Querying the IoT Using Multiresolution Contexts

Querying the IoT Using Multi-Resolution Contexts

Claudia Diamantini, Antonino Nocera, Domenico Potena, Emanuele Storti and Domenico Ursino

***Abstract*—People's daily life is increasingly intertwined with smart devices, which are more and more used in dynamic contexts. Therefore, searching and exploiting the wealth of information produced by the Internet of Things (IoT) requires novel models including a representation of the actual context of use. The definition of context is inherently difficult, due to the variety of application scenarios and user needs. In this paper, we propose a general model for devices' contexts representing context components at different resolutions (or levels of granularity). This enables the definition of a multi-resolution context-based algorithm for querying the IoT, according to given preferences and contexts that can be tightened or relaxed depending on the given application goal. Experimental results show how the proposed approach outperforms traditional solutions by increasing the retrieval of relevant results while keeping precision under control.**

***Index Terms*—Query processing, Context modeling, Semantics, Multidimensional systems, Internet of Things**

I. Introduction

Searching the Internet of Things (IoT) [1] or its Multi-IoT (MIoT) extension [2]–[4], has recently received considerable interest. The number of devices endowed with connection capabilities is increasing. This opens the doors to a variety of disruptive applications, but also raises the need for more effective mechanisms to discover devices and resolve queries for their content. In particular, [5] distinguishes a *discovery* activity, a sort of crawling that is able to identify various collections of IoT resources, and a *searching* activity, devoted to identifying a subset of discovered IoT content as search results of a given query. In this paper we focus on the latter, assuming a given device's network (i.e. the contact list of a device in the IoT) as the collection of resources. A characteristic that differentiates IoT resources from others like the web, is that the former are far more mobile and dynamic. Mobile devices, changing their position continuously, also rapidly change their neighborhood and the environment they are immersed in. A number of criteria for network building has been devised with the aim to make the IoT more and more autonomous [6]: *(i) proximity*, *(ii) homogeneity*, *(iii) ownership*, (iv) *co-working* in a given application and *(v) sociality* (users' devices are connected to each other in some social networks).

As for this last criterion, nowadays a lot of applications for smart devices exploit information coming from their owners' online social networks, e.g. to retrieve their contact lists. Moreover, since devices become more and more autonomous, the way they interact with each other is changing as well. This gives the IoT a more social connotation [7], but results in more complex scenarios in which heterogeneous peers can interact, contexts can rapidly change, as well as the meaning and the relevance of the information exchanged. In these dynamic environments, existing content-based search strategies can be of limited use. To better illustrate the issue, a *motivating scenario* is reported hereby: Laura likes to keep fit and make outdoor workout, in particular running in the countryside. She also needs to check outdoor conditions (temperature, humidity, cloud coverage or raining) before going to her preferred sites in the neighborhood, because her health status requires specific weather conditions. She may obtain coarse-grained weather information from weather services, however she could also obtain more timely and localized information by asking for the values of position, temperature, and brightness measured by her network of devices (more specifically, by the network of the device she uses for querying). Let us assume that one day Laura enters a bus where other 50 persons have a smart device. Traditional proximity criteria would lead to connect Laura's devices to other devices nearby. However, it turns out that most people on the bus are children coming back from school, who spend most of their time at home or doing indoor workout. It is then clear that when Laura wants to know about weather conditions, information about temperature or brightness obtained at query time from these new contacts will probably be of little use. Indeed, they would have been acquired in a context (studying at home or running in the gym) different from Laura's desired one (running in the countryside).

The availability of a search mechanism for scoring devices on the basis of their context would thus help to reduce the number of queries and/or filter out irrelevant answers. It becomes immediately apparent, however, that the way in which context similarity is established is critical, as it possibly leads to loose potentially useful information. For instance, a query asking for devices located at a very specific longitude and latitude is likely to have no answer, but such a geographical precision is obviously unnecessary to query brightness or temperature. Furthermore, even slightly changing a user's goal, the relevance of the information provided by a given device can greatly change: referring again to the previous scenario, discriminating between indoor and outdoor location may be sufficient in some situations, but a more precise location of a device, discriminating between city and country areas, or

Claudia Diamantini, Domenico Potena, Emanuele Storti and Domenico Ursino are with the Department of Information Engineering, Polytechnic University of Marche, 60131 Ancona, Italy (email: $c.diamantini@univpm.it$, $d.potena@univpm.it$, $e.storti@univpm.it$, $d.ursino@univpm.it$)

Antonino Nocera is with the Department of Electrical, Computer and Biomedical Engineering, University of Pavia, 27100 Pavia, Italy, (email: $a.nocera@unipv.it$)

among different cities, can be better for Laura's goal.

In this paper, we address these issues by proposing a general model for devices' contexts whose main feature is the representation of context components at different levels of granularity, or resolution, which enables the definition of a multi-resolution context-based algorithm for querying the IoT. The search is performed according to given preferences and contexts that can be tightened or relaxed depending on the given application goal. The novel contributions of this paper are summarized as follows:

- (1) the *context model* enabling the search mechanism is represented by a set of dimensions described through different granularity levels.
- (2) On the top of the model, we define the notions of similarity and compatibility among contexts at different levels of precision, that underpin the *multi-resolution querying mechanisms*.
- (3) The abstract query model is discussed in the perspective of modern IoT networks, often characterized by highly heterogeneous devices and network characteristics. Different *query execution models* are analyzed and their pros and cons discussed.

Finally, a comprehensive assessment of the efficiency and the effectiveness of the multi-resolution approach is provided, by means of both analytical and experimental evaluations. The ideas reported in the present paper have been preliminarily presented in [8], [9]. With respect to previous work, the present paper considerably extends and formalizes the context model and the context-based querying algorithm, introduces the discussion on the query execution models, and provides a comprehensive analytical and experimental evaluation.

The rest of the paper is structured as follows: Section II organizes and discusses related literature. Section III introduces the context model. An algorithm for query answering is illustrated in Section IV. Section V provides an analysis of the characteristics and performances of different execution models, also comparing them with baseline approaches. Finally, Section VI draws conclusions, enlightening also future work.

II. Related Work

A. Query answering in IoT networks

Query answering over an IoT network and discovery of IoT devices and their content are relatively new areas of investigation that are increasingly gaining interest. In a recent work [5], the term "Internet-of-Things Search Engines" (IoTSE) has been proposed to collectively account for all the approaches aiming at information discovery over an IoT network. In the last years, some effort has been put by researchers and practitioners to develop several approaches mostly differing in the type of content retrieved (e.g., device metadata, device functions, data streams) and objective, e.g. device selection or scoring to retrieve a ranked list of devices satisfying some criteria, as we do in our work.

Search approaches start from a query on metadata/content (R), sensing data streams (D), functionalities (F), static information of IoT content (S) or a mix of them. The first class (R) is the most frequent one and includes, among others, approaches to query devices starting from their identificators (returning their description) [10], or from textual data (returning relevant things), practically implementing a distributed low-powered text-retrieval system [11]. To extend the expressive capabilities of queries, [5] proposes a representation of devices and their content based on a heterogeneous graph, where each edge stands for a correlation relation between two nodes (e.g., a device and its content, or two devices). A meta-path is then defined as a path in the graph connecting nodes, which can be returned as an answer to a query on the graph. Our approach shares some aspects with the described work, such as the capability to specify queries on both metadata (i.e., device properties) and content (i.e. in terms of the context), although the reference to a multidimensional representation is novel with respect to the mentioned literature. Another family of approaches (D) are focused on querying low-level sensing data, instead of device metadata or content (e.g., [12]) Finally, other examples include querying specific metadata properties, such as the set of functions or capabilities provided by devices. This functionality, which is also enabled by our approach, is provided for instance by [13], which uses an ontology to model all high-level functionalities.

B. Context models and languages

Context modeling and context-aware systems spread in the research literature as soon as mobile devices have been introduced. One of the first characterizations of context is given in [14] where authors mention three important aspects, namely where you are, with whom you are, and what resources are nearby. Location has been the first modeled and exploited element in applications dealing with smart spaces and tourism [15]. Considering the limited features associated with the location element (basically, positioning), context models were developed at a very low level, and mainly coded into the application logic as taxonomies [16]. The need for a more general and comprehensive definition of the notion of context has been recognized e.g. in [17], where a context is defined as any information about an entity relevant to the interaction between a user and an application.

Research effort has been put also on models and languages to design contexts. Knowledge representation approaches try to decouple context representation from applications, developing proper models and languages to design and characterize richer context models as first-class citizens. Models for context representation have been extensively investigated in the literature (see surveys in [12], [18]), and can be simply defined as sets of attribute-value pairs [19]–[21], also using mark-up languages [22] and possibly formalized as context ontologies. The elective languages for the specification of ontologies are logic-based, e.g. RDF and OWL, which allow context-related entities to be modeled as a conceptualization of a domain, providing a rich semantic representation and reasoning capabilities, support to sharing and reuse among applications (e.g., [23]–[26]). Another example is [27] where the Context Aware Sensor Configuration Model (CASCOM) is proposed to simplify the process of configuring IoT middleware platforms,

easing the retrieval process for data consumers, especially non-technical personnel. The richness of ontologies has complexity as a shortcoming, which can be critical for many applications like in the IoT scenario, where computing abilities are limited, as also recognized in [28], [29]. Geometrical spaces are also considered to model contexts, e.g. Context Spaces [30].

Most of these models do not consider context values at different granularity levels. Ontologies considers hierarchical structures (e.g. specialization) only for the definition of sub-concepts, while an exception can be found in [31], where an ontology has been adopted for richer models of spatial information with levels of granularity. In Context Spaces, a context is represented as a point or region in a multidimensional space. Dimensions are *flat* sets of admissible values. In fact, a model taken as an abstraction is the relational database model. In contrast, in this paper we take as abstraction the multidimensional model typical of data warehouses, where dimension values are organized into a lattice defined by a roll-up partial order, representing increasing aggregation levels. As a consequence, reasoning with a multi-resolution perspective is enabled. The Context Dimension Tree hierarchical model is proposed in [32] for wireless sensor network applications. In the model, leaf nodes represent simple values of the dimension, while structured values can be represented by nesting one more level of the dimension, thus defining a tree structure. The hierarchy in this proposal is thus devoted to represent complex context elements, whereas in our proposal we deal with values at different granularity. As will be clear later, by managing granularity of context dimensions we overcome the need to define in advance which information is part of context and which is not, as this can be decided by the application at run time, hence contributing to overcome limitations of knowledge-based context representation techniques.

.

C. Social Networks and Social Internet of Things

In the past literature, the task of defining contexts in IoT has been often intertwined with that of empowering the IoT of social features. Indeed, the idea of studying device contexts has risen from the need of improving the quality of the interactions among them.

The idea of bringing social network concepts into the Internet of Things is something that finds its roots back in 2001. In fact, in [33], the authors describe the idea of filtering contacts based on context proximity. An important aspect of the proposal is that the authors actually suggest two forms of proximity that can be used to trigger the creation of new relationships, namely spatial proximity and context proximity. The former is the classical notion of proximity considering the coexistence of objects in the same location and a distance metric, whereas the latter combines both physical location with other metrics, such as the movement patterns. The idea is mainly to allow the establishment of connections for devices worn by the same person. A more complex idea is presented in [34], in which objects are designed as social entities and are referred with the name of *blogjects*, i.e. object that can blog. Still on this context, the paper presented in [35] puts emphasis on the possibility of using social networks as an easy-access channel to improve communication among objects and among humans and objects.

Of course, the evolution of the IoT towards social networks can also lead to consider objects as a mean to find new powerful ways to share contents and update human status in those systems. This is exactly the idea described in [36], in which a new design is proposed to support elder citizens to maintain their content in social sites up-to-date by leveraging smart devices. As a natural consequence of these research efforts, more recently, several authors have started to propose unified frameworks considering virtual profiles of both humans and smart devices as entities collaborating to reach goals and produce results [37]–[39].

The concept of *Social Internet of Things (SIoT)* has been introduced in [6]. The paper discusses both architectural issues to enable smooth integration of objects in existing social networks and policies to manage the (social) interaction among objects. A discussion on how social features may impact optimization and control of direct IoT communications is reported in [40]. Other work, instead, seeks to exploit this new social nature of objects for disparate application scenarios. One of the first attempts in this direction is the work described in [41], in which a new strategy for exploiting the SIoT for recommendation services is obtained by leveraging the SIoT as a mean to share data cross-application.

All the approaches described in this section are, in principle, related to ours. Indeed, our approach belongs to the last set of works, trying to leverage the social nature of objects to improve their interoperability bringing advantages to their owners. However, in our paper, we seek to define a strategy to improve the quality of connections among objects by using social information and the semantic nature of the interaction between humans and objects. To the best of our knowledge this is the first solution proposed in this setting.

III. A MODEL FOR DEVICES IN AN IOT NETWORK

This section is devoted to introducing the model for devices and their contexts. First of all, let us notice that modern devices are composed of a bundle of sensors. Each device as a whole has a set of properties, e.g. a brand and an owner. Also, a device and each component sensor inherit some properties from other sensors. For instance, the position measured by a GPS sensor is by direct extension the position of each sensor of the device it belongs to and of the device itself.

Definition 1: **Device Model.** A device is a triple $\Delta = (S, P, c)$. $S = \{S_1, \ldots, S_n\}$ is a set of simple sensors responding to a single physical stimulus. $P = \{(p_1, v_1), \ldots, (p_k, v_k)\}$, is a set of device's properties, where $p_i \in \Sigma$ is a signature of property names, and $v_i \in Dom(p_i)$ is one of the possible values in the domain associated with p_i. $c = (\iota_1, \iota_2, \ldots, \iota_m)$ is a tuple of sensor values organized according to the context model defined in the following.

We hasten to notice that, for reasons that will be clear later, the dimension m of the vector c does not need to be equal to the number of sensors n.

As a possible definition of Σ we consider $\Sigma = \{$*owner*, *brand*, *model*, *measure*$\}$. The first three properties in the list

describe non-exhaustive, self-explaining characteristics of a device. The *measure* property is introduced to list the kind of measures a device can provide, thus describing device's capabilities. For example, the pairs $(measure, Position)$, and $(measure, Time)$ express the capability of a device to give information about current location and time. We also like to introduce a further kind of capability, expressed by the pair $(measure, Goal)$. It declares the ability of a device to provide information about the current activity the device is used for (e.g. "running", "sleeping", or "reading"). Typical values of $Goal$ are not raw data coming from sensors, like a time, hence $Goal$ is not a measure in a strict sense. However, those values derive from a classification of the activity performed by processing sensor data. For instance, step counters can distinguish between "running" or "walking" by considering the pace at which steps are recorded, or sleep trackers can be able to classify the different sleep stages.

Properties in P are basically fixed. On the other hand, device's sensors produce data whose values vary over time, and can be exploited to define the device's context c, according to the model introduced in the next subsection.

A. Context of a device

The proposed context model is based on a set of dimensions $\mathcal{D} = \{D_1, \ldots, D_m\}$, where each dimension is defined by a lattice of admissible values. This structure is borrowed from the notion of dimension typical of data warehouses. At the best of our knowledge, its adoption as a model for contexts has never been proposed before. We provide the following definitions of dimension and roll-up, elaborated from [42]:

Definition 2: **Dimension.** A dimension D consists of:

- a scheme $Sc(D)$, made of:
  - a finite, non-empty set of levels $L = \{l_1, \ldots, l_w, \top\}$,
  - a partial order $\preceq_L$: $L \times L$. If $l_i \preceq_L l_i$ we say that l_i rolls-up to l_j,
- an instance $I(D)$, defined by:
  - a set of instances, or members, $M_i = \{\iota_1^i, \ldots, \iota_k^i\}$ for each level l_i,
  - a family of roll-up functions $\rho^{l_i \to l_j} : M_i \to M_j$ for each pair of levels $l_i \preceq_L l_j$.

The partial order $\preceq_L$ defines a lattice, where the level $\top$ is the top: $\forall l_i, l_i \preceq_L \top$. Its instance consists of a unique special element, denoted by *all*. Similarly, one of the levels l_i is the bottom: $\forall l_j, l_i \preceq_L l_j$, and includes raw sensor values as its instances. In the following, we will use the symbol $\preceq_L$ to denote the direct order relation, and the notation $\preceq_L^{(s)}$ to explicit transitivity when needed, i.e. $l_i \preceq_L^{(s)} l_j$ if $\exists \{l'_1, \ldots l'_s\} \subseteq L : l_i \preceq_L l'_1 \ldots \preceq_L l'_s \preceq_L l_j$, for $s \geqslant 1$. On the other hand, $l_i \preceq_L^{(0)} l_j$ means that $l_i \preceq_L l_j$. Furthermore, with some abuse of notation, we will see the family of roll-up functions as a unique relation denoted by $\rho(\iota_p^i, \iota_q^j)$. Finally, let us introduce the following notion of distance.

Definition 3: **Roll-up distance between members.** Given a dimension $D \in \mathcal{D}$, let $\iota_p^i, \iota_q^j \in D$ be two instances of levels l_i, l_j, with $l_i \preceq_L^{(s)} l_j$. The roll-up distance between ι_p^i and ι_q^j, denoted by $dist(\iota_p^i, \iota_q^j)$, is defined as the number of steps over the roll-up relation necessary to move from l_i to l_j: $dist(\iota_p^i, \iota_q^j) = s + 1$. Of course, $dist(\iota_p^i, \iota_q^j) = 0$ if $l_i = l_j$, $dist(\iota_p^i, \iota_q^j) = \infty$ if $l_i \not\preceq_L l_j$.

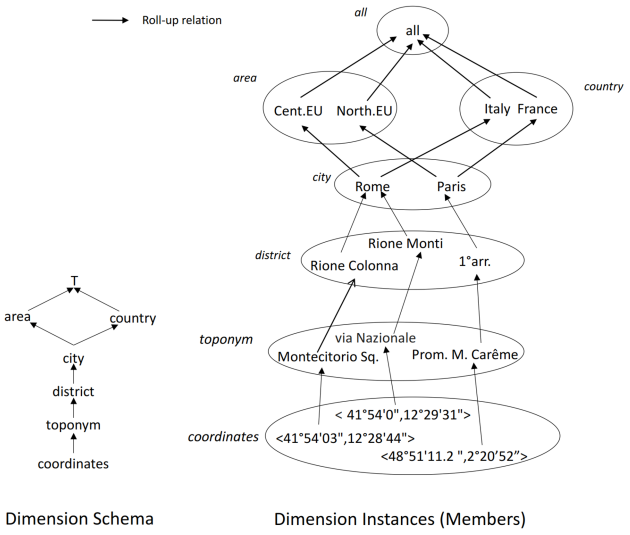

Fig. 1: Schema and instance of the Position dimension

Each level of a dimension represents a level of detail or granularity at which it is sensible to consider data for a given application. In the definition of a device's context, dimensions are built upon data coming from sensors, i.e. each D_i is one of the values of the *measure* type property (e.g. *Position* and *Time*). Figure 1 shows an example of the *Position* dimension.

Besides roll-up, the following relations are defined:

Definition 4: **Identity and compatibility of instances.** Given a dimension $D \in \mathcal{D}$, let $\iota_p^i, \iota_q^i \in D$ be two instances of level l_i of D,

- $id(\iota_p^i, \iota_q^i)$: if ι_p^i and ι_q^i are the same value or synonyms, e.g. id(*Montecitorio square*, *Montecitorio Sq.*);
- $cpt(\iota_p^i, \iota_q^i)$: if ι_p^i, $\iota_q^i \in M_i$ and $\exists \iota_r^j \in M_j$: $\rho(\iota_p^i, \iota_r^j) \wedge \rho(\iota_q^i, \iota_r^j)$, $l_i \preceq_L l_j$. In practice, $cpt(\iota_p^i, \iota_q^i)$ if ι_p^i and ι_q^i roll-up to the same higher-level member.

The identity relation is introduced to account for possible heterogeneities in the representation of members by different devices. Compatibility accounts for a sort of similarity between members that are mapped to the same member at the higher (i.e. coarser) level. For instance, for the dimension in Figure 1, cpt(*Rione Colonna*, *Rione Monti*), cpt(*Italy*, *France*), whereas it does not hold cpt(*Rome*, *Paris*).

Note that relations actually depend on the given dimension schemas. For instance, if we substituted the two levels *country* and *area* with a unique level *continent* in the *Position* dimension schema, then *Rome* and *Paris* would turn out to be compatible. Different dimension schemas can be suitably defined for different applications and must thus be agreed among devices. In this sense, dimension hierarchies, together with identity relation, define a knowledge base for the network and the given application. Aspects related to mechanisms for dimension schema management and integration are out of the scope of the present paper. We refer to work in data warehouse integration dealing with these issues (see e.g. [42], [43]).

We are now ready to define the notion of context.

Definition 5: **Context.** The context C of a device consists of:

- a schema $Sc(C)$, defined by a set of dimensions $\mathcal{D}_C \subseteq \mathcal{D}$;
- an instance $I(C)$ (or c for short) defined as a tuple $c = (\iota_1^*, \iota_2^*, \cdots, \iota_c^*) \subseteq \times_{D \in \mathcal{D}_C} M_{*D}$, where $M_{*D} = \bigcup_{i=1}^{w_D} M_{iD}$, is the union of the members of each level l_i of the dimension D. In the following we will refer to ι_j as any instance of any level of dimension D, dropping superscript to simplify notation.

To make an example, considering the context schema $\{Position, Time, Goal\}$, a possible instance is: $\iota_{Position}$= *Montecitorio square*, ι_{Time}= *July* 22^{nd} *2020 [11:00-12:00]*, ι_{Goal} = *Running*. A second example is $\iota_{Position}$= *Rome*, ι_{Time}= *July 2020*, ι_{Goal} = *Running*. We can also consider $\iota_{Position}$= *all*, ι_{Time}= *all*, ι_{Goal} = *all* as a context instance. We will denote such a context by c_{all}. In other terms, a context can be specified at any level of detail for each dimension: the higher the level, the coarser its specification, with the c_{all} instance meaning somewhere, sometime, some goal.

With this observation in mind, it is simple to see that, although in principle the number of dimensions forming a device's context cannot exceed the number of measures the device is able to produce, the context schema of any device can be generalized to $\mathcal{D}$ assuming $\iota_k = all$ for any $D_k \in \mathcal{D} \backslash \mathcal{D}_C$, meaning that the device is not able to provide a specific value for the missing measures. Hence, we can consider homogeneous context schemas for any device equal to $\mathcal{D}$, that is the set of measures provided by existing sensors.

The specification of the device's context at different levels of detail is suitably exploited by query answering techniques discussed in the next section. In particular, these techniques will make use of the following relations between context instances, which are based on the relations between dimension instances defined before.

Definition 6: **Identity, roll-up, and compatibility of contexts.** Given two context instances $c = (\iota_1, \iota_2, \cdots, \iota_m)$ and $c' = (\iota'_1, \iota'_2, \cdots, \iota'_m)$:

- $id_C(c, c')$, if $id(\iota_i, \iota'_i)$, $1 \leqslant i \leqslant m$; for instance c=(*Montecitorio square*, [10:00-11:00], *Running*) c'=(*Montecitorio sq.*, [10:00-11:00], *Running*).
- $\rho_C(c, c')$, if $id(\iota_i, \iota'_i)$ or $\rho(\iota_i, \iota'_i)$, $1 \leqslant i \leqslant m$ and $id_C(c, c')$ does not hold; for instance c=(*Montecitorio square*, [10:00-11:00], *Running*), c'=(*Montecitorio sq.*, *Morning*, *Running*). We define the distance $dist(c, c')$ between c and c', as $dist(c, c') = \sum_{i=1}^{m} dist(\iota_i, \iota'_i)$, where $dist(\iota_i, \iota'_i) = 0$ if $id(\iota_i, \iota'_i)$. We will also say that c is a context *finer than* c' or, viceversa, that c' is a context *coarser than* c.
- $cpt_C(c, c')$, if either $id(\iota_i, \iota'_i)$ or $\rho(\iota_i, \iota'_i)$ or $cpt(\iota_i, \iota'_i)$, $1 \leqslant i \leqslant m$ and neither $id_C(c, c')$ nor $\rho_C(c, c')$ holds; for instance, c=(*Rione Colonna*, [09:00-10:00], *Running*) c=(*Rione Monti*, [09:00-10:00], *all*).

We like to end the section noting that, besides *Time* and *Position*, the model is suited to represent as context dimensions any factor that can be measured or, generally speaking, exposed by a device and typically taken into account in the IoT literature (e.g., [12]). For instance, one may want to know the temperature measured by devices located in a clear sky area. In this case the context is represented by the cloud coverage dimension, whose values can easily be organized in a hierarchy of ranges. Goals, as well as other more abstract context factors or situations, can also be part of the context provided that a classification exists, and a lattice of classes can be defined.

IV. Query answering over an IoT network

In this section, we discuss a Context-Based Query Answering (CBQA) algorithm, aimed at retrieving the subset of devices in a user network $\mathcal{N} = \{\Delta_1, \ldots, \Delta_N\}$ that satisfy the requirements expressed by a user query, sorting results by relevance. We refer to a disjunctive query in the form $q = \langle (x_1 \vee x_2 \vee \ldots \vee x_t), Z \rangle$, where each x_i is a context of interest and $Z = \{(p_1, v_1), (p_2, v_2), \cdots, (p_p, v_p)\}$ represents the set of properties that must be satisfied by the devices. Referring to our example scenario, Laura's device may ask a query to check outdoor conditions: *q=⟨(Via Cassia 1081 Roma, [8:00-9:00])∨(Via della Marcigliana Roma, [8:00-9:00]),{(measure, Position), (measure, Temperature),(measure, Brightness)}⟩*.

The query is satisfied by a device if its context matches (at least) one of the query contexts $x_1, \ldots, x_t$[1]. Without loss of generality, we discuss hereby only the case t=1, i.e. a query specifying one context. The discussion can be generalized by considering the disjunction of contexts as disjunction of queries, i.e. $q = \langle x_1 \vee \ldots \vee x_t, Z \rangle = q_1 \vee \ldots \vee q_t$, with $q_i = \langle x_i, Z \rangle$. Indeed, by referring to $A(q)$ as the answer set of a query q, the following holds: $A(q) = \bigcup_{i=1}^{n} A(q_i)$.

The algorithm returns a ranked list of devices satisfying the query $\Delta_O = \{(\Delta_i, r_i) : \Delta_i \in \mathcal{N}, r_i \in [0, 1]\}$, where r_i represents the relevance of the device with respect to the query. As reported in Figure 2, the steps of CBQA are the following:

- *Context & properties evaluation.* For each device $\Delta_i \in \mathcal{N}$, the algorithm evaluates whether its properties and context match the user query q. If so, the device is added to the output list (lines 3-8).
- *Query relaxation.* If no device is retrieved in the previous step, the query q is rewritten as $q' = (x', Z)$, where x' is a context coarser than x: $\rho_C(x, x')$ (lines 9-16).

Context and properties evaluation will be discussed in detail in Subsection IV-A. Query relaxation is a key feature of the approach enabled by the proposed hierarchical context definition. It allows to reduce the precision of the specified context (e.g., a square), thus accepting in principle less precise measures (e.g., the square's district), in order to increase answering capability. For many applications, measures provided by these devices are still acceptable, although not as precise as initially wanted. As a matter of fact, moving from a given context to a coarser context, the loss of precision depends on the variability of a measure with respect to dimensions. For instance, Time is

[1]A device can be in only one context at a time; thus, it can match more than one query context only if the contexts are identical or there is a roll-up relation between them. We consider this situation for the sake of generality, being the practical interest of this kind of query limited.

1. **CBQA**($\mathcal{N}, q, \theta_r, k$):
2. $\Delta_O \leftarrow [\ ]$
3. **for all** $\Delta_i \in \mathcal{N}$ **do**
4. r_i=k*EVALUATE(Δ_i,q)
5. **if** $r_i \geqslant \theta_r$ **then**
6. $\Delta_O \leftarrow \langle \Delta_i, r_i \rangle$
7. **end if**
8. **end for**
9. **if** $\Delta_O \neq \varnothing$ or $q.x = c_{all}$ **then**
10. **return** Δ_O
11. **else**
12. $q^{'}$=REWRITE(q)
13. $k^{'}$=$k * \gamma_1$
14. $\theta_r^{'}$=$\theta_r * \gamma_2$
15. **return** CBQA($\mathcal{N}, q^{'}, \theta_r^{'}, k'$)
16. **end if**

Fig. 2: CBQA

constant inside each time zone. Thus, if an application seeks the value of time, e.g. at street level, moving to upper levels like city, region or state does not reduce measure precision and device relevance for that query. Similarly, Brightness varies smoothly with respect to Position, while measures like Temperature or Humidity show a complex pattern with respect to Position, as they are affected also by environmental factors like the presence of artificial heating systems. For lack of space, here we simply reduce the relevance of a device by a constant factor $\gamma_1 \in (0, 1]$ at each relaxation (line 13), assuming in practice the same amount of precision loss for each measure and each context dimension. Under this assumption, the relaxation strategy in the REWRITE function (line 12) simply needs to randomly choose one dimension, moving one level up in the dimension lattice (while in the general case it is sensible to start relaxing dimensions with a smaller impact on measure precision). Other more complex relaxation strategies can be put in place, managing statistical information in order to improve answer probability or device recall. Besides context, the algorithm allows to relax the threshold controlling the degree of relevance by a factor $\gamma_2 \in (0, 1]$ (line 14). The $CBQA$ algorithms is then called again, recursively, with the new q', k', and θ' (line 15). Recursion ends when either a solution is found, or a solution does not exist (line 9).

It is easy to see that, if $\theta_r = 0$, the algorithm always finds at least the trivial solution, formed by all devices in the user network. In fact, eventually, the query context is relaxed to c_{all} for which $\rho(c_i, c_{all})$ always holds for any device Δ_i. However, devices retrieved may not have the necessary capabilities or other requested properties. At the other extreme, when $\theta_r = 1$, only devices perfectly matching query properties can be retrieved. In case no device is retrieved, the algorithm stops when reaching the maximum context relaxation $x = c_{all}$. Different stopping criteria can be defined, for instance by limiting the number of relaxation steps that should be performed

A. Context & properties evaluation

This subsection discusses the context and property evaluation phase. Given a device Δ_i and a query q, context and properties evaluation is aimed at determining: *(i)* whether the device should be returned as output and, in the affirmative case, *(ii)* to what extent it is relevant to the query. Evaluation is first done by comparing the current device's context c_i with the query context x, and then the device properties P_i with the query properties Z, as shown in Figure 3:

- First, the function checks whether the device context c_i is identical to x ($id_C(x, c_i)$) or finer than x ($\rho_C(c_i, x)$). The rationale under this condition is that if a query asks for a context specified at a certain level of granularity, then any device with a context specified at a finer level is a candidate device. For instance, devices located in *Montecitorio Square* should be retrieved when looking for devices in *Rome*. If this happens, the set Z of desired properties is compared to the set of properties P_i of the device under consideration, to evaluate its relevance. Among possible suitable metrics, here we consider the Jaccard metric $r_i = J(P_i, Z)$=$\frac{|P_i \cap Z|}{|P_i \cup Z|}$, which measures the number of property pairs of P_i in common with Z, divided by the total number of pairs in their union.
- Conversely, if the device context is neither identical or finer than the query context, the function checks if a compatibility relation holds (cpt_C). In this case, the relevance value r_i is equal to the Jaccard index weighted by a factor $\alpha_{cpt} \in (0, 1)$, i.e. $r_i = \alpha_{cpt} * J(P_i, Z)$, which accounts for the precision loss of results.

1: **eval**(c_i, P_i, x, Z)
2: $r_i \leftarrow 0$
3: **if** $id_C(c_i, x) \vee \rho_C(c_i, x)$ **then**
4: $r_i \leftarrow J(P_i, Z)$
5: **else**
6: **if** $cpt(c_i, x)$ **then**
7: $r_i \leftarrow \alpha_{cpt} * J(P_i, Z)$
8: **end if**
9: **end if**
10: **return** r_i

Fig. 3: Context & properties evaluation

B. Query execution models

A reasonable concern may arise regarding how the abstract model can be actually implemented in modern IoT networks, more and more characterized by a high heterogeneity of computation, network and energy capacity of involved entities. Modern IoTs include devices ranging from simple smart-sensors (e.g., smart meters, smart temperature sensors) to more sophisticated devices like smartphones or smart driving cars [41]. Of course, this can introduce some limitations to the applicability of our approach: clearly, a low-capacity device can hardly be used for querying the IoT, even if it can still be involved in providing information to build answers to other devices' queries (see also [12] for a discussion). In the

following, we investigate strategies for coping with this issue, discussing local and distributed execution models. We hasten to notice that the discussion will be focused on computation and storage capacity, and network traffic, which primarily constraint other factors like power consumption or connection quality.

Local evaluation: according to this execution model, context and properties of every device are fetched, and context & properties evaluation is performed at the *querying* device. To this aim, both the context c_i and the properties P_i of the device at hand must be available locally. According to the kind of storage distribution, namely distributed or local, the querying device needs to retrieve context and properties from Δ_i or to look up for the needed information in an internal repository or cache, respectively. If queried devices have static contexts (e.g., weather stations that do not change location or goal and return a value a few times a day) the local storage model allows to speed up query answering and limit network traffic. Instead, for devices with highly dynamic contexts (e.g., wearable devices, such as fitbands or smart watches), a distributed storage is more efficient as it avoids continuous updates of the contexts status. Intermediate solutions are also possible, e.g. storing contexts for a subset of devices only, and adopting policies for the storage refresh frequency.

The function `EVALUATE_L` for the local evaluation is described in Figure 4. According to the local or distributed storage policy, the function `GET_CONTEXT_AND_PROPERTIES` (line 10) is executed by looking up the needed information in the local storage or directly querying the device by calling a proper service, respectively. Once information is obtained, the `EVAL` function is called (line 11). In the following, we use the symbol $CBQA_L$ (resp., $CBQA_{LS}$) to denote the variant using local computation without a local storage (resp., with local computation and local storage). In this model, the querying device needs to be provided with considerable computational (and possibly storage) capacity, but queried devices need minimal specifications.

1: **evaluate**(Δ_i, q)
2: **if** computation=local **then**
3: **return** EVALUATE_L(Δ_i, q)
4: **else**
5: **return** EVALUATE_D(Δ_i, q)
6: **end if**
7:
8: **evaluate**$_L$(Δ_i, q)
9: $r_i \leftarrow 0$
10: $\langle c_i, P_i \rangle \leftarrow$ GET_CONTEXT_AND_PROPERTIES(Δ_i)
11: $r_i \leftarrow$ EVAL(c_i, P_i, x, Z)
12: **return** r_i
13:
14: **evaluate**$_D$(Δ_i, q)
15: $r_i \leftarrow$ REQUEST_EVAL$_{\Delta_i}(q)$
16: **return** r_i

Fig. 4: Local and distributed policies

Distributed evaluation: in a distributed execution model, context & properties evaluation is performed at the *queried* device. As such, there is no need for the querying device to cache contexts/properties of the other devices in the network. The function `EVALUATE_D` is reported in Figure 4 and involves the call of a `REQUEST_EVAL` function, which calls a device service, that we assume to be available, performing the actual `EVAL` function with the local context and properties. Only devices with a relevance greater than zero will be returned as a result. In the following, we refer to the $CBQA$ variant using a distributed evaluation model by $CBQA_D$.

A distributed execution model facilitates the execution of queries by devices with limited capacity. In fact, the burden of context computation is put on the queried device: this makes a partial computation to derive its relevance to the query, according to its properties and current context. In this case, also a device like a smart watch or simpler, may perform queries, as the only required functionality is to send query messages and receive answers. Further analyses are presented in the next section.

TABLE I: Main features of the baseline algorithms and CBQA

Approach	Context	Computation	Storage
Baseline$_1$	no	local	no
Baseline$_2$	yes[a]	local	no
CBQA$_{LS}$	yes	local	yes
CBQA$_L$	yes	local	no
CBQA$_D$	yes	distributed	no

[a]With no knowledge base

V. Evaluation

In this section we compare the three implementations of the supervised algorithm discussed in the previous section, namely $CBQA_L$, $CBQA_{LS}$ and $CBQA_D$ with two baseline algorithms capturing the general behavior of existing approaches:

- $Baseline_1$. The algorithm has no notion of context: referring to the case study presented in the Introduction, it is as if Laura asked for the values of Position, Temperature and Brightness to her whole network of devices. We implement this algorithm by considering the query $q = \langle \varnothing, Z \rangle$.
- $Baseline_2$. In this case, a *flat* context is considered. This means that only an identity of contexts can be checked, while the lack of a proper knowledge base and related reasoning mechanisms prevents the evaluation of roll-up or compatibility relations among contexts.

Table I summarizes the main features of the approaches. In the following, we first propose an analytic evaluation of the efficiency based on *performance indicators* measuring traffic parameters that are recognized important in the literature (e.g. [1], [21]). Then, we discuss quantitative results enlightening the efficiency and query answer capability of the proposed multi-resolution mechanism. Answering capability can be either meant as the probability to obtain an answer for a given query (i.e. at least one device is retrieved), or as device recall capability, i.e. the number of devices retrieved. In fact, for some applications, values from just one device may suffice,

while in other cases one may want to retrieve as many values as possible. In the following, we will take into account both figures of merit.

A. Analytical evaluation

Let $q = \langle x, Z \rangle$ be a context query, and N be the size of the network in terms of number of devices. We analyze the performances of the five approaches with respect to (1) number of devices contacted per query, (2) network traffic, and (3) execution time. The main outcomes of the analysis are reported below and summarized in Tables II and III:

- *Number of devices contacted per query.* Each user query needs to be evaluated. All approaches require to contact all N devices in the user network in order to answer the query, with the exception of $CBQA_{LS}$, that manages a local storage of device contexts and properties; hence, it does not require any context query to be launched over the network. On the other hand, all the N devices need to be contacted at setup time, or whenever a refresh of the storage content is scheduled.
- *Network traffic*. It is expressed in terms of the overall size of the responses obtained from the devices contacted by a context query. As reported in Table II, for $Baseline_2$ and $CBQA_L$, every contacted device returns its context and properties to the querying device. Hence, the traffic generated by one response from a device is equivalent to the size of the context and properties, which we estimate to be comparable among devices. Therefore, the overall traffic is proportional to the number N of devices contacted. On the other hand, as for $CBQA_D$, the evaluation is performed in a distributed fashion by each contacted device. As a consequence, only the result is transferred, hence reducing the total traffic. Finally, algorithm $CBQA_{LS}$, as already mentioned, generates traffic only at setup time. Since this analysis focuses on the traffic generated by the retrieval of contexts, $Baseline_1$ is not taken into account here, as the notion of context is not available for this approach.
- *Execution times*, expressed as a sum of different contributions, namely: t_{req} (time for sending out a query), t_{retr} (time to obtain the response from a device), t_{eval} (time to evaluate whether a device context and properties satisfy the query). Both $Baseline_2$ and $CBQA_L$ share the same estimation of the execution time, given by the time necessary to launch N queries and wait for the responses, plus the time for the evaluation of N contexts/properties. However, it has to be noted that for $Baseline_2$ the evaluation time is shorter, as the only context evaluation enabled by this approach is on the identity of contexts. Unlike the mentioned approaches, the query time for $CBQA_{LS}$ is only due to the evaluation of the N contexts, as they are already available in the local storage. On the other hand, $CBQA_D$ is able to grant a shorter time as the evaluation time is done in parallel by the contacted devices.

TABLE II: Comparison among the baseline algorithms and $CBQA$ in terms of number of devices contacted per query and network traffic

Approach	# contacts		Network traffic	
	Setup	Query	Setup	Query
Baseline_2	0	N	0	size(c+P)*N
CBQA_L	0	N	0	size(c+P)*N
CBQA_{LS}	N	0	size(c+P)*N	0
CBQA_D	0	N	0	size(r)*N

TABLE III: Comparison among the baseline algorithms and $CBQA$ in terms of execution time

Approach	Execution time	
	Setup	Query
Baseline_1	0	$N * (t_{req} + t_{retr})$
Baseline_2	0	$N * (t_{req} + t_{retr}) + N * t_{eval}$[a]
CBQA_L	0	$N * (t_{req} + t_{retr}) + N * t_{eval}$
CBQA_{LS}	$N * (r_{req} + t_{retr})$	$N * t_{eval}$
CBQA_D	0	$N * t_{req} + t_{eval} + N * t_{retr}$

[a]Evaluation time considers only checking identity among contexts and matching properties.

B. Experimental evaluation

The present subsection is devoted to numerically assess the efficiency and effectiveness of the multi-resolution approach. Given the goal of the subsection, experiments focus on the abstract `EVAL` function in Algorithm 3, since results are independent from the specific computation and storage scenario at hand and are valid for all $CBQA$ variants. For simplicity reasons, experiments do not take into account the match between query and device properties (which, by the way, is a step shared by $CBQA$ and baselines).

First, we introduce the dataset, the procedure to generate arbitrary synthetic knowledge bases with the desired characteristics, and the experimental setting. Then we show results, discussing how the answering capability of $CBQA$ varies depending on the size of the network, the knowledge base and the rewriting mechanism. We will adopt the following measures for assessing answering capability:

- *Average device recall*. It is defined by $\frac{\sum_{q \in Q} |\Delta_{O_q}|}{|Q|}$, where $|\Delta_{O_q}|$ is the number of devices returned by $CBQA$ in response to a query q, and Q is the set of possible queries;
- *Query answer probability*. It is defined by $\frac{|\{q : |\Delta_{O_q}| > 0\}|}{|Q|}$ i.e. the fraction of queries for which $CBQA$ returns at least one device.

We also provide the execution time of a query on a device.

The code is available at the GitHub project page[2] and is released under the GNU General Public Licence v.3, while the dataset is under the Open Data Commons Attribution (ODC-BY) License. Experiments were performed on a cluster of 10 servers, each running Linux Ubuntu 18.04 LTS 64 bit, with 4x2198MHz CPU and 8 GB RAM.

Dataset: We refer to a dataset of 10000 devices, each with a single sensor. Each device belongs to one or more IoT networks, the largest of which includes 2000 devices. Given the unavailability of public data of IoT networks, in

[2]https://github.com/KDMG/MIOT/tree/master/ContextBasedQueryAnswering

our experiments we built a dataset by following the ideas expressed in [44], according to which one of the main factors used to build links in IoT is the proximity of nodes. Therefore, we started from a real-world repository available at http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html, which regards radio taxi routes in the city of Porto from July 1st 2013 to June 30th 2014. Each route contains a sequence of PoIs (Point of Interests) corresponding to GPS coordinates of a vehicle. As done in [45], we mapped each route to a smart device and we used information about proximity (computed by using PoIs inside available routes) to establish connections among different devices. Indeed, under the assumption that the proximity of nodes is the main factor to create links among smart objects to build an IoT [44], we leveraged the paths followed by radio taxis to simulate smart object movements and, hence, to estimate their chance of being in proximity one with the other. The idea of using radio taxi routes makes sense in our scenario as among the possible typologies of smart objects we consider also smart cars, and, in any case, the most used smart objects typically belong to the personal area networks of people. Because people often move in a city by taxi, using a repository of taxi routes can be a good indicator of the possible movements of smart objects.

Knowledge base: The knowledge base contains the instance of the context model, i.e. the set of possible values for each dimension of a context. Such dimensions are defined as hierarchies[3] automatically initialized by varying two parameters for each dimension, namely: *(i)* the number of levels (*lev*), and *(ii)* the branching factor (*br*). The former determines the depth of the tree and the latter its width. To make an example, with number of levels=3 and branching factor=3, the dimension hierarchy will include 1 root element at level 1, 3 elements at level 2 and 9 elements at level 3 (i.e. 13 elements in total). In the general case, the size of a dimension $|D_i|$, in terms of number of elements, is given by $|D_i| = \sum_{k=0}^{lev-1} br^k$.

The overall number of possible contexts $|C|$, when we consider n dimensions, is given by all the possible combinations of elements for each dimension, namely $|C| = |D_1| \times \ldots \times |D_n|$. In the case of three dimensions of the same size, $|C| = (\sum_{k=0}^{lev-1} br^k)^3$. In the above example, $|C| = 13^3 = 2197$.

Experimental settings are as follows:

- *Knowledge base initialization*: a specific number of levels and branching factor is defined, ranging from 3 to 5 for both parameters.
- *Device initialization*: a device is picked from the dataset, and the set $\mathcal{N}$ of all the devices belonging to its network is extracted. In our tests, the following procedure is repeated with 20 devices belonging to the sets $\mathcal{N}_1, \ldots, \mathcal{N}_{20}$, whose sizes range from N_1 =100 to N_{20}=2000:
  - For each device $\Delta_i \in \mathcal{N}_j$, a context c is randomly assigned to Δ_i. The assignment is done according to a strategy following a power law. Specifically, we randomly assign a context instance from a small subset of C (i.e., 20% its size) to the large majority of devices (i.e., 80% in our tests), which hereafter we name C_s. In this way, we aim at modeling the most plausible context similarity of devices belonging to the same network. The procedure assigns a random context from the remaining set $C - C_s$ (corresponding to the remaining 80% of C) to the rest.
  - A query $q_c = \langle x, Z \rangle$ is defined by assigning a context x from C_s (as discussed, since Z is not the focus of the experiments, it is set to void).
  - The query q is launched on the network.

The procedure is repeated for a number of queries equal to $|C|$, in order to perform a comprehensive evaluation. Results are then averaged over all queries. Experiments are focused on three different scenarios, according to the relative sizes of the knowledge base and of the network, namely:

- $|C_s| \approx N$: the number of contexts is smaller than or comparable to the network size (from 0.22x up to a 4.4x); here, we consider 3 dimensions, 3 levels and a branching factor equal to 3, for a total of 2.197 different contexts (and, consequently, possible queries) and $|C_s| = 439$. In this scenario the probability that two devices share the same context is not negligible.
- $|C_s| > N$: the number of contexts is greater than the network size (from 3x up to 60x); in this case, we consider 3 dimensions, 3 levels and a branching factor 5, for a total of different contexts and queries equal to 29.791. $|C_s| = 5.958$. In this scenario the probability that two devices share the same context is low.
- $|C_s| \gg N$: the number of contexts is much greater than the network size (from 177x to 3500x). In this case, we assume 3 dimensions, 5 levels and a branching factor equal to 3, for a total of different contexts and queries equal to 1.771.561, with $|C_s| = 354.312$. In this scenario the probability that two devices share the same context is negligible.

Finally, for each scenario we also evaluated t_{eval}, i.e. the execution time of a query on a device. For this test, we set up a commodity hardware (2x2198MhZ, 2 GB RAM) to simulate the computation capabilities of an IoT device. Given a knowledge base, a query for each context has been launched and execution times have been averaged.

Results: experimental results in the three scenarios are shown in Figure 5. For each network size, query answering probability and average device recall of $CBQA$ are reported in the top and bottom figures respectively. In the figures, the id line reports the performance when only devices with contexts identical to that of the query are considered. It also corresponds to $Baseline_2$ performance. The ρ line is obtained by considering devices with both identical and finer contexts. This corresponds to the first step of the algorithm. Finally, the cpt line adds the contribution given by compatible devices, thus providing the overall performance of $CBQA$ with no rewriting. The overall contribution given by one or more (up to five) rewritings is also reported. When a rewriting does not provide further contribution, the corresponding line is omitted.

As a general observation, we can note very low answering

[3]This choice does not limit the generality of experiments, since a lattice-based dimension can always be transformed into a hierarchy-based, and allows us to consider the worst-case scenario in terms of dimension of the search space for given experimental parameters.

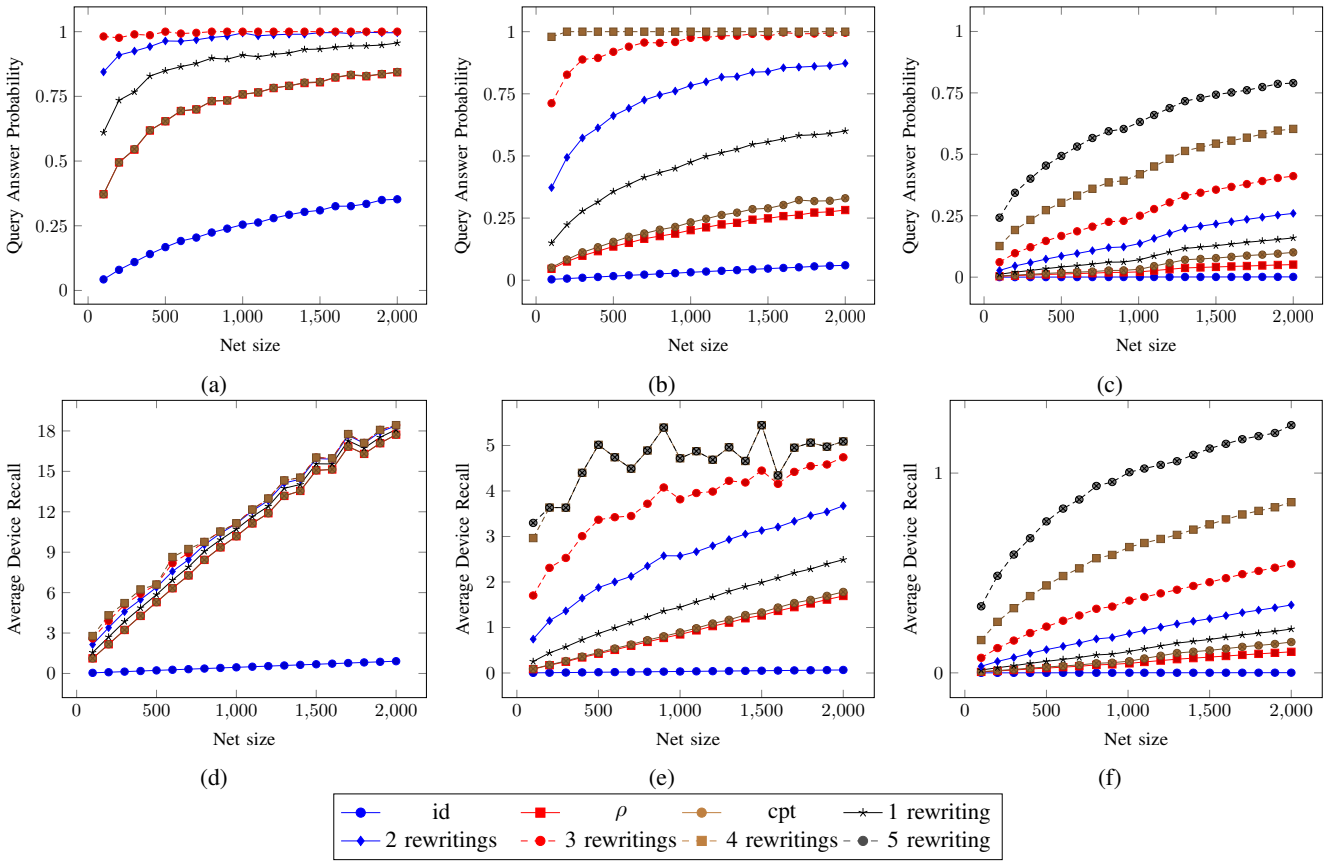

Fig. 5: Query answer probability and average device recall with 3 dimensions: 3 levels and branching factor 3 (a) and (d); 3 levels and branching factor 5 (b) and (e); 5 levels and branching factor 3 (c) and (f)

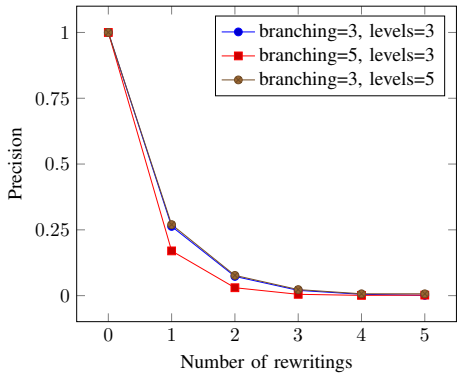

Fig. 6: Evaluation of precision against the number of query rewritings

capability, especially device recall, in each of the three scenarios when only identical contexts are considered. This result is expected, since the fraction of queries that can be answered in this way is equal to the number of contexts assigned to devices, that is a tiny portion of the set of potential queries. This enlightens the limit of approaches based on a flat definition of contexts, like $Baseline_2$. In all cases, we can appreciate the gain provided by $CBQA$ over $Baseline_2$, while it seems that the advantage of considering compatible contexts is limited. In detail, for what concerns the three scenarios:

$|C_s| \approx N$: as shown in Figure 5d, a good recall is obtained even without rewriting, since with a small number of contexts the likelihood to find devices whose contexts roll-up to a common context increases. Also note that rewritings have a bigger impact on query answering probability than on recall. In fact, even a single rewriting increases the query answer probability by at least 40%, almost doubling the score in the case of smaller networks (see Figure 5a). After 3 rewritings, more than 95% of queries are answered.The execution of a query on a device takes 0.15 ms on average.

$|C_s| > N$: figures 5b and 5e show that with the increase of the number of contexts, both query answering probability and average device recall are low without rewriting. For what concerns device recall, the contribution of each rewriting is greater than in the previous scenario, even for small networks. When query answering probability is considered, gains ranging from 82% to 195% are obtained by one rewriting, whereas at least 98% of queries are answered after 4 rewritings. The execution time of a query is 2.21 ms on average.

$|C_s| \gg N$: as shown in Figure 5f, in this scenario, which is also the most typical one, only in very few cases devices with an identical, finer or compatible context can be retrieved, although the contribution of ρ is the most relevant. It should be noted that in this scenario the maximum number of possible rewritings is 12, so less than a half of the possible rewritings already allows to reach satisfactory query answering proba-

bility (79%) and at least one device retrieved, on average, for largest networks. In all cases the advantage of rewriting is striking: the minimum gain in terms of query answering probability obtained on all networks by using rewritings ranges from 58%, in the case of 1 rewriting, to 683% in the case of 5 rewritings. Similar trends also occur when recall is considered. The execution of a query takes 290.65 ms on average.

We can also observe that:

Network size: the increase in the number of devices in a network has an obvious positive impact, on average, on results. In particular, in every test, the larger the size, the greater the percentage of queries with a response and the average number of devices retrieved.

Size of knowledge base: the number of dimensions, the number of levels and the branching factor determine the overall number of contexts that can be defined. Although we can assume that, in real settings, most contexts are specified by values at the bottom levels (e.g., at street or square level, rather than at city or region levels), the larger the knowledge base, the larger the space of possible contexts. This implies that: *(i)* the execution of the algorithm will take longer, and *(ii)* a smaller number of devices can be retrieved with identical, finer or compatible contexts. In other terms, the size of the network and of the knowledge base have an inverse effect on the answering capability.

Number of query rewritings: by rewriting a query q to q', the likelihood of obtaining an answer increases in all cases, as for the number of retrieved devices. Indeed, the number of contexts finer or compatible with that of q' is obviously larger than in the case of q. The size of the knowledge base (and, specifically, the number of levels) clearly has an impact on the gain. In fact, with smaller knowledge bases, a single rewriting is enough to significantly increase the chance to find at least one solution (less steps are needed to reach the root context).

The increase of answer capability comes at the price of a reduced answer precision. As already noted, evaluating answer precision is not simple, as it depends both on the similarity of the retrieved device's context with respect to the query context, and on the variability of the requested measures over context dimensions. Let's limit the analysis to a simpler notion of context precision, defined as the ratio between the number of "correct" contexts (i.e. either identical or finer than the query context) and the number of contexts considered when rewriting is applied. Figure 6 shows that in our experimental setting the decrease of context precision follows a geometric decay with respect to the number of rewritings, whose slope depends on the branching factor. Since in general lower levels of dimension hierarchies typically include more members than higher levels, this suggests that precision drops faster for queries asking for very fine-grained contexts. However, even in this case, the precision of a measure whose values are almost constant on the set of sibling members is not significantly affected, like e.g. in the case of time or cloud coverage with respect to position.

Finally, as to the comparison with $Baseline_1$, the latter always contacts all the $\mathcal{N}$ devices corresponding to running $CBQA$ with the context set to *all* for each dimension. Hence it provides in principle the same query answering capability and context precision. However, recall that with $Baseline_1$ there is no way to assess the precision of results in practice, since contexts are not actually taken into account. Conversely, it is possible to conclude that our approach allows the retrieval of a much larger number of relevant results (i.e., finer or compatible) than $Baseline_2$ as it evaluates identical contexts only. As a consequence, by exploiting the knowledge base, and specifically roll-up relations, a much higher recall can be obtained with $CBQA$ with the same precision. Furthermore, as already noted, by exploiting roll-up relation and the notion of rewriting, our approach can optionally decide to lower precision in order to improve query answering capability.

VI. Conclusions

In this paper we have presented a model for devices' contexts whose main feature is the representation of context components at different levels of granularity. On top of this model, we have introduced multi-resolution querying mechanisms to retrieve a set of devices according to given preferences and context requirements, that can be flexibly tightened or relaxed depending on the desired performances and the given application goal. Results enlighten the advantages of the approach with respect to existing solutions that either do not manage contexts at different granularity levels, or do not consider the notion of context at all, adding reasonable overheads for computational, storage and network costs.

The pioneering approach proposed in the paper required to make some simplifying assumptions, opening up a number of interesting research directions. First, we need to explore the intriguing issue of the relation between measure precision and context precision. A preliminary idea is to extend the model by associating to each measure a set of functions that describe the variability of the measure with respect to each dimension. Then, we will have to develop a suitable approach to exploit this information in $CBQA$.

Second, a limitation of this study is that here we assume that objects keep track of all the new contacts that can be established according to criteria based on *proximity*, *device homogeneity*, and relations among the corresponding *owners*. Of course, in many interesting applications, the list of contacts of an object must be limited, to both control network traffic and cope with the limited resource capabilities of smart objects. In this paper, we trivially assume that only timely contacts are preserved and that old and inactive ones can be safely discharged. Nevertheless, a more refined strategy for filtering new contacts to guide the construction and evolution of the IoT network appears necessary, as well as to identify unreliable sources. Therefore, future work will be devoted to defining criteria for network building and maintenance.

Finally, different IoT scenarios like the Multi-IoT, and specific applications domains, like Smart Building Networks or Internet of Vehicles, should be studied to tailor the approach and understand the specific pros and cons.

Acknowledgments

We thank the Editor-in-Chief and the reviewers for their comments, which allowed us to significantly improve the

quality of the paper. Experiments have been performed on an Iaas platform provided by the H2020 project Helix Nebula Science Cloud.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] G. Baldassarre, P. Lo Giudice, L. Musarella, and D. Ursino, "The MIoT paradigm: Main features and an "ad-hoc" crawler," *Future Generation Computer Systems*, vol. 92, pp. 29–42, 2019.

[3] P. Lo Giudice, A. Nocera, D. Ursino, and L. Virgili, "Building Topic-Driven Virtual IoTs in a Multiple IoTs Scenario," *Sensors*, vol. 19, no. 13, p. 2956, 2019, mDPI.

[4] D. Ursino and L. Virgili, "Humanizing IoT: defining the profile and the reliability of a thing in a Multi-IoT scenario," *Towards Social Internet of Things: Enabling Technologies, Architectures and Applications. Studies in Computational Intelligence*, vol. 846, pp. 51–76, 2020, springer Nature.

[5] N. K. Tran, Q. Z. Sheng, M. A. Babar, L. Yao, W. E. Zhang, and S. Dustdar, "Internet of things search engine," *Commun. ACM*, vol. 62, no. 7, pp. 66–73, Jun. 2019.

[6] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The Social Internet of Things (SIoT)–when social networks meet the Internet of Things: Concept, architecture and network characterization," *Computer networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[7] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a social structure to the Internet of Things," *IEEE Communications Letters*, vol. 15, no. 11, pp. 1193–1195, 2011, iEEE.

[8] C. Diamantini, A. Nocera, D. Potena, E. Storti, and D. Ursino, "Find the Right Peers: Building and Querying Multi-IoT Networks Based on Contexts," in *Proc. 13th International Conference on Flexible Query Answering Systems (FQAS'19)*, Amantea, Italy, 2019, lecture Notes in Artificial Intelligence. Springer.

[9] C. Diamantini, A. Nocera, D. Potena, E. Storti, and D. Ursino, "Multi-dimensional contexts for querying IoT networks," in *Proc. 27th Italian Symposium on Advanced Database Systems, CEUR Workshop Proceedings Volume 2400*, Grosseto, Italy, 2019.

[10] S. Mayer and D. Guinard, "An extensible discovery service for smart things," in *Proceedings of the Second International Workshop on Web of Things*. ACM, 2011, p. 7.

[11] H. Wang, C. C. Tan, and Q. Li, "Snoogle: A search engine for pervasive environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 8, pp. 1188–1202, 2009.

[12] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, 2014, iEEE.

[13] M. Mrissa, L. Médini, and J. Jamont, "Semantic discovery and invocation of functionalities for the web of things," in *2014 IEEE 23rd International WETICE Conference*. IEEE, 2014, pp. 281–286.

[14] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *1994 First Workshop on Mobile Computing Systems and Applications*, Dec 1994, pp. 85–90.

[15] D. Abowd, A. K. Dey, R. Orr, and J. Brotherton, "Context-awareness in wearable and ubiquitous computing," *Virtual Reality*, vol. 3, no. 3, pp. 200–211, Sep 1998.

[16] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *IEEE Network*, vol. 8, pp. 22–32, 1994.

[17] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, Jan. 2001.

[18] O. Cabrera, X. Franch, and J. Marco, "Ontology-based context modeling in service-oriented computing: A systematic mapping," *Data & Knowledge Engineering*, vol. 110, pp. 24–53, 2017.

[19] S. Ahn and D. Kim, "Proactive context-aware sensor networks," in *Proceedings of the Third European Conference on Wireless Sensor Networks*, ser. EWSN'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 38–53.

[20] S. Baek, E. Choi, J. Huh, and K. Park, "Sensor information management mechanism for context-aware service in ubiquitous home," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, pp. 1393–1400, 2007.

[21] C. Hou, H. Hsiao, C. King, and C. Lu, "Context discovery in sensor networks," in *ITRE 2005. 3rd International Conference on Information Technology: Research and Education, 2005.*, June 2005.

[22] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran, "Composite capability/preference profiles (cc/pp): Structure and vocabularies, ver. 1.0," W3C, Tech. Rep., January 2004.

[23] A. Agostini, C. Bettini, and D. Riboni, "Hybrid reasoning in the care middleware for context awareness," *International Journal of Web Engineering and Technology*, vol. 5, no. 1, pp. 3–23, 2009.

[24] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *Knowledge Engineering Review*, vol. 18, no. 3, pp. 197–207, 2003.

[25] H. Chen, T. Finin, and A. Joshi, "Soupa: standard ontology for ubiquitous and pervasive applications," in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004*, 2004.

[26] T. Strang, C. Linnhoff-Popien, and K. Frank, "Cool: A context ontology language to enable contextual interoperability," in *Distributed Applications and Interoperable Systems*, J. Stefani, I. Demeure, and D. Hagimont, Eds. Springer Berlin Heidelberg, 2003, pp. 236–247.

[27] C. Perera and A. V. Vasilakos, "A knowledge-based resource discovery for internet of things," *Knowledge-Based Systems*, vol. 109, pp. 122 – 136, 2016.

[28] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 161–180, 2010.

[29] P. Bhargava, S. Krishnamoorthy, and A. Agrawala, "Rocomo: A generic ontology for context modeling, representation and reasoning in a context-aware middleware," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. ACM, 2012, pp. 584–585.

[30] A. Padovitz, S. W. Loke, and A. Zaslavsky, "Towards a theory of context spaces," in *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004.*, 2004, pp. 38–42.

[31] I. Millard, D. De Roure, and N. Shadbolt, "The use of ontologies in contextually aware environments," in *Proceedings of First International Workshop on Advanced Context Modelling, Reasoning and Management*, 2004, pp. 42–47.

[32] F. A. Schreiber, L. Tanca, R. Camplani, and D. Viganò, "Pushing context-awareness down to the core: more flexibility for the perla language," in *Electronic Proc. PersDB 2012 Workshop (Co-located with VLDB 2012)*, 2012, pp. 1–6.

[33] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen, "Smart-its friends: A technique for users to easily establish connections between smart artefacts," in *international conference on Ubiquitous Computing*. Springer, 2001, pp. 116–122.

[34] J. Bleecker, "A manifesto for networked objects—cohabiting with pigeons, arphids and aibos in the Internet of Things," *http://research.techkwondo.com/blog/julian/185*, 2006.

[35] M. Kranz, L. Roalter, and F. Michahelles, "Things that twitter: social networks and the Internet of Things," in *What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing (Pervasive 2010)*, 2010, pp. 1–10.

[36] E. Nazzi and T. Sokoler, "Walky for embodied microblogging: sharing mundane activities through augmented everyday objects," in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 2011, pp. 563–568.

[37] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable web of things," in *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 2010, pp. 702–707.

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

[39] H. Ning and Z. Wang, "Future Internet of Things architecture: like mankind neural system or social organization framework?" *IEEE Communications Letters*, vol. 15, no. 4, pp. 461–463, 2011.

[40] Q. Du, H. Song, and X. Zhu, "Social-feature enabled communications among devices toward the smart iot community," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 130–137, 2019.

[41] Y. Saleem, N. Crespi, M. H. Rehmani, R. Copeland, D. Hussein, and E. Bertin, "Exploitation of social iot for recommendation services," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 359–364.

[42] R. Torlone, "Two approaches to the integration of heterogeneous data warehouses," *Distrib. Parallel Databases*, vol. 23, no. 1, pp. 69–97, Feb. 2008.

[43] M. Banek, B. Vrdoljak, A. M. Tjoa, and Z. Skocir, "Automated integration of heterogeneous data warehouse schemas," *International Journal of Data Warehousing and Mining*, vol. 4, no. 4, pp. 1–21, 2008.
[44] I. Guedalia, J. Guedalia, R. P. Chandhok, and S. Glickfield, "Methods to discover, configure, and leverage relationships in internet of things (iot) networks," 2018, uS Patent 9,900,171.
[45] S. Nicolazzo, A. Nocera, D. Ursino, and L. Virgili, "A privacy-preserving approach to prevent feature disclosure in an IoT scenario," *Future Generation Computer Systems*, vol. 105, pp. 502–519, 2020.