The MIoT paradigm: Main features and an “ad-hoc” crawler

Giorgio Baldassarre, Paolo Lo Giudice, Lorenzo Musarella,
Domenico Ursino

PII: S0167-739X(17)32627-4
DOI: https://doi.org/10.1016/j.future.2018.09.015
Reference: FUTURE 4450

To appear in: *Future Generation Computer Systems*

Received date : 13 November 2017
Revised date : 12 March 2018
Accepted date : 5 September 2018

The MIoT paradigm: main features and an "ad hoc" crawler

Giorgio Baldassarre[1], Paolo Lo Giudice[2], Lorenzo Musarella[2], and Domenico Ursino[3]

[1] SISAL, Milano,
[2] DIIES, University "Mediterranea" of Reggio Calabria,
[3] DII, Polytechnic University of Marche
{giorgio.baldassarre@sisal.it, paolo.lo.giudice@unirc.it, d.ursino@univpm.it}

Abstract

The Internet of Things (IoT) is currently considered the new frontier of the Internet, and a lot of research results about this topic can be found in the literature. One of the most effective ways to investigate and implement IoT is based on the use of the social network paradigm: Social Internet of Things (SIoT) is an excellent attempt in this direction. In the last years, social network researchers have introduced new paradigms capable of capturing the growing complexity of this scenario. One of the most known of them is Social Internetworking System, which models a scenario comprising several related social networks. In this paper, we investigate the possibility of applying the ideas underlying Social Internetworking System to IoT, and we propose a new paradigm, called MIoT (Multiple Internets of Things), capable of modelling and handling the increasing complexity of this last context. Furthermore, in order to facilitate knowledge extraction and exploitation in presence of a huge number of things, we also propose a crawler specifically designed for an MIoT. Finally, through an experimental campaign, we show that classical crawlers are not adequate for MIoTs, whereas our own is well suited and outperforms all of them in this context.

Keywords: Internet of Things, MIoT paradigm, Social Internetworking System, Cross Nodes, Cross Edges, Crawling Strategies, Cross Node Driven Search

1 Introduction

The Internet of Things can be considered as an evolution of the Internet, based on the pervasive computing concept [9]. In the past, several strategies to implement the IoT paradigm and to guarantee ubiquitous computing have been proposed [24, 66, 16]. One of the most effective of them is based on the use of the social networking paradigm [7, 10, 8]. In this case, IoT is represented as a social network and, thanks to this association, Social Network Analysis-based models can be used to empower IoT. One of the most advanced attempts in this direction is SIoT (Social Internet of Things). In SIoT, things are empowered with social skills, making them more similar to people [7, 10]. In particular, they can be linked by five kinds of relationship, namely: *(i)* parental object relationship; *(ii)* co-location object relationship; *(iii)* co-work object relationship; *(iv)* ownership object relationship; *(v)* social object relationship. If: *(i)* a node is associated with each thing, *(ii)* an edge is associated with each relationship between things, and, finally, *(iii)* all the nodes and the edges linked by the same relationship are seen as joined together, SIoT can be modeled as a set of five pre-defined networks. Here, some nodes belong to only one network (we call them inner-nodes), whereas other ones belong to more networks (we call them cross-nodes).

The idea underlying SIoT is extremely interesting and, as a matter of fact, has received, and is still receiving, a lot of attention in the literature. However, we think that, in the next future, the number of relationships that might connect things could be much higher than five, and relationships could be much more variegate than the ones currently considered by SIoT. As a consequence, we think that a new paradigm, taking into account this fact, is in order.

In [11, 43], we introduced the concept of Social Internetworking System (SIS, for short) as a system comprising an undefined number of users, social networks and resources. The SIS paradigm was thought to extend the Single Social Network paradigm by taking into account that: *(i)* a user can join many social networks, *(ii)* these joins can often vary over time, and *(iii)* the presence of users joining more social networks can favor the cooperation of users, who do not join the same social networks. We think that the key concepts of SIS can also be applied to things (instead of to users) and to relationships between things and, in this paper, we propose the MIoT (Multiple Internets of Things) paradigm. The core of the SIS paradigm is modeling users and their relationships as a unique big network and, at the same time, as a set of related social networks connected to each other thanks to those users joining more than one social network. In this paper, we propose to extend the ideas underlying the concept of SIS to IoT. The MIoT paradigm arises as a result of this objective.

Roughly speaking, an MIoT can be seen as a set of things connected to each other by relationships of any kind and, at the same time, as a set of related IoTs, one for each kind of relationship. Actually, a more precise definition of MIoT would require the introduction of the concept of instance of a thing in an IoT. According to this concept, the instance of a thing in an IoT represents a virtual view of that thing in the IoT. Having this in mind, an MIoT can be seen as a set of related IoTs, one for each kind of relationship into consideration. The nodes of each IoT represent the instances of the things participating to it. As a consequence, a thing can have several instances, one for each IoT to which it participates. As will be clear in the following, the existence of more instances for one thing plays a key role in the MIoT paradigm because it allows the definition of the cross relationships among the different IoTs of the MIoT.

Differently from SIoT, in the MIoT paradigm, the number of relationships is not defined a priori. In an MIoT, there is a node for each thing; furthermore, there is an edge between two nodes if the corresponding things are linked by a relationship. If more kinds of relationship exist between two things, then more edges exist between the corresponding nodes, one for each kind of relationship. All the nodes linked by a given kind of relationship, together with the corresponding edges, form an IoT of the MIoT.

Observe that, under this MIoT definition, SIoT can be seen as a specific case of MIoT in which the number of the possible kinds of relationship is limited to 5 and these kinds are pre-defined. IoTs are interconnected thanks to those nodes corresponding to things involved in more than one kind of relationship. We call *cross nodes* (*c-nodes*, for short) these nodes and *inner nodes* (*i-nodes*, for short) all the other ones. Then, a c-node connects at least two IoTs of the MIoT and plays a key role to favor the cooperation among i-nodes belonging to different IoTs. As a consequence, differently from

SIoT, the nodes of an MIoT are not all equal: c-nodes will presumably play a more important role than i-nodes for supporting the activities in an MIoT.

Note that the MIoT paradigm can be seen as an attempt to address an open issue evidenced in [8] about some improvements that should be made on the SIoT paradigm. Among these improvements, two very relevant ones evidenced in this paper are the following:

- defining inter-objects relationships; this issue requires a correct representation of a smart object and the definition of both methods and tools to crawl and discover other (possibly heterogeneous) objects with which interactions can be established;
- modeling the new social networks thus obtained, characterizing them and defining new algorithms to perform their analysis.

The MIoT paradigm already mentioned, and the crawling strategy, which we present below, taken together, can represent an answer to these exigencies of improvement.

From a more applicative point of view, having some IoTs that can "communicate" through c-nodes can lead to some beneficial synergies. For instance, assume that an environment-related IoT can communicate with a home-related IoT through a cross node. Assume that the former IoT evidences an abnormal presence of dioxin in a place located some kilometers away from the home (for instance, owing to a fire of a plastic deposit). Assume, also, that this IoT is evidencing that the wind direction is pushing the dioxin towards the home. The home-related IoT could be "informed" through a cross node about this fact and could close all windows before the arrival of the dioxin.

Once an MIoT has been defined, it is possible to apply Social Network Analysis-based techniques on it to extract powerful knowledge concerning its things, their relationships, the IoTs formed by them, etc. However, in order to perform knowledge extraction, especially when the number of the things to investigate is huge, an important pre-requisite is having a good approach to crawl the underlying graph. Crawling is also extremely useful in a second family of applications, based on the exploration of the "neighborhood" (i.e., things and relationships) of a given thing (think, for instance, of the case in which a new thing is added to the Internet of Things and wants to create relationships with other things). There are also a lot of further possible applications of crawling, already known in the literature [44, 59], and that can be extended to the Internet of Things.

In the literature, several crawling strategies for single social networks have been proposed. Among them, the most representative ones are: *(i)* Breadth-First Search (BFS, for short) [65], which moves in breadth by exploring the neighborhood of each node; *(ii)* Random Walk (RW, for short) [38], which moves in random directions; *(iii)* Metropolis-Hastings Random Walk (MH, for short) [57, 32, 51], which moves in random directions, disfavoring high-degree nodes. These strategies were largely investigated for single networks, and their pros and cons have been highlighted in [21, 34].

However, we have seen that, in an MIoT, there exist two different kinds of node, and none of the previous strategies considers this fact, as they were developed for crawling a single network. We argue that a new strategy, capable of distinguishing c-nodes from i-nodes and of performing a right tradeoff between breadth, depth and randomicity, is in order. Therefore, a second objective of this paper is addressing this issue. In fact, we propose a new crawling strategy, called *Cross Node Driven Search*

(CDS, for short). CDS is centered on c-nodes; in fact, it allows users to privilege the visit of c-nodes over the one of i-nodes, if necessary, and to tune how much c-nodes should be privileged over i-nodes.

To prove the correctness of CDS, we tested it against the three main classic strategies mentioned above. In carrying out this task, we defined, and, then, used different metrics aimed to evaluate the quality of each crawler under consideration. The results of these experiments confirm our assumption about the inadequacy of the classic crawling strategies for an MIoT and, by contrast, the suitability of the new CDS strategy in this context.

This paper is organized as follows: in Section 2, we illustrate related literature. In Section 3, we present the MIoT paradigm. In Section 4, we describe the CDS crawler and illustrate the experimental campaign, which we performed to test it. In Section 5 we propose a comparison between our model and approach and other, more or less conventional, ones. Finally, in Section 6, we draw our conclusions and have a look at some future developments of our research efforts in this area.

2 Related Literature

Several years have passed since the IoT paradigm was introduced [6, 9, 41, 49]. During this period, the term "Internet of Things – IoT" has been associated with a huge variety of concepts, technologies and solutions. For instance, in the last few years, new technologies, such as Big Data [12] and Social Networking, have been applied to IoT and have changed, and are currently changing, the very definition of this term. What IoT will become in the future depends on the evolution of these technologies [60].

The current research on IoT focuses on the capability of connecting every object to the Internet. This way of thinking IoT led to the Web of Things (hereafter, WoT) paradigm [25, 24, 27] and to the application of Social Networking to the IoT domain [8]. In the next future, these technologies will be combined with other ones, such as Information Centric Networks [56, 66, 67, 4, 50, 5, 47] and Cloud [16, 58, 30]. As a matter of fact, the strengths of these last ones are exactly the features necessary to overcome the weaknesses of the current IoT concept [64]. Some examples of this combination can be already found in the literature [18, 2?, 63, 62].

Significant efforts have been made to apply the Social Networking ideas to the IoT domain. Actually, the implementation of reliable IoTs [7] passes through the definition of a complex architecture capable of managing services. In this research direction, the authors of [48] propose CASCOM, a model devoted to simplify the interaction between consumers and data in an IoT context. It is also necessary that this complex architecture enables a complete connectivity among things [33], guarantees quick reactions to frequent state variations and, finally, ensures a good scalability.

Furthermore, as IoT is based on the Internet, it must address the same security issues characterizing this network [28]. Therefore, the development of new architectures capable of fulfilling security and privacy requirements is in order [68].

The first attempts to apply Social Networking to the IoT domain can be found in [23, 42, 31, 26]. In these papers, the authors propose to use human social network relationships to share services provided by a set of things.

An important step forward is performed in [7], where the SIoT paradigm is introduced. Here, the authors propose an approach to creating relationships among things, without requiring the owner intervention. Thanks to this idea, things can autonomously crawl the network to find services and resources of their interest provided by other things. In [10], the same authors clearly highlight what are the main strengths of SIoT. Specifically: *(i)* the SIoT structure can be dynamically modified to ensure network navigability and to find new things; *(ii)* scalability is guaranteed, like in human social networks; *(iii)* a level of trustworthiness between things can be established; *(iv)* the past social network approaches can be redefined to solve problems typical of the IoT context [45].

Today, the connection level of humans and things is continuously increasing, so that it appears reasonable to start to investigate the "network of networks" scenario, thus passing from Social Networking to Social Internetworking. One of the most interesting attempts in this direction is Social Internetworking System (hereafter, SIS); it regards the connection of several human networks to form a network of human networks [11, 43]. The strength of SIS resides in the fact that this structure is capable of interconnecting users joining different social networks. In this new scenario, concepts and tools of Social Network Analysis can be adapted to evaluate the main features concerning the interactions between users belonging to the same network or to different networks. This new paradigm aims at guaranteing a tradeoff between the autonomy of each network of the SIS and the possibility of increasing power, efficiency and effectiveness, obtained through the interaction of the networks of the SIS. To the best of our knowledge, no architecture similar to SIS has been proposed for networks of things yet.

In [8], the authors point out that there are still several open issues that must be investigated in the SIoT paradigm. In particular, making things capable of establishing heterogeneous social relationships requires specific investigations and new approaches. Among them, the most relevant ones for our context are: *(i) Defining inter-objects relationships.* This task requires a correct digital representation of a smart object and the definition of a methodological and technological solution capable of crawling and discovering other (possibly heterogeneous) objects, with which interactions can be established. *(ii) Modeling the new social graphs thus obtained*, in such a way as to characterize them and to define new algorithms for performing their analysis.

Crawling represents a key issue for the implementation of the IoT paradigm. The necessity of addressing this issue is mentioned in many papers (e.g., [8, 39, 55, 20, 17], to cite a few). In spite of this high demand, just few papers addressing this problem can be found in the past literature on IoTs. Most of the approaches proposed in these papers focus on the creation of search engines conceived to operate on IoT [39, 40] or, more often, on the Web of Things [59, 15]. In [59], an accurate survey on this last research area is presented.

In [19], the authors propose a geo-based crawler for IoT aiming at minimizing inter-site communication costs. Every site uses its own crawler that is provided with some predefined rules for fetching and parsing the Web. In [17], a framework to automatize the search, and the next classification, of services belonging to a digital health ecosystem, is proposed. This framework exploits both a focused web crawler, which explores the network, and a social classification system. In [36], the authors propose an approach aimed at improving the existing web crawlers, when they operate on IoT, and to catch up the fingerprints of the IoT nodes. This approach is based on an incremental crawler, which periodically classifies nodes in such a way as to ensure the highest classification accuracy for the most important ones.

In [35], the crawling problem is approached from a different perspective. Indeed, one of the main problems in a network of things is battery consumption. To avoid it, in most cases, sensors perform a working-sleeping duty cycle. The authors of [35] model the crawling problem as a scheduling one and define a sleep-aware schedule method called EasiCrow. This method is well suited to crawl sensors with an asynchronous sleeping cycle. In [54], the authors, starting from the assumption that things are becoming the major producers and consumers of data, propose a system to extract data from different sources. Once data has been acquired, this system provides suitable interfaces allowing both humans and machines to share and dynamically search the services of their interest.

3 The MIoT paradigm

We define an MIoT $\mathcal{M}$ as a set of m Internets of Things (see Figure 1 for a schematic representation of it)[1]. Formally speaking:

$$\mathcal{M} = \{\mathcal{I}_1, \mathcal{I}_2, \cdots, \mathcal{I}_m\}$$

where $\mathcal{I}_k$ is an IoT.

Let o_j be an object of $\mathcal{M}$. We assume that, if o_j belongs to $\mathcal{I}_k$, it has an instance ι_{j_k}, representing it in $\mathcal{I}_k$. As pointed out in the Introduction, in this paper, the instance ι_{j_k} indicates a virtual view (or, better, a virtual agent) representing o_j in $\mathcal{I}_k$. For instance, it provides all the other instances of $\mathcal{I}_k$, as well as the users interacting with $\mathcal{I}_k$, with all necessary information about o_j. Interestingly, this information is represented according to the format and the conventions adopted in $\mathcal{I}_k$.

In $\mathcal{M}$, *a set MD_j of metadata* are associated with an object o_j. We define a rich set of metadata of an object, because these play a key role in favoring the interoperability of IoTs and of their objects, which is the main objective of an MIoT. As a consequence, MD_j consists of three different subsets:

$$MD_j = \langle MD_j^D, MD_j^T, MD_j^O \rangle$$

Here:

- MD_j^D represents the set of *descriptive metadata*. It denotes the type of o_j. For representing and handling descriptive metadata, a proper taxonomy, such as the one defined by the IPSO Alliance [1], can be adopted.

- MD_j^T represents the set of *technical metadata*. It must be compliant with the object type. In other words, there is a different set of metadata for each object type of the taxonomy. Also in this case, the IPSO Alliance provides a well defined set of technical metadata for each object type. It is worth pointing out that, in principle, we could have allowed much richer descriptive and technical metadata. However, we did not make this choice because we preferred to relate our definition of metadata to an international IoT standard, such as the one defined by the IPSO Alliance. Furthermore, as will be clear in the following, our approach needs mainly operational metadata. As a consequence, making descriptive and technical metadata more complex would have added a useless level of complexity to our model.

[1] In this paper, the term "IoT" is intended according to the new trends that characterize this research field [8]. These trends suggest that, with the explosion of the number of available things, it is not realistic to talk about a unique Internet of Things. By contrast, it is more appropriate to consider several IoTs, each consisting of a (social) network of things.

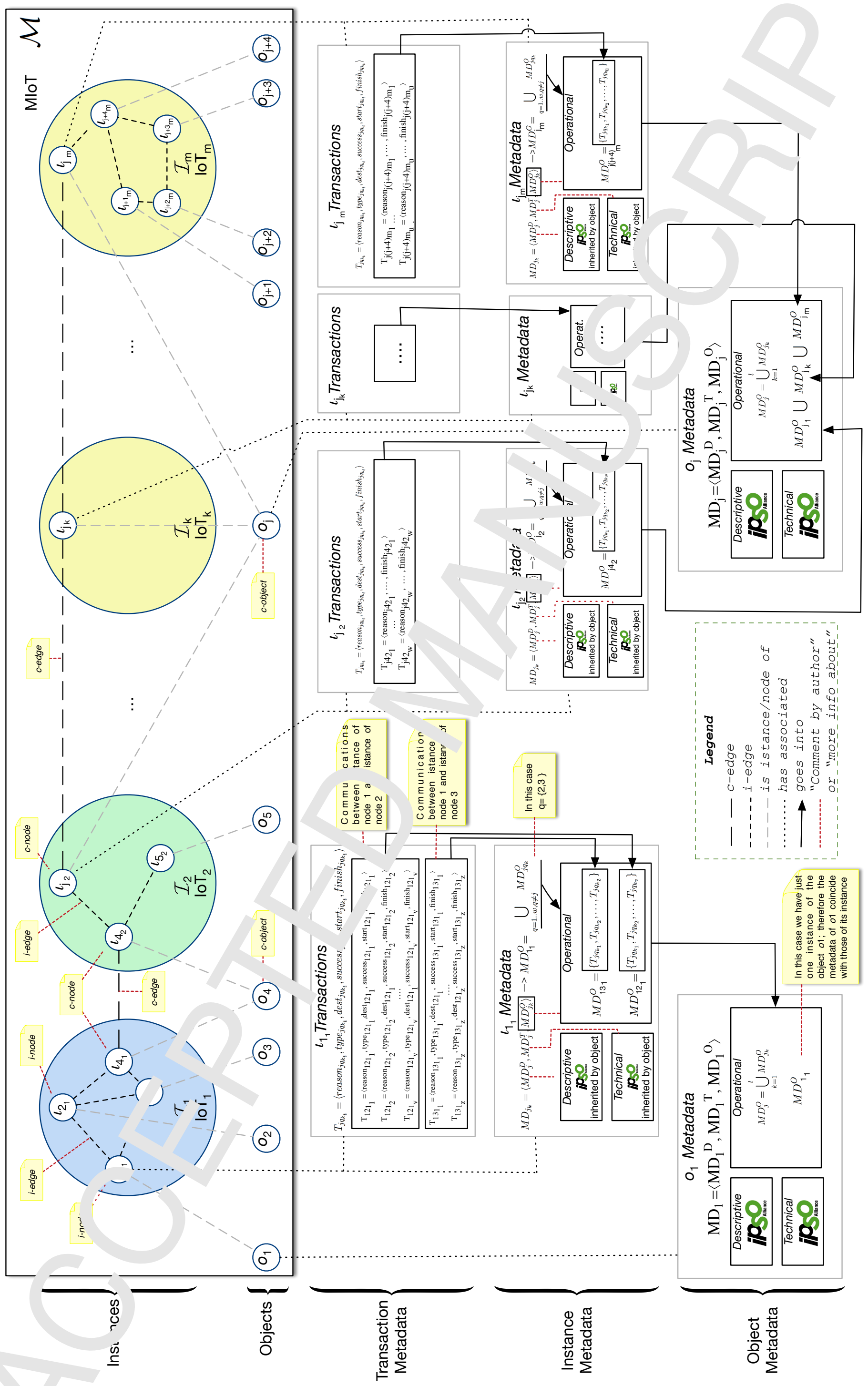


Figure 1: Schematic representation of the proposed MIoT structure

- MD^O_j represents the set of *operational metadata*. It regards the behavior of o_j. The operational metadata of an object o_j is defined as the union of the sets of the operational metadata of its instances. Specifically, let $\iota_{j_1}, \iota_{j_2}, \ldots, \iota_{j_l}$, $l \leq m$, be the instances of o_j belonging to the IoTs of $\mathcal{M}$. Then:

$$MD^O_j = \bigcup_{k=1}^{l} MD^O_{jk}$$

MD^O_{jk} is the set of the operational metadata of the instance ι_{jk}. In order to understand the structure of MD^O_{jk}, we first have to analyze the structure of $MD^O_{jq_k}$, i.e. the set of operational metadata between two instances ι_{j_k} and ι_{q_k}, of the objects o_j and o_q, in the IoT $\mathcal{I}_k$.

Specifically, $MD^O_{jq_k}$ is given by the set of metadata associated with the transactions between ι_{j_k} and ι_{q_k}. In particular:

$$MD^O_{jq_k} = \{T_{jq_{k_1}}, T_{jq_{k_2}}, \ldots, T_{jq_{k_v}}\}$$

where $T_{jq_{k_t}}$, $1 \leq t \leq v$, represents the metadata of the t-th transaction between ι_{j_k} and ι_{q_k}, assuming that v is the current number of transactions between the two instances.

$T_{jq_{k_t}}$ can be represented as follows:

$$T_{jq_{k_t}} = \langle reason_{jq_{k_t}}, type_{jq_{k_t}}, inst1_{jq_{k_t}}, inst2_{jq_{k_t}}, success_{jq_{k_t}}, start_{jq_{k_t}}, finish_{jq_{k_t}} \rangle$$

where:

– $reason_{jq_{k_t}}$ denotes the reason causing the transaction, chosen among a set of default values.

– $type_{jq_{k_t}}$ indicates the transaction type (e.g., unicast, multicast, and so forth).

– $inst1_{jq_{k_t}}$ and $inst2_{jq_{k_t}}$ denote the two instances involved in $T_{jq_{k_t}}$. Observe that a transaction between ι_{j_k} and ι_{q_k} could be part of a longer path whose source and/or target nodes could be different from ι_{j_k} and ι_{q_k}. In principle, the source and/or the target nodes of a transaction could belong to an IoT different from $\mathcal{I}_k$. In this last case, it is necessary to reach $\mathcal{I}_k$ from the source, and/or to reach the target from $\mathcal{I}_k$, through one or more cross nodes, if possible.

– $success_{jq_{k_t}}$ denotes if the transaction succeeded.

– $start_{jq_{k_t}}$ is the timestamp associated with the beginning of the transaction.

– $finish_{jq_{k_t}}$ is the timestamp associated with the end of the transaction (its value is NULL if $T_{jq_{k_t}}$ failed).

In our model, the direction of a transaction is not considered. Furthermore, the parameter v, i.e., the number of transactions for each pair of instances, varies when moving from a pair of instances to another.

Observe that we have made our model powerful enough to represent and handle all the transactions between two instances of each IoT. Having all these detailed historical data at disposal could help the analysis of the real "social" behavior of each object. Furthermore, these data could be exploited in many applications; think, for instance, of the computation of the trust and reputation of each object, the investigation of objects with similar or complementary behaviors, and so forth. On the other hand, maintaining a full history of transactions may be very expensive and useless in many real life applications; in some cases, suitable data summarizations could be enough. As a consequence, when passing from the abstract model definition to real life applications, the transaction representation could be removed, extended or restricted on the basis of a tradeoff between costs and benefits for the current application.

We are now able to define the set of the operational metadata MD^O_{jk} of an instance ι_{j_k} of $\mathcal{I}_k$. Specifically, let $\iota_{1_k}, \iota_{2_k}, \ldots, \iota_{w_k}$ be all the instances belonging to $\mathcal{I}_k$. Then:

$$MD^O_{jk} = \bigcup_{q=1..w, q \neq j} MD^O_{jq_k}$$

In other words, the set of the operational metadata of an instance ι_{j_k} is given by the union of the sets of the operational metadata of the transactions between ι_{j_k} and all the other instances of $\mathcal{I}_k$.

Given an instance ι_{j_k}, relative to an object o_j and an IoT $\mathcal{I}_k$, we define the metadata MD_{j_k} of ι_{j_k} as:

$$MD_{j_k} = \langle MD^D_j, MD^T_j, MD^O_{j_k} \rangle$$

In other words, the descriptive and the technical metadata of an instance ι_{j_k} coincide with the ones of the corresponding object o_j. Instead, the operational metadata of ι_{j_k} is a subset of the operational metadata of o_j that comprise only those ones regarding the transactions, which ι_{j_k} is involved in.

It is possible to associate a graph:

$$G_k = \langle N_k, A_k \rangle$$

with $\mathcal{I}_k$. Here, N_k indicates the set of the nodes of $\mathcal{I}_k$. There is a node n_{j_k} for each instance ι_{j_k} of an object o_j in $\mathcal{I}_k$. A_k denotes the set of the edges of $\mathcal{I}_k$. There is an edge $a_{jq_k} = (n_{j_k}, n_{q_k})$ if there exists a link between the instances ι_{j_k} and ι_{q_k} of the objects o_j and o_q in the IoT $\mathcal{I}_k$.

Also the overall MIoT $\mathcal{M}$ can be represented as a graph:

$$\mathcal{M} = \langle N, A \rangle$$

Here:

- $N = \bigcup_{k=1}^{m} N_k$;
- $A = A_I \cup A_C$, where:
  - $A_I = \bigcup_{k=1}^{m} A_k$;

– $A_C = \{(n_{j_k}, n_{j_q}) | n_{j_k} \in N_k, n_{j_q} \in N_q, k \neq q\}$; observe that n_{j_k} and n_{j_q} are the nodes corresponding to the instances ι_{j_k} and ι_{j_q} of the object o_j in $\mathcal{I}_k$ and $\mathcal{I}_q$.

In other words, an MIoT $\mathcal{M}$ can be represented as a graph whose set of nodes is the union of the sets of nodes of the corresponding IoTs. The set A of the arcs of $\mathcal{M}$ consists of two subsets, A_I and A_C. A_I is the set of the inner arcs of $\mathcal{M}$ and is the union of the sets of the arcs of the corresponding IoTs. A_C is the set of the cross arcs of $\mathcal{M}$; there is a cross arc for each pair of instances of the same object in different IoTs. We call:

- *i-edge* an edge of $\mathcal{M}$ belonging to A_I;
- *c-edge* an edge of $\mathcal{M}$ belonging to A_C;
- *c-node* a node of $\mathcal{M}$ involved in at least one c-edge;
- *i-node* a node of $\mathcal{M}$ not involved in any c-edge;
- *c-object* an object having at least one pair of instances whose corresponding nodes are linked by a c-edge; clearly, any object with at least two different instances is a c-object.

It is worth pointing out that, as mentioned in the Introduction, there is a strict correlation between the MIoT paradigm and the concept of Social Internetworking System (hereafter, SIS) already presented in the literature [11]. In particular: *(i)* the concept of c-edges shares several features with the one of "me"-edge in a SIS; *(ii)* the concept of c-node is similar to the one of bridge in a SIS; *(iii)* a c-object corresponds to a user joining more social networks.

3.1 An example of an MIoT

Since the MIoT paradigm is new, in the Internet there is no known case study or real example about it yet. As a consequence, to provide the reader with an example, and, at the same time, to have a testbed for our experiments, we constructed an MIoT starting from some open data about things available on the Internet. In particular, we derived our data from *Thingful* [2]. This is a search engine for the Internet of Things, which allows us to search among a huge number of existing things, distributed all over the world. Thingful also provides some suitable APIs allowing the extraction of all the data we are looking for.

In order to construct our MIoT, we decided to work with 250 things whose data was derived from Thingful. Given the huge number of things available in Thingful, it could appear that the number of things composing our testbed is excessively limited. However, we observe that:

- This was the first attempt to construct a real MIoT and, then, it was extremely important for us to have full control of it in order to verify if we were proceeding well. A full human control with a much higher number of nodes was not possible.
- We wanted to fully analyze the behavior, the strengths and the weaknesses of our crawler and to understand, step by step, its way of operating vs the ones of other crawlers. Again, a full human verification of these aspects was not possible with a larger testbed.

- As it will be clear in the following, our approach to obtaining the testbed is fully scalable. As a consequence, an interested researcher can apply it to construct a much larger testbed, if necessary.

We considered three dimensions of interest for our MIoT, namely:

a. *Category*: It specifies the application field which a given thing operates in. The categories we have chosen were five, namely *home*, *health*, *energy*, *transport*, and *environment*. Each category originated an IoT. Each thing was assigned to exactly one category.

b. *Coastal distance*: It specifies the coastal distance (i.e., the distance from any sea, lake or river) of each thing. The distance values we have set were:

   - *near*, for things distant less than 20 kilometres from the coast, for the categories *environment* and *energy*, and less than 5 kilometres, for the other three categories;
   - *mid*, for things whose minimum distance from the coast was between 20 and 105 kilometres, for the categories *environment* and *energy*, and between 5 and 25 kilometres, for the other three categories;
   - *far*, for things whose minimum distance from the coast was higher than 105 kilometres, for the categories *environment* and *energy*, and higher than 25 kilometres, for the other three categories.

   An IoT was created for each distance value. The different coastal distance values for *environment* and *energy*, on the one hand, and for the other three categories, on the other hand, have been determined after having analyzed the distribution of the involved categories of things against the coastal distance, in such a way as to produce a uniform distribution of each category of things in the three IoTs related to the coastal distance dimension.

c. *Altitude*: it specifies the altitude of the place where the thing is located. The altitude values we have defined were: *plain* (corresponding to an altitude less than 500 meters), *hill* (corresponding to an altitude between 500 and 1000 meters), and *mountain* (corresponding to an altitude higher than 1000 meters). An IoT was created for each altitude value.

As a consequence, our MIoT consists of 11 IoTs. We associated an object with each thing; therefore, we had 250 objects. In principle, for each object, we could have associated an instance for each dimension. However, in order to make our testbed closer to a generic MIoT, representing a real scenario, where it is not said that all the objects have exactly the same number of instances, we decided not to associate three instances with each object. Instead, we associated only one instance (distributed uniformly at random among the three dimensions, and based on the features of the things of the IoTs of a given dimension) to 200 of the 250 objects. Analogously, we associated two instances (distributed by following the same guidelines mentioned above) to 35 of the 250 objects. Finally, we associated three instances, one for each possible dimension, to 15 of the 250 objects. At the end of this phase, we had 315 instances, distributed among the 11 IoTs of our MIoT as shown in Table 1.

IoT	*Number of instances*
a.home	22
a.health	22
a.energy	22
a.transport	22
a.environment	22
b.near	14
b.mid	38
b.far	53
c.plain	44
c.hill	50
c.mountain	6

Table 1: Number of instances present in the IoTs of our MIoT

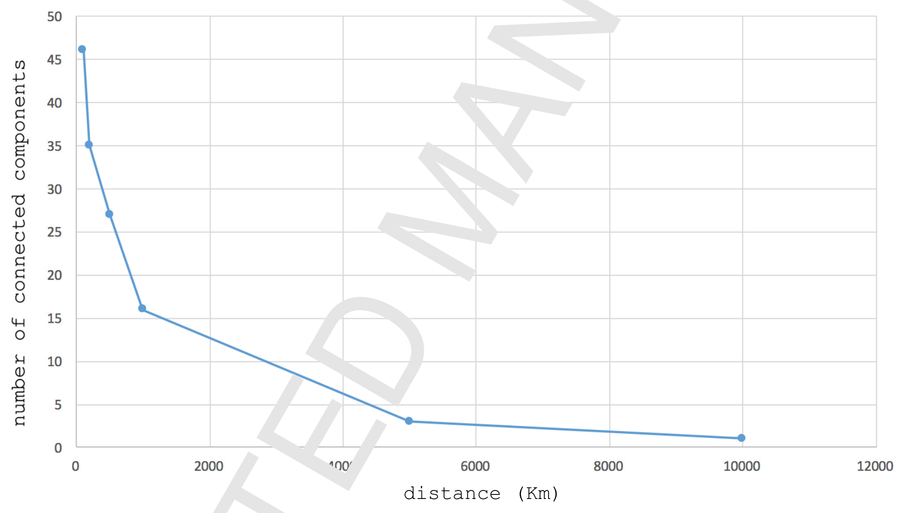


Figure 2: Distribution of the number of connected components of the instances of our MIoT against distances

To complete our MIoT and its network representation, we had to define a policy to create *i-edges*. In fact, it was clear that our MIoT should have had a node for each instance and a c-edge for each pair of instances referring to the same object. Therefore, the last decision regarded how to define i-edges. Given our scenario, it appeared reasonable to consider distances among things as the leading parameter for the creation of i-edges. To carry out this last task, we have preliminarily computed the distribution of the number of connected components possibly created from our instances against the maximum possible distance. Obtained results are reported in Figure 2. Based on this figure, in order to obtain a balanced number of connected components, we decided to connect two instances of the same IoT if the distance of the corresponding things was lesser than 1000 kilometres.

After this last choice, our MIoT was fully defined. In order to help the reader to mentally portray it, in Figure 3, we provide a graphical representation. The interested reader can find the corresponding

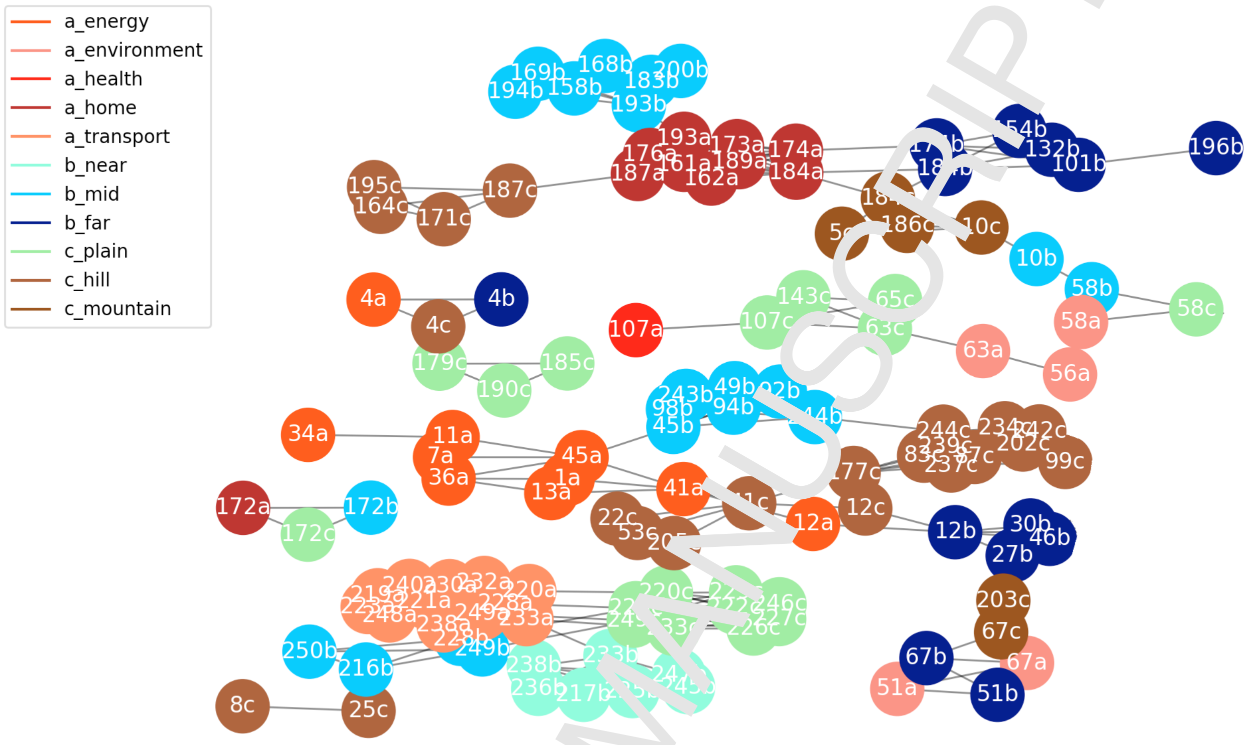


Figure 3: Graphical representation of our MIoT

dataset (in the `.csv` format) at the address `www.barbiana20.unirc.it/miot/datasets/miot1`. The password to type is "`za.12&;lq74.#`".

3.2 Why use the MIoT paradigm?

In the Introduction, we have specified that the MIoT paradigm goes in the direction suggested by some authors, who observe that it is no longer possible to think of a single global Internet of Things [8].

In this section, we present a case study aiming at comparing the classical vision of a unique global Internet of Things with the new MIoT-based vision of several Internets of Things connected to each other through cross nodes and cross edges. In our opinion, this case study can help the reader to be convinced of the practicality of the MIoT paradigm.

First, we must clarify that a slavish comparison between the previous vision of IoT and the MIoT-based vision is not possible, because this last paradigm associates more instances with the same object, one for each network joined by it. By contrast, the classical global IoT-based vision considers only objects and does not allow the existence of more instances of the same object. In other words, the global IoT-based vision returns a coarser model of the involved things and their relationships, incapable of verifying if the same object shows different features or behaviors in different subnetworks

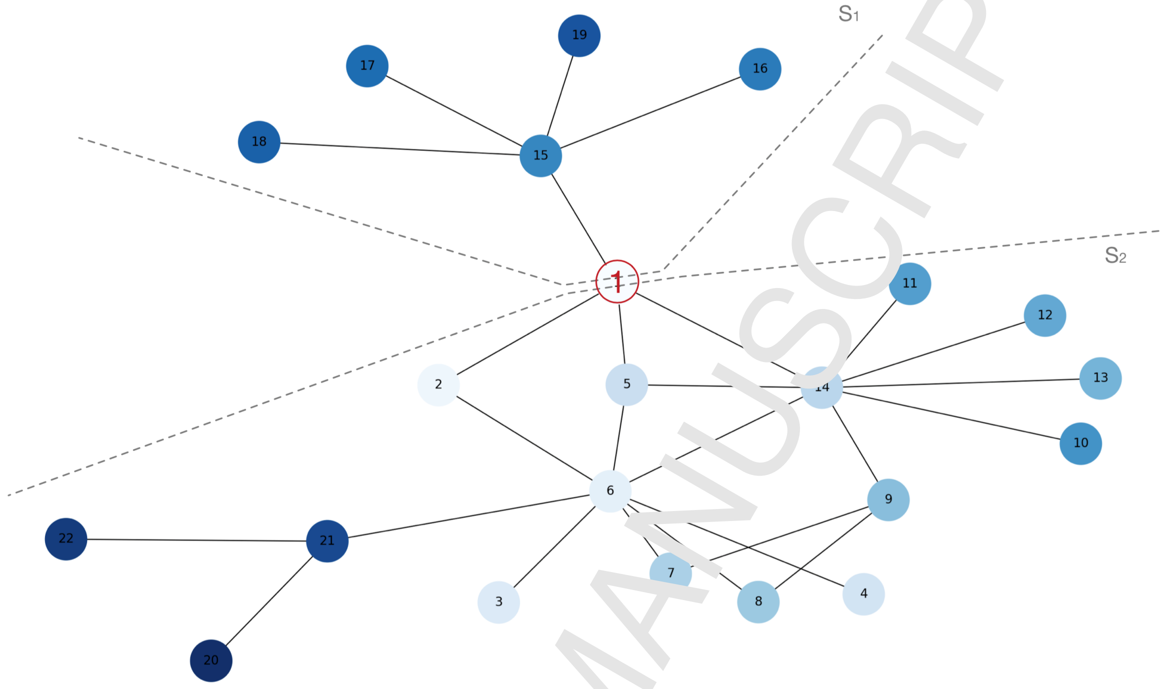


Figure 4: Our case study

of the global network. Vice versa, this verification is not only possible, but also natural, in the MIoT paradigm. Indeed, it is sufficient to investigate the different features and behaviors of the various instances of the same object in the IoTs they belong to.

After having made this important premise, which already represents a justification of the usefulness of the MIoT paradigm, we start by presenting our case study by which we aim at showing that the global IoT-based vision can provide imprecise information about the features and the roles of the corresponding things.

Since the global IoT-based vision does not consider object instances, in this case study we assume that all the instances of a cross object have been merged in a unique c-node.

With this consideration in mind, let us consider Figure 4. Here, we report a set of nodes each associated with an object. If we consider the global IoT-based vision, all these nodes form a unique IoT where it is possible to distinguish two quite separated subnetworks, called S_1 and S_2 in the figure, connected only thanks to the object represented by Node 1. If we consider the MIoT-based vision, we have two IoTs connected, by means of the object represented by Node 1, to form an MIoT.

Let us focus our attention on this node. Clearly, it is the most important node of this scenario because it is the only one allowing the communication and the cooperation between the nodes of the subnetwork S_1 and the ones of the subnetwork S_2.

However, if we compute the classical centrality measures for the nodes of this network, we have that the rank of Node 1 is not very high in any centrality measure (see Table 2). In other words, if we adopt the global IoT-based vision, no centrality measure is capable of capturing the importance

Nodes	*Betweenness Centrality*	*Degree Centrality*	*Closeness Centrality*	*Eigenvector Centrality*
1	0.39 (3)	0.19 (4)	0.44 (4)	0.30 (4)
2	0.07 (6)	0.09 (8)	0.41 (5)	0.20 (6)
3	0.00 (11)	0.05 (11)	0.33 ()	0.13 (14)
4	0.00 (12)	0.05 (12)	0.33 ()	0.13 (15)
5	0.07 (7)	0.14 (6)	0.47 (3)	0.34 (3)
6	0.52 (1)	0.38 (1)	0.48 (2)	0.34 (2)
7	0.01 (9)	0.09 (9)	0.34 ()	0.19 (7)
8	0.01 (10)	0.09 (10)	0.34 ()	0.19 (8)
9	0.04 (8)	0.14 (7)	0.37 (6)	0.23 (5)
10	0.0 (13)	0.04 (13)	0.35 (9)	0.13 (10)
11	0.0 (14)	0.04 (14)	0.35 (10)	0.13 (11)
12	0.0 (15)	0.04 (15)	0.35 (11)	0.13 (12)
13	0.0 (16)	0.04 (16)	0.35 (12)	0.13 (13)
14	0.48 (2)	0.38 (2)	0.52 (1)	0.49 (1)
15	0.35 (4)	0.23 (3)	0.35 (7)	0.11 (16)
16	0.0 (17)	0.05 (17)	0.26 (17)	0.03 (19)
17	0.0 (18)	0.05 (18)	0.26 (18)	0.03 (20)
18	0.0 (19)	0.05 (19)	0.26 (19)	0.03 (21)
19	0.0 (20)	0.05 (20)	0.26 (20)	0.03 (22)
20	0.0 (21)	0.05 (21)	0.26 (21)	0.04 (17)
21	0.18 (5)	0.14 (5)	0.35 (8)	0.15 (9)
22	0.0 (22)	0.05 (22)	0.26 (22)	0.04 (18)

Table 2: Betweenneess Centrality, Degree Centrality, Closeness Centrality and Eigenvector Centrality, and the corresponding ranks, for all the nodes of the case study of Figure 4

of this node. By contrast, the MIoT paradigm is capable alone of intrinsically evidencing the key role played by Node 1, without the need of computing any centrality measure.

With regard to this last observation, we are also aware that, in a real scenario, where the IoTs composing an MIoT are many and the number of c-objects is high, it could be extremely challenging to define a new MIoT-oriented centrality measure. This should be capable of determining the most relevant nodes in an MIoT taking also (but not exclusively) into account if they are c-nodes or not. In the future, we plan to investigate the possibility to define such a measure.

4 CDS: a crawler tailored for MIoTs

4.1 Motivations underlying CDS

As pointed out in the Introduction, in real cases, when the number of involved things is huge, in order to investigate the main features of an MIoT and to extract useful knowledge from its data, a crawling strategy is mandatory. This strategy must be able to consider not only the instances and their connections in a single IoT (i.e., i-nodes and i-edges), but also the instances of the same objects (along with the corresponding connections) in different IoTs (i.e., c-nodes and c-edges). Furthermore, it must take into consideration that c-nodes and i-nodes have different nature and that c-nodes are more important than i-nodes in an MIoT, which implies that it must be possible to privilege c-nodes over i-nodes, if necessary. Finally, it must allow users to specify how much c-nodes must be privileged over i-nodes. Observe that this problem has a correspondence with the one of finding a crawler specifically tailored for a Social Internetworking Scenario and, therefore, a crawler privileging "me"-edges over intra-network edges and bridges over intra-network nodes.

In the past, several crawling strategies operating in a *single network* (and, therefore, in a *single IoT*) have been proposed. Among them, three very popular ones are Breadth First Search (BFS, for short), Random Walk (RW, for short) and Metropolis-Hastings Random Walk (MH, for short). BFS implements the classical Breadth First Search visit. RW selects the next node to visit uniformly at random among the neighbors of the current node. Both BFS and RW tend to favor power nodes (i.e., nodes having high outdegrees). As a consequence, both of them present bias in some network parameters [34]. MH is a more recent crawling strategy, conceived to unfavor power nodes in such a way as to remove the bias, in BFS and RW, caused by their tendency to favor this kind of node. It was shown that MH performs very well in a single network [21], especially for the estimation of the average degree of nodes. At each iteration, MH randomly selects a node n_j from the neighbors of the current node n_i. Then, it randomly generates a number p, belonging to the real interval $[0, 1]$. If $p \leq \frac{outdeg(n_i)}{outdeg(n_j)}$, where $outdeg(n_i)$ and $outdeg(n_j)$ are the outdegrees of n_i and n_j, it selects n_j as the new current node. Otherwise, it maintains n_i as the current node. The higher the outdegree of a node, the higher the probability that MH discards it. The way of proceeding of MH has been specifically conceived to reach the goal of disfavoring high-degree nodes in such a way as to remove the bias caused by them, as explained above.

In the past, BFS, RW and MH were deeply studied for single networks and it was found that none of them is always better than the other ones. However, no investigaton about the application of these strategies in a set of related IoTs (of which, SIoTs and MIoTs are specific cases) has been carried out. Thus, there is no evidence that they are still valid in this new context. Rather, it is easy to foresee that they will show some weaknesses, since they do not take into account the main actors of related IoTs, i.e., the instances of the same things in different IoTs and their connections (which represent c-nodes and c-edges in the MIoT paradigm).

We expect that these instances and their connections play a crucial role in crawling a set of related IoTs, since they allow different IoTs to be crossed, thus evidencing the main actors of related IoTs, i.e. c-nodes and c-edges allowing their interconnections. These nodes and edges are not "standard" ones, due to their role. As shown in Section 3.2, we cannot see a set of related IoTs just as a unique huge IoT. By contrast, its nature, specificities and behavior must be strongly considered by a crawling strategy that aims to be effective and efficient for a set of related IoTs.

As it will be described in the next section, this original intuition has been fully confirmed by our experimental campaign, which clearly highlights the drawbacks of BFS, RW an MH when passing from a single IoT to a set of related IoTs.

4.2 Description of CDS

In the design of CDS, we start by analyzing some aspects limiting BFS, RW and MH in a set of related IoTs (and, therefore, also in an MIoT), in such a way as to overcome them.

BFS performs a Breadth First Search of a local neighborhood of the current node. Now, the average distance between two nodes of a single IoT is generally less than the one between two nodes of different IoTs. In fact, to pass from an IoT to another, it is necessary to cross a c-node and, since, in real cases, c-nodes are (much) less numerous than i-nodes, it could be necessary to generate a long path before reaching one of them. As a consequence, the local neighborhood considered by BFS includes one or a small number of IoTs.

To overcome this problem, a Depth First Search, instead of a BFS, could be performed. For this purpose, the way of proceeding of RW and MH should be included in our crawling strategy. However, since, generally, there is a limited number of c-nodes in an IoT, the simple choice to go in-depth blindly does not favor the crossing from an IoT to another. A solution that addresses the above issues could consist in the implementation of a "non-blind" Depth-First Search that favors c-nodes in the choice of the next node to visit. This is exactly the strategy we have chosen, and the name we give to it, i.e., Cross Node Driven Search (CDS, for short), clearly reflects its way of proceeding.

Observe that this problem has a correspondence with the one of finding a crawler specifically tailored for a Social Internetworking Scenario and, therefore, a crawler privileging "me"-edges over intra-network edges and bridges over intra-network nodes.

However, following exactly the strategy mentioned previously would make it impossible to explore (at least partially) the neighborhood of the current node because the visit would proceed in-depth very quickly and, as soon as a c-node is encountered, there is a cross to another IoT. The overall result of this strategy would be an extremely fragmented crawled sample. To avoid this problem, given the current node, our crawling strategy explores a fraction of its neighbors before performing an in-depth search of the next node to visit.

To formalize our crawling strategy, we need to introduce the following parameters:

- *inf* (i-node neighbors fraction). It represents the fraction of the i-node neighbors of the current node that should be visited. It ranges in the real interval $(0, 1]$. When *inf* tends to 1, CDS behaves as BFS. By contrast, when *inf* tends to 0, CDS behaves as MH and RW[2]. In all these cases, CDS inherits all the strengths and the weaknesses of the corresponding strategies. Intermediate values of *inf*, suitably determined (see Section 4.3), allow CDS to maximize the pros and to minimize the cons of BFS, RW and MH.

- *cnf* (c-node neighbors fraction). It represents the fraction of the c-node neighbors of the current node that should be visited. It ranges in the real interval $(0, 1]$. It allows the tuning of the number of IoT crossings performed by CDS. The higher its value, the higher this number. Clearly, an excessive number of crossings could return a sample involving many IoTs of the MIoT but with a very little number of connections between each pair of IoTs. This could cause, in the Multiple-Network context, the same problem caused by RW in the Single-Network scenario. As a consequence, also for this parameter, a tradeoff is necessary.

For instance, in a configuration where $inf = 0.15$ and $cnf = 0.30$, CDS visits 15% of the i-node neighbors of the current node and 30% of the c-node neighbors of the current node.

[2]To be extremely accurate and precise, this is true if the parameter *cnf* (that we introduce below) is fixed to 1, in case we want to visit the whole MIoT, or to 0, in case we want to restrict our visit to just one IoT of the MIoT.

We are now able to formalize our crawling strategy. We report its pseudocode in Algorithm 1.

Algorithm 1 CDS

```
Notation   We denote by $I(n)$ a function returning the number of i-node neighbors of the node $n$ and by $C(n)$ a function returning
    the number of c-node neighbors of $n$.
Input   $\mathcal{M}$: an MIoT composed of $m$ IoTs; $n_{it}$: a non-negative integer; $cnf$, $inf$: a real number in the range [0,1]; SeenNodes,
    VisitedNodes, VisitedCNodes: a set of nodes
Output   SeenNodes; VisitedNodes;
Variable   $v, w$: a node
Variable   $p$: a real number in the range [0,1]
Variable   $c$: an integer number
Variable   NodeQueue: a queue of nodes
 1: $NodeQueue := \emptyset$
 2: select a seed node $s$ (not already present in $VisitedNodes$) from $\mathcal{M}$ uniformly at random
 3: insert $s$ in NodeQueue
 4: while $n_{it} > 0$ do
 5:     extract a node $v$ from NodeQueue
 6:     insert $v$ in VisitedNodes
 7:     insert all the nodes adjacent to $v$ in SeenNodes
 8:     if $(C(v) \geq 1)$ then
 9:         clear NodeQueue
10:         $c := 0$
11:         while $((c < \lceil cnf \cdot C(v) \rceil)$ and $(n_{it} > 0))$ do
12:             let $w$ be one c-node neighbor of $v$ not in VisitedCNodes selected uniformly at random
13:             generate a number $p$ in the real interval $[0, 1]$ uniformly at random
14:             if $\left(p \leq \frac{C(v)+I(v)}{C(w)+I(w)}\right)$ then
15:                 insert $w$ in NodeQueue and in VisitedCNodes
16:                 $c := c + 1$
17:                 $n_{it} := n_{it} - 1$
18:             end if
19:         end while
20:     end if
21:     if $(I(v) \geq 1)$ then
22:         $c := 0$
23:         while $((c < \lceil inf \cdot I(v) \rceil)$ and $(n_{it} > 0))$ do
24:             let $w$ be one of the i-node neighbors of $v$ selected uniformly at random
25:             generate a number $p$ in the real interval $[0, 1]$ uniformly at random
26:             if $\left(p \leq \frac{I(v)}{I(w)}\right)$ then
27:                 insert $w$ in NodeQueue
28:                 $c := c + 1$
29:                 $n_{it} := n_{it} - 1$
30:             end if
31:         end while
32:     end if
33:     if $((n_{it} > 0)$ and $(NodeQueue = \emptyset))$ then
34:         goto 37
35:     end if
36: end while
37: if $(n_{it} = 0)$ then
38:     return $SeenNodes, VisitedNodes$
39: else
40:     return $CDS(\mathcal{M}, n_{it}, cnf, inf, SeenNodes, VisitedNodes, VisitedCNodes)$
41: end if
```

CDS receives: *(i)* an MIoT $\mathcal{M}$, consisting of m IoTs; *(ii)* a non-negative integer n_{it}, denoting the number of iterations that must be still performed; *(iii)* cnf and inf; *(iv)* three sets of nodes, called $SeenNodes$, $VisitedNodes$ and $VisitedCNodes$, whose semantics will be clear in the following. It returns $SeenNodes$ and $VisitedNodes$ after having updated them.

It exploits: *(i)* a function $I(n)$ returning the number of i-node neighbors of the node n; *(ii)* a function $C(n)$ returning the number of c-node neighbors of the node n; *(iii)* two support nodes v and w; *(iv)* a support real number p in the real interval $[0, 1]$; *(v)* a support counter c; *(vii)* a support queue $NodeQueue$ of nodes.

First CDS selects a seed node s (not already present in the list $VisitedNodes$ of the nodes already visited) from $\mathcal{M}$ uniformly at random, and inserts it in $NodeQueue$. Then, it starts a cycle that ends when the number n_{it} of iterations to be still performed is 0.

During each iteration, CDS extracts a node v from $NodeQueue$ and inserts it in $VisitedNodes$. At the same time, it inserts all the node neighbors of v in the list $SeenNodes$.

At this point, it computes $C(v)$ to verify if there exist c-node neighbors of v. In the affirmative case, it clears $NodeQueue$[3] and starts to examine these nodes until to either the number of examined c-nodes reaches the maximum value established through mf or there are no available iterations.

During each of these internal iterations, CDS selects a node w, among the c-node neighbors of v not already present in the set $VisitedCNodes$ of the already visited c-nodes; the selection of w is performed uniformly at random. Then, it generates a real number p in the range $[0, 1]$ uniformly at random. If $p \leq \frac{C(v)+I(v)}{C(w)+I(w)}$, then w is inserted in both $NodeQueue$ and $VisitedCNodes$, c is increased of 1 and n_{it} is decreased of 1. Note that the last condition implements the strategy of MH into CDS, in such a way as to let CDS to inherit the pros of MH.

After having processed the c-node neighbors of v, CDS starts to process the i-node neighbors of v in an analogous way. In particular, it selects a node w among the i-node neighbors of v uniformly at random. Then, it generates a number p in the real interval $[0, 1]$ uniformly at random and, if $p \leq \frac{I(v)}{I(w)}$, it inserts w into $NodeQueue$, increases c of 1 and decreases n_{it} of 1.

CDS terminates the external cycle started at row 4 when $n_{it} = 0$ or when there are no nodes that can be visited starting from the current seed. In the former case, it returns $SeenNodes$ and $VisitedNodes$. In the latter case, it recursively calls another instance of itself in such a way as to re-start all the previous tasks from another seed node not already visited in the past.

4.3 Experimental campaign

We carried out our experiments on the testbed presented in Section 3.1. In particular, we performed two kinds of experiment, namely:

- *setting of CDS*: in this case, we aimed to choose the most suitable values of the input parameters of CDS;
- *evaluation of CDS*: in this case, we compared CDS with BFS, RW and MH to quantitatively determine its strengths and weaknesses.

In the next subsections, we present each of these experiments.

[3]Observe that this task is performed to privilege c-nodes over i-nodes and to favor crossings from one IoT to another. Indeed, if $NodeQueue$ would have not been cleared, there was the risk to remain in the same IoT or, in any case, to visit a very small number of IoTs.

Iterations	10	20	30	40	50	60	70	80	90	100
Seen nodes	50	78	107	150	165	163	183	187	181	198
Visited nodes	11	21	34	48	59	68	94	102	105	125
IoT Crossings	4	9	14	17	24	24	33	40	30	43
Visited IoTs	5	6	7	9	9	9	10	10	10	10

Table 3: Number of seen nodes, number of visited nodes, number of IoT crossings and number of visited IoTs against the number of iterations performed by CDS

4.3.1 Setting of CDS

As pointed out in Section 4.2, CDS needs three input parameters that can be used to make it more responsive to our needs. These parameters are: *(i)* inf, i.e. the i-node neighbors fraction that should be visited; *(ii)* cnf, i.e. the c-node neighbors fraction that should be visited; *(iii)* n_{it}, i.e. the maximum number of iterations.

We recall that our testbed consists of 315 nodes; 200 of them are i-nodes, whereas 115 of them are c-nodes.

First, we computed the variation of the number of seen and visited nodes, IoT crossings and visited IoTs against the variation of the number of performed iterations. Obtained results are reported in Table 3.

From the analysis of this table, we can see that:

- after 20 iterations, 24.76% of all nodes are seen, 6.67% of all nodes are visited and 54.55% of IoTs are visited;
- after 50 iterations, 52.38% of all nodes are seen, 18.73% of all nodes are visited and 81.81% of IoTs are visited;
- after 70 iterations, 58.10% of all nodes are seen, 29.84% of all nodes are visited and 90.91% of IoTs are visited;
- after 100 iterations, 62.85% of all nodes are seen, 39.68% of all nodes are visited and 90.91% of IoTs are visited.

Taking into account these observations, as well as the trends of the corresponding measures reported in Figure 5, we observe that setting the number of iterations to 70 (or, more formally, setting $n_{it} = 0.22 \cdot |N|$) is a good tradeoff between the capability of sampling the highest possible number of the MIoT nodes and the effort required to perform this task.

After having set $n_{it} = 70$, we computed the variation of the number of seen and visited nodes, IoT crossings and, finally, visited IoTs against the variation of the values of inf and cnf. In particular, we considered five possible values of inf (i.e., $inf = 0$, $inf = 0.25$, $inf = 0.50$, $inf = 0.75$, and $inf = 1$) and five possible values of cnf (i.e., $cnf = 0$, $cnf = 0.25$, $cnf = 0.50$, $cnf = 0.75$, and $cnf = 1$). Obtained results are reported in Table 4.

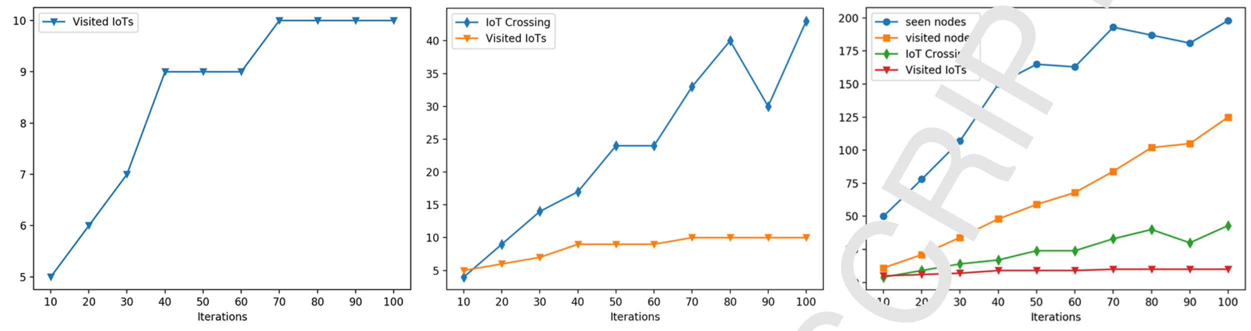


Figure 5: Trends of the number of seen nodes, visited nodes, IoT crossings and visited IoTs against the number of iterations performed by CDS (trends are separated in the first two graphs and put together in the last one)

From the analysis of this table, we can see that the best values for the four parameters are found when inf is low and cnf is high. This is totally in line with the semantics of these two coefficients, as well as with the role that they play in CDS. In particular, we observe that, if we consider the four parameters overall, the best pair of values is $inf = 0.25$ and $cnf = 0.75$.

4.3.2 Evaluation of CDS

In this experiment, we compared CDS with BFS, RW and MH. In this activity, the first preliminary task was to find reasonable metrics for evaluating the performances of crawlers that operate on a set of related IoTs. For this purpose, first we extended to the Multiple-Network context the metrics designed for evaluating the performances of crawlers that operate on a Single-Network context. Then, we introduced some other metrics specific for a set of related IoTs.

This section illustrates all our efforts in this direction and the results we have obtained. Specifically, it is organized in three subsections. The first presents our basic evaluation measures. The second describes a combined evaluation measure introduced by us. Finally, the last presents the results of the test that we have performed by means of these measures.

Basic evaluation measures

The basic evaluation measures that we designed for our experimental campaign are the following:

- *Cross Node Ratio (CNR)*: This is a real number, in the interval $[0, 1]$, defined as the ratio of the number of crawled c-nodes to the number of all the c-nodes of the MIoT.
- *IoT Crossings (IC)*: This is a non-negative integer and denotes how many times the crawler switches from one IoT to another.
- *Visited IoTs (VI)*: This is a positive integer and measures how many different IoTs are visited by the crawler.

Seen nodes					
	$inf = 0$	$inf = 0.25$	$inf = 0.50$	$inf = 0.75$	$inf = 1$
$cnf = 0$	152	144	159	132	161
$cnf = 0.25$	200	178	201	212	171
$cnf = 0.50$	189	183	206	196	170
$cnf = 0.75$	199	212	204	172	201
$cnf = 1$	208	174	181	181	194
Visited nodes					
	$inf = 0$	$inf = 0.25$	$inf = 0.50$	$inf = 0.75$	$inf = 1$
$cnf = 0$	55	55	56	54	56
$cnf = 0.25$	64	61	65	65	62
$cnf = 0.50$	65	64	70	67	63
$cnf = 0.75$	71	70	69	62	70
$cnf = 1$	70	63	66	65	68
IoT crossing					
	$inf = 0$	$inf = 0.25$	$inf = 0.50$	$inf = 0.75$	$inf = 1$
$cnf = 0$	23	20	22	19	24
$cnf = 0.25$	29	26	31	31	26
$cnf = 0.50$	30	28	[illegible]	32	37
$cnf = 0.75$	36	37	34	26	35
$cnf = 1$	35	25	[illegible]	29	33
Visited IoT					
	$inf = 0$	$inf = 0.25$	$inf = 0.50$	$inf = 0.75$	$inf = 1$
$cnf = 0$	9	8	8	8	10
$cnf = 0.25$	10	9	10	10	9
$cnf = 0.50$	9	9	10	9	9
$cnf = 0.75$	9	10	10	9	10
$cnf = 1$	10	9	9	9	9

Table 4: Number of seen nodes, visited nodes, IoT crossings and visited IoTs against the variation of inf and cnf

- *Unbalancing (UB)*: This is a non negative real number defined as the standard deviation of the fraction of nodes discovered for each IoT w.r.t. the overall number of nodes discovered in the sample. UB ranges from 0, corresponding to the case in which each IoT is sampled with the same number of nodes, to a maximum value, corresponding to the case in which all sampled nodes belong to the same IoT.
- *Degree Bias (DB)*: This is a real number defined as the root mean squared error, for each IoT of the MIoT, of the average node degree estimated by the crawler and the one estimated by MH, which is considered the best crawling strategy for the estimation of the degree of a network node in the literature [3[illegible], 21]. If the crawled sample does not cover one or more IoTs, then these are not considered in the computation of DB.

If we consider the parallelism between MIoTs and Social Internetworking, we have that, in a Social Internetworking System: *(i)* CNR would return the ratio of the number of bridges discovered to the number of all the nodes in the sample; *(ii)* IC would measure how many times the crawler switches from one social network to another; *(iii)* VI would return how many different social networks are visited by the crawler; *(iv)* UB would represent the standard deviation of the percentages of nodes discovered for each social network w.r.t. the overall number of nodes discovered in the sample; *(v)* DB would denote the root mean squared error, for each social network of the SIS, of the average node degree estimated by the crawler and the one estimated by MH.

As for CNR, IC and VI, the higher their value, the higher the performance of the crawling strategy. By contrast, as far as UB and DB are concerned, the lower their values and the higher the performance of the crawling strategy. Observe that VI allows the evaluation of the crawler's capability of covering many IoTs of the MIoT. With regard to this measure, a further consideration is in order. Indeed, one could think that a fair crawling strategy should sample different IoTs proportionally to their respective overall size. Actually, this crawler behavior could result in incomplete samples in case of a high variance of these sizes. In fact, it could happen that some small IoTs would be not represented, or would be insufficiently represented, in the sample. CNR and IC are related to the coupling degree of the IoTs of the MIoT, whereas DB is related to the average degree.

A combined evaluation measure

Besides some separated metrics, each capturing an important aspect of the crawling strategy, it is certainly important to define a synthetic measure capable of capturing a sort of "overall" crawler behavior. Furthermore, this overall measure should allow users to tune the importance of the five metrics in it, which could be different in different application cases. A reasonable way to do this consists in defining the overall metric as a linear combination of the five ones introduced above, where the coefficients reflect the importance that users want to associate with them. We call *Overall Crawling Quality* (OCQ, for short) this measure and define it as:

$$OCQ = w_{CNR} \cdot \frac{CNR}{CNR_{max}} + w_{IC} \cdot \frac{IC}{IC_{max}} + w_{VI} \cdot \frac{VI}{VI_{max}} + w_{UB} \cdot (1 - \frac{UB}{UB_{max}}) + w_{DB} \cdot (1 - \frac{DB}{DB_{max}})$$

Here, CNR_{max}, IC_{max}, VI_{max}, UB_{max} and DB_{max} are the upper bounds of CNR, IC, VI, UB and DB, which, in a comparative experiment, can be set to the maximum value obtained by the crawlers into consideration. Furthermore, w_{CNR}, w_{IC}, w_{VI}, w_{UB} and w_{DB} are real numbers belonging to the interval $[0, 1]$ such that their overall sum is 1.

Before reasoning about the possible values of the five weights of OCQ, we point that the defined metrics are not completely independent of each other. In fact, if $CNR = 0$, then IC and VI are also 0. Furthermore, the value of CNR influences the values of both VI and UB. As a consequence, it is reasonable to assign different weights to the five metrics by associating the highest weights with the most influential ones. To perform this task, we defined an algorithm that is based on the Kahn's approach for topological sorting of graphs [29]. This algorithm uses a data structure called Metric Dependency Graph. This graph has a node n_i for each metric M_i; there exists an edge from n_i to n_j if the metric M_i influences the metric M_j. Each node has associated a weight. Initially all the node weights are set to 0.20 (see Figure 6). Our algorithm starts from a node with no outgoing edges and splits the corresponding weight (in equal parts) between itself and the nodes it depends on. Clearly, if a node has no incoming edge, it maintains its weight. After the split of the weight, our algorithm removes all the incoming edges from the corresponding nodes and repeats the previous tasks until all the nodes of the graph have been processed.

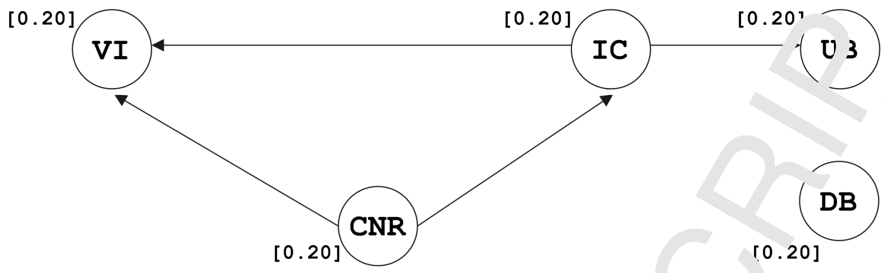


Figure 6: Our Metric Dependency Graph

It is worth pointing out that the node processing order could be not unique, if there exists more than one node with no outgoing edges. However, it is possible to prove that the final metric weights returned by our algorithm do not depend on the adopted node processing order.

It is possible to formalize the previous algorithm in a closed formula allowing us to compute the weight w_i associated with each node n_i of the Metric Dependency Graph. In particular, we have:

$$w_i = \frac{\frac{1}{1+indeg(n_i)} \cdot \left(w + \sum_{n_i \in OSet(n_j)} w_j\right)}{\sum_{k=1}^{5} w_k}$$

Here, $indeg(n_i)$ is the indegree of n_i, w is a number representing the initial weight of n_i (that, in our case, is 0.20 for all the five nodes) and $OSet(n_j)$ is the set of the nodes reachable from n_j through its outgoing edges. This formula indicates that w_i consists of two components; the former is the initial weight w; the latter represents the weight gained by n_i thanks to the fact that other nodes depend on it. In turn, n_i splits its weight among the nodes it depends on and itself; this is handled by the term $1 + indeg(n_i)$. The denominator of the formula is used to normalize w_i in the interval $[0, 1]$.

By applying the previous formula to our five metrics we obtained the following weight values: $w_{CNR} = 0.45$, $w_{IC} = 0.18$, $w_{VI} = 0.07$, $w_{UB} = 0.10$ and $w_{DB} = 0.20$.

Test results

We are now ready to analyze the performances of CDS, BFS, RW and MH when applied on an MIoT. For this activity, we used the testbed described in Section 3.1. We applied BFS, RW and MH to each MIoT by regarding it as a unique graph. Furthermore, in order to make the MIoT graph totally compliant with the inputs classically received by BFS, RW and MH, we considered a "condensed version" of the MIoT graph by putting just one node for each c-object. We run CDS with $inf = 0.25$ and $cnf = 0.75$, which, as pointed out in Section 4.3.2, are the parameter settings that guarantee the maximum number of IoT crossings. We report the obtained results in Table 5.

We recall that the higher the values of CNR, IC and VI and the lower the values of DB and UB, the better the performances of the strategies into examination.

From the analysis of Table 5, we can observe that, as far as CNR, IC, VI and UB are concerned, CDS outperforms BFS, RW and MH. For instance, the value of CNR obtained by CDS is about 230% (resp., 273%, 296%) better than the one of BFS (resp., RW, MH).

	CDS	BFS	RW	MH
CNR	0.211	0.064	0.057	0.053
IC	9.133	6.400	2.333	2.333
VI	27.000	10.933	7.467	7,467
DB	3.476	0.844	0.026	0
UB	0.142	0.269	0.236	0.199

Table 5: Values of the five metrics obtained by CDS, BFS, RW and MH

	CDS	BFS	RW	MH
Configuration A	0.695	0.433	0.383	0.409
Configuration B	0.747	0.410	0.39[illegible]	0.407

Table 6: Values of OCQ obtained by CDS, BFS, RW and MH for the two weight configurations into examination

The only metric for which CDS shows a worse performance than the other strategies is DB. In fact, as for this metric, the value obtained by MH is 0. This was expected because DB is measured having the value of MH as the reference one since, in the literature, it is well known that MH guarantees the best Degree Bias among all crawling strategies [[illegible]4, 21]. BFS and RW obtain values of DB near to the ones of MH, whereas CDS shows the worst performance, even if it is still acceptable. The results obtained by CDS for DB were also expected because the purpose of this crawler is to privilege c-nodes over i-nodes. As a consequence, when a c-node is encountered, the node queue is cleared (see Line 4 in Algorithm 1) in such a way as to stimulate the IoT crossings and, ultimately, the visit of c-nodes, which is the main objective of our crawler. Clearing the node queue produces a distortion because several nodes directly connected to the current one will not be put in the set of visited nodes. In turn, this produces an effect in the degree bias and, ultimately, the worst performance of CDS, as far as the value of DB is concerned. However, observe that these results are obtained with the default configuration of CDS (i.e. $inf = 0.25$ and $cnf = 0.75$). Actually, if necessary, it is possible to configure CDS in such a way that it behaves as RW and MH, which present the best values of DB. In fact, as seen in Section 4.2, this behavior can be obtained by making inf tend to 0.

Since there is one parameter for which CDS shows the worst results w.r.t. the other three crawlers, it is particularly important the computation of the values of OCQ, because this parameter summarizes the overall performance of the crawlers into examination. We computed the values of OCQ for both the configuration that sets all the metric weights to 0.20 (we call it "Configuration A" in the following) and the one that takes the parameter dependencies into account ($w_{CNR} = 0.45$, $w_{IC} = 0.18$, $w_{VI} = 0.07$, $w_{UB} = 0.10$ and $w_{DB} = 0.20$ - we call it "Configuration B" in the following). In Table 6, we report the obtained results (we recall that the higher the value of OCQ and the better the performance of the corresponding crawler).

From the analysis of this table we can observe that, in both cases, CDS outperforms BFS, RW and MH. Interestingly, in the configuration taking the Metric Dependency Graph into account, CDS obtains even better results than in the other one.

In our opinion, these results clearly evidence that, in an MIoT scenario:

- The crawling strategies defined for single networks do not perform well because they do not consider the important differences existing between c-nodes and i-nodes and between c-edges and i-edges.
- A cross node centered crawler, like CDS, shows very satisfying results and, certainly, indicates a way to go for further crawler strategies specifically designed to operate on a set of related IoTs.

5 Analytical Discusssion

In this section, we propose an analytical discussion aiming at comparing our model and approach with other, more or less conventional, ones. We start by observing that, in the last years, the interest and the attention towards IoTs and sensor networks are enormously increased. This has led, and is currently leading, to a large variety of models and approaches. Some, very common and particularly interesting, families of approaches that can be recognized are the ones based on:

- fuzzy logic;
- neural networks;
- hierarchical models.

In the following, we present a comparison between our approach and each of these families.

Fuzzy logic based approaches allow the possibility that a thing belongs to more sets simultaneously [33, 46, 53, 3]. Also in our model, an object can belong to more IoTs, thanks to its instances. However, differently from fuzzy logic based approaches, in our case, when there is the instance of an object in an IoT, this means that the object surely belongs to that IoT. Instead, in fuzzy logic based approaches, an object belongs to a given IoT with a certain plausibility.

Neural network based approaches can exploit the potentialities of a highly dynamic structure, such as neural network [14, 52]. The dynamism of the support data structure certainly represents an analogy with our approach, which is based on an equally dynamic structure, i.e. social network. However, even if these two support data structures are graph based, they have totally different objectives. Indeed, neural networks are well suited for performing classifications and for handling non-linear scenarios. Social Networks are centered on node cooperation, node centralities and information diffusion. Furthermore, in an MIoT, there is no need to handle non-linearity.

Hierarchical approaches are certainly a bit more different from the MIoT paradigm than the other two families considered above [37, 61]. In fact, they mainly aim at detecting (more or less) hidden relationships among objects at different abstraction levels. Even if such a family of approaches is quite far from the current MIoT paradigm, it could represent a good starting point for an evolution of our model. Indeed, the current MIoT architecture consists of only two levels of control. Increasing the hierarchy length and, therefore, the granularity level, would allow the definition of more instances of one object in the same IoT, which could provide our model with a higher refinement capability.

Finally, to the best of our knowledge, the approach most similar to ours is the one described in [13]. In fact, analogously to what happens in an MIoT, in this approach an object is described by means of an ennuple. This choice allows an ordered representation of an object, its activities and its instances. However, very differently from our approach, the one of [13] models data coming from an IoT as a big data stream. This forces a kind of sampling allowing only the registration of the probability that a given object is in a given condition or in a given place. Interestingly, the approach of [13] provides the user with a strong support for data cleaning and integration. Instead, the MIoT paradigm does not address this issue because it assumes that cleaning and integration tasks have been performed before the construction of the MIoT graph.

6 Conclusion

In this paper, we have presented the MIoT paradigm, aimed to introduce some ideas typical of Social Internetworking Systems in IoT. We have seen that an MIoT can be considered as a set of things connected to each other by means of several kinds of relationship not defined a priori. At the same time, an MIoT can be seen as a set of related IoTs, one for each kinds of relationship existing among things.

We have also seen that, in several applications, it is extremely useful a crawling strategy well suited for an MIoT. In fact, we have shown that, in an MIoT, the classical crawling strategies, conceived for single networks (i.e., BFS, RW and MH), perform badly because they are not able to distinguish c-nodes from i-nodes and c-edges from i-edges. Finally, we have presented CDS, a crawler specifically tailored for an MIoT, and we have shown that it outperforms BFS, RW and MH.

In our opinion, this paper is not to be intended as an ending point. By contrast, it is a starting point for addressing many challenges in the context of IoT, based on the ideas to adopt Social Internetworking, instead of the much simpler Social Networking paradigm. For instance, we plan to investigate new forms of centralities specifically suited for an MIoT. In fact, we argue that, analogously to what happened for crawlers, the classical centrality measures are not adequate in presence of a set of related IoTs, when it is necessary to distinguish between c-nodes and i-nodes. The new centrality measures should be centered on the main actors of MIoTs, i.e., c-nodes.

References

[1] IPSO Alliance. *https://www.ipso-alliance.org/*, 2018.

[2] Thingful: A Search Engine for the Internet of Things. *https://thingful.net/*, 2018.

[3] F. Ali, S. Islam, D. Kwak, P. Khan, N. Ullah, S. Yoo, and K. Kwak. Type-2 fuzzy ontology–aided recommendation systems for IoT–based healthcare. *Computer Communications*, 2017. Elsevier.

[4] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro. Named data networking for IoT: An architectural perspective. In *Proc. of the European Conference on Networks and Communications (EuCNC'2014)*, pages 1–5, Bologna, Italy, 2014. IEEE.

[5] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. Aguiar, and A. Vasilakos. Information-centric networking for the internet of things: challenges and opportunities. *IEEE Network*, 30(2):92–100, 2016. IEEE.

[6] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer networks*, 54(15):2787–2805, 2010. Elsevier.

[7] L. Atzori, A. Iera, and G. Morabito. SIoT: Giving a social structure to the Internet of Things. *IEEE Communications Letters*, 15(11):1193–1195, 2011. IEEE.

[8] L. Atzori, A. Iera, and G. Morabito. From "smart objects" to "social objects": The next evolutionary step of the Internet of Things. *IEEE Communications Magazine*, 52(1):97–105, 2014. IEEE.

[9] L. Atzori, A. Iera, and G. Morabito. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks*, 56:122–140, 2017. Elsevier.

[10] 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*, 56(16):3594–3608, 2012. Elsevier.

[11] F. Buccafurri, V.D. Foti, G. Lax, A. Nocera, and D. Ursino. Bridge Analysis in a Social Internetworking Scenario. *Information Sciences*, 224:1–18, 2013. Elsevier.

[12] H. Cai, B. Xu, L. Jiang, and A. Vasilakos. IoT-based big data storage systems in cloud computing: perspectives and challenges. *IEEE Internet of Things Journal*, 4(1):75–87, 2017. IEEE.

[13] L. Chen, M. Tseng, and X. Lian. Development of foundation models for Internet of Things. *Frontiers of Computer Science in China*, 4(3):376–385, 2010. Springer.

[14] Y. Chen, Z. Zhen, H. Yu, and J. Xu. Application of Fault Tree Analysis and Fuzzy Neural Networks to Fault Diagnosis in the Internet of Things (IoT) for Aquaculture. *Sensors*, 17(1):153, 2017. Multidisciplinary Digital Publishing Institute.

[15] B. Christophe, V. Verdot, and V. Toubian. Searching the 'web of things'. In *Proc. of the International Conference on Semantic Computing (ICSC'2011)*, pages 308–315, Palo Alto, CA, USA, 2011. IEEE.

[16] S. Distefano, G. Merlino, and A. Puliafito. Enabling the cloud of things. In *Proc. of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS'2012)*, pages 858–863, Taichung, Taiwan, 2012. IEEE.

[17] H. Dong, F. Hussain, and E. Chang. A framework for discovering and classifying ubiquitous services in digital health ecosystems. *Journal of Computer and System Sciences*, 77(4):687–704, 2011. Elsevier.

[18] I. Farris, R. Girau, L. Militano, M. Nitti, L. Atzori, A. Iera, and G. Morabito. Social virtual objects in the edge cloud. *IEEE Cloud Computing*, 2(6):20–28, 2015. IEEE.

[19] C. Fetzer, P. Felber, E. Riviere, V. Schiavoni, and P. Sutra. Unicrawl: A practical geographically distributed web crawler. In *Proc. of the International Conference on Cloud Computing (CLOUD'2015)*, pages 389–396, New York, NY, USA, 2015. IEEE.

[20] R. Gaur and D.K. Sharma. Review of ontology based focused crawling approaches. In *Proc. of the International Conference on Soft Computing Techniques for Engineering and Technology (ICSCTET'2014)*, pages 1–4, Nainital, India, 2014. IEEE.

[21] M. Gjoka, M. Kurant, C.T. Butts, and A. Markopoulou. Walking in Facebook: A case study of unbiased sampling of OSNs. In *Proc. of the International Conference on Computer Communications (INFOCOM'10)*, pages 1–9, San Diego, CA, USA, 2010. IEEE.

[22] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013. Elsevier.

[23] D. Guinard, M. Fischer, and V. Trifa. Sharing using social networks in a composable web of things. In *Proc. of the International Conference on Pervasive Computing and Communications (PERCOM 2010)*, pages 702–707, Mannheim, Germany, 2010. IEEE.

[24] D. Guinard, V. Trifa, F. Mattern, and E. Wilde. From the internet of things to the web of things: Resource-oriented architecture and best practices. *Architecting the Internet of Things*, pages 97–129, 2011. Springer.

[25] D. Guinard, V. Trifa, and E. Wilde. Architecting a mashable open world wide web of things. *Technical Report of the Institute for Pervasive Computing, ETH Zürich, Zürich, Switzerland*, 663, 2010.

[26] L. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Proc. of the International Conference on Ubiquitous Computing (Ubicomp'2001)*, pages 116–122, Atlanta, GA, USA, 2001. Springer.

[27] I. Ishaq, D. Carels, G. Teklemariam, J. Hoebeke, F. Abeele, E. Poorter, I. Moerman, and P. Demeester. IETF standardization in the field of the Internet of Things (IoT): a survey. *Journal of Sensor and Actuator Networks*, 2(2):235–287, 2013. Multidisciplinary Digital Publishing Institute.

[28] Q. Jing, A. Vasilakos, J. Wan, J. Lu, and D. Qiu. Security of the Internet of Things: perspectives and challenges. *Wireless Networks*, 20(8):2481–2501, 2014. Springer.

[29] A.B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.

[30] S. Karnouskos. Smart houses in the smart grid and the search for value-added services in the cloud of things era. In *Proc. of the International Conference on Industrial Technology (ICIT'2013)*, pages 2016–2021, Cape Town, Western Cape, South Africa, 2013. IEEE.

[31] M. Kranz, L. Roalter, and F. Michahelles. Things that Twitter: social networks and the Internet of Things. In *Proc. of the International Workshop on Pervasive Computing (Pervasive 2010)*, pages 1–10, Helsinki, Finland, 2010.

[32] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about Twitter. In *Proc. of the First Workshop on Online Social Networks (WOSN'2008)*, pages 19–24, Seattle, WA, USA, 2008.

[33] H. Kumarage, I. Khalil, Z. Tari, and A. Zomaya. Distributed anomaly detection for industrial wireless sensor networks based on fuzzy data modelling. *Journal of Parallel and Distributed Computing*, 73(6):790–806, 2013. Elsevier.

[34] M. Kurant, A. Markopoulou, and P. Thiran. On the bias of BFS (Breadth First Search). In *Proc. of the International Teletraffic Congress (ITC'2010)*, pages 1–8, Amsterdam, The Netherlands, 2010. IEEE.

[35] M. Li, H. Chen, X. Huang, and L. Cui. EasiCrawl: A Sleep-Aware Schedule Method for Crawling IoT Sensors. In *Proc. of the International Conference on Parallel and Distributed Systems (ICPADS'2015)*, pages 148–155, Melbourne, Australia, 2015. IEEE.

[36] X. Li, Y. Wang, F. Shi, and W. Jia. Crawler for Nodes in the Internet of Things. *ZTE Communications*, 3:009, 2015.

[37] L. Liu, H. Ma, D. Tao, and D. Zhang. A hierarchical cooperation model for sensor networks supported cooperative work. In *Proc. of the International Conference on Computer Supported Cooperative Work in Design (CSCWD'06)*, pages 1–6, Nanjing, China, 2006. IEEE.

[38] L. Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdos is Eighty*, pages 1–46. 1993. Springer.

[39] H. Ma and W. Liu. Progressive Search Paradigm for Internet of Things. *IEEE MultiMedia*, Forthcoming. IEEE.

[40] T. Maekawa, Y. Yanagisawa, Y. Sakurai, Y. Kishino, K. Kamei, and T. Okadome. Context-aware web search in ubiquitous sensor environments. *ACM Transactions on Internet Technology*, 11(3):12, 2012. ACM.

[41] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012. Elsevier.

[42] H. Ning and Z. Wang. Future internet of things architecture: like mankind neural system or social organization framework? *IEEE Communications Letters*, 15(4):461–463, 2011. IEEE.

[43] A. Nocera and D. Ursino. PHIS: a system for scouting potential hubs and for favoring their "growth" in a Social Internetworking Scenario. *Knowledge-Based Systems*, 36:288–299, 2012. Elsevier.

[44] C. Olston and M. Najork. Web crawling. *Foundations and Trends*, 4(3):175–246, 2010. Now Publishers, Inc.

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

[46] A. Patel and T.A. Champaneria. Fuzzy logic based algorithm for Context Awareness in IoT for Smart home environment. In *Proc. of the International Conference on Region 10 Conference (TENCON'16)*, pages 1057–1060, Singapore, 2016. IEEE.

[47] C. Perera, Y. Qin, J. Estrella, S. Reiff-Marganiec, and A. Vasilakos. Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys*, 50(3):32, 2017. ACM.

[48] C. Perera and A. Vasilakos. A knowledge-based resource discovery for Internet of Things. *Knowledge-Based Systems*, 109:122–136, 2016. Elsevier.

[49] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the Internet of Things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014. IEEE.

[50] Y. Qin, Q. Sheng, N. Falkner, S. Dustdar, H. Wang, and A. Vasilakos. When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications*, 64:137–153, 2016. Elsevier.

[51] A.H. Rasti, M. Torkjazi, R. Rejaie, and D. Stutzbach. Evaluating Sampling Techniques for Large Dynamic Graphs. *Univ. Oregon, Tech. Rep. CIS-TR-08-01*, 2008.

[52] C. Razafimandimby, V. Loscri, and A.M. Vegni. A neural network and iot based scheme for performance assessment in internet of robotic things. In *Proc. of the International Conference on Internet-of-Things Design and Implementation (IoTDI'16)*, pages 241–246, Orlando, USA, 2016. IEEE.

[53] A.F. Santamaria, A. Serianni, P. Raimondo, F. De Rango, and M. Froio. Smart wearable device for health monitoring in the internet of things (IoT) domain. In *Proc. of the International Conference on Proceedings of the summer computer simulation conference (SCSC'16)*, page 36, Montreal, Canada, 2016. Society for Computer Simulation International.

[54] A. Shemshadi, Q. Sheng, and Y. Qin. Thingseek: A crawler and search engine for the internet of things. In *Proc. of the International Conference on Research and Development in Information Retrieval (SIGIR'2016)*, pages 1149–1152, Pisa, Italy, 2016. ACM.

[55] A. Shemshadi, L. Yao, Y. Qin, Q. Sheng, and Y. Zhang. Ecs: A framework for diversified and relevant search in the internet of things. In *Proc. of the International Conference on Web Information Systems Engineering (WISE'2015)*, pages 448–462, Miami, Florida, USA, 2015. Springer.

[56] I. Stojmenovic and S. Olariu. Data-centric protocols for wireless sensor networks. *Handbook of sensor networks: algorithms and architectures*, pages 417–456, 2005. Wiley.

[57] D. Stutzback, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *Proc. of the International Conference on Internet Measurements (IMC'2006)*, pages 27–40, Rio De Janeiro, Brasil, 2006. ACM.

[58] K. Tei and L. Gurgen. ClouT: Cloud of things for empowering the citizen clout in smart cities. In *Proc. of the World Forum on Internet of Things (WF-IoT'2014)*, pages 369–370, Seoul, South Korea, 2014. IEEE.

[59] N. Tran, Q. Sheng, M. Babar, and L. Yao. Searching the Web of Things: State of the Art, Challenges, and Solutions. *ACM Computing Surveys*, 50(4):55, 2017. ACM.

[60] C. Tsai, C. Lai, and A. Vasilakos. Future Internet of Things: open issues and challenges. *Wireless Networks*, 20(8):2201–2217, 2014. Springer.

[61] M. Tubaishat, J. Yin, B. Panja, and S. Madria. A secure hierarchical model for sensor network. *ACM Sigmod Record*, 33(1):7–13, 2004. ACM.

[62] J. Wan, J. Liu, Z. Shao, A. Vasilakos, M. Imran, and K. Zhou. Mobile crowd sensing for traffic prediction in internet of vehicles. *Sensors*, 16(1):88, 2016. Multidisciplinary Digital Publishing Institute.

[63] J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. Vasilakos. Software-defined industrial internet of things in the context of industry 4.0. *IEEE Sensors Journal*, 16(20):7373–7380, 2016. IEEE.

[64] K. Xu, Y. Qu, and K. Yang. A tutorial on the internet of things: From a heterogeneous network integration perspective. *IEEE Network*, 30(2):102–108, 2016. IEEE.

[65] S. Ye, J. Lang, and F. Wu. Crawling online social graphs. In *Proc. of the International Asia-Pacific Web Conference (APWeb'10)*, pages 236–242, Busan, Korea, 2010. IEEE.

[66] Y. Zhang, D. Raychadhuri, L. Grieco, E. Baccelli, J. Burke, R. Ravindran, G. Wang, A. Lindgren, B. Ahlgren, and O. Schelen. Requirements and Challenges for IoT over ICN. *https://tools.ietf.org/html/draft-zhang-icnrg-icniot-requirements-00*, 2015. IETF Internet-Draft.

[67] Y. Zhang, D. Raychadhuri, R. Ravindran, and G. Wang. ICN based Architecture for IoT. *https://tools.ietf.org/html/draft-zhang-iot-icn-challenges-02*, 2013. IRTF contribution.

[68] J. Zhou, Z. Cao, X. Dong, and A. Vasilakos. Security and privacy for cloud based IoT: Challenges. *IEEE Communications Magazine*, 55(1):26–33, 2017. IEEE.

Biographies

Domenico Ursino received the MSc Degree in Computer Engineering from the University of Calabria in July 1995. He received the PhD in System Engineering and Computer Science from the University of Calabria in January 2000. From January 2005, he is an Associate Professor at the University Mediterranea of Reggio Calabria. From November 2015, he is a Vice Rector, Responsible for Information Technology, of his University. His research interests include Social Network Analysis, Social Internetworking, Source and Data Integration, Ecosystems consisting of Internet of Things, Innovation Management, Knowledge Extraction and Representation, Biomedical Applications, Recommender Systems, Data Lakes. In these research fields, he published more than 170 papers.

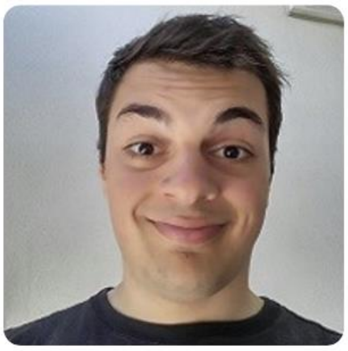
Paolo Lo Giudice received the MSc Degree in ICT Engineering from the University Mediterranea of Reggio Calabria in October 2016. He is currently a PhD Student in ICT Engineering at the same University. His research interests include Social Network Analysis, Social Internetworking, Source and Data Integration, Ecosystems consisting of Internet of Things, Innovation Management, Knowledge Extraction and Representation, Biomedical Applications, Data Lakes. He is an author of 12 papers.

Lorenzo Musarella received the MSc Degree in ICT Engineering from the University Mediterranea of Reggio Calabria in October 2017. He is currently a PhD Student in ICT Engineering at the same University. His research interests include Social Network Analysis, Social Internetworking, Source and Data Integration, Ecosystems consisting of Internet of Things, Innovation Management, Knowledge Extraction and Representation, Biomedical Applications, Data Lakes. He is an author of 4 papers.

Giorgio Baldassarre received the MSc Degree in ICT Engineering from the University Mediterranea of Reggio Calabria in October 2017. He is currently a consultant at SISAL. His research interests include Social Network Analysis, Social Internetworking, Internet of Things.

Photos

Giorgio Baldassarre



Paolo Lo Giudice

Lorenzo Musarella

Domenico Ursino

Highlights

- A new paradigm for modeling the Internet of Things as a set of related networks, instead of on a single network
- A complete formalization of a model for representing and handling a scenario consisting of Multiple Internets of Things
- A new crawler specifically tailored for handling a Multiple Internets of Things context