A Novel Indexing Method for Scalable IoT Source Lookup

Seyed Amir Hoseinitabatabaei, *Member, IEEE,* Yasmin Fathy, *Member, IEEE,*
Payam Barnaghi, *Senior Member, IEEE,* Chonggang Wang, *Senior Member, IEEE,*
and Rahim Tafazolli, *Senior Member, IEEE*

Abstract—When dealing with a large number of devices, the existing indexing solutions for the discovery of IoT sources often fall short to provide an adequate scalability. This is due to the high computational complexity and communication overhead that is required to create and maintain the indices of the IoT sources particularly when their attributes are dynamic. This paper presents a novel approach for indexing distributed IoT sources and paves the way to design a data discovery service to search and gain access to their data. The proposed method creates concise references to IoT sources by using Gaussian Mixture Models (GMM). Furthermore, a summary update mechanism is introduced to tackle the change of sources availability and mitigate the overhead of updating the indices frequently. The proposed approach is benchmarked against a standard centralized indexing and discovery solution. The results show that the proposed solution reduces the communication overhead required for indexing by three orders of magnitude while depending on IoT network architecture it may slightly increase the discovery time.

Index Terms—Internet of Things, Source Indexing, Probabilistic Model, Gaussian Mixture Model, Distributed Discovery.

1 Introduction

How to find a *thing* in the Internet of Things (IoT) haystack? This is a critical question that IoT users and developers are facing and will face in the future [1]. The existing solution to the source discovery in IoT is very similar to the web search engines, where the attributes of each item (web pages in web domain) are crawled to fill the index repository (ies). User queries that are received from a search engine interface are evaluated against the indices and sources associated with the closest index matches are fetched for data (the standard architecture for IoT source discovery is elaborated in section 1.1). Examples of such search platforms for IoT includes GSN [2], LSM [3], and Microsoft SensorMap [4].

As elaborated in [5] the index (address) provided for IoT resource should at least cover (the) location and functionality (type) of the device. Using this information, a variety of approaches such as IPv6 [5], hashing methods [6], Object Name Service ONS [7] are adapted to make a unique index, for each IoT item.

Indexing method has a significant impact on the performance of the search engine. In recent years, a variety of approaches for indexing the IoT resources have been introduced with various objectives such as reducing the complexity of the search mechanism, e.g. "Hexastore" [8], addressing a variety of query types e.g. [9], and enabling to consider user priorities e.g. [10]. However, only a small number of existing solutions have taken the impact of the indexing process on the scalability of the search mechanism into their design considerations.

Such solutions are beneficial when indexed attributes are static. Otherwise, when for example billions of IoT devices are moving around, i.e. changing one of their indexed attributes, new indices are to be continuously generated, imposing a significant load to the address generator nodes and the rest of the network only to maintain the vital indices. Experts in IoT domain has recently identified this issue as a significant threat to the applicability of the existing IoT source discovery methods [6], [11] and [12]. Reports on the performance of real-world platforms such as [3] have also emphasized on the congestion and bottleneck problems as a result of index update issues. Recent advances in providing more effective indexing by clustering the indexed attributes e.g. [13] and [14] have been able to reduce the size of indices significantly. However, such approaches either make strong assumptions about the distribution of the indexed attributes or impose excessive computation burden to retrain the clusters when frequent updates are required.

In this paper, we take an initial step towards mitigating the communication overhead of maintaining the indices and introduce a Distributed Scalable Indexing System A.K.A DSIS. In short, we make the following contributions.

We propose to replace the conventional indices with a set of new indices that are based on the mathematical representation of the distribution of the attributes of the IoT sources. We discuss that such approach significantly reduces the communication overhead for maintaining the indices while providing enough information to accomplish the search process. We further elaborate on how such indices can be generated using Gaussian Mixture Models. Required steps for constructing and maintaining the indexing parameters from the "location" and "type" attributes of resources are

- *S.A. Hoseinitabatabaei, Y.Fathy, P.Barnaghi, and R.Tafazolli are with the Department of Electrical Engineering, University of Surrey, Guildford, Surrey,United Kingdom.*
  E-mail: a.taba, y.fathy, p.barnaghi, r.tafazolli @surrey.ac.uk
- *Chonggang Wang is with Interdigital corp. E-mail: Chonggang.Wang@InterDigital.com*

detailed and further information for extending the approach to indexing multiple attributes are provided.

To mitigate the computation cost for the update of the new index parameters, we introduce a novel approach for incremental updating of the constructed indexing parameters namely, Variation Compensation Vectors. A detailed description of the update mechanism algorithm is provided.

The proposed indexing and the update mechanisms are evaluated through extensive simulations, and the results are presented. Also, the impact of the indices overlaps on the performance of the search process is investigated and discussed.

Finally, we provide an example of the implementation of the proposed solution in the distributed networks. A novel source lookup mechanism enhanced with our indexing method is deployed in a hierarchical IoT network. Accuracy, indexing efficiency and search complexity of the proposed solution are investigated through several simulations and benchmarked against a conventional centralized approach. Results are presented and discussed.

The remainder of this paper is structured as follows. Section 2 provides a summary of the related works and the state of the art solutions. Section 3 introduces the fundamental concepts behind the proposed approach including attribute summarization for indexing and updating mechanisms. Section 4 details the underlying mathematical formulation. Simulations setup is presented in section 5 and the initial evaluation results are reported in Section 6 where accuracy and sensitivity of the proposed solution (are) investigated. Section 7 provides an example for the implementation of DSIS in a hierarchical network and introduces a hierarchical search mechanism. Finally, concluding remarks and future works are provided in Section 8.

1.1 IoT source discovery

The architecture of the IoT systems and Machine-to-Machine (M2M) networks are discussed in several existing works including the European Telecommunications Standards Institute (ETSI) M2M architecture [15]. We have adopted the network architecture of the ETSI M2M in our design. Fig. 1 shows a general architecture for IoT source lookup indexing and the key entities that are typically sensing and collection, source lookup indexing, and query. In sensing and collection process, the published data and services from IoT sources (e.g. device and sensors) are collected. Each source is connected to a gateway (GTW) or an Information Repository (IR). The indexing of the data attributes of IoT sources is then constructed and stored at Discovery Servers (DSs), while the actual data are stored at GTWs. Source lookup indexing is built on top of DSs where each DS is responsible for selecting the most appropriate gateways that have answers for user/application queries. Queries are composed of type, location and other attributes [16].

For the sake of simplification of the presentation, we limit our discussions to exact queries. For example, Get temperature (i.e. type) in (-25.382708 and -49.265506) (i.e. location). In this example, the lookup mechanism identifies the GTW in which the queried Thermometer is registered and publishes its measurements.

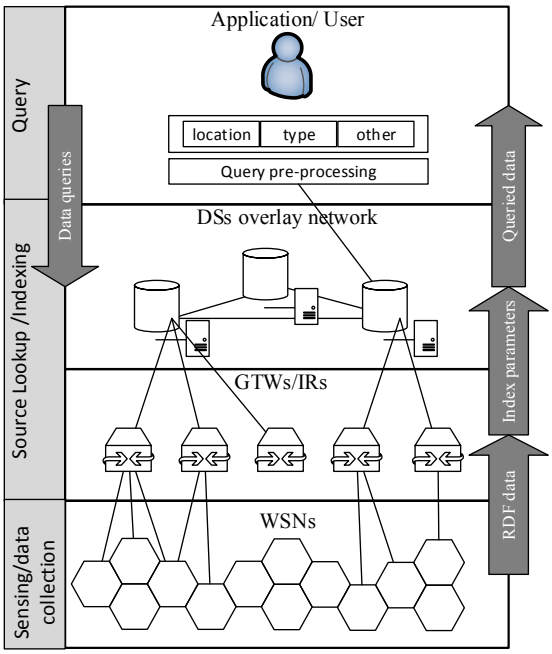

Fig. 1. Simulation results for the network.

2 Related Works

While the indices must provide sufficient information for DSs to address the queries, they should be generated in a way that allows for a dynamic update with a minimum computation overhead despite the number of IoT sources. Furthermore, the traffic load associated with the communication of indices between gateways and DSs and even between DSs should not scale with the number of sources. The existing indexing mechanisms typically fail to suffice these requirements. In what follows we review some examples of the indexing solutions for IoT source discovery.

Evdokimov in [17] assumes that IoT sources are represented by a numerical identification i.e. Object ID (OID). Information about the sources is stored in distributed Information Services (ISs) which are similar to the IR shown in Fig. 1. Discovery servers (DSs) process user queries for a specific source (i.e. OID) and provides a link to the Information Service (IS) which is expected to have the requested information. A centralized architecture is utilized for DSs. Other works have adopted the same concept for publishing the indices to the DSs, but instead of a centralized DS they use distributed networks of DSs to have a more scalable query processing. For example, The Bridge project utilizes Lightweight Directory Access Protocol (LDAP) [18] and the studies in [19] and [20] considers a peer to peer (P2P) architecture.

The main shortcoming of these solutions lies in the fact that the computation burden of generating the indices and the communication load for propagating the indices from DSs to ISs scales with the number of sources attributes that are to be indexed, making them prone to overflow, when a large number of indices are generated at the same time.

A recent study in [21] has proposed a new distributed service discovery mechanism for IoT sources which expands the preceding architectures by introducing a mechanism to support a flexible identification scheme using multidimen-

sional attributes and range queries. The multidimensional attributes are first mapped into a one-dimensional domain, and then they are indexed based on a Prefix Hash Table (PHT) structure [22]. However, still, data and the generated indices are associated in a one-to-one fashion which, as discussed earlier, may limit the scalability of the system. Also, by mapping the data attributes of sources to one dimension, the search mechanism has to look for data within a wider range of unique item, making the attribute matching required for the query processing more challenging.

Linked Stream Middleware (LSM) [3] focuses on search and discovery of the sensors and actuators. LSM provides a framework for providing a semantic description (i.e. RDF descriptions) for the sensors and actuators data and allows for SPARQL-like queries across both resources and the harvested data. The sensory data in LSM is annotated and transformed into RDF triples. The triples are then stored in a repository which is capable of executing SPARQL queries. The main shortcoming of the LSM framework is the lack of scalability due to the centralized architecture. The query execution time is shown to increase drastically with the increase in the number of provided triples. Moreover, triple storages are not suitable for intensive applications and insertion of many new data into the triple storage creates a bottleneck for the system.

Another approach is to store sources information in an Extensible Markup Language (XML) file and select a specific XML field to construct indices [23]. However, the time for building indices is affected by the increasing number of connected devices which makes this approach not scalable.

Among few studies that have taken the initial steps for reducing the impact of indexing process on the scalability of the search mechanism, [13] presents an approximation technique for query processing in WSNs. It is a tree-based structure in which a Gaussian model of sensor's data is stored at each child node, and an aggregated Gaussian model of children nodes is stored at their parents to facilitate traversing the tree to find an approximated answer for user queries. However, the work assumes that the sensor's data has a Gaussian distribution which is not the case in all IoT environments and applications and models are to be retrained for updating the models. The later imposes excessive computation burden when frequent updates are required.

Chu [14] propose a multi-indexing approach for services provided by IoT sources in which functionality description, spatial and temporal attributes are used to answer user range queries. The work initially starts by clustering a group of similar services into the same cluster based on their functional similarities for their descriptions. The main drawback of this approach is that the computation time to cluster services provided by IoT sources scales with their number. It also does not provide any effective updating mechanism when either a new service connects or an existing one disconnects. As a result, the clusters are to be rebuilt every time that the attributes of the available services changes.

In this work, we propose a novel approach for the Distributed Source Indexing, namely DSIS. DSIS replaces the standard indices that were generated in one-to-one association with the sources with a new set of indices that form the one-to-many association, hence improving on the state of the art approaches for reducing communication overhead, while allowing for accurate[1] processing of queries. Different from [13] and [14] our solution benefits from a novel index update mechanism that mitigates the computation and communication overhead for maintaining the indices. Also, unlike [13], DSI does not require any prior assumption on the attributes.

3 METHODOLOGY

DSIS is comprised of two mechanisms: first is a probabilistic indexing that exploits conditional probabilities to reduce the size of indices and second is an updating process to keep the indices up-to-date with minimum overhead. In what follows we explain our rationale for developing these mechanisms.

3.1 Probabilistic Indexing

As was mentioned earlier, DSs process the queries to find the GTW in which the queried data resides. We assume that there is an IoT network comprises of a set of DSs (see Fig. 1) such that N_D refers to the number of DSs in that network.

Let's assume that A is the ensemble of all possible data attributes fields where A=$\{a_1,...a_n\}$. Each element of A represents an attribute field which can take any value from its value set a_i=$\{ai_1,...ai_n\}$. A_D^n is a member of an ensemble of attribute fields that are available at the n^{th} DS (DSn) in the network. IoT source is represented by a finite number of values for different attribute fields e.g. s=$\{a1_1,...,an-1_{n-1},an_n\}$.

Query processing at a given DS e.g. DSn involves an index matching process where each attribute value in the query is compared with the possible counterparts at A_D^n.

Matching each attribute reduces the search space for the remaining attributes to a subset of values that have been associated with the matched attribute value. The average complexity of processing a query at n^{th} DS can be estimated by the joint entropy of the attribute values of that query in A_D^n. The average number of binary search processes that are required to resolve a query based on our example source index s can be calculated as follows.

$$H(s) = -\sum_k p(a1_k)log(p(a1_k))... \\ -\sum_j p(an_j|a1_1..an-1_3)log(p(an_j|a1_1..an-1_3)) \quad (1)$$

H is the entropy and represents the average number of binary matches that are required to resolve the query. $p(ai_k)$ is the likelihood of ai_k over all the entries for attribute ai at A_D^n. The summations in (1) are reminiscent of the reduction of the searching space when the queried attributes indices are sequentially matched with their counterparts at A_D^n.

Equation 1 implies that conditional probabilities are the key parameters in resolving the query. Inspired by this observation, we propose a new approach to replace direct index matching process by indirect evaluation of query attributes on mathematical models of conditional probabilities

1. We refer to the accuracy in the handling of queries as the correctness of identified gateway/data repository for queried items given the constructed GMM models

that are trained over the data attribute space. By doing so, the number of indices that are to be retained across the IoT network is reduced to a few model parameters.

According to (1), if all possible combinations of the conditional probabilities are in hand the optimal order in which the attributes of the query are to be matched to achieve the fastest convergence can be calculated. However, calculating probabilities for all possible combinations is computationally costly. To mitigate that, we (order) the attributes and then sequentially calculate the conditional probabilities between each pair of these attributes. This process is hereby referred to as *attribute summarization*.

The order in which the attributes are modeled has a direct impact on the efficiency of the summary process. Ideally, attributes should be selected in a way that the number of model parameters on the overall attribute values is minimized. In section 4, we introduce a heuristic approach to find an appropriate order of attributes.

Replacing matching process with estimations from probability functions may cause false positive errors. For example, the probability distribution function may not exactly fit the actual distribution of the data attributes and does not fall to zero at the missing attribute values. In fact, the accuracy of the query processing depends on the closeness of fit of the models to the real probability distribution of the data attributes. We circumvent this problem by using Gaussian Mixture Models (GMM) and assigning a threshold for minimum acceptable probability values.

GMM is a parametric probability density function which is represented as the weighted sum of Gaussian component densities. Herein, we assume that readers are familiar with the basics of GMM, an introduction to GMM readers is provided in [24]. As a generative learning algorithm, GMM assumes a probabilistic pattern, dependent on certain parameters, between data and classes and through the learning process specifies a joint distribution over data and recognized classes.

Reasons for selecting GMM for probability density estimations are two folds; the first reason is that GMM can approximate any probability distribution with a reasonable accuracy provided that the number of Gaussian components is sufficiently large, and the parameters of the model are chosen correctly [25]. The second reason is that the GMM can be implemented as a complete unsupervised technique which is crucial for autonomous processing and discovery of large-scale distributed IoT data. It should be noted that GMM is just an example of various parametric probability density estimation models that can be used in our proposed approach for probabilistic index lookup and any other models that satisfy the above mentioned criteria can be implemented and used in the same manner as we describe for GMM. Comparison of the performance of different probability density estimation models remain for future works.

The model parameters are trained at GTWs, a GTW may train multiple GMM for different data attributes. GMM parameters are then forwarded to DSs across the network. DSs maintain an account for each of the registered GTW and its associated GMMs. Also, model parameters that are received from various GTWs at a DS can be aggregated to create a generic probabilistic model that represents the distribution of overall indices at that DS. Such generic models can be shared with other DSs in a distributed overlay network. When a DS receives a query, the GMMs are employed to attain a probabilistic estimation of the presence of the queried data at each GTW. DS may opt to forward the query to GTW/s for which the GMM model/s have given the highest probability value or pass it to other DSs. False positive errors are controlled by defining a threshold for minimum allowable probability values. Indeed routing the queries between DSs is dependent on the architecture of the overlay network. In Section 7, we provide an example of implementation of our technique in a hierarchical network of DSs.

Locating the GTW of a queried source by following the highest probability estimation is only possible when the models are not overlapped. Meaning that the estimated probability from the GMM that is provided by the correct GTW is always higher than estimations from other GMMs of other GTWs. However, depending on the distribution of the attributes of the sources, the models may often be found to be overlapping in real-world scenarios. In section 6 we investigate the impact of the overlaps on an accuracy of identifying the GTWs and DSs and in section, 7.2 we show how a search mechanism can cope with this problem.

3.2 Index Update Method

Due to the dynamicity of the attributes of the IoT resources, the model parameters are to be updated from time to time. Indeed repeating the learning process is computationally expensive and may delay the query processing. In this regard, we have envisioned two types of update methods with different computation costs and frequencies. The first is a lightweight process that is triggered in relatively short intervals depending on the dynamicity of the attributes of the data. In this process, the available model parameters that describe the distribution of the conditional probabilities are considered to be more or less a valid representation of the overall data and will be only adapted to the new variations that have happened since the last interval. In section 4, we will introduce Variation Compensation Vectors (VCVs) which convey the required information for adaptation of the models from GTWs to DSs. The second type of update process is triggered at longer intervals; when the proportion of the changed attributes is comparable with the primary attributes that were used to construct the initial model. Therefore, the existing model parameters are no longer sufficient for representing the data. In this case, a new distribution model is obtained.

Fig. 1 summarizes the proposed data flow layers across the network. Each arrow shows a particular type of data flow between network nodes. GTWs construct the GMM models and propagate the model parameters to the DSs overlay network. Updates to the models are provided from the GTWs in the form of VCVs. The upper layer shows the exchange of the queries and model parameters between DSs. User queries are received at DSs and forwarded from DSs to the selected GTWs.

4 Approach

This section elaborates our approach for the formulation of the probabilistic indexing approach and the updating

mechanism.

4.1 Attribute Summarization

The primary objective of the attribute summarization is to minimize the number of model parameters that are required to represent the presence and dependency of data attributes. In what follows we present a heuristic approach for estimating an appropriate order of the attributes.

Attribute summarization is an iterative process. Through each iteration, initially, two attributes are selected as primary and secondary attributes. Then the distribution of a subset of values of the primary attribute which are associated with the first value of the secondary attribute is estimated using GMM. If the size of the subset is not enough for distribution estimation, the exact attribute values are used instead. The process continues to cover all the values within the secondary attributes. The next iteration starts with replacing the secondary attribute with the primary attribute and adding a new attribute that has not been analyzed before as the secondary attribute. The summarization process terminates when all the data attributes are covered.

To find the ideal order, at the first step the ratio between the average number of values from primary attribute, which are associated with a distinct value for the secondary attribute is to be calculated. This measure provides an estimation of the number of GMMs that are to be created for each combination of primary and secondary attributes. This ratio can be calculated for of all the possible combinations of primary and secondary attributes and based on that the order of attributes which minimizes the total number of GMMs can be identified. However, to reduce the computation cost, and yet estimate an appropriate order of attributes, once the ratios are calculated, the attribute combinations are sorted by the highest ratio to the lowest one. We ensure that the attributes with a greater number of distinct values are modeled at early iterations resulting in significant improvements of the of model parameter to data attributes. The estimated distributions (along with the remaining attribute values) are linked to each other to form a consistent representation of the attribute space. It should be noted that our approach, may not converge in an optimal order. The optimal order can be estimated using Viterbi algorithm or Exhaustive search which tend to add more computation burden.

The number of distinct values for categorical attributes are typically limited. Therefore, our approach tends to push them towards the upper layers of the attribute summarization process and represents them with discrete values instead of GMM.Through the evaluation of the DSIS that is presented in Section 6 and 7, we have represented each resource by type and location attributes. Type attributes have less diversity in comparison with the location attributes and are to be placed on top of the hierarchy for attribute selection. In this manner, each GTW maintains some GMM models each of which is trained over the location attributes of a particular type of resource. In our simulations, a GTW can take up to 20 different types of sensors resulting in up to 20 different GMM models.

The next step is to develop the GMM models. We have summarized the parameters that are used for the equations in this section in Table 1. GMM estimations are defined by three parameters: φ is the mixing proportion of the components, E is the expected value of the attributes and σ is the covariance matrix of attributes. Given that the number of required components is estimated by one of the aforementioned techniques e.g. [24], the GMM calculates the model parameters through an Expectation Maximization process. It can be shown that the EM algorithm monotonically improves the likelihood of the model for describing the data distribution. In our problem, we aim to estimate $p(ax|\varphi, E, \sigma, ay_p)$ which describes the distribution of a data attribute field i.e. ax given certain values of another attribute i.e. ay_p and GMM parameters. By adopting the conventional EM algorithm (i.e. [24]) GMM parameters are trained over data. Detailed formulation of the training phases of GMMs is provided in Appendix A.

TABLE 1
Summary of Parameters

Parameter	Description
φ	mixing proportion of the GMM components
E	the expected value of the attributes
σ	the covariance matrix of attributes
C	the total number of mixture components
ax_l	the vector of attribute values for the l^{th} data entry
w_{lk}	a posterior probability of k^{th} in a Gaussian mixture component (Mc) given a_{xl}
φ_k	a mixing proportion of the k^{th} Gaussian mixture component (Mc)
E_k	the estimated mean of the k^{th} Gaussian mixture component (Mc)
σ_k	the estimated covariance of the k^{th} component
$\grave{\varphi}_k^j$	the new mixing proportion of the k^{th} Gaussian component of the j^{th} GMM model
L_j	the total number of samples that are used for constructing the j^{th} GMM model
$\tilde{E}$	the expected value of the attributes
$\tilde{E^2}$	the sum of the squared attribute values that will be used to estimate the corrected covariance matrix of attributes
$\tilde{L}$	the number of modified data points

Following the above attribute selection process, GMM models are trained over the attributes of data items at the gateways. Next, the parameters of the models alongside with the number of total data samples that are calculated during the training phase (i.e. σ, E, φ, L) are forwarded to the DS that is assigned to each gateway. A DS may receive several GMM models from different gateways. The models are aggregated in the DS to form more generic GMM models, representing the entire data items that are referenced at that DS.

The aggregated model comprises the Gaussian components of the initial models, provided by gateways, but with different mixing proportions. The new mixture ratios are calculated as follows.

$$\grave{\varphi}_k^j = \frac{L_j \varphi_k^j}{\sum_h L_h} \tag{2}$$

Where $\grave{\varphi}_k^j$ is the new mixing proportion of the k^{th} Gaussian component of the j^{th} GMM model and L_j is the total number of samples that are used for constructing the j^{th} GMM model. Before commencing the aggregation process, DSs should ensure that the models are homogeneous. This

is achieved by preserving the order of the selected attributes in learning the conditional probabilities.

The order in which attributes are used during the learning process is not maintained across GTWs. In the case of the model discrepancy, DSs can utilize the GTWs' model parameters to re-sample the data and train new models before aggregating them. Re-sampling should be performed in a way that preserves the statistical properties of the original model. In essence error margins defined by central limit theorem can be used to assess the accuracy of re-sampling.

These models can be shared between DSs in the network. Depending on the network architecture models may be aggregated several times to construct a global model for the distribution of the data. Through the query processing, the estimated probability of the queried data based on the aggregated models would be the key parameter to find the DS which is associated with the gateway that hosts the queried data. An example of the data discovery in distributed networks is presented in section 7.

Another remaining challenge is that the model parameters may quickly become obsolete due to the variation of the data attributes, e.g. when the resources (i.e. sensor nodes) are highly mobile, and their location attributes changes frequently or the scenarios that several resources join and leave the network at a rapid pace. Thereby, there is a need for an efficient mechanism that updates the parameters across the network and meanwhile imposes a minimum computation and communication overhead. The update process is detailed in the following section.

It is worth noting that variables that are not directly presented in a metric space such as textual variables will go through a pre-processing step to attain a metric representation, before training GMMs. Pre-processing is a standard process in text mining, particularly when textual data have to be clustered with distance-based clustering algorithms. We refer the interested readers to the survey in [26] that describes some conventional algorithms which are used for distance-based clustering of text documents.

4.2 Index Updating Mechanism

Variations of the attributes over time are updated in the probabilistic references through the summary updating process. At the GTWs, the summary updating mechanism periodically produces new models over the newly modified data and sends the calculated parameters to DSs. Subsequently, DSs update their models with these new parameters. We refer to this type of update process as "Complete-Update" and its period is represented by T. Due to the volume of the data, complete-update may impose a considerable computation cost and cannot be frequently executed. On the other hand, performing updating process in long intervals is also not desired as it degrades the performance of the query processing due to the variations that take place within the update intervals. To resolve this problem and to achieve more accurate references during the update intervals Temporary-Updates are introduced. We represent the update period of the temporary updates with t and assume that $t < T$. Temporary updates take advantage of a novel parameter adaptation mechanism namely, the Variation Compensation.

Different from the complete ones, temporary updates are not iterative and are only performed over the portion of the data that is modified. In this regard, temporary-update imposes a less computational burden. Throughout the temporary-update, the numbers of Gaussian components are assumed to be fixed, and the existing model parameters are still a close approximation of the correct model. Based on these assumptions, the model parameters are adapted to the recent changes of the data attributes. The adaptation process is based on combining the parameters of the recently obsoleted models with compensating parameters that are calculated from the data items that are changed after the last update (complete-update or temporary-update). These parameters are referred to as Variation Compensation Vectors or VCVs. The modified model after each temporary-update is used as a base for the next updates. The temporary-update can start after a complete update and be repeated several times until the next complete update. The existing model is then made obsolete, and a new model is provided by the complete-update.

The process for calculating the VCVs depends on how the stored data items have changed. Scenarios for the alternation of the attributes may include: addition of new resources in WSNs, removal of resources, and variation of the attribute values of the existing resources e.g. change of the location attributes due to the movement of the device. In what follows, we elaborate our approach for calculating the VCVs.

Our solution for adapting the model parameters is performed in two steps. The first step is to calculate VCVs as sufficient statistics of the variations. This step is similar to the E-step of GMM training. We denote the parameters that are calculated at this step with "$\sim$". This step is always performed at the GTWs. The second step combines the VCVs that were derived from the first step with the parameters of the original model to create an adapted model. We denote the parameters that are calculated at this step with "$\wedge$". The second step is taken at the DSs.

The main difference between DSIS and the well known Maximum A Posteriori (MAP) [27] method lies in the second step. MAP tends to combine the parameters of the original model and the one from the training set with an emphasis on the characteristics of the new training data items. Whereas, DSIS strives to utilize the original and new parameters (that are derived at the first step) to generate an estimation of a GMM as if it was trained on the entire updated dataset. It should be noted that the original model parameters (i.e. $\sigma,\ E,\ \varphi$ are now providing suboptimal results due to the modified portion of the data.

In the scenario of adding new resources, the formulation of the first step would be as follows:

$$\tilde{\omega}_k^l = Pr(Mc = k|ax_l; \varphi, E, \sigma) \tag{3}$$

$$\tilde{\varphi}_k = \frac{1}{\tilde{L}}(\sum_{i=1}^{\tilde{L}} \tilde{\omega}_k^i) \tag{4}$$

$$\tilde{E}_k = \frac{\sum_{l=1}^{\tilde{L}} \tilde{\omega}_k^l ax_l}{\tilde{L}\varphi_k} \tag{5}$$

$$\tilde{E}_k^2 = \frac{\sum_{l=1}^{\tilde{L}} \tilde{\omega}_k^l ax_l {ax_l}^t}{\tilde{L}\varphi_k} \tag{6}$$

For the modified portion of the data, $\tilde{\omega}$ is a posterior probability of Gaussian mixture components (Mc) for a given attribute vector ax_l. $\tilde{\varphi}$ is the mixing proportion of the components, $\tilde{E}$ is the expected value of the attributes and $\tilde{E^2}$ is the sum of the squared attribute values that will be used to estimate the new covariance matrix of attributes. Finally, $\tilde{L}$ is the number of modified data points.

Once the VCVs are calculated, gateways update these parameters at DSs. The adopted parameters are then calculated by combining the previous model parameters and updated VCVs as follows.

$$\hat{\varphi}_k = \frac{L * \varphi_k + \tilde{L} * \tilde{\varphi_k}}{L + \tilde{L}} \tag{7}$$

$$\hat{E}_k = \frac{L * \varphi_k * E_k + \tilde{L} * \tilde{\varphi_k} * \tilde{E_k}}{L * \varphi_k + \tilde{L} * \tilde{\varphi}_k} \tag{8}$$

$$\hat{\sigma}_k = \frac{(\sigma_k * \sigma_k^t + E_k * E_k^t) * L * \varphi_k + \tilde{E}_k^2 * \tilde{L} * \tilde{\varphi}_k}{L * \varphi_k + \tilde{L} * \tilde{\varphi_k}} - \hat{E_k} * \hat{E_k^t} \tag{9}$$

A closer look at Equations 7 - 9 indicates that the estimated parameters are reminiscent of Maximization step of the GMM training. Here instead of original dataset (with L data items), the updated dataset (i.e. L+$\tilde{\text{L}}$ data elements) is used. The M-step of the EM algorithm maximizes the likelihood of the posterior probabilities. Given that the posterior probability is provided from the original model the second step of the update process calculates the model parameters in a way that maximizes the likelihood function of the model.

It should be noted that the original model parameters that were calculated in a similar manner (see Appendix A for Equations 15 - 17) are now providing suboptimal results due to the modified portion of the data.

Fig.2 provides an illustrative example of the update process. Fig.2.a resembles the initial model that is trained in a GTW. The model has two Gaussian mixing components. Fig.2.b shows the situation where some new data are added, and the model is updated based on the temporal update mechanism. As described in Section 4, the number of components is preserved in temporary updates. Fig.2.c demonstrates the model after a complete update which is a new model trained on the entire dataset. Different from the original model, the new model takes advantage of five Gaussian components to estimate the probability distribution of the data. As is demonstrated in Fig.2, the likelihood of the models improves after each step of the updating process. In case of removal (i.e. when the resources leave and their data become unavailable), after calculating the sufficient statistic form (3-6) the updated parameters are calculated as follows.

$$\hat{\varphi_c} = \frac{(L * \varphi_k - \tilde{L} * \tilde{\varphi_k})}{L - \tilde{L}} \tag{10}$$

$$\hat{E_c} = \frac{L * \varphi_k * E_k - \tilde{L} * \tilde{\varphi_k} * \tilde{E_k}}{L * \varphi_k - \tilde{L} * \tilde{\varphi}_k} \tag{11}$$

$$\hat{\sigma_c} = \frac{(\sigma_k * \sigma_k^t + E_k * E_k^t) * L * \varphi_k + \tilde{E}_k^2 * \tilde{L} * \tilde{\varphi}_k}{L * \varphi_k - \tilde{L} * \tilde{\varphi_k}} - \hat{E_k} * \hat{E_k^t} \tag{12}$$

Where $\hat{\varphi}$, $\hat{E}$ and $\hat{\sigma}$ are the updated components mixing proportion, expected value and covariance matrix. In the case of the variation of the attribute values of existing resources, first the old attributes are removed according to (3-6) and (10-12) and then the new attributes are added based on (3-9). Pseudo codes of both attribute summarization and index updating mechanisms are provided in Appendix E.

4.3 Complexity of Attribute Summarization and Index Updating Mechanisms

Appendix A details the formulation of Expectation-Maximization (EM) process required for training GMMs as summary of the attributes and elaborates on GMM training complexity.

As for the DSIS attribute summarization, the total time complexity of training GMMs on the given set of attributes O($A * (I * K * \text{L} * D^2 + I * K * D^3)$), where A is the number of attributes, K is the number of mixture components, L is the number of data points,D is the dimension of attributes and I is the number of iterations of EM algorithm. Index updating mechanism described in 4.2 is not iterative, therefore the computation complexity, including calculating VCVs and adaptation of models with VCVs, is in the order of O(A*(K*$\tilde{L}$ *D^2+K*D^3)). $\tilde{L}$ is the number of modified data points.

4.4 Privacy and Security Remarks

DSIS relies on the veracity of the information that is exchanged between IoT devices, GTWS and DSs and at the same time interoperability of these nodes in the network. Solid security measures should be considered to protect the data communications from various security threats in real-world applications. Below we describe some of the security issues.

The communications among the nodes are in clear text and therefore are not secure. No embedded mechanism is defined for nodes (incl. GTWs, IRs and DSs) to authenticate each other before accepting/rejecting any messages from upper/lower/same tier nodes. It is not yet considered how a malicious node which impersonates a valid one can be detected and compartmentalized. Edge of the network needs to consider protective mechanisms against DoS/DDoS (Distributed Denial-of-service attack) whereby an attacker manages to target the DSs with an abundant number of fake or valid queries in an attempt to disrupt the whole system through resource overloading. Using NIDS (Intrusion Detection Systems) e.g. [28] in the key points of the architecture (i.e. gateways and the DSs) can help to resolve this issue.

DSIS indices are created based on IoT resource attributes. Attributes that contain private information should be excluded or sufficiently encrypted and anonymized before be sent to the GTWs.

In [29] and [30], an overview of the existing works on securing IoT networks and the issue of trust in exploiting IoT resources is discussed. In-depth analysis of DSIS security and privacy implications remains for future works. Appendix D provides a set of recommendations on resolving node failure problem.

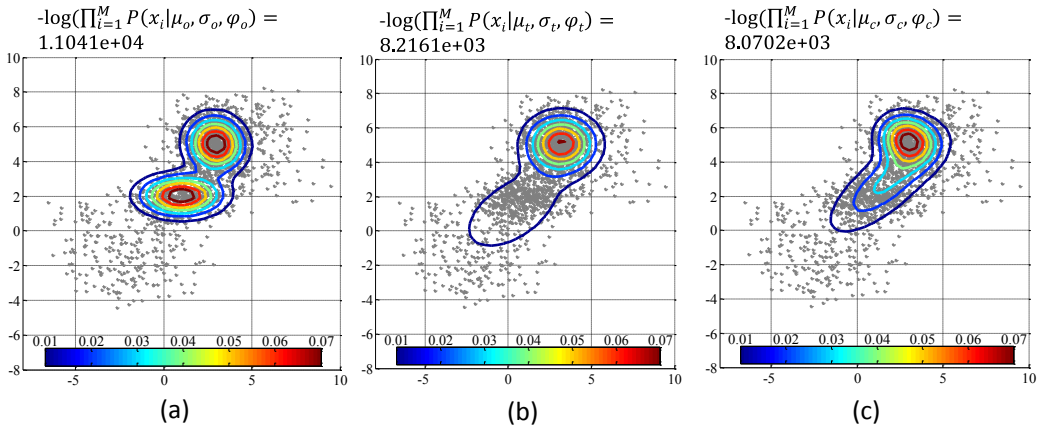

Fig. 2. Demonstration of the summary updating mechanism, in this figure number of data points (M), is 1000. (a) Original model with two Gaussian Mixture components, μ_o, σ_o and φ_o are GMM components of the original model data attributes. Once the original is trained, data points are randomly displaced therefore the original model does not give an accurate representation of the new data distribution; (b) The original model is adopted according to the proposed compensation mechanism (temporary update), μ_t, σ_t and φ_t are GMM components of the model after the temporary update; (c) a new GMM is trained over the modified data (complete update); the new model comprises five components, μ_c, σ_c and φ_c are GMM components of the model after the complete update.

5 Simulation Setup

Two sets of simulations are performed to evaluate the performance of the DSIS solution. The first round of simulations are presented in section 6 and aim to study the impact of the GMMs overlap. Model overlap is expected to be the primary reason for false positive errors. Herein, we investigate how the overlap between the models influence both the accuracy of locating the queried data points based on the probability estimations from GMMs, and the effectiveness of the update process. The impact of the other critical parameters including the network architecture and scalability characteristics are studied in Section 7.

Similar to the recommendation for essential attributes of IOT data in [5], "type" and "location" are utilized as the primary source attributes to construct the GMM models in our simulations. However, the presented analysis applies to any other type of attributes. Queries are presumed to include the type and location attributes of the requested source. Through this work, we assume queries are pre-processed and mapped into a set of valid data attributes using the state of the art techniques such as [31]. User queries are received at DSs across the network.Upon reception of a user query, DSs uses DSIS indices to find the GTW/IR where the specified source stores its measurements.

The evaluations are performed on a simulated network of DSs, GTWs, and WSNs. Connections between these nodes are formed based on the structure presented in Fig. 1. The network configuration in section 6 is agnostic to the network architecture and could be part of a large scale hierarchical structure or a peer-to-peer distributed network of DSs. In Section 7 we elaborate on the extension of this network to a large-scale hierarchical network.

6 Evaluation of Overlap Impact

In this set of simulations, we have three DSs each are connected to three gateways each of which receiving data from three WSNs. Each WSN is populated with 1000 sensors randomly chosen from 20 different types (i.e. overall 18000 sensors). DSs are connected in a peer-to-peer fashion. One of the DSs serves as a mediator (i.e. DSm), which receives the queries (e.g. from the rest of the network) and passes it to the two other DSs.

Through each Monte-Carlo run, sensors are randomly generated within a given radius (i.e. rG = 25 nmi) of the associated gateway. Location and type of the sensors are registered at the gateways and are later used to generate the GMMs. Gateways are randomly positioned within a given radius (i.e. r_D) of the DSs. The relative distance of DSs varies through the simulations and is represented by d Each DS stores the GMMs that are received from its allocated GTWs. Afterward, the DSs aggregate the models according to the procedure that was explained in Section 4.1 and pass the resulted models to the DSm, where these models are used to distribute the user queries between the DSs.

To study the impact of GMMs' overlap on the performance of the proposed solution, we have changed the values for r_D and d while r_G was fixed. It is worth mentioning here that our simulation study utilizes the geographical area as a metric to assess the overlap (for the location data. Similarly, if GMMs were trained on a different attribute, the relevant metric of the space that is used for training GMMs should be considered to assess the level of overlap used.

6.1 DS and GTW Resolution Using DSIS

In this set of simulations, we strive to locate the queried data points using DSIS indices. Herein, each sensor is queried from the DSm. As mentioned earlier queries are forwarded from DSm to the other DSs and subsequently to gateways based on the likelihood of retaining the queried data. Simulations are repeated ten times for each pair of r_D and d values and the results are averaged.

Fig. 3 demonstrates the success rate of DS recognition process with respect to different r_D and d values and the results when the errors from DS and GTW identification steps are combined. The difference between corresponding

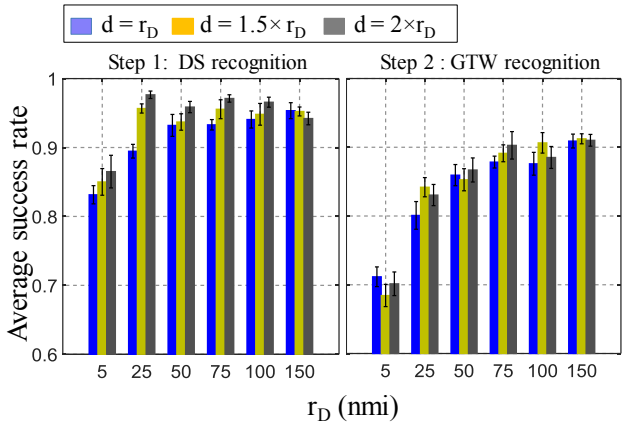

Fig. 3. DS and GTW recognition accuracy versus different r_D and d values

bars on the left and right parts of the figure represents the GTW identification error.

Increasing r_D reduces the chance of overlap between GTWs GMMs within the same DS. On the other hand, Increasing d reduces the chance of overlap between DSs GM models, which should improve the rate of correct DS identification. Results presented in Fig.3 are aligned with these expectations. However, the magnitude of improvement is dependent on the separation distance of the DSs.

The DS identification process shows a sudden jump at the initial values of the r_D and fluctuates around the same level for the rest of the values. In case of the DS identification, the lowest accuracy has been 83.2% which was obtained when r_D = 5nmi and d = 5nmi and the highest value has been 95.3% at r_D = 150nmi and d = 150nmi. When GTW recognition error is added, results show a clear sensitivity to the variation of the r_D and improves almost monolithically with higher values of r_D. Herein, the identification accuracy starts with 68% and when r_D = 5nmi and d = 7.5nmi and reaches to at 91.2% when r_D = 150nmi and d = 225nmi.

The results indicate that using the aggregated models at DSm, the correct DSs can be resolved accurately. Particularly, when the distance between DSs is equal or greater than GTWs radius. In the case of GTWs, correct identification of the GTWs hinges on the density of the DSs and higher success rates can be achieved at lower densities of GTWs within DSs.

The overlap between the model has a clear impact on the accuracy of the DS and GTWs identification. Excessive model overlap of the GMM models at r_D= 5nmi when compared with moderate overlap situation at r_D= 50nmishows over 10% reduction in the accuracy of the DSs and GTWs recognition.

Comparing the results at DS and GTW level implies that at DS level resolutions has been performed with a higher accuracy and has been less prone to the overlapping impact. Query processing methods could benefit from such an improvement in accuracy of the identification process that occurs when aggregated GMMs are employed. In section 7, we introduce an example of such query processing methods.

Through the evaluations, we observed that increasing the size of the area for which the GMMs are trained reduces the sensitivity of the models to individual data points. This in return, results in confusion of the identification process. When r_D increases the aggregated GMM at DSs must cover a broader area which results in slight degradation of the DS identification success rates at the highest value of r_D (r_D = 150 nmi). As an example solution to moderate this effect, in Section 7 we propose a method that replaces the data point query with a range query of appropriate size before the estimation of the likelihood values. To summarize, the proposed solution has been shown to be capable of recognizing the correct DSs and GTWs accurately, provided that the overlaps between generated models are controlled.

6.2 Update Mechanism

The update process is evaluated as follows. First, some sensors with random types and locations are added to each WSNs. The newly added sensors are then queried from DSm at the following phases: a) before any update, b) after the temporary-update and c) after the complete-update. Similar to the previous subsection simulations are repeated ten times for each pair of r_D and d values, and the results are averaged. Herein, we present the results when 250 sensors are added which is equivalent to 25% of the WSNs population. Evaluation of other proportions including 10% and 50% of the WSNs population have shown similar trends, and their results are not presented for redundancy reasons. Overall 4400 sensors are added to the network at each round of simulations. The corresponding results are shown in Fig.4.

It is worth reminding the reader that after phase (c) new GMMs are trained over the entire data set therefore the results of this phase show the maximum achievable performance using the proposed solution. In this manner, we utilize the results after phase (c), to benchmark the query processing performance after phase (a) and (b).

Compared to the results after phase (c), using the obsoleted models i.e. phase (a) increases the average error of DS recognition by 5%, which is equivalent to ~ 205 extra miss detections. GTW recognition has on average added 150 extra misses.

A comparison between the results of query processing after phase (a) and (c) indicates that the proposed indexing solution is intrinsically resilient to the sensors variation effects. That is down to the fact that GMM parameters represent the statistical properties of the whole sensors and their variation with respect to a relatively small number of updated attributes data are not significant. The degradation in the performance of the query processing of phase (a) becomes more evident when the r_D and d are close to their highest values and the distribution of the nodes within the network is sparse. Such situations have been effectively recovered when the temporary-update process is applied.

A cross comparison of the results from all phases confirms the usefulness of the update methods in improving the success rate. Results after phase (b) shows a noticeable reduction in miss detection error. Where the average increase in DS recognition error reduces to 1.1%, which is equivalent to only 46 extra miss detections and when GTW recognition is added the average additional miss detections reaches to 102 sensors. The latter observation implies that

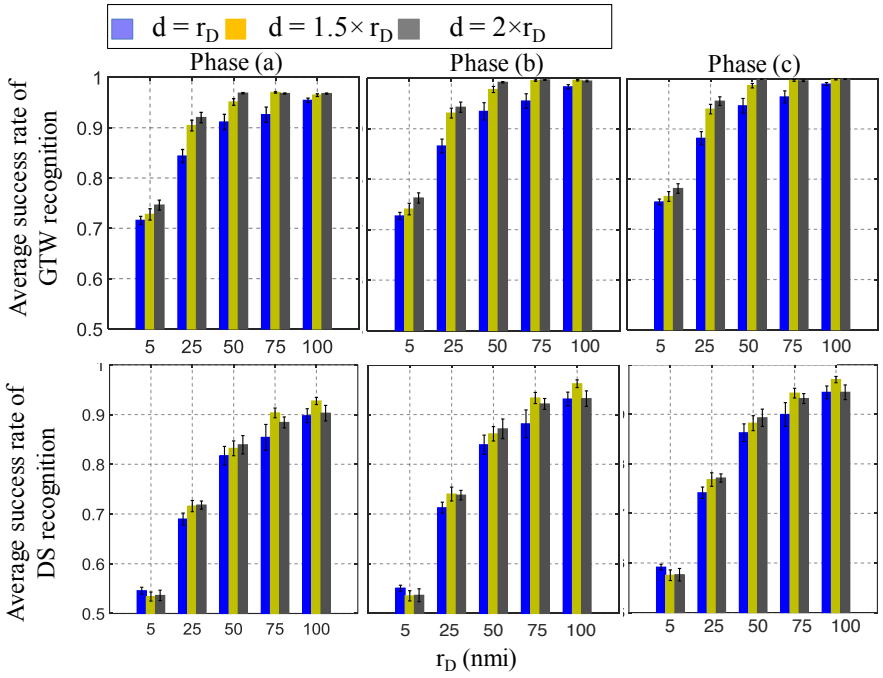

Fig. 4. Update mechanism evaluation when 4400 sensors are added to the network.Phase(a): before updating the references.Phase(b): when references are updated by the temporary-update method. Phase(c): when references are updated by complete-update method.

the temporary-update method has been more efficient in improving the recognition of DSs rather than GTWs.

Another interesting phenomenon is that the temporary-update effect is dependent on the r_D values. Fig.4 shows that the highest improvement is obtained where r_d 25 nmi and is almost equal to success rate that is achievable after the complete-update.However, different values of d do not have a significant impact on the improvement of the GTW and DS identification accuracy after temporary-updates. Complimentary evaluations on update process are provided in Section 7.

The proposed method was shown to cope with variations of data attributes, but the level of resiliency depends on r_D and d. The degradation in query processing success rate as a result of using old models has been effectively compensated after applying the temporary-update mechanism. Our evaluations confirm that the temporary-update could be an adequate substitute for the complete-update when a small margin of error is acceptable.

7 Distributed Data Discovery

In this section, we aim to evaluate the scalability characteristics of our technique. In doing so, first, we elaborate on the integration of our method in a hierarchical network as an example of distributed networks. We introduce a query processing method that is accustomed to the characteristics of DSIS. Scalability properties are studied based on the impact of node densities at different layers of the network on the query processing performance and the number of parameters that are required to maintain the indices across the network. Later on, we evaluate the performance of the enhanced hierarchical data discovery and benchmark it against a centralized approach from the state of the art techniques. It should be noted that our proof of concept implementation, does not cover the detailed network management procedures.

7.1 Distributed Network

Our example of distributed network follows a hierarchical architecture. Fig.5 represents a schematic diagram of our proposed hierarchical structure with three layers. Nevertheless, our proposed approach is flexible regarding the number of layers in the hierarchy.

DSIS in our hierarchical architecture is implemented as follows. DSs are arranged in a hierarchical overlay network with three layers as shown in Fig.5. One DS rests at the top (i.e. DS Level 3 or DSl_3) and is communicating with a particular number of DSs at the lower level (i.e. DS Level 2 and 1 denoted as DSl_2 and DSl_1). Each DS at the 2nd layer is connected to some DSs at the 1st tier. DSs at the first layer are connected to GTWs which receives sensory information from WSNs. WSNs provide GTWs with attributes of the available resources. Each GTW maintains a SQL database which stores the attributes of WSNs sources. GTWs create GMM models over the attribute space and forward the resulted model parameters to their designated DSs. DSs at each level aggregates the received models and pass the aggregated model parameters to the next upper layer. Queries are routed from DSs to GTWs that might have responses for the queries according to our proposed searching mechanism that is elaborated in the next section. The identified GTW runs an SPARQL query to verify if the IoT source that is specified by the query is registered at the identified GTW or not. If the query is successful, the user is provided with a link to the data source and if not the search mechanism is notified to continue the search in a different GTW.

7.2 Search Mechanism

Our search mechanism takes advantage of the DSIS likelihood measures to forward the queries from DSs to the desired GTWs. DSs utilize two types of models to route the queries across DSs overlay network. The first model is the DS (aggregated) GMM model. The second model is the set of GMM models from lower layer nodes. A related point to consider is that the second model may also include the aggregated models of other DSs from lower layers. Queries are forwarded from DSs overlay network to GTWs following the most probable path (i.e. the node that shows the highest likelihood for the requested data source). However, identifying the most likely node is not as straightforward as it may seem. Below, we remark on four major challenges and explain how they are tackled in this study.

First, according to our overlay network architecture, each DS can only estimate the presence of the queried attributes within its underlying nodes. However, forwarding queries requires a global understanding of the probability distributions across the network. Our solution is to start the process from the top node of the hierarchy. According to the example architecture provided in Fig.5, this is equivalent to starting the query processing at the 3rd layer. The black arrows that are shown in Fig.5 illustrate an example of the paths that the query may traverse to find a GTW that has the highest likelihood of containing the requested data source.

Second, as more and more GMM are aggregated at higher layers in the network, the aggregated models become less sensitive to the probability of the individual data items. As shown in 2 the likelihood of individual data items in the aggregated models, continually reduces in proportion to the number of data items that are referenced. This in return may mislead the route identification mechanism at the higher layers.

To resolve the issue, we expand the point queries to range queries at higher layers. More specifically, some data items are extrapolated from the queried data point and added to the query, before initiating the probability estimation at each node. The extrapolated items are normally distributed around the queried data item, and their quantity is in proportion to the number of referenced data at that node. A natural selection for a standard deviation of the extrapolation is the expected value of the standard deviation of the Gaussian components that are generated at the GMM model of each node. As the query traverses the network towards the lower layers, models are becoming more sensitive to individual data items, and in this regard, the extrapolation radius should be reduced. Therefore a division factor is introduced that is multiplied by the standard deviation of the extrapolation and it varies with the layer number.

Our experimental evaluations resulted in selecting the following values: the division factor is set to 1 for layer two which makes the standard deviation of extrapolation equal to the expected value of the standard deviation of the models. In layer 1, the deviation factor is set to 10, and at the GTWs, it is set to 100.

It is worth mentioning that range queries at each step add a minor computation burden for calculating the cumulative probability of the range which is in proportion to the number of extrapolated data points.

Third, although most of the queries are expected to be answered by following the most probable nodes, there will be still cases in which the queried data are not found in the first identified GTW (or first attempt). This problem arises when the GMM models have overlaps, or the requested data does not exist. If the *initial attempt* is not successful, the DS of layer one, which has forwarded the query, tries other underlying GTWs in the order of their probability estimation of the queried data.

If the new attempts are not successful, the query is processed again from the upper layer. Other DSs are queried according to their probability estimation for the requested source. Search for the desired resource (e.g. to find a GTW that contains the inquired data) continues to another layer in a similar manner and terminates by finding the source or exceeding the maximum limit of the hops allowed. The blue arrows in Fig.5 resemble the additional searches after the failure of the first attempt to respond a query. The second attempt would be to search the other GTWs associated with the same DS. In the case of subsequent failures, the 4th attempt will try to query the most suitable GTWs of second most probable DS (which are again determined based on their probability estimations).

Forth, identifying the most likely path based on evaluating the next hop requires a fundamental assumption that the likelihood values monolithically increase from higher layers to lower layers. A more generic solution could be to use the Viterbi algorithm to find the optimal path. Given a sequence of probabilities in different nodes at different layers, the Viterbi algorithm offers an iterative solution to identify the most probable sequence of nodes with worse case complexity of $O(S^2T)$. Where S is the number of nodes and T is the length of the sequence (i.e. fixed to four in our evaluations). However, our assessment shows that our simple approach is sufficient to identify the GTWs in our simulation environment accurately.

7.3 Simulation Settings

A portion of the earth surface with an area of approximately 4800 km^2 is taken for generating the simulated network. DSs of level 2 and 3 are presumed to be independent of any physical location, DSl_1s are allocated with a random location within the simulation area.GTWs are distributed uniformly around the location of DSl_1s. The maximum allowable dispersion of the GTW's locations varies between 50 km to 150 km. Also, the sink nodes (top nodes) in WSNs follow a uniform random distribution in the proximity of their designated GTW. Maximum dispersion of the sink nodes varies between 10 km to 50 km. The distribution of the sensors locations is also uniform centered by their designated WSN sink nodes. The maximum dispersion that is allowed in this case is a random value between 1 to 10 km.

Through our MATLAB simulations, at the start of each Monte-Carlo run, random numbers of sensors, WSNs, GTWs and DSs are generated and linked based on the network architecture. The number of DSs and GTWs may vary depending on the simulation scenarios. Table 2 summarizes the initial state of the simulated environment. The component capacity is the maximum number of nodes from

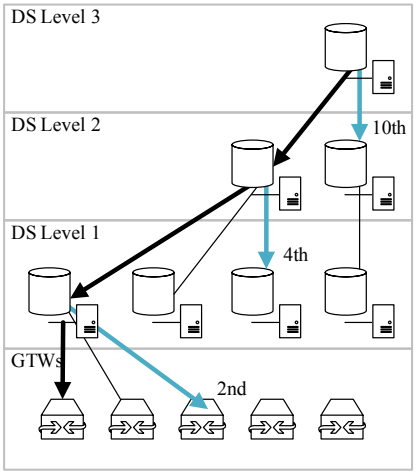

Fig. 5. The proposed hierarchical overlay network structure, thick black arrows shows the most probable path and blue arrows are other subsequent attempts

TABLE 2
Specifications of initial conditions of the simulations

Component	WSN	GTWs	DSl_1	DSl_2	DSl_3
Capacity (initial specifications)	1000 Sensors	1-5 WSNs	1-5 GWs	1-5 DSl_1s	4.0 DSl_2s

the lower layer that is connected to the specified component.

7.4 Evaluation Metrics

We use different metrics to evaluate the performance of the proposed solution. The first metric is the Query Success Rate (QSR) that shows the percentage of the queries that are resolved after a given number of search attempts. Ideally, most of the queries should be responded in the first attempt based on following the highest probability estimations throughout the network. It also contributes to understanding the overhead of the query processing.

The Indexing Efficiency (IE) is another metric that is defined as the ratio between the total number of GMM components that are generated from all IoT source attributes to the total number of actual sources that are referenced at the lowest level of the network. This metric denotes the efficiency of using our modeling approach for referencing the IoT sources. IE should have values between zero and one, the closer IE value to zero the better the indexing efficiency. In conventional solutions such as [3], IE metric is equal to one.

The Computation Efficiency (CE) metric is used to benchmark the computation time of DSIS approach against a standard centralized approach similar to the solution described in [3]. To simulate this solution, we store all the IoT sources attributes in a large centralized repository. The processing time is calculated as the time required for performing SPARQL queries on the centralized repository. The computation efficiency is calculated based on the processing time of the proposed approach and the baseline model (i.e. centralized) as follows:

$$CE = \frac{T_b - T_{tot}}{T_b} \tag{13}$$

where T_b and T_{tot} represent the median of the measured processing time for the baseline and the proposed approach, respectively. T_{tot} includes the total time required for finding the GTWs (now, route processing time) and the time needed to process each SPARQL query at the GTWs (now referred as the processing at the reduced table). The route processing time comprises the extrapolation time and the time required for evaluating the GMM models on each IoT source attribute at all the DSs that the query traverses. We study the trend of the IE and CE parameters on the increase of the various network entities to verify the scalability of the approaches on the baseline solution.

7.5 Analysis and Results

The first category of simulations evaluates the accuracy and scalability of the proposed DSIS against the centralized baseline model. Each simulation scenario of this class investigates the effect of the population of a particular network entity (i.e. WSNs and GTWs) on the performance of IoT source indexing. The initial values for each object were described in Table 2. Each simulation is repeated 5-10 times, and the results are averaged.

The second category of simulations is dedicated to the summary updating mechanism. The performance of searching mechanism is compared and analyzed before and after the updating indices processes. A simulation scenario number (SSN) is allocated to each simulation to facilitate the comparison of the simulation scenarios.

7.5.1 WSNs Density Effect

The first set of simulations investigates the effect of the capacity of WSNs on the query processing performance of IoT indexed sources. This includes four simulation sets each of which specifies a different value for the maximum number of sensors that can be generated at the WSNs starting from 1000 sensors to 4000 sensors. Table 3 represents the average number of the network entities that were generated through the simulations.

Table 4 shows the simulation results. It can be observed that the majority of the queries are addressed at the first attempt. The success rate results also indicate that GMM models at GTWs are sufficient to represent the distribution of attributes of IoT sources even when the number of IoT sources increases. Success rates also indicate that the mis-detection of the DSl_1 inside DSl_2 is more significant than mis-detection of GTWs inside DSl_1. This observation can be utilized to provide more sophisticated query processing and enhance the QSR at the first attempt.

The trend of CE and IE parameters shows an improvement with the capacity of the WSNs. It can be seen that our approach outperforms the centralized baseline model timing when the WSNs capacity exceeds 2000 sensors.

Fig.6 demonstrates the distribution of the timings of query processing for the baseline solution in comparison with our proposed solution. It can be observed that in our approach, route processing accounts for the majority of the processing time. The blue boxes comprise the distribution of

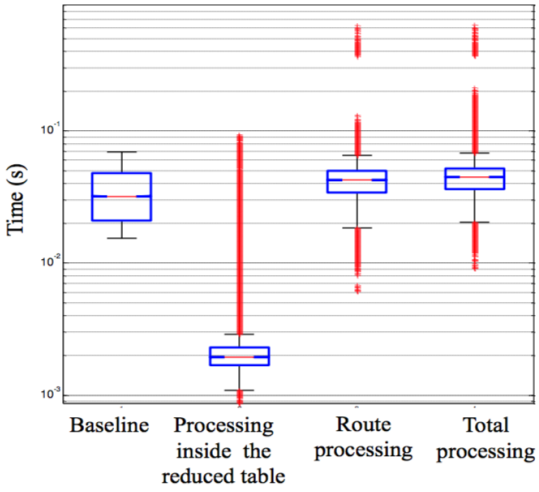

Fig. 6. Comparison of the computation time between the baseline model from [3] and the proposed solution. The WSN capacity is set to 2000, and the rest of the entities are generated based on their default value from Table 2.

TABLE 3
Simulation environment summary

SSN	WSN capacity	Sensors	WSN	GTW	DSl_1	DSl_2
1	1000	57709.0	201.8	38.3	12.0	4.0
2	2000	115210.0	194.8	39.8	11.8	4.0
3	3000	147330.0	141.2	31.0	11.2	4.0
4	4000	215290.0	215.0	36.0	12.3	4.0

the 25 to 75 percentile of the data, and the red dots are the outliers that are placed above the 95 percentile or below the 5 percentile values.

The improvement of CE values is down to the fact that by increasing the number of sensors the baseline model has to process the queries within significantly larger data entries. However, in the case of our approach query processing at the databases is limited to the sensors that are available at a particular GTW.

Furthermore, the variation of the IE parameter values on the WSN capacity implies that our approach tends to be more efficient when the WSNs capacity increases. This is because the number of generated GMM model components at the GTWs does not grow proportionally with the number of IoT sources.This observation confirms that our approach scales better with the number of sensors in compression with the baseline model.

Fig.7 shows the relationship between the number of queries and the number of routing processes attempts that have been made to address them for the different capacities of WSNs. Due to the negligible standard error of the results error bars are not plotted. It can be observed that the increase in capacity of the WSNs has improved the accuracy of the route identification at initial attempts. As mentioned earlier this could be down to the fact that GMM models tend to provide a better fit to the distributions of the sensors when the capacity of WSNs increases.

7.5.2 GTWs and DSs Density Effect

The detailed results of the evaluation of the node densities at GTW and DS levels are provided in Appendix B, Here, we

TABLE 4
WSNs capacity effect on query success rate

SSN	Query success rate (%)			Efficiency	
	First attempts	First DSl_1	First DSl_2	CE	IE
1	94.0	94.2	98.3	-1.42	0.0018
2	98.0	98.1	99.4	-0.28	0.0007
3	98.6	98.7	99.5	0.24	0.0006
4	99.4	99.5	99.7	0.13	0.0001

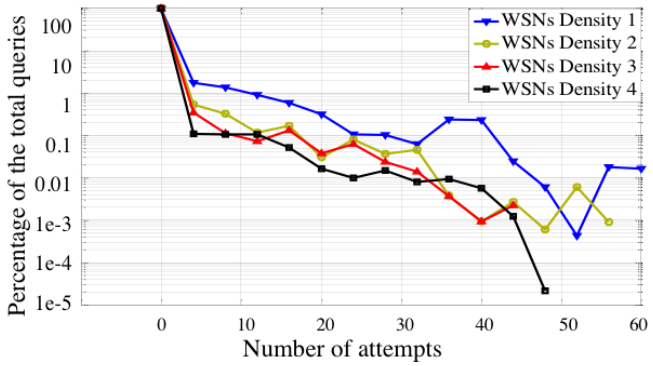

Fig. 7. Distribution of the queries over the number of query attempts for different WSN capacities, Capacities 1 to 4 are equal to 1000, 2000, 3000 and 4000 sensors respectively.

provide some remarks on the assessment results. Evaluation of GTWs density revealed that the number of WSNs within the GTWs has almost no impact on the success rates. Similar to the WSNs density effect, the CE and IE indicators improve when numbers of WSNs within the GTWs increases. We also examined the dependency of query processing performance on DSl_1, DSl_2, and DSl_3 capacities. It was observed that increase in the DSL1s capacity brings the initial level of misdirecting errors towards the higher layers of the network. This phenomenon may root in the effect of overlap between GMMs of GTWs that was explained in section 6. Changing DSl_2s and DSl_3s capacity didn't have any significant impact on the query processing performance. Another notable observation was that, in all evaluation scenarios, the CE parameter improved with increasing the number of sensor nodes. This phenomenon implies that the proposed approach has better scalability characteristics in comparison with the benchmarking approach.

7.6 Evaluation of Summary Updating Mechanism

The last set of simulations is dedicated to the evaluation of the updating mechanism. Through these simulations, once the simulation environment is set-up and the models are propagated through the network, some new sensors are added to the network. Next, the references are updated based on the proposed summary updating mechanisms. The effect of updates mechanisms is then investigated depending on the number of attempts that are required to locate the new data items. The simulation environment comprises 2 DSL_2 and the rest of the network entities are generated based on the configurations that are provided in 2. The simulation environment is also reduced to 122 km^2. Fig.8 shows the number of attempts before and after updates when the population of the added sensors is equal to 10, 22, 26, and 34 percent of the original population.

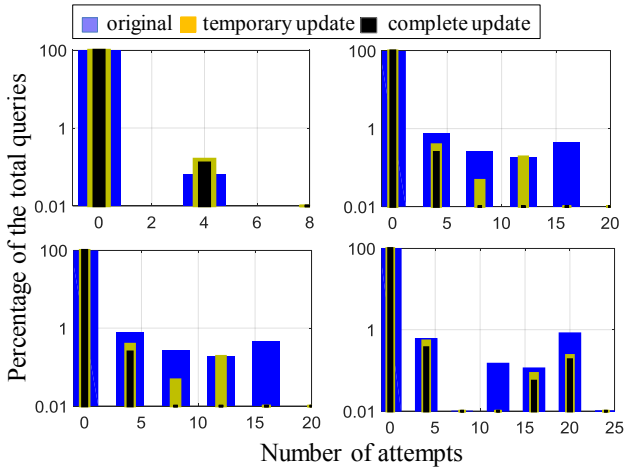

Fig. 8. DSL_2: Distribution of the queries over the number of attempts for different amount of added sensors, including (a) 10 percent (b) 22 percent (c) 26 percent and (34) percent of original population.

Apart from Fig.8 (a), which corresponds to the situation when the population of the added sensors equals 10% of the original population, the other results confirm that the query processing accuracy improves after both temporary-update and complete-update and the improvements are more significant after the complete-update. The close performance of the models at Fig.8(a) indicates that the original model is sufficient for addressing the queries when the difference between the population of the added sensors and the original population is negligible.

8 Conclusions

This paper a novel approach for indexing distributed IoT sources, called DSIS. DSIS employs GMMs to create concise indices to search for the data. DSIS benefits from a novel summary updating mechanisms to address the need for a frequent update of the references. Evaluation of the proposed solution indicates that if the overlaps between generated models are controlled, indices provided from DSIS are sufficient to accurately recognize the correct DS and GTW in which the queried data are stored. Evaluation of the update mechanisms, confirms that the temporary-update could be an effective substitute for the complete-update when a small margin of error is acceptable. Several simulations were then conducted to inspect the performance of the approach in terms of scalability of the indexing, and accuracy and reliability of the query processing. The results were also benchmarked against a widely used centralized approach. The results show that the proposed solution can locate the data within the simulated network with a high accuracy and low communication load (over 94% accuracy in the first attempt). In comparison with the baseline, our distributed approach shows better scalability properties by improving the computation efficiency, indexing efficiency and preserving the query processing accuracy when the population of different network entities are increased.

Overall the probabilistic indexing is an effective solution to resolve the scalability issues of the existing indexing mechanisms. However, such solution may not be suitable for small-scale networks where the gain on communication overhead for reducing the number of indices is marginal, and the excess of delay in data discovery on the discriminative solutions is noticeable. Temporary and complete updates may also become challenging when a majority of the indices change in a high rate, in the case of small scale networks, performing the learning and adaptation processes for updating models may happen to be less efficient than maintaining the original indices.

Future work will focus on evaluations with a large number of attributes. Privacy, security and trust are also other key issues that are important in indexing and discovery of IoT resources. In this paper, we have focused on developing scalable solutions for indexing and discovery of resources in distributed and dynamic IoT networks. However, providing access to resources and selecting appropriate devices/resources to obtain IoT data or service will also require information about reliability and trust; the devices/resources should also implement and deploy adequate security and privacy protection mechanisms that can control sharing their data/services in large-scale open networks. These additional selection criteria will require a suitable ranking mechanism that can extend the proposed indexing and discovery framework.

References

[1] A. Zaslavsky and P. P. Jayaraman, "Discovery in the internet of things: The internet of things (ubiquity symposium)," *Ubiquity*, vol. 2015, no. October, pp. 2:1–2:10, Oct. 2015. [Online]. Available: http://doi.acm.org/10.1145/2822529

[2] A. Salehi, M. Riahi, S. Michel, and K. Aberer, "Gsn, middleware for stream world," in *10th international conference on mobile data management (MDM 2009), May*, 2009, pp. 18–20.

[3] D. Le-Phuoc, H. N. M. Quoc, J. X. Parreira, and M. Hauswirth, "The linked sensor middleware–connecting the real world and the semantic web," *Proc. of the Semantic Web Challenge*, vol. 152, 2011.

[4] S. Nath, J. Liu, and F. Zhao, "Sensormap for wide-area sensor webs," *Computer*, vol. 40, no. 7, pp. 90–93, Jul. 2007. [Online]. Available: http://dx.doi.org/10.1109/MC.2007.250

[5] "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645 – 1660, 2013.

[6] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: a geographic hash table for data-centric storage," in *Proc. of the 1st ACM int. workshop on Wireless sensor networks and applications*. ACM, 2002, pp. 78–87.

[7] B. Fabian and O. Gunther, "Distributed ons and its impact on privacy," in *2007 IEEE International Conference on Communications*, June 2007, pp. 1223–1228.

[8] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: sextuple indexing for semantic web data management," *Proc. of the VLDB Endowment*, vol. 1, no. 1, pp. 1008–1019, 2008.

[9] A. Bhattacharya, A. Meka, and A. K. Singh, "Mist: Distributed indexing and querying in sensor networks using statistical models," in *Proc. of the 33rd int. conf. on Very large data bases*. VLDB Endowment, 2007, pp. 854–865.

[10] C. Perera, A. Zaslavsky, P. Christen, M. Compton, and D. Georgakopoulos, "Context-aware sensor search, selection and ranking model for internet of things middleware," in *Mobile Data Management (MDM), 2013 IEEE 14th Int. Con. on*, vol. 1. IEEE, 2013, pp. 314–322.

[11] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[12] P. Barnaghi and A. Sheth, "On searching the internet of things: Requirements and challenges," *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 71–75, Nov 2016.

[13] A. Meliou, C. Guestrin, and J. M. Hellerstein, "Approximating sensor network queries using in-network summaries," in *Information Processing in Sensor Networks, 2009. IPSN 2009. Int. Conf. on*. IEEE, 2009, pp. 229–240.

[14] C. Du, Z. Zhou, L. Shu, X. Jia, and Q. Wang, "An effective iot services indexing and query technique," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE Int.Conf.on and IEEE Cyber, Physical and Social Computing*. IEEE, 2013, pp. 777–782.
[15] M. Chen12, J. Wan, and F. Li, "Machine-to-machine communications: architectures, standards and applications," 2012.
[16] P. Barnaghi, W. Wang, L. Dong, and C. Wang, "A linked-data model for semantic sensor streams," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE Int. Conf. on and IEEE Cyber, Physical and Social Computing*. IEEE, 2013, pp. 468–475.
[17] S. Evdokimov, B. Fabian, S. Kunz, and N. Schoenemann, "Comparison of discovery service architectures for the internet of things," in *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE Int. Con. on*. IEEE, 2010, pp. 237–244.
[18] M. Wahl, T. Howes, and S. Kille, "Lightweight directory access protocol (v3)," 1997.
[19] N. Schoenemann, K. Fischbach, and D. Schoder, "P2p architecture for ubiquitous supply chain systems," 2009.
[20] B. Fabian, T. Ermakova, and C. Muller, "Shardis: A privacy-enhanced discovery service for rfid-based product information," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 3, pp. 707–718, Aug 2012.
[21] F. Paganelli and D. Parlanti, "A dht-based discovery service for the internet of things," *Journal of Computer Networks and Communications*, vol. 2012, 2012.
[22] S. Ramabhadran, S. Ratnasamy, J. M. Hellerstein, and S. Shenker, "Prefix hash tree: An indexing data structure over distributed hash tables," in *Proc. of the 23rd ACM symposium on principles of distributed computing*, vol. 37, 2004.
[23] W. T. Lunardi, E. de Matos, R. Tiburski, L. A. Amaral, S. Marczak, F. Hessel, and M. Thielo, "Cobasen (context-based search engine) framework: Discovery of computing devices in iot environments."
[24] D. Zhang, H. Guo, and B. Luo, "An algorithm for estimating number of components of gaussian mixture model based on penalized distance," in *Neural Networks and Signal Processing, 2008 Int. Conf. on*. IEEE, 2008, pp. 482–487.
[25] M. Shi and A. Bermak, "An efficient digital vlsi implementation of gaussian mixture models-based classifier," *Very Large Scale Integration (VLSI) Systems, IEEE Trans. on*, vol. 14, no. 9, pp. 962–974, 2006.
[26] C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," in *Mining Text Data*, C. C. Aggarwal and C. Zhai, Eds. Springer US, 2012, pp. 77–128.
[27] E. W. Kamen and J. K. Su, *Introduction to Optimal Estimation*. Springer, 1999.
[28] Y. Fu, Z. Yan, J. Cao, O. Kon, and X. Cao, "An automata based intrusion detection method for internet of things," vol. 2017, pp. 1–13, 01 2017.
[29] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things," *Journal of Network and Computer Applications*, vol. 42, pp. 120 – 134, 2014.
[30] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10 – 28, 2017.
[31] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *Services Computing, IEEE Trans. on*, vol. 3, no. 3, pp. 223–235, 2010.
[32] D. Reynolds, "Gaussian mixture models," *Encyclopedia of biometrics*, pp. 827–832, 2015.

S.Amir Hosseinitabatabaei is a Senior Research Fellow at the Institute for Communication Systems (ICS) at University of Surrey. Amir received his PhD in electronic engineering from University of Surrey in 2013. His research interests include pervasive and mobile computing, Internet of Things and mobile communications. Contact him at: s.hoseinitabatabaei@surrey.ac.uk.

Yasmin Fathy is a PhD candidate at Institute of Communication Systems (ICS) at the University of Surrey. She obtained her MSc in Artificial Intelligence (AI) from AI Lab at Vrije Universiteit Brussel (VUB) in Belgium. Her research interests include Internet of Things (IoT), big data analytics, machine learning and stream processing. Contact her at: y.fathy@surrey.ac.uk.

Payam Barnaghi is a Reader in Machine Intelligence at the University of Surrey. His research interests include machine learning, Internet of Things, semantic web, web services, information centric networks, and information search and retrieval. Contact him at: p.barnaghi@surrey.ac.uk.

Chonggang Wang is a member of Technical Staff at InterDigital Communications. His research interests include IoT, Mobile Communication and Computing, and Big Data Management and Analytics. He is the founding Editor-in-Chief of IEEE Internet of Things Journal. Contact him at: cgwang@ieee.org.

Rahim Tafazolli is a Professor of Mobile and Satellite Communications and the Director of Institute of Communication Systems (ICS)/5G Innovation Centre at the University of Surrey. He has been active in research for more than 20 years, has authored and co-authored more than 600 papers in refereed international journals and conferences. Contact him at: r.tafazolli@surrey.ac.uk.

APPENDIX A
GMM TRAINING PROCESS

Gaussian Mixture Model (GMM) is a parametric distribution estimation model. Data distribution is estimated as a weighted sum of a set of Gaussian components, where each component has a mean value E, a covariance matrix σ and mixing weight φ. The model parameters are estimated by exploiting the Expectation Maximisation (EM) algorithm on the training dataset.

By adopting the conventional EM algorithm for GMMs training process in our problem can be formulated as follows.

E-step (expectation): Here the posterior probabilities are calculated based on the current guess of parameters:

$$w_k^l = Pr(Mc = k|ax_l; \varphi, E, \sigma) = \frac{p(ax_l|Mc = k; E, \sigma)\ p(Mc = k; \varphi)}{\sum_{h=1}^{C}\ p(ax_l|Mc = h; E, \sigma)p(Mc = h; \varphi)} \quad (14)$$

In Equation 14, Mc is the set of mixture components and C represents the total number of mixture components (i.e. $\|Mc\|$), a_{xl}, represents the vector of attribute values for the l^{th} data entry, w_k^l is a posterior probability of k^{th} Gaussian mixture component (Mc) given ax_l. M-Step (maximisation): at this step model parameters are calculated in way that maximizes the log-likelihood of $p(ax|\varphi, E, \sigma)$.

$$\varphi_k = \frac{1}{L}(\sum_{l=1}^{L} \omega_k^1) \quad (15)$$

$$E_k = \frac{\sum_{l=1}^{L} \omega_k^1 ax_l}{L\varphi_k} \quad (16)$$

$$\sigma_k = \frac{\sum_{l=1}^{L} \omega_k^1 ax_l {ax_l}^t}{L\varphi_k} - E_k E_k^t \quad (17)$$

φ_k is mixing proportion of the k^{th} Gaussian component. E_k is the estimation of the mean of the k^{th} Gaussian component. σk is the estimated covariance of the k^{th} component. L is the total number of data entries. The algorithm then iterates between E and M steps until it converges (i.e. until there are no significant changes on the likelihood function). Further details on GMM can be found at [32]. Figure 9 represents the flowchart of GMM training process.

Computation Complexity

In the training phase, GMM algorithm runs iteratively until the termination conditions are satisfied. The major computation burden is at Maximization step. The time complexity of matrix manipulations (e.g. inversion, matrix determinant, and matrix decomposition) is O(D^3), where D is the dimensionality of the attribute data space (e.g. D = 2 for our location attribute). In case of posterior probabilities, the time complexity is O(D^2) for each pair of ax_l and Mc . Therefore, the total time complexity of GMM training would be O($I * K * N * D^2 + I * K * D^3$), where K is the number of mixture components, N is the number of data points and I is the number of iterations of EM algorithm.

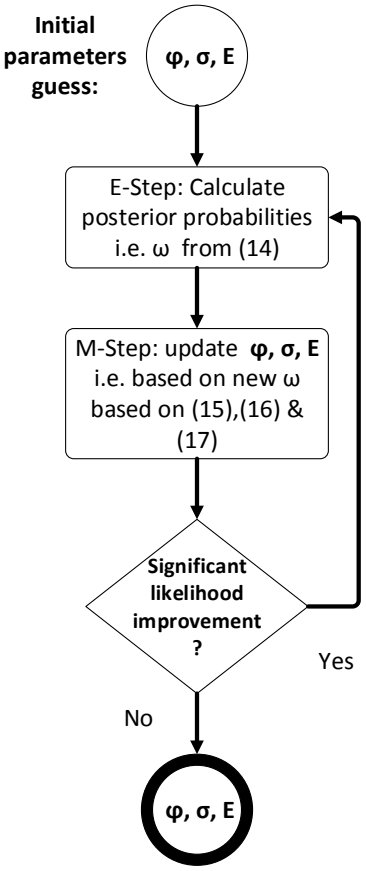

Fig. 9. GMM training flowchart.

APPENDIX B

Evaluation of GTW's density revealed that the number of WSNs within the GTWs has almost no effect over the success rates. Similar to the WSNs density effect, the CE and IE parameters improve when higher numbers of WSNs increases the number of sensors at the GTWs. This includes 4 simulation sets each of which specifies a different value for range of the capacity of the GTWs, starting from 2-7 to 3-8, 4-9 and 6-11 WSNs. Table 5 represents the average number of the entities that were generated through the simulations. Simulation results are presented in Table 6.

Results in Table 6 show that the GTW capacity has almost no effect over the success rates among these simulations. However, the success rates have improved in comparison with WSN capacity, which has similar specification but less number of GTWs. This can be explained by the effect of increase in the number of sensors that were referenced at the GTW which as was explained earlier tend to improve the goodness-of-fit of the GMM components. Similar phenomenon has been observed when the capacities of WSNs were analysed. Analysis of the required number of route processing attempts in Fig.10 also verifies with this observation. Improvement of the CE values can be explained by the effect of increase in the number of sensors over the baseline model as was described earlier. Comparison of the IE values also suggests that the number of sensors is a more critical parameter than the number of WSNs at GTWs for

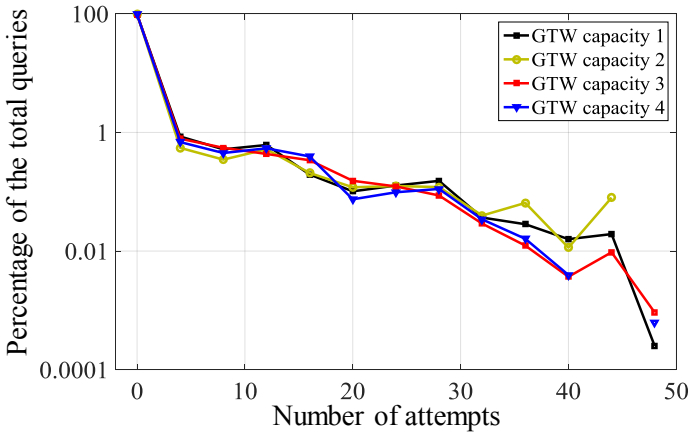

Fig. 10. GTW Density Effect: Distribution of the query over the number of query attempts for different GTW capacities, Capacities 1 to 4 are equal to 2-7, 3-8, 4-9 and 6-11 WSNs respectively.

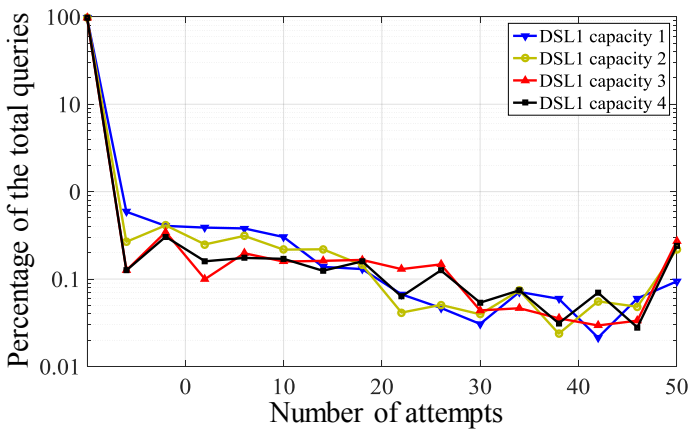

Fig. 11. DSL_1: Distribution of the queries over the number of query attempts for different DSL_1s capacities, Capacities 1 to 4 are equal to 2-7, 3- 8, 4-9 and 6-11 GTWs respectively.

the number of generated Gaussian components.

TABLE 5
GTW Density Effect: Simulation environment summary

SSN	GTW capacity	Sensors	WSN	GTW	DSL1	DSL2
1	2-7	90496.0	364.6	36.2	11.8	4.0
2	3-8	92355.0	464.0	35.0	10.3	4.0
3	4-9	98964.0	519.0	33.0	10.7	4.0
4	6-11	177850.0	538.2	39.2	14.4	4.0

TABLE 6
GTW Density Effect: GTW capacity effect on query success rate

SSN	Query success rate (%)			Efficiency	
	First attempts	First DS1	First DS2	CE	IE
1	97.0	97.2	99.0	-0.62	0.0009
2	97.6	97.8	99.1	-0.61	0.0009
3	97.2	97.4	98.9	-0.59	0.0009
4	97.3	97.5	98.9	-0.12	0.0004

APPENDIX C
DISCOVERY SERVER LEVELS DENSITY CAPACITY EFFECT

The following simulations examine the dependency of query processing performance to the three level overlays of DSL_1, DSL_2 , and DSL_3 capacity.

DS-Level 1

The following simulations examine the dependency of query processing performance to the DSL_1 capacity. This includes 4 simulation sets each of which specifies a different values for range of the capacity of the DSL_1s, starting from 2-7 to 3-8, 4-9 and finally 6-11 GTWs. Table 7 represents the average number of the entities that where generated through the simulations. Simulation results are provided in Table 8.

In Fig.11, DSL_1 capacity has a clear impact on the number of attempts that are required for addressing the queries after the initial attempt. After the initial attempt,

TABLE 7
DSL_1: Simulation Environment Summary

SSN	DSL1 capacity	Sensors	WSN	GTW	DSL1	DSL2
5	2-7	82670.0	336.4	53.6	11.0	4.0
6	3-8	118820.0	397.6	76.4	12.8	4.0
7	4-9	126210.0	488.6	85.0	12.2	4.0
8	6-11	158400.0	490.6	110.0	12.2	4.0

increase in DSL_1s capacity lowers the number queries that are addressed under the fist selected DS_1 and at GTW level (e.g. $\approx$ 5-10 attempts) and in- creases the amount of queries that are addressed at higher layers. This phenomenon implies that the increase in the DSL_1 capacity has shifted the source of misdirecting errors of the queries, towards the higher layers of the network.

It was earlier mentioned that the query time of the baseline model, increases with the number of IoT sources, which in turn has improved of CE value between SSN 5 and 6 in Table 8. At SSN 7 and 8, according to Fig.11, the queries that are not addressed by the initial attempts require considerably more query attempts in comparison with lower capacities. Query routing process overhead of the queries that are not addressed at the first attempt in turn degrades the CE parameter.

DS-Level 2

Next set of simulations analyse the impact of variation of DSL_2 capacity on the query processing performance. This includes 4 simulation sets each of which specifies a different value for range of the capacity of the DSL_2, starting from

TABLE 8
DSL_1: Gateway Capacity Effect On Query Success Rate

SSN	Query success rate (%)			Efficiency	
	First attempts	First DS1	First DS2	CE	IE
5	96.9	97.0	98.7	-1.12	0.0010
6	97.6	97.8	99.0	-0.3	0.0006
7	98.0	98.1	99.0	-0.4	0.0006
8	97.2	98.2	99.0	-1.01	0.0005

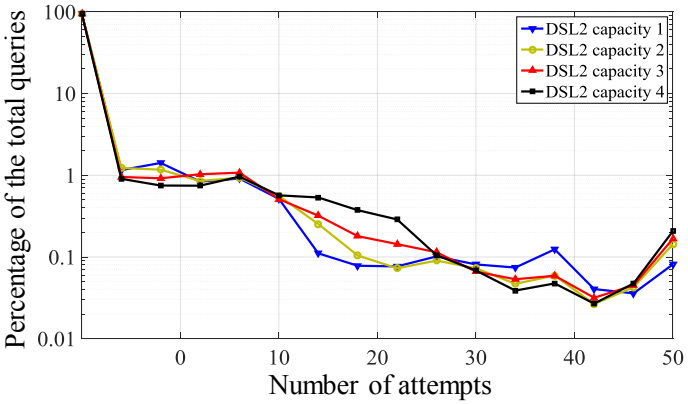

Fig. 12. DSL_1: Distribution of the queries over the number of query attempts for different DSL2s capacities, Capacities 1 to 4 are equal to 2-7, 3-8, 4-9 and 6-11 DSL1s respectively.

TABLE 9
DSL_2: Simulation Environment Summary

SSN	DSL2 capacity	Sensors	WSN	GTW	DSL1	DSL3
9	2-7	95316.0	323.2	63.7	20.6	4.0
10	3-8	103220.0	490.8	68.3	22.3	4.0
11	4-9	116600.0	478.7	76.4	25.3	4.0
12	6-11	137320.0	472.7	88.5	28.9	6.0

2-7 DSL_1 to 3-8, 4-9 and 6-11 DSL_1. Table 9 summarizes the average number of the entities that where generated through the simulations and simulation results are tabulated in Table 10. Query success rate results indicates that the proposed approach is agnostic to the capacity of the DS_2. These observations alongside with results of SSN 1, and which has similar specification but less capacity for DSL_2, suggest that lower layer configuration such as WSNs density may have a more significant impact over the performance of the query processing.

Another interesting phenomenon is the trend of the IE parameter. Similar to effect of DS_1 capacity, with the increase of DS_2 capacity more and more GTWs and subsequently sensors and GMM models are created. Therefore, IE parameter shows the trade-off between the numbers of model components to describe the available IoT sources and size of the network. Improvement on CE value with the increase of DSL_2 capacity, complies with our previous claim on shortcoming of the baseline model for processing queries for higher order of sensors and emphasis on scalability of our approach.

Fig.12 shows the number of attempts required for processing the queries at each capacity range of DSL_2. While the ratio for the initial attempts that are typically made under the first DS_1, are similar among different traces, there is a region between 20-40 attempts that shows a significant divergence between traces. This divergence can be due to the DSL_2 capacity effect where increasing the more attempts to resolve the queries that were not ad- dressed under the first DS.

TABLE 10
DSL_2: Gateway Capacity Effect On Query Success Rate

SSN	Query success rate (%)			Efficiency	
	First attempts	First DS1	First DS2	CE	IE
9	94.3	94.3	99.0	-0.91	0.0008
10	94.2	94.3	99.0	-0.84	0.0008
11	94.2	94.3	99.0	-0.72	0.0007
12	94.2	94.3	99.0	-0.61	0.0007

DS-Level 3

The final set of the first category of simulations, explores the effect of the variation of DSL_3 capacity over the query processing performance. This includes 4 simulation sets each of which specifies a different value for the capacity of the DSL_3 including 2, 3, 5 and 6 DSL_2. Table 11 summarizes the average number of the entities that where generated through the simulations and results are shown in Table 12. Improvement on CE value with the increase of the number of DSL_2, complies with our previous observations on the effect of total number of sensors and emphasis on scalability of our approach. IE value has also remained unchanged at different capacity of DSL_3 implying that the number of created Gaussian components tends to increase in proportion with the number of sensors in these scenarios.

Fig.13 shows the number of attempts required for processing the queries at each capacity range of DSL_3s. It can be observed that the order of attempts after the first DSL_1, (i.e $\approx$>15 attempts) increases with the size of the network.

TABLE 11
DSL_2: Simulation Environment Summary

SSN	DSL3 capacity	Sensors	WSN	GTW	DSL1	DSL2
13	2	24187.0	87.3	16.6	6.2	2.0
14	3	41659.0	174.1	27.9	9.3	3.0
15	5	58441.0	218.0	39.8	13.4	5.0
16	6	78920.0	361.0	53.0	18.6	6.0

APPENDIX D
IMPLEMENTATION RECOMMENDATIONS

Adding a new DS

Once a new DS is added to the network, depending on the layer of the DS, a group of GTWs/DSs at the lower layer send their GMMs to the new DS. As a lesson learned from our observations in section 5, these GTWs/DSs should be selected in a way that the overlap between aggregated models at the upper layer is minimized.

TABLE 12
DSL_2: Gateway Capacity Effect On Query Success Rate

SSN	Query success rate (%)			Efficiency	
	First attempts	First DS1	First DS2	CE	IE
13	92.3	92.5	98.5	-3.06	0.0018
14	93.8	94.0	98.7	-2.41	0.0016
15	95.2	95.4	98.5	-2.19	0.0018
16	94.2	94.3	98.5	-1.90	0.0018

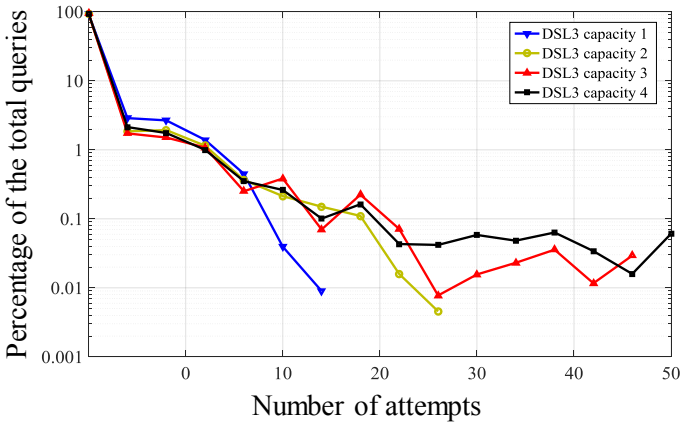

Fig. 13. DSL_2: Distribution of the queries over the number of query attempts for different DSL_3s capacities, Capacities 1 to 4 are equal to 2, 3, 5 and 6 DSL_2s respectively.

Node failure

In order to cope with node failure, each network node may maintain a copy of its GMM in its neighbour. Upon failure of that node, neighbour nodes can aggregate a part/whole of the copied GMM model to their own model and associate themselves with related links to the upper and lower layers.

APPENDIX E
DSIS PSEUDO CODES

The proposed index summarization method and update mechanism are summarized in Fig.14.

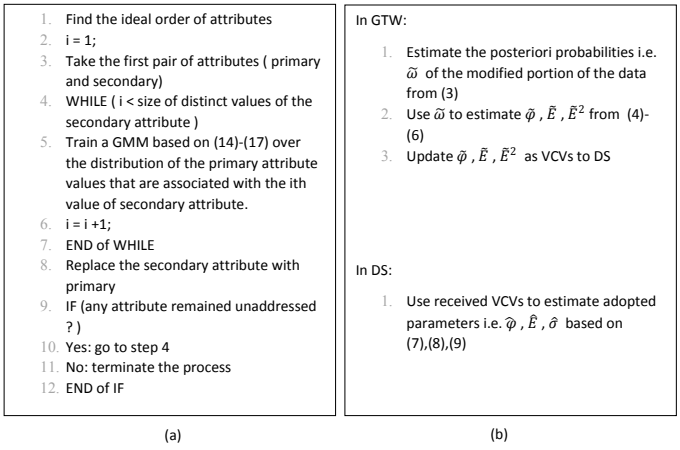

Fig. 14. Pseudo codes of the proposed solution. a) attribute summarization method. b) index updating method