proportional to their odds-ratio score
Boley [19]	Intersection of the top ranking terms according to term frequency and document frequency
Markov chain [20]	Method that allow to chose consecutively query terms from a first term
Poisson distribution [21]	Generate query using frequency words from a series of independent Poisson processes
Query expansion [22]	Process of reformulating query given by a user

Another approach is the term-frequency method, which classifies terms based on their frequency in a cluster of documents [15]; only the terms above a given threshold are considered, according to a uniform distribution, to compose a query. The authors in [16] propose a variant of this method called probabilistic term-frequency, where all the terms in the cluster are considered for a query, but with a probability proportional to their frequency.

The authors in [17] propose an approach based on odds ratio; this score measures how strong the probability of a term appearing in relevant documents is w.r.t. the probability of the same term appearing in non-relevant documents. Only the term with a ratio higher than a threshold can be selected to generate a query. Similarly to [16], the authors in [18] improve the odds ratio method with a probabilistic approach by assigning a probability proportional to the odds ratio score and by selecting the terms according to this probability.

Another approach is proposed by Boley in [19]. The query terms are selected in accordance to an intersection of two sets: the Text Frequency (TF) word list, which refers to words frequency in a selected text and the Document Frequency (DC) word list, which considers the terms' frequency in all the documents of a cluster.

Authors in [20] show additional developments in query generations using Markov chains. To estimate the query, they calculate the probability that a word "translates" to a query term, and then they chose a document containing the selected word, based on the number of times the word appears in it. The Markov chain alternates the choices between words and documents iteratively until the query is generated.

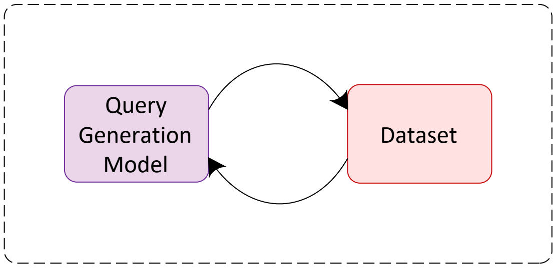

Figure 1: Proposed system to develop and test IoT management algorithms

In [21], the authors study a new family of query generation models based on the Poisson distribution. These models calculate the frequency of each term independently with a Poisson distribution. To rate a document, the authors estimate a multivariate Poisson model based on the document, and then give a score to it based on the likelihood of the query given by the estimated Poisson model.

In the last years, query generation techniques have developed methods to expanse the query and change it to reformulate a given query with relevant information so as to improve retrieval performance and to take into account the user profile [24]. A common technique consists of expanding the original query with synonyms: the new terms are chosen based on a probabilistic approach. In [22], the authors propose a relevance model, which assigns a relevance probability to each word of the collection to measure how relevant a synonym could be for the terms in the original query.

All the cited techniques might not adapt directly to the IoT, but can be used as a starting point for the design of a query generation model for the IoT. In particular, in this paper we focus on the probabilistic methods and adapt them to the IoT scenario. Moreover, we validate the obtained model by making use of a real set of queries, obtained from the Lysis IoT platform [25].

3. Introduction to the proposed solutions

This paper aims to provide the two essential elements needed to develop and test management algorithms in an IoT ecosystem, namely a query generation system and a dataset of objects with realistic behavior. As it is depicted in Figure 1, even though these two elements can cooperate, i.e. the query generation model can be tested by using the dataset, they exist and can be used independently. The details about their functionalities will be better explained in Section 4 (for the query generation model) and Section 5 (for the dataset).

In the following, we present the reference scenario and the needed notation to describe it.

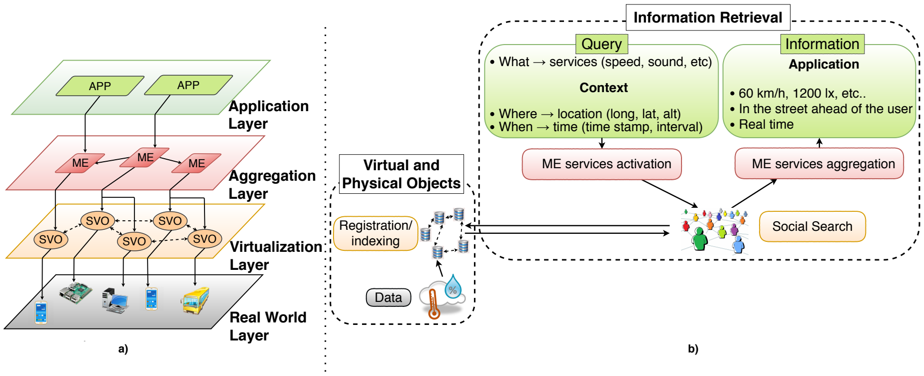

Figure 2: Reference SIoT architecture and query generation model.

3.1. *Reference Scenario*

The SIoT provides the objects with some capabilities typical of humans' behavior when looking for and providing information in their social communities. Accordingly, social relationships are created among objects, which are used when the peers are looking for help [3]. As in most of the IoT architectures, the owner has the control on which social interactions the objects are allowed to perform and which information and services can be shared with other peers. The applications installed by owners in their cloud space and that rely on their objects' capabilities often need to look for services provided by other objects. This results in queries that are managed by the SIoT by making use of objects' social connections through word of mouth.

The focus of the paper is the analysis and modeling of this query generation process. To this, we consider the reference SIoT architecture shown in Figure 2a), which is based on four levels [26]: Application, Aggregation, Virtualization and Real World. The lower layer is made up of the "things" of the real world, which have the role to sense the physical environment and provide data to the higher layers. The Virtualization layer is made of Social Virtual Objects (SVOs), which represent the digital counterparts of any entity of the real world enhanced with social capabilities, fully describing their characteristics and the services they are able to provide [27]. The Micro Engine (ME), which is the main entity of the Aggregation layer, is a mash-up of one or more SVOs and other MEs, and it is responsible for getting and processing information from SVOs into high-level services requested by applications at the higher level. Finally, the Application layer is installed in the Cloud and partially in the devices, so that applications can be deployed and executed exploiting one or more MEs.

Figure 2b) illustrates a generic service query in the SIoT, which highlights all the components involved in the process. The whole process starts when the application layer triggers some processing that requires to look for other services and then generates a relevant query. The query specifies *what* services are required and it is enhanced with context parameters, which represent the application requirements, such as a specific time (*when*) or a specific place (*where*).

The generated query is then handled at the Aggregation Layer, where the needed MEs for data elaboration are activated. After this, the query is taken over by the SVO of the device that triggered the process, which navigates its social network in order to search for other SVOs that can offer the data related to the desired services by the application. Indeed, in the SIoT, each SVO maintains information related to its friends and to the services that the corresponding physical object can provide. In this sense, SVOs can be seen as atomic registration/indexing servers. However, it is not the focus of the paper to design an indexing mechanism of data.

In order to better explain the query process, an explanatory example is presented here. Suppose that an object installed the application *RealTimeTrafficEvaluationApp*, which evaluates the traffic of a specific street in real-time, i.e. within a limited time interval with respect to the current time. Accordingly, the object creates a query with the list of services needed to execute the application and the requirements related, in this case, with the reference location and time. The aggregation layer than activates the MEs associated with the services and passes the query to the SVO, which looks for the objects, among its social network, that can execute the services fulfilling the desired requirements. Once they are found, the aggregation layer processes the result and provides the requested information, i.e. the real-time traffic condition in the specified street, to the user.

When all the services are retrieved, they are forwarded again to the Aggregation Layer which composes them through the activated MEs and finally provides the result of the application back to the device that triggered the request.

The depicted scenario where objects collaborate by mashing their services has great potentials as this allows for the deployment of powerful applications. This is the case of objects (e.g. cars) that share information to decide on the best route to get to a destination, objects that perform collaborative spectrum sensing and objects that need to send alarms to all the people in a given area to reach humans nearby, just to cite few examples. Reaching the right device(s) with whom interact is a key task in this context, and the SIoT provides a potentially effective approach to this by relying on the created social network. However, to evaluate the relevant performance, there is the need to model the generation of the query characterizing these scenarios, which should help in conducting a proper system performance evaluation. Such a model should describe which object (with relevant characteristics) would typically need to retrieve information from any other objects with other relevant features. Whereas the query model that is proposed in the following is adopted to evaluate the performance of the SIoT paradigm, it can be adopted for other IoT architectures as well. Finally, since our goal is to model the objects behavior when requesting services at the application layer, in this paper we do not consider how the query is handled.

3.2. Nodes and network modeling

In our modeling, the set of nodes in the SIoT, i.e., the set of SVOs, is represented by $\mathcal{N} = \{n_1, ..., n_i, ...n_I\}$ with cardinality I, where n_i represents a generic SVO. Its physical counterpart can be static or mobile with position $\boldsymbol{L}_i = \left[l_i^a, l_i^b\right]$, which can then be fixed or varying over time. In our problem setting, SVOs create social relations so let the resulting social network be described by an undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{E} \subseteq \{\mathcal{N} \times \mathcal{N}\}$ is the set of edges, each representing a social relation between a couple of nodes.

The modeling of our problem can not overlook the different typologies of objects in a network, since objects with different profiles can provide different services and are interested in different applications [28]. We then define the following sets: $\mathcal{T} = \{t_1, ..., t_x, ...t_X\}$ as the set of possible typologies of objects, such as smartphones, cars, traffic lights, and others. For every typology t_x, we define a set $\mathcal{B}_x = \mathcal{B}(t_x) = \{b_{x1}, ..., b_{xy}, ...b_{xY}\}$ as the set of possible brands inside the typology t_x, while the set $\mathcal{M}_{xy} = \mathcal{M}(b_{xy}) = \{m_{xy1}, ..., m_{xyz}, ...m_{xyZ}\}$ represents the set of possible models for typology t_x and brand b_{xy}. All the possible models available in the network can then be described by the set $\bar{\mathcal{M}} = \{\cup_{\forall xy} \mathcal{M}_{xy}\}$, which allows us to define the following 2-tuple $\Gamma = \left\langle \mathcal{N}, \bar{\mathcal{M}} \right\rangle$, which associates to every node n_i the corresponding model of the device and thus enables also to infer the typology and the brand. This tuple will be useful to enable the creation of the parental relation (i.e. the POR defined in Section 2.1), which is based on these characteristics.

Then, we need to define the applications in the network, which are those that are requested during the querying process and the possible services provided by the nodes and that can satisfy the queries. Let $\mathcal{A} = \{a_1, ..., a_w, ...a_W\}$ be the set of possible applications that can be installed by the devices in our scenario. However, applications do not run on all the devices but only on those they are meant to, so a single device will only have a subset of applications installed on it; we can then define the matrix $\boldsymbol{O} = [o_{iw}]$ where the generic element o_{iw} is equal to 1 if node n_i can potentially install application a_w and 0 otherwise.

Then, we define $\mathcal{S} = \{s_1, ..., s_j, ...s_J\}$ as the set of services that can be performed by any node in the network and that can be used to compose the applications in $\mathcal{A}$. Thus, we can define the matrix $\boldsymbol{D} = [d_{ij}]$, where the generic element d_{ij} is equal to 1 if node n_i can provide service s_j and 0 otherwise.

It is true that both the matrices of installed applications and available services, namely $\boldsymbol{O}$ and $\boldsymbol{D}$, should be related to the typology of the node, since it is the typology that determines the possible uses for an object. However, this approach is too simplistic since different nodes can offer different services and run different applications based on external characteristics related to their owner, such as privacy settings. Let us consider two users which own a smartphone each: one of them is willing to share all the smartphone's services while the other one only one or two of them; similarly, even if the set of applications they can install on their smartphone is the same, they have decided to install different applications based on their interests.

To model how an application generates a query, let's recall that a query

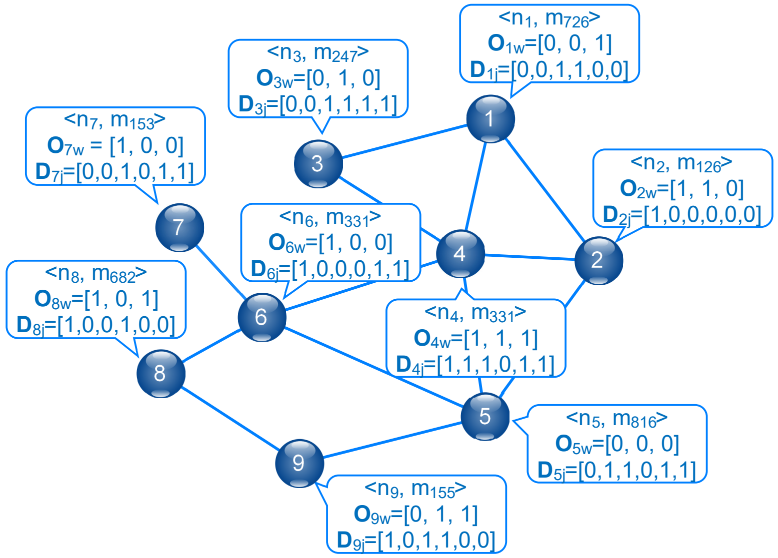

Figure 3: Representation of the network nodes.

only specifies the needed services and their requirements, and it is the aggregation layer that combines them to fulfill the request of the application. To this, we can model the query as the tuple ♀ $= \langle \mathcal{Q}^{serv}, \mathcal{Q}^{req} \rangle$, where $\mathcal{Q}^{serv} = \{q_1^{serv}, ..., q_h^{serv}, ...q_H^{serv}\}$ is the set of atomic queries representing the individual services needed to fulfill the application requests using a node's social network, while $\mathcal{Q}^{req} = \{q_1^{req}, ..., q_k^{req}, ...q_K^{req}\}$ is the set of requirements. The goal of a query generation model is then to calculate the probability to generate a specific query ♀. In our modeling, we make the assumption that the number of atomic queries matches the number of services to be found; nonetheless, based on the particular search mechanism implemented in the (S)IoT, the number of queries can be lower w.r.t. the number of services, since a query can be used to find two or more services at the same time. However, the modeling of the search engine is not considered in this paper.

Figure 3 provides a simple example of a generic SIoT graph $\mathcal{G}$, where $I = 9$ and each node is characterized by a tuple $\Gamma_i = \langle n_i, m_{xyz} \rangle$, which defines for node n_i its model m_{xyz}, from which we can infer the typology t_x and the brand b_{xy}. In our example, we can notice how nodes can share the same typology, as it is the case of nodes n_2, n_7, n_9, since they have the same first digit of the $\bar{\mathcal{M}}$ set, and even the same brand, as n_7 and n_9 are described by m_{153} and m_{155} respectively. In particular, if nodes belong to the same typology, brand and model, such as the case of nodes n_6 and n_4, they are then able to create a POR.

In this example, each SVO can have up to 3 applications installed, as indicated by the number of columns of matrix $\boldsymbol{O}$, and it is capable of providing up to six services, as shown by the column dimension of $\boldsymbol{D}$. Suppose that a

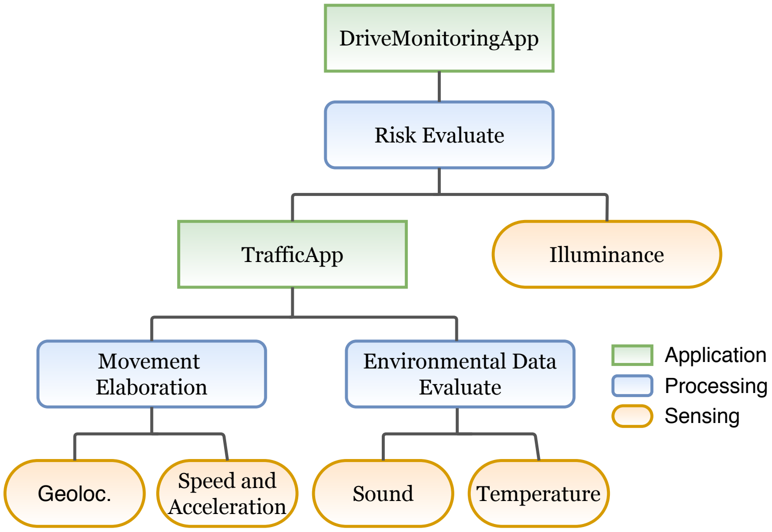

Figure 4: Decomposition of the *DriveMonitoringApp* application into services.

user, which owns node n_1, is interested in the *DriveMonitoringApp* application that monitors and evaluates his/her driving behavior and the related risks and then installs it in n_1. To provide the requested application, which is indicated in our example as o_{13}, to the user, the related SVO will have to search for the needed services, which are shown by the orange balloons in Figure 4 and that are indicated as the services s_1, s_2, s_3, s_5 and s_6 in our example scenario. Node n_1 will then generate a query $\qoppa$ with $\mathcal{Q}^{serv} = \{q_1^{serv}, q_2^{serv}, q_3^{serv}, q_4^{serv}, q_5^{serv}\}$ and $\mathcal{Q}^{req} = \emptyset$ to look for the five services among its friends. When all the services are retrieved, they are sent to the aggregation layer, which provides the necessary processing capabilities (blue balloons in Figure 4). Please note that in some cases the node could be able to provide some of the services by itself as the case of node n_1, which can provide service s_3 ($d_{13} = 1$).

As we will see in the next section, the query generation model is more complicated than this example, since it has to take into account space and time requirements.

4. Query Generation Model and Simulation

In the next subsection, we illustrate our query generation model whereas in the second subsection we evaluate the model performance in a real IoT scenario.

4.1. Query Generation Model

In the IoT, the number of possible applications is huge, but not all the types of things can install the same set of applications and even the same application installed in the same object can generate queries with different requirements.

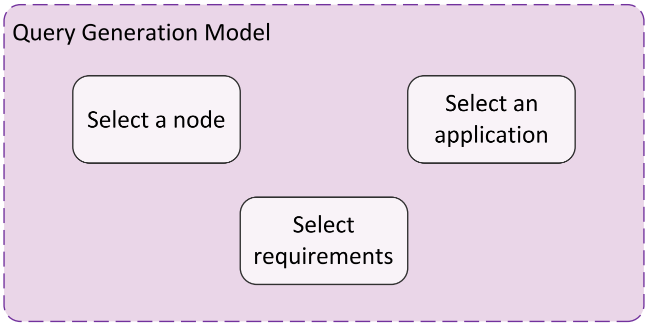

Figure 5: Query generation model functionalities

When studying the IoT, and in particular the Social IoT, it is difficult to evaluate the performance of service search mechanisms, i.e. how (S)IoT systems can fulfill application requests. This is due to the lack of query generation models, that are needed to understand which application can generate a query and with which requirements.

As described in Section 3.1, the goal of a query generation model is to compute the probability that a query $\mathcal{Q}$ will be generated; the composition of the atomic queries in $\mathcal{Q}^{serv}$ represents the set of services needed by the application. The choice of the application that will generate a query, and that will then determine the services to search, depends on the particular object in which the application is installed. Figure 5 shows the main functionalities that characterize the query generation model. According to this picture, based on the chosen application and on which node it is installed, the model has to generate the set of query requirements $\mathcal{Q}^{req}$, which are applied to the set of atomic queries.

Applications and nodes are highly intertwined: choosing a node determines which applications can be installed on that node, and selecting an application fixes the possible nodes in which the application can be installed. In order to obtain the probability to generate a specific set of atomic queries $\mathcal{Q}^{serv}$, that corresponds to application $\mathcal{A} = a_w$, we have to compute the joint density function of nodes $\mathcal{N}$ and application $\mathcal{A}$ as follows:

$$p_{\mathcal{A},\mathcal{N}}(a_w, n_i) = p(\mathcal{A} = a_w \cap \mathcal{N} = n_i) = \begin{cases} 0 & \text{if } o_{iw} = 0 \\ p_i(\mathcal{Q}^{serv}) & \text{if } o_{iw} = 1 \end{cases} \tag{1}$$

where $p_i(\mathcal{Q}^{serv})$ is the probability that node n_i, which potentially installed application a_w, generates the set of atomic queries $\mathcal{Q}^{serv}$. For $o_{iw} = 1$, it can also be written in terms of conditional distributions:

$$\begin{aligned} p_i(\mathcal{Q}^{serv}) &= p(\mathcal{N} = n_i | \mathcal{A} = a_w) * p(\mathcal{A} = a_w) = \\ &= p(\mathcal{A} = a_w | \mathcal{N} = n_i) * p(\mathcal{N} = n_i) \end{aligned} \tag{2}$$

Eq. 2 shows the double nature of the query generation process, which can begin both by selecting an application or a node.

The probability that the set of atomic queries $\mathcal{Q}^{serv}$ is generated by any node in $\mathcal{N}$ is then defined as

$$P(\mathcal{Q}^{serv}) = \sum_i p_i(\mathcal{Q}^{serv}) \tag{3}$$

The application selection greatly influences the difficulty of the search operations, since applications can have different levels of intricacy, ranging from simple ones, which only need one or two services, to complex ones, with nested applications and multiple services. Moreover, not all applications require information with the same frequency. To this, in Section 5.3, we will test several different distributions for the applications' frequency, namely $p(\mathcal{A} = a_w)$, to evaluate how the SIoT network reacts in terms of navigability.

The choice of the node affects both its geographical and social position. The first one is important since it influences the requirements of the query, while the position of the node in the social network impacts on the number of friends selectable and thus on the number of friends a node can rely upon when looking for services. Since there is no particular constraint in the choice of a node, i.e., every node has the same probability to trigger an application request, $p(\mathcal{N} = n_i)$ follows a uniform distribution.

Once the query for services has been generated, it is important to know which requirements are needed for the specific application, namely to generate the set of requirements for the query. Indeed, different nodes requesting the same application can also specify different attributes or characteristics for it. The set of possible requirements can be quite large, ranging from the accuracy of the sensed data to their precision; however, not all the requirements are always needed: the *only* ones that need to be declared, either explicitly or implicitly, are space and time. For example, an application that needs temperature measurements as inputs could be requested in different areas, such as in a room or a park (space requirement) and for different time intervals, as it is the case for historical or real-time data (time requirement). The minimum set of requirements can then be expressed as $\mathcal{Q}^{req} = \{q_{s1}^{req}, q_{s2}^{req}, q_t^{req}\}$, where q_{s1}^{req} and q_{s2}^{req} indicate the space requirements, namely for the x and y-coordinates, while q_t^{req} expresses the time requirement.

As suggested in [29], to describe the concept of interest in a specific point in space, the best distribution should be normal: to this, we describe the space requirements as a 2-dimensional normal distribution, where the probability density function can be expressed as follows:

$$f_{wi}(q_{s1}^{req}, q_{s2}^{req}) = \frac{1}{2\pi\sigma_{q_{s1}^{req}}\sigma_{q_{s2}^{req}}} * exp\left(-\frac{1}{2}\left[\frac{\left(q_{s1}^{req} - l_i^a - \mu_{q_{s1}^{req}}\right)^2}{\sigma^2_{q_{s1}^{req}}} + \right.\right.$$
$$\left.\left. + \frac{\left(q_{s2}^{req} - l_i^b - \mu_{q_{s2}^{req}}\right)^2}{\sigma^2_{q_{s2}^{req}}}\right]\right) \quad (4)$$

where $\mu_{q_{s1}^{req}}$, $\sigma^2_{q_{s1}^{req}}$ and $\mu_{q_{s2}^{req}}$, $\sigma^2_{q_{s2}^{req}}$ are the mean and variance values for the x and y-coordinates respectively.

All these values are application dependent, i.e. they depend on the particular application a_w at hand, but we have decided not to show such dependence in the above formula to keep it clean. In particular, when the mean values, $\mu_{q_{s1}^{req}}$ and $\mu_{q_{s2}^{req}}$, are both equal to 0, then the distribution is centered on the current position of the node n_i, namely l_i^a and l_i^b, i.e a node is looking for information around itself.

Also, the time requirement can be modeled using the time interest of applications, as suggested in [30], since objects require information mostly in real-time and less as we move farther in time, i.e. historical data. We modeled such behavior as an exponential distribution as follows:

$$f(q_t^{req}) = \begin{cases} 0 & \text{if } q_t^{req} > 0 \\ \lambda_a * exp\left(\lambda_a q_t^{req}\right) & \text{if } q_t^{req} \leq 0 \end{cases} \quad (5)$$

where λ_a is a constant, depending on the particular application at hand. The requirement for $q_t^{req} = 0$ means that the application is needed in real-time, while the values of $q_t^{req} < 0$ indicate that historical data are requested. Whenever a SVO receives a request with a temporal requirement, it will check if its stored data can satisfy the requirements, otherwise, it has to contact the physical objects to retrieve the data; however, in some cases, the SVO would not contact its physical counterpart, in order to avoid consuming resources.

Once the query has been generated, the goal of the SIoT system will be to find all the services in ♀ starting from the SVO of the node with the selected application, making use of its social relations to crawl the network.

As an example of query generation, let us consider the following flow: the system chooses randomly an object among the available ones, e.g. a car. This car can be interested in several applications, so the model has to pick one of them, based on how frequently they require information, e.g. the *DriveMonitoringApp* showed in Figure 4: the resulting set of atomic queries is then $\mathcal{Q}^{serv} = \{$ Geoloc., Speed and Acceleration, Sound, Temperature, Street Lights $\}$. The final step is to set the requirements for the application, that will be inherited by every ser-

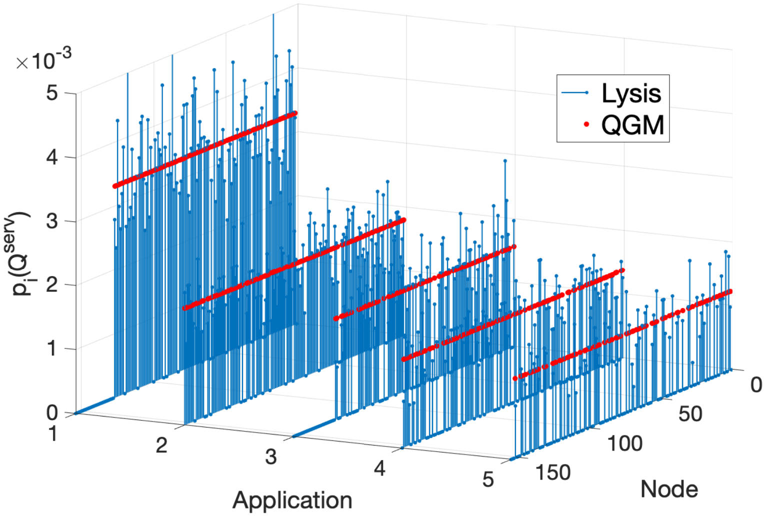

Figure 6: Query probability distribution for the Lysis data and for the Query Generation Model (QGM).

vice in $\mathcal{Q}^{serv}$: as spatial requirement, the car chooses an area of [100 m x2 km] around itself (i.e. in the road ahead), while as time requirement, the car selects $q_t^{req} = 0$ thus asking for the information to be obtained continuously in real-time.

The goal of the SIoT system will then be to find all the services in $\mathcal{Q}^{serv}$ starting from the SVO of the selected car, making use of its social relations to crawl the network.

4.2. Model Simulation

In order to simulate and validate the query generation model proposed, a set of real IoT queries is required. These data are obtained by the Lysis platform [25]: a collection of more than 11000 queries from 154 devices over a period of 7 months, from April 2017 to October 2017 (a complete description of the data is available here[1]). The network is composed of two types of nodes: smartphones and Raspberry boards; based on the typology, the devices can require up to five different applications.

Figure 6 illustrates how the proposed query generation model, displayed by red dots and labeled QGM, matches the probability for each node to generate a specific set of atomic queries $\mathcal{Q}^{serv}$ obtained from the Lysis dataset, represented with blue lines. Moreover, since not all the devices can install the same applications, then it will happen that some nodes will *never* require a given service and then *never* generate the corresponding query and thus $p_i(\mathcal{Q}^{serv}) = 0$.

---

[1] http://www.social-iot.org/index.php?p=downloads

Table 2: Requirements for the Lysis Applications.

App	$\mu_{q_{s1}^{req}}$	$\mu_{q_{s2}^{req}}$	$\sigma_{q_{s1}^{req}}$	$\sigma_{q_{s2}^{req}}$	$\mathbf{H_1}$	λ_a	$\mathbf{H_2}$
1	0	0	0	0	0	0.50	0.099
2	0	0	0.1	0.1	0.019	2.00	0.116
3	0	0	0.3	0.3	0.013	0.50	0.108
4	0.45	-0.45	0.1	0.1	0.012	10.0	0.118
5	0.70	-0.70	0.1	0.1	0.018	20.0	0.159

In our model, the nodes follow a uniform distribution, while the applications' frequency is proportional to the number of services needed by each application, i.e. that the first app requires more services than the last app. To evaluate the performance of our model, we made use of an f-divergence measure, namely the Hellinger distance [31], to quantify the similarity of the two probability distributions. Unlike other f-divergence measures, the Hellinger distance is a bounded metric: given two probability distribution P and Q, the maximum distance 1 is achieved if P and Q are completely divergent, while a distance $H(P, Q) = 0$ means that the two probability functions are completely overlapping and hence identical. In our case, the value of the Hellinger distance is equal to 0.0047, so we can conclude that our model is able to generate an almost identical distribution w.r.t. the real data.

Table 2 shows the parameters used to describe the space and time requirements for each of the five applications. The two columns labeled as H_1 and H_2 indicates the Hellinger distance between the real data from the Lysis dataset and our requirement distributions for space and time, respectively. The maximum value of the Hellinger distance is under 0.16 thus indicating a very good approximation of our model.

The values of the model's parameters, namely $\mu_{q_{s1}^{req}}$, $\mu_{q_{s2}^{req}}$, $\sigma_{q_{s1}^{req}}$, $\sigma_{q_{s2}^{req}}$ and λ_a, are computed by applying linear regression to a small set of interactions for each application (around 1% of the total number of requests).

5. Data Analysis and Simulations

This Section presents a dataset of profiles for the objects in a Smart City environment, based on the FIWARE Data Models [32]. The dataset is then used to construct a social network of objects, which is analyzed in Subsection 5.2. Finally, the last Subsection assesses the performance of the network when tested with the query generation model in terms of navigability.

5.1. Dataset

The main functionalities required to create a dataset are illustrated in Figure 7. As it will be better explained in the rest of the subsection, these functionalities are in charge of creating: objects' information (e.g. owner, typology, brand, model), traces of the positions and timestamps of the devices, the list of all the applications that can be installed by the objects, objects' profiles (expressed

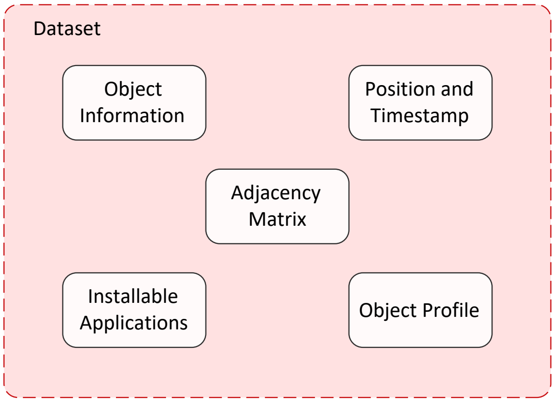

Figure 7: Dataset functionalities

as the set of the available services) and an adjacency matrix with the social relationships for each object.

The first step to construct a dataset is to obtain the profile of the objects. To this, we extracted objects' information and positions from the SmartSantander project [33], which is experimental research in support of typical applications and services for a smart city. We have classified all the available devices according to the data models proposed in the FIWARE Data Models. This enables the portability of the dataset among different platforms. These models consider both static and mobile objects and are mostly located in the city center of the city.

Each of the three public mobile categories of objects, namely buses, taxis and garbage trucks, moves in an independent way: buses' movement is created according to the list of bus stops, which are available from the Servicio Municipal de Tranportes Urbanos de Santander (TUS) [34]; taxis can start from 1 out of 3 taxi stations around the city; garbage trucks start from the landfill and cover all the city.

However, a complete Smart City scenario must also consider devices from private users. To this, we introduce 4000 users in the city, so that each user owns a certain number of devices. The devices' distribution is based on the ownership report of the Global Web Index in 2017 [35] calculated over 50000 users aged among 16 and 64 years old and it is shown in Table 3; some of these devices are considered mobile, i.e. they are carried by the users during their movements, while others are static and are then left at the user's home.

To simulate the mobility of these 4000 users, we rely on the well-known mobility model *Small World In Motion* (SWIM) [36]. SWIM can generate synthetic data, which can create mobility traces able to mimic human social behaviors. In fact, it has been proven that the SWIM mobility model allows obtaining accurate matching between the output of the model and the most popular mobility traces available in CRAWDAD [37], generating data with the same statistical properties, such as in terms of inter-contact time between people. The simu-

Table 3: Distribution Ownership Devices over 50.000 Users Aged 16-64.

Mobile Devices	**Ownership (%)**
Smartphone	91
Car	55
Tablet	40
Smart Fitness	22
Smartwatch	5
Static Devices	**Ownership (%)**
Pc	84
Printer	53
Home sensors	15

lation area needs to match the city center of Santander, so since SWIM only considers areas of interest of unitary square, we had to scale down the city center (which roughly has an area of 4 km x 4 km) and then modify the model to avoid users to move towards uninhabited areas, such as the sea.

The simulator requires some additional parameters. The user perception radius, set to 0.015, indicates the distance within which a user, or in our case a device, can *see* all other users/devices; this parameter is set according to the communication range of a Wi-Fi connection [38] specifically scaled considering that the simulation area of SWIM is a unitary square. The parameter α, which can have values in the range $[0; 1]$, is used to determine whether the users prefer to visit popular sites (smaller values) rather than nearby ones (bigger values). It has been set to 0.9. The entire simulation covers a time-lapse of ten days.

Following the network modeling proposed in Section 3.2, the dataset as a total number of devices equals $I = 16216$, 14600 of which are private and 1616 are public. The resulting network comprehends a total of $X = 16$ typologies of objects and to each of the typologies owned by private users a brand and a model selected randomly among $Y = 12$ brands and $Z = 24$ models have been assigned. We suppose that the municipality bought all the objects inside an object's typology with the same brand and model, so only the category is needed to classify public objects.

The devices of the smart cities, compared to the dataset in [4], are able to provide $J = 18$ services, which can be arranged to provide $W = 28$ different applications for the users.

A complete description of the data obtained in this paper is available for tests here[2] and includes objects' information (such as owner or typology), traces on the positions and timestamps of the devices, the list of all the applications we envision in a Smart City scenario, objects' profiles (expressed as set of available services and possible applications requests) and an adjacency matrix with the social relationships for each object.

---

[2] http://www.social-iot.org/index.php?p=downloads

5.2. Network Analysis

Based on object movements and profiles, each device can create its own set of relations with other devices. All the relations depend on the rules set in the system: as explained in [39], these rules have a direct impact on the overall navigability of the network: for the overall network to be navigable, i.e. to enable a node to easily reach any other node in the network, all, or the most of, the nodes must be connected, i.e., a giant component must exist in the network, and the effective diameter must be low. Moreover, the distribution of the number of connections each node has with its peers, namely the degree distribution, should be close to a power-law distribution. This results in a scale-free network and indicates the presence of hubs, i.e. nodes with a large number of connections w.r.t. the average, in the network. With this goal in mind, in the following, we discuss the characteristics of the obtained resulting network. The only relation we did not consider in these experiments is the C-WOR since it has been demonstrated from [39] that its contribution to the navigability of the network can be negligible.

All relationships, except for the OOR, are created using as a starting point [4]. An overview of the relations and their differences is illustrated below:

- The *Ownership Object Relationship* (OOR) is created between devices that belong to the same owner. To avoid too many relations, objects will create a relation only if they are in the communication range of each other. We assume that private devices use one out of three possible technologies: LoRa, Wi-Fi and Bluetooth.

- The *Parental Object Relationship* (POR) is created among two objects in the same category, brand and model. Since the reasoning behind the POR is to create long-distance links, two devices owned by private users, with the same typology, brand and model, will establish a relationship only if their distance is greater than a threshold, which is set to 3.8 km in order to reduce the number of relationships. For the public devices, a node is elected as a hub and all the other nodes with the same model will create a POR with the hub.

- Devices located in the same place can create a *Co-Location Object Relationship* (C-LOR). These relationships are created between a static device and a mobile one and do not take into account the contact duration but only the number of meetings between the two objects. A number of meetings equal to 10 has given an appropriate number of relations.

- The *Social Object Relationship* is a relation type that can be created among mobile devices and it is based on three parameters, namely the number of meetings (N), the meeting duration (T_M) and the interval between two consecutive meetings (T_I). These parameters are set to $N = 3$, $T_M = 15$ minutes and $T_I = 3$ hours, respectively.

- Mobile public object have hardly any chance to create SORs, so in order to include them in the SIoT network, we introduce another specific type

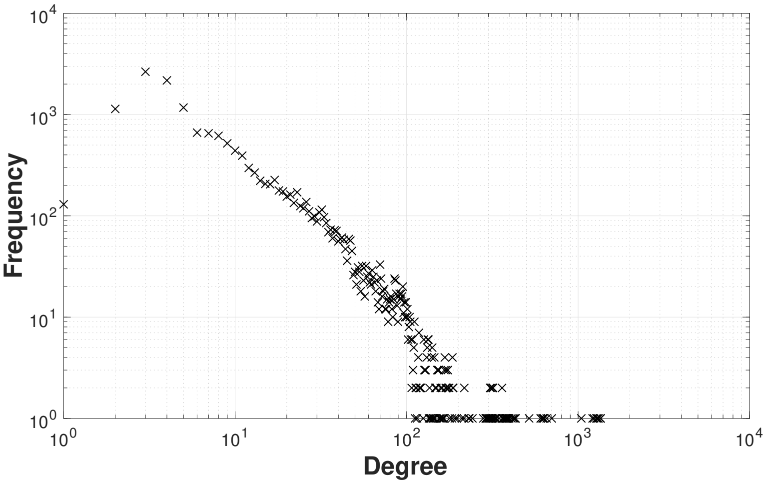

Figure 8: SIoT Degree Distribution.

Table 4: Relationships' parameters.

Parameters	**OOR**	**POR**	**C-LOR**	**SOR**	$\mathbf{SOR_2}$	**SIoT**
Number of relationships	58173	21245	27440	21245	20910	146117
Giant component (%)	8.99	4.17	51.28	24.06	15.23	100
Average degree	50.01	2.00	6.59	10.89	16.93	18.02
Average path length	2.15	1.99	27.31	4.34	3.01	4.22
Diameter	5	2	69	8	7	8

of SOR. This SOR, which we called $\mathbf{SOR_2}$, uses the same parameter of the SOR but with less stringent constraints; in particular, we set them to $N = 2$, $T_M = 2$ minutes and $T_I = 1$ hour.

The resulting distribution for the network is shown in Figure 8, while Table 4 shows the main network parameters for each relation and the whole network. We can notice that the SIoT degree distribution can be approximated to a power-law distribution, thus indicating its navigability. This is due to the presence of C-LOR and SOR, while the OOR and POR, which originate from other parameters, i.e., nodes characteristics and number of devices owned by a user, deviate from such a distribution: however, these relationships are still important since they connect groups of nodes so that the majority of nodes have more than one connection.

The average degree indicates the average number of edges connected to each node: OOR is the relation that creates the greatest number of friendships, however, it only creates small clusters of highly interconnected objects and thus the dimension of the giant component, the highest percentage of nodes belonging to the largest finite fraction of the entire graph's nodes, is low. Similar reasoning also applies to the POR: since the goal of the POR is to create long-distance links, the relation is created only if the distance between two devices is greater than a threshold so that the resulting number of relations is lower w.r.t. OOR. Finally, OOR and POR are able to create a highly connected cluster has can be inferred by the low values of the average path length, which is the average number of steps along the shortest paths for all possible pairs of network nodes. In order to connect the public mobile devices (buses, taxies and garbage trucks) we had to add another type of relation, that we called SOR_2: this relation makes use of the same parameter of the SOR but considering less stringent requirements. The contribution of the SOR_2 to the navigability of the network is the same as C-LOR and SOR: since all these relations create short distance links among devices regardless of their characteristics, they are able to connect the cluster of objects created by OORs and PORs. The resulting SIoT network then comprehends a giant component with all the devices where the longest shortest path between any two nodes, i.e. the diameter of the network, is still low.

This result is important since it ensures that every query generated by any object in the network can be fulfilled.

5.3. SIoT network navigability

The navigability in a network indicates how a node can reach any other peer and thus represents a fundamental parameter both for the generation of a network and to understand the average distance between cooperating nodes. To test the navigability of our dataset and query generation model, we have chosen the object typology with the highest number of requested applications, the smartphones, and analyze 1000 processes of the query generation model. All the results are shown with a 95% confidence interval around the mean value, i.e. that 95% of the values from the distribution lie within ±1.96 standard deviations.

The network's response is calculated in terms of average distance, i.e. the average number of hops needed to find all the required services, computed on the number of services of the application. This is done to avoid disparity among applications that require a different number of services: if, for example, the application depicted in Figure 4 is satisfied in three hops, that means that the five services composing it are found by the search engine in 15 hops, and then with an average of three hops each. This is also justified by the fact that the services can be found in parallel; however, in this paper we do not implement any specific searching mechanism, i.e. we are not using any mechanism for a node to navigate the network on its own with local information. On the other hand, we compute the distance among two nodes in terms of global network navigability, i.e., routing is performed by assuming that each object has a view about the global social network topology.

In order to compare the performance of the query generation model for the SIoT, we also created two other networks with similar characteristics: a

Table 5: Characteristics of the Random, Barabási-Albert and SIoT networks.

Parameters	**Random**	**BA**	**SIoT**
Number of relationships	145852	146449	146117
Average degree	17.99	18.06	18.02
Average path length	3.68	3.17	4.22
Diameter	5	5	8

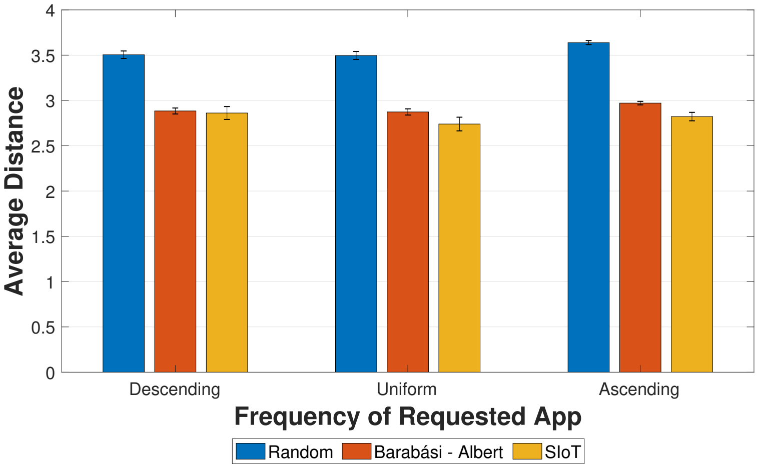

Figure 9: Average number of hops needed to solve a query. The applications' frequency changes based on the number of services requested by the application itself.

Random network and a Barabási-Albert network, which is able to generate scale-free networks based on preferential attachments [40]. The characteristics of the three networks are shown in Table 5: we can see how, at a global level, the SIoT has a higher average path length and diameter w.r.t the other two networks.

The first set of simulations focuses on the impact of the applications' frequency. Queries are then generated with a frequency related to the number of services needed by the application and without any kind of requirements, i.e. the network has to find all the nodes that can provide the required service. The results are shown in Figure 9.

We can notice that the SIoT network is able to outperform the other two networks, independently of the frequency. This is due to the fact that the SIoT relations are created to connect nodes with similar interests, so as to facilitate the discovery of information. Moreover, the impact of the applications' frequency is negligible. This result can be explained considering that the final goal of a search engine is to find the services needed by an application and that the same services can be arranged in several ways thus providing different

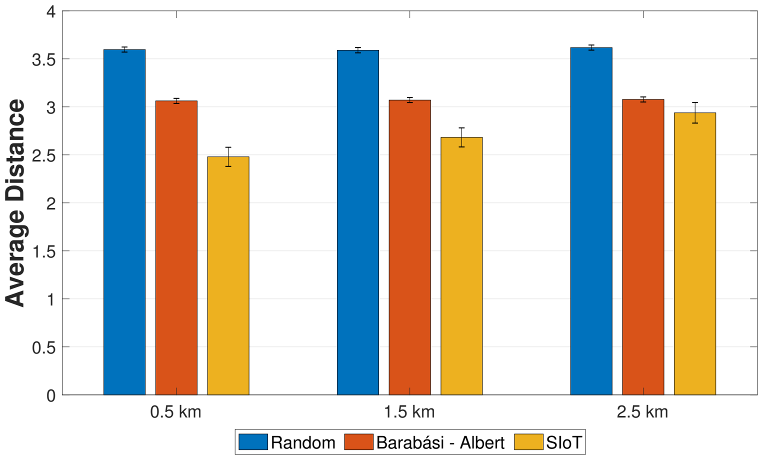

Figure 10: Average distance for different mean values of the space requirement.

applications: this means that even by changing the frequency, the services that need to be found are mostly the same. In the following, we will consider that all applications generate queries with the same frequency.

The second set of experiments consists of the analysis related to the space requirement: we first evaluate the impact of the mean values for the x and y-coordinates and then we investigate the effects of the variance.

Figure 10 shows the hop distance when nodes request applications that are located, on average, 0.5, 1.5 and 2.5 km away from them. This value is calculated as the Euclidean distance between the requester and the possible providers. We can note that there is a difference of almost half a hop between the two extreme cases, namely 0.5 and 2.5 km; even if the SIoT envisages the creation of long-distance links, such as the PORs, the greatest number of relations are created with nearby devices, so the best results can be obtained when a node looks for services in its vicinity. This is justified by Figure 11, which shows the average number of friends created within 1, 2 and 3 km from a node. We can note that, w.r.t. the other two networks, in the SIoT the greatest number of relations are created with nearby devices, so the best results can be obtained when a node looks for services in its vicinity.

To test the impact of the variance, shown in Figure 12, we consider different values that can cover respectively 50, 100 and 500 meters. As expected, the bigger the variance, the bigger the number of nodes that can provide the requested services and thus is simpler for the search engine to quickly find them.

The third set of simulations focuses on the time requirements, i.e. how fresh the information a node is requesting must be. As explained before, the search mechanism is performed at the virtual level, where the virtual counterparts store the information provided by the physical objects: however, this information can

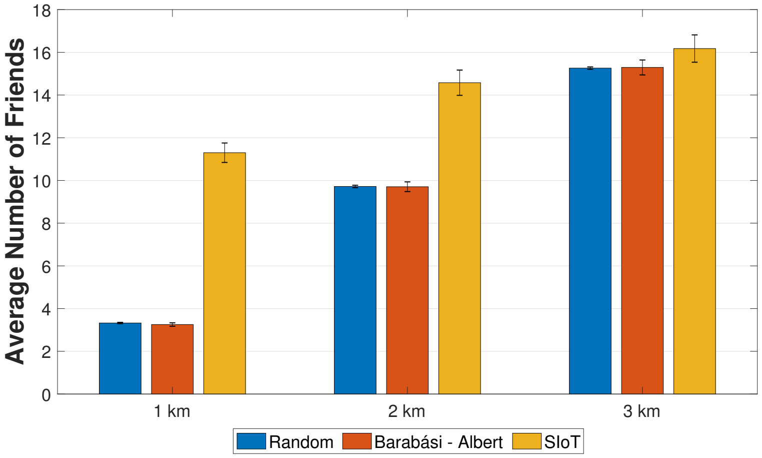

Figure 11: Average number of friends for objects within an area of 1, 2 and 3 km radius.

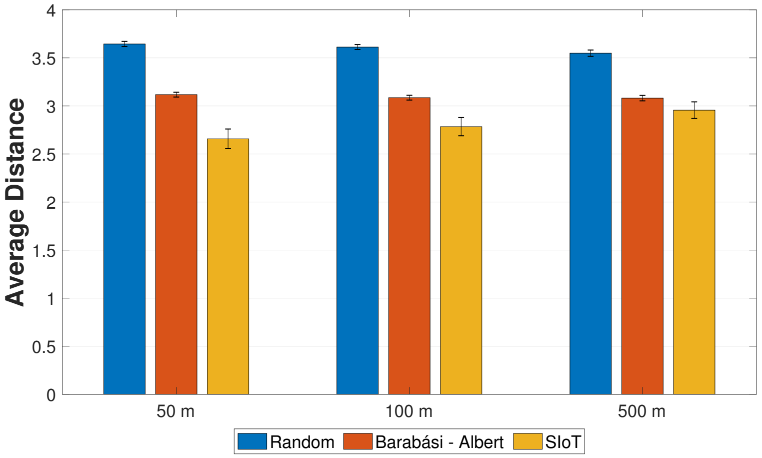

Figure 12: Average distance for different variance values of the space requirement.

Table 6: Synchronization time for all types of objects

Typologies	**Sync. Time**
Car, Indicator, Smart Fitness, Street lighting and Waste Management	7 minutes
Alarms, Home sensors, Parking, Smartphone, Smartwatch, Tablet and Transportation	17 minutes
Environment, Pc and Weather	23 minutes
Point of Interest and Printer	29 minutes

not be always synchronized with the ones sensed by the objects due to energy and bandwidth constraints. Based on the characteristics of the objects, every typology has a different synchronization time (see Table 6), so it may happen that the information found by the search engine is not fresh enough. In this case, the SVO interacts with the physical object, and then consumes its resources, to ask for additional reading in order to satisfy the query. Synchronization times are chosen as prime numbers to avoid that a large number of objects upload information to their corresponding SVOs at the same time. At the start of our simulation (time 0), all the devices synchronize their data with the corresponding SVO, and then they follow the synchronizations depicted in Table 6. So at any point in time, when a request with a temporal constraint arrives, we are able to compute if the SVO can satisfy it with its information or it has to contact the physical device.

The following results are shown only for the SIoT network; indeed, the network is created considering only space parameters, so there are no further differences among the networks. Figure 13 shows the average distance to satisfy a query looking for information generated within 1, 5 or 10 minutes. We run 100 query processes and each process is repeated 10 times. Relaxing the time constraint, as it happens with a 10 minutes requirement, leads to results similar to those obtained without any requirement for the query; on the other hand, the number of hops increases when the application is requested within a short time. As we approach the real-time requirement, $q_t^{req} = 0$, we can note that some points start to be missing from the curves: in particular, when requesting applications with a 1-minute requirement, the corresponding curve has no data for processes 12, 74 and 77. This means that the search engine has not been able to find any SVO satisfying the query in any of the 10 runs.

We then decided to analyze the number of times an SVO has to contact its physical counterpart during the 10 runs to satisfy the query. Figure 14 shows the corresponding results: as expected, with 1-minute requirement and during processes 12, 74 and 77, the SVO had to contact the physical object for all 10 runs and 6.43 times on average over the 100 processes, while with the 10-minutes requirement it is always possible to find an SVO with the required service. Finally, with the 5-minutes requirement, the physical objects are contacted on average less than once for each process (0.77 times).

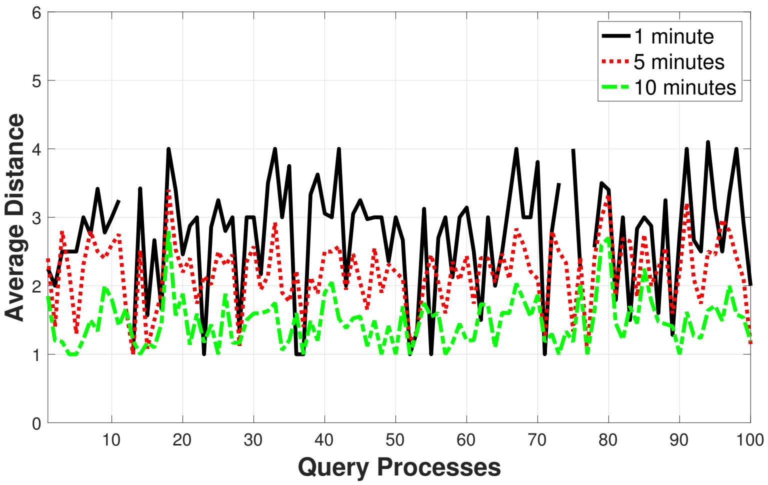

Figure 13: Average distance for different values of the time requirement.

The last set of simulations concerns the performance of the network to satisfy a complete query with both space and time requirements. To this, we decide to create a generic query requesting an application 1.5 kilometer away from the requester (mean value) with a range of 200 meters (variance value) and with information related to the last 5 minutes (time value). Figure 15 on the left axis shows the distance to solve the query, which is 3.16 hops over the 100 processes.

Finally, on the right axis of Figure 15, we try to analyze if hubs are involved in the search process by plotting the average degree of the intermediate nodes between requester and provider, i.e. the degree of the nodes forwarding the query. Given that the global network navigability returns the best possible path, it is also able to find the best intermediate nodes, i.e. the hubs in the network. By analyzing the degree of such nodes, we provide hints to the development of local routing algorithms, that should make use of these nodes. The average degree for the requesters is 17.43 friends, which is in line with the average degree of the network, while the average degree for the providers is 50.25 friends. However, when studying the degree of the intermediate nodes we find that its value is 149.08 connections thus confirming that the hubs are a crucial part of the search mechanisms in the SIoT.

6. Conclusions

In this paper, we have proposed a query generation model that can be used to analyze the performance of search and discovery mechanisms in the SIoT. To define the model, we have generated a dataset, which is based on real IoT objects, available in the city of Santander, and makes use of people mobility

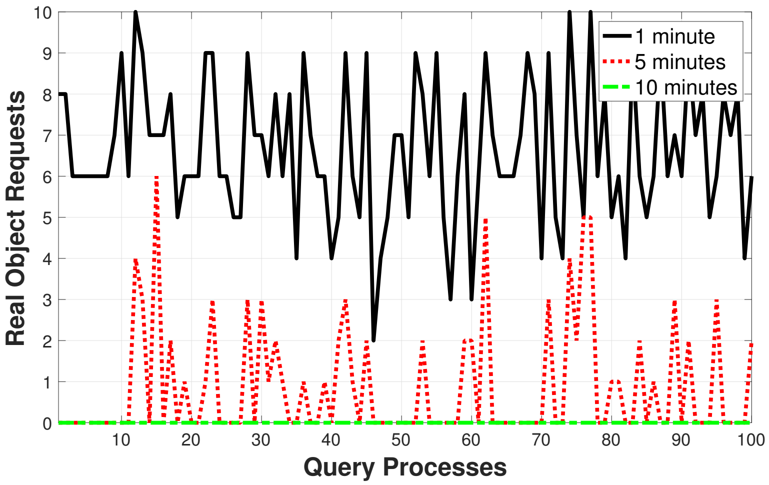

Figure 14: Number of request to real world objects due to the time requirement.

models. The dataset and the resulting social net- work are made available to the research community in order to test several (S)IoT management algorithms.

Moreover, we introduce a query generation model, which is able to generate applications requests from any given node. Our simulations have proven that, if opportunely tuned, our model is able to generate a query probability distribution almost identical to the one obtained from real data.

Moreover, through experimental analysis we were able to com- pare the SIoT networks obtained from our dataset with two other network, namely a Random network and a Barabási-Albert network. Even if, at the global level, the SIoT network shows worse parameters w.r.t. the other two networks, when tested with our query model, the SIoT network is able to outperform them: in particular, we tested the average distance in terms of the number of hops between requester and provider to respond both to simple queries with no requirements and even to more complex queries with space and time requirements.

Acknowledgements

This work was partially supported by Italian Ministry of University and Research (MIUR), within the Smart Cities framework (Project Netergit, ID: PON04a200490), within the PON R&I 2014-2020 framework (Project AIM (Attrazione e Mobilit Internazionale), ID: AIM1873058-1) and by the Italian Ministry of Economic Development (MiSE, Project INSIEME, HORIZON 2020, PON 2014/2020 POS. 395).

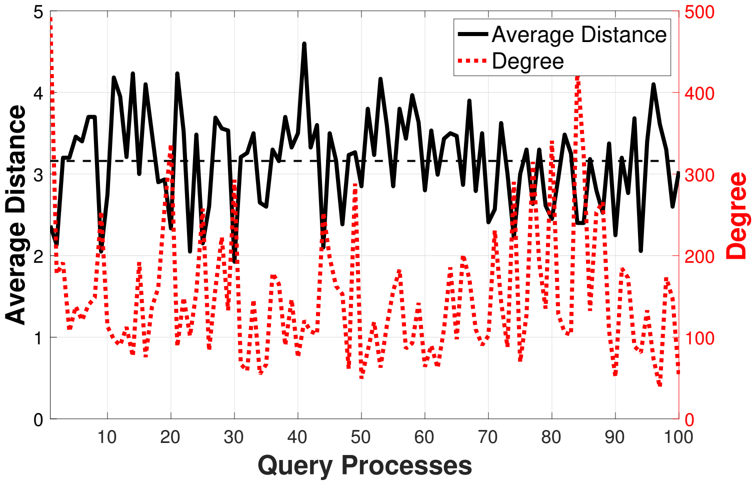

Figure 15: Average Distance and average of intra-nodes degree for queries with both requirements, *Space* and *Time*.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE Communications Surveys & Tutorials 17 (4) (2015) 2347–2376.

[2] K. Li, L. Tian, W. Li, G. Luo, Z. Cai, Incorporating social interaction into three-party game towards privacy protection in iot, Computer Networks 150 (2019) 90–101.

[3] L. Atzori, A. Iera, G. Morabito, M. Nitti, The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization, Computer Networks 56 (16) (2012) 3594–3608.

[4] C. Marche, L. Atzori, M. Nitti, A dataset for performance analysis of the social internet of things, in: 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), IEEE, 2018, pp. 1–5.

[5] M. Nitti, R. Girau, L. Atzori, Trustworthiness management in the social internet of things, IEEE Transactions on knowledge and data engineering 26 (5) (2014) 1253–1266.

[6] H. Xia, C.-q. Hu, F. Xiao, X.-g. Cheng, Z.-k. Pan, An efficient social-like semantic-aware service discovery mechanism for large-scale internet of things, Computer Networks 152 (2019) 210–220.

[7] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, N. Crespi, The cluster between internet of things and social networks: Review and research challenges, IEEE Internet of Things Journal 1 (3) (2014) 206–215. doi:10.1109/JIOT.2014.2318835.

[8] S. Büttcher, C. L. Clarke, G. V. Cormack, Information retrieval: Implementing and evaluating search engines, Mit Press, 2016.

[9] N. Ramachandran, V. Perumal, S. Gopinath, M. Jothi, Sensor search using clustering technique in a massive iot environment, in: Industry Interactive Innovations in Science, Engineering and Technology, Springer, 2018, pp. 271–281.

[10] P. Barnaghi, A. Sheth, On searching the internet of things: Requirements and challenges, IEEE Intelligent Systems 31 (6) (2016) 71–75.

[11] W. T. Lunardi, E. de Matos, R. Tiburski, L. A. Amaral, S. Marczak, F. Hessel, Context-based search engine for industrial iot: Discovery, search, selection, and usage of devices, in: 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), IEEE, 2015, pp. 1–8.

[12] N. K. Tran, Q. Z. Sheng, M. A. Babar, L. Yao, W. E. Zhang, S. Dustdar, Internet of things search engine: Concepts, classification, and open issues, arXiv preprint arXiv:1812.02930.

[13] M. Liu, D. Li, Q. Chen, J. Zhou, K. Meng, S. Zhang, Sensor information retrieval from internet of things: Representation and indexing, IEEE Access 6 (2018) 36509–36521.

[14] S. Wan, Y. Zhao, T. Wang, Z. Gu, Q. H. Abbasi, K.-K. R. Choo, Multi-dimensional data indexing and range query processing via voronoi diagram for internet of things, Future Generation Computer Systems 91 (2019) 382–391.

[15] Y. Yang, J. O. Pedersen, A comparative study on feature selection in text categorization, in: Icml, Vol. 97, 1997, pp. 412–420.

[16] R. Jones, R. Ghani, Automatically building a corpus for a minority language from the web, in: Annual Meeting-Association for Computational Linguistics, Vol. 38, UNKNOWN, 2000, pp. 29–36.

[17] D. Mladenic, M. Grobelnik, Feature selection for unbalanced class distribution and naive bayes, in: ICML, Vol. 99, 1999, pp. 258–267.

[18] R. Ghani, R. Jones, D. Mladenic, Online learning for web query generation: Finding documents matching a minority concept on the web, in: Web Intelligence: Research and Development, Springer, 2001, pp. 508–513.

[19] D. Boley, M. Gini, R. Gross, E.-H. S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, J. Moore, Document categorization and query generation on the world wide web using webace, Artificial Intelligence Review 13 (5-6) (1999) 365–391.

[20] J. Lafferty, C. Zhai, Document language models, query models, and risk minimization for information retrieval, in: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2001, pp. 111–119.

[21] Q. Mei, H. Fang, C. Zhai, A study of poisson query generation model for information retrieval, in: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2007, pp. 319–326.

[22] V. Lavrenko, W. B. Croft, Relevance-based language models, in: ACM SIGIR Forum, Vol. 51, ACM, 2017, pp. 260–267.

[23] G. L. Somlo, A. E. Howe, Using web helper agent profiles in query generation, in: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM, 2003, pp. 812–818.

[24] H. K. Azad, A. Deepak, Query expansion techniques for information retrieval: A survey, Information Processing & Management 56 (5) (2019) 1698–1735.

[25] R. Girau, S. Martis, L. Atzori, Lysis: A platform for iot distributed applications over socially connected objects, IEEE Internet of Things Journal 4 (1) (2016) 40–51.

[26] R. Girau, S. Martis, L. Atzori, Lysis: A platform for iot distributed applications over socially connected objects, IEEE Internet of Things Journal 4 (1) (2017) 40–51.

[27] M. Nitti, V. Pilloni, G. Colistra, L. Atzori, The virtual object as a major element of the internet of things: a survey, IEEE Communications Surveys Tutorials PP (99) (2015) 1–1. doi:10.1109/COMST.2015.2498304.

[28] I. Grønbæk, Architecture for the internet of things (iot): Api and interconnect, in: Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on, IEEE, 2008, pp. 802–807.

[29] Q. Z. Sheng, E. Stroulia, S. Tata, S. Bhiri, Service-Oriented Computing: 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings, Vol. 9936, Springer, 2016.

[30] V. Issarny, G. Bouloukakis, N. Georgantas, B. Billet, Revisiting service-oriented architecture for the iot: a middleware perspective, in: International conference on service-oriented computing, Springer, 2016, pp. 3–17.

[31] I. Sason, S. Verdu, f-divergence inequalities, IEEE Transactions on Information Theory 62 (11) (2016) 5973–6006.

[32] Fiware data models (2018).
URL `https://www.fiware.org/developers/data-models/`

[33] L. Sanchez, et al., Smartsantander: Iot experimentation over a smart city testbed, Computer Networks 61 (2014) 217–238.

[34] TUS-Santander, Tus santander (2018).
URL `http://www.tusantander.es`

[35] G. W. Index, GWI Social Q1 2017 (2017).

[36] S. Kosta, A. Mei, J. Stefa, Small world in motion (swim): Modeling communities in ad-hoc mobile networking, in: Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on, IEEE, 2010, pp. 1–9.

[37] J. Leguay, A. Lindgren, J. Scott, T. Riedman, J. Crowcroft, P. Hui, Crawdad trace upmc/content/imote/cambridge (v. 2006–11–17) (2006).

[38] R. Girau, S. Martis, L. Atzori, Neighbor discovery algorithms for friendship establishment in the social internet of things, in: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), IEEE, 2016, pp. 165–170.

[39] C. Marche, L. Atzori, A. Iera, L. Militano, M. Nitti, Navigability in social networks of objects: The importance of friendship type and nodes' distance, in: 2017 IEEE Globecom Workshops (GC Wkshps), IEEE, 2017, pp. 1–6.

[40] L. Zhu, L. Wang, X. Zheng, Y. Xu, The barabasi and albert scale-free network model, Journal of Intelligent & Fuzzy Systems 35 (1) (2018) 123–132.

Claudio Marche received the M.Sc. degree in telecommunication engineering with full marks in 2018 from the University of Cagliari. Since graduation, he has been working as Researcher in the Department of Electrical and Electronic Engineering at the University of Cagliari, in the MCLab research group. He is currently a Ph.D. student in Electronic and Computer Engineering at the University of Cagliari. His current research interests include Internet of Things (IoT), Social Internet of Things (SIoT) and Trustworthiness for IoT.

Luigi Atzori is currently a Professor with the Department of Electrical and Electronic Engineering, University of Cagliari, Italy, where he also leads the laboratory of Multimedia and Communications with around 15 affiliates. He is also the head of the CNIT Research Unit in Cagliari. His research interests are in multimedia communications and computer networking (wireless and wireline), with emphasis on multimedia QoE, multimedia streaming, NGN service management, service management in wireless sensor networks, and architecture and services in the Internet of Things. He served as a technical program chair for various international conferences and workshops. He has been an Area Editor and a Guest Editor for many journals, including ACM/Springer Wireless Networks Journal, the IEEE Communications Magazine, Springer Monet, Signal Processing: Image Communications Journals (Elsevier), the IEEE IOT JOURNAL, and Ad Hoc Networks (Elsevier). He served as a reviewer and panelist for many funding agencies, including H2020, FP7, Cost, and Italian MIUR.

Virginia Pilloni is an Assistant Professor at the University of Cagliari. In 2013 she was awarded with the PhD degree at the same university. From November 2011 to April 2012 she was a visiting PhD student at the Centre for Communication System Research at the University of Surrey. She has been involved in several national and international research projects, among which DEMANES (EU 7FP) and QoE-NET (EU H2020-MSCA-ITN-2014). She was in the organizing committee of 2018 QoMEX 2018 and chair of the 1st and 2nd Workshop on Smart Building and Internet of Things (2019 SBIoT at 2019 IEEE ICCCS and 2020 SBIoT to be held at 2020 IEEE GIoTS). She served as a reviewer for H2020 projects. Her main research interests are on the improvement of performance of Internet of Things nodes through task allocation, and on efficient scheduling of appliances in Smart Building scenarios.

Michele Nitti is an Assistant Professor at the University of Cagliari, Italy since 2015. In 2013, he has been a visited student at the Department of Management, Technology and Economics at ETH Zurich, Switzerland. He served as a technical program chair for various international conferences (IEEE BMSB 2017, IEEE IoT VT Summit 2020) and workshops (IEEE ICCCS 2019, IEEE GIoTS 2020) and he has been keynote invited speaker at the International Workshop of Trends and Challenges in Social Internet of Things (TC-SIoT2018) at PIMRC 2018 and at the Topical Area Track ”Green Technologies - Environment, Sustainability the Circular Economy” held during the IEEE World Forum on the Internet of Things 2019. Currently, he is a member of the editorial board for the Computer Networks Journal and co-founder of an academic spin-off (GreenShare s.r.l.) which works in the mobility sector. His main research interests are in architecture and services for the Internet of Things (IoT), particularly in the creation of a network infrastructure to allow the objects to organize themselves according to a social structure (IoT), Social Internet of Things (SIoT) and Trustworthiness for IoT.