A Distributed In-network Indexing Mechanism for the Internet of Things

Yasmin Fathy, Payam Barnaghi, Shirin Enshaeifar and Rahim Tafazolli
Institution for Communication Systems (ICS), Electronic Engineering Department, University of Surrey
Guildford, Surrey, GU2 7XH, United Kingdom
{*y.fathy, p.barnaghi, s.enshaeifar, r.tafazolli*}*@surrey.ac.uk*

***Abstract*—The current Web and data indexing and search mechanisms are mainly tailored to process text-based data and are limited in addressing the intrinsic characteristics of distributed, large-scale and dynamic Internet of Things (IoT) data networks. The IoT demands novel indexing solutions for large-scale data to create an ecosystem of system; however, IoT data are often numerical, multi-modal and heterogeneous. We propose a distributed and adaptable mechanism that allows indexing and discovery of real-world data in IoT networks. Comparing to the state-of-the-art approaches, our model does not require any prior knowledge about the data or their distributions. We address the problem of distributed, efficient indexing and discovery for voluminous IoT data by applying an unsupervised machine learning algorithm. The proposed solution aggregates and distributes the indexes in hierarchical networks. We have evaluated our distributed solution on a large-scale dataset, and the results show that our proposed indexing scheme is able to efficiently index and enable discovery of the IoT data with 71% to 92% better response time than a centralised approach.**

***Keywords*-Internet of Things (IoT); Wireless Sensor Networks (WSN); distributed Indexing; real-world data**

I. Introduction

The Internet of Things (IoT) is a substantial part of the future Internet. The term IoT was coined over a decade ago. However, with the prevalence of low-cost sensors and network-connected devices, the IoT has gained momentum in recent years to push the boundaries between physical and digital worlds. The IoT bridges the gap between ubiquitous network-enabled devices that record physical world observations and measurements and provide services, applications, and autonomous interactions and the network (e.g. Internet). The data generated by sensory devices are often spatio-temporal data streams that change over time. This demands dealing with the data drift problem in which data characteristics and descriptive statistics change over time. IoT data streams are different from the conventional data streams [1]. They demand different requirements and more dynamic and adaptive solutions.

The Wireless Sensor and Actuator Networks (WSANs) play a central role in IoT infrastructure to allow autonomous data collection and resource management. However, the large-scale network-enabled devices impose challenges in collecting, querying and processing their data and services on the Internet. Indexing IoT resources allows fast and efficient search and discovery for their data and services. However, continuous scanning for the entire connected devices is inefficient and computationally intensive. Effective distributed, scalable and efficient indexing solutions are required to support data discovery and accessing for large-scale IoT networks.

The problem of indexing IoT resources comprises two key elements: building an indexing structure and developing a query processing scheme. The processing scheme is to identify the key attributes (e.g. location and type) in which users can use these attributes to search (query) for a data resource. Most of the existing indexing IoT frameworks and solutions rely on using pre-defined resource links or centralised data repositories which make them not scalable for discoverable large-scale and uncoordinated networks of devices and resources [2, 3].

The remainder of this paper is structured as follows. Section II discusses the background and related work. Section III details and discusses our architecture with a focus on distributed indexing mechanism. Evaluation and experimental results that are conducted and analysed in our simulated environment against a centralised baseline approach are discussed in Section IV. The conclusions, possible extensions and future work are discussed in Section V.

II. Background and Related Work

Distributed indexing for IoT resources is to discover a set of connected devices in an efficient manner to find and have fast access to a resource that can have the requested data. Many indexing approaches have been proposed and developed such as [4, 5]. For example, Geographic Hash Table (GHT) [4] relies on a hash function that maps information type into geographic coordinates; key-value pairs are used to construct a distributed index wherein a key is an event type name (e.g. high temperature), and the value is the data location. This allows GHT to group nodes with the same type of information together, albeit they might be far away. However, GHT only supports binary events (i.e. either an event occurs or it does not) and exact queries. Greenstein et al. [5] extend GHT by introducing a Distributed Index for Features in Sensor Networks (DIFS) to support range queries. Indexes are constructed as a tree-based structure in which each node in the tree stores information about a certain range of values within a geographical area. The non-root nodes in DIFS tree structure can have several parents. However, DIFS is susceptible to have a distance sensitivity problem if some of the parent nodes of a child node in the tree are located far away in different geographic areas. Moreover, constructing and updating DIFS structure are costly [6]. This is because every node in the tree

978-1-5090-4130-5/16/$31.00

should be aware of the boundary of the entire geographical area. Distributed Index for Multi-dimensional data (DIM) is described in [7]. The work relies on dividing the whole sensor field into partitions (i.e. zones) and preserving data-locality by hashing multi-attribute events into geographic zones. This allows constructing a multi-dimensional search tree in which each geographical area is represented by multi-attribute events to support routing multi-dimensional range queries. However, routing algorithms are resource intensive and are not well scaled with large-scale sensor networks [8].

Meliou et al. [9] introduce a probabilistic model-based approach for constructing in-network spanning tree to minimise the communication overhead while processing queries in WSNs. Each node in the tree has a constructed Gaussian model by aggregating Gaussian models of its child nodes. A summarisation mechanism is used to summarise data from different tree levels for answering approximate user queries with a specified accuracy. However, "Select" query statements are received by all sensors in the network, and the model can not answer spatial queries (i.e. query of sensor observation and measurement data from specific locations). Another distributed indexing solution is proposed in [10]. The indexing mechanism assumes that there is a one-to-one relationship between the nodes (resources and services) and their indexes. However, this results in a high complexity for updating the indexes.

The IoT should support mechanisms for discovery and query of IoT resources and their data. By IoT resource, we refer to any resource (e.g. service, ubiquitous device) in the WSN/Internet that publishes its data on the network. Various IoT Discovery Services (DSs) have been surveyed in [11]. Discovery services are mainly to find the data providers (resources) for the requested queries given the key search attributes (e.g. type and location). Most of the existing DSs in the IoT are centralised [12]. Some other discovery services offer limited functionalities. For example, Linked Sensor Middleware (LSM), Global Sensor Networks (GSN) middleware, and other similar solutions usually provide limited and mainly centralised search and discovery mechanisms [13].

A Distributed Hash Table (DHT)-based discovery service is introduced in [14]. DHT locates distributed data efficiently. However, it supports only exact match for a given key. There are also solutions such as Wolfram Data Drop[1] and Thingful[2]. Thingful relies on adding meta-data and the description of resources manually. The search and discovery functions in Wolfram Data Drop are based on specific data streams for each instance (user performs operations on data from a specific databin[3]). Other solutions such as Dyser and SenseWeb/SenseMap [15] rely on a centralised indexing approach which hinders their scalability for distributed networks.

Overall, scalability and efficient update of the indexes in dynamic IoT networks are the main drawbacks of these solutions. This motivates us to compare our proposed distributed indexing mechanism with a centralised solution. Our main contribution is a novel distributed and efficient indexing mechanism in a hierarchical distributed network to allow discovery of IoT data. It is worth mentioning that wireless related issues (e.g. transfer delay) are not within the scope of the current work.

III. Proposed Approach

We have designed and developed an architecture for distributed indexing of the IoT data. Figure 1 shows the key components of the architecture design. The following describes a step-by-step description of the indexing and discovery process:

(i) Data are published by various IoT resources.
(ii) Every IoT data resource has a type (e.g. temperature), location (i.e. longitude, latitude and altitude) and value (e.g. 12°C) attributes that can be queried.
(iii) Every IoT data resource has only one type (dew temperature, relative humidity, visibility, wind direction, temperature, or wind speed).
(iv) Data can be archived in Information Repositories (IRs) or can be accessed directly via the resources.
(v) Gateways (GWs) are intermediary nodes that have access to individual resources.
(vi) Discovery Services (DSs) aggregate indexes of GWs together (only references to GWs with some associated information (e.g. set of types for the connected devices)).
(vii) The query is composed of type, location, and time attributes.
(viii) The indexing process is based on resources which allow discovery of their data.

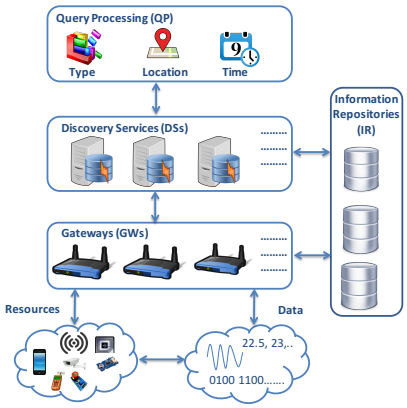

Figure 1. The architecture for distributed indexing

[1]http://www.wolframalpha.com/
[2]http://thingful.net/
[3]is a unique identifier for a Wolfram Data Drop instance (e.g. sensor)

In the proposed architecture, Query Processing (QP) receives user queries and forwards them to DSs. DSs are responsible for routing and locating a set of related GWs that might have a resource that provides a response to the requested query, or DS can search historical data that are archived in IRs. The proposed indexing and discovery framework consists of three main layers as shown in Figure 1. We have constructed and evaluated our model by indexing approximately 6 million data records from different resources. Each resource has a set of attributes (e.g. location, type, time). These are explained in the following.

A. Real-world Dataset

We use a set of weather sensory data that are collected from Automated Surface Observing System (ASOS)[4]. The data are gathered from weather stations that are located in different countries. The dataset has the following attributes; source-name (i.e. station name), geographical co-ordinates; longitude (e.g. 153.26) and latitude (e.g. -28.83), country (we currently use data from 7 different countries; Belgium, United Kingdom, Canada, Australia, Egypt, Japan and the United States), time-stamp (e.g. $2014-01-0100:03:00$), air temperature in Celsius (e.g. 26.0), and dew point temperature in Celsius (e.g. 17.0). Dew point is used to measure atmospheric moisture. The dataset has also relative humidity (e.g. 57.6%), wind direction in degrees from north (e.g. 80.0), and wind speed in knots (e.g. 5.0).

We remove the invalid values and apply interpolation to fill missing values. We use Google Elevation API[5] to get altitude values given longitude and latitude values. We then convert the geo-graphical co-ordinate system (longitude, latitude, altitude) into Cartesian co-ordinates (X, Y, Z) for clustering. We take into account that the geographical points in our dataset are based on World Geodetic System (WGS-84) [16] (it assumes earth is an ellipsoid rather than a sphere). The conversion between the two co-ordinate systems are performed using the following equations:

$$\begin{aligned} X &= (N+h)\ cos\phi\ cos\lambda \\ Y &= (N+h)\ cos\phi\ sin\lambda \\ Z &= ((1-e^2)N+h)\sin\phi \end{aligned} \tag{1}$$

Where λ, ϕ, h are longitude, latitude and altitude (in radian), respectively, and N is the vertical radius of curvature (in metres) and can be obtained by:

$$N = \frac{a}{\sqrt{(1-e^2 sin^2\phi)}} \tag{2}$$

Where $e = 0.081819190842622$ and $a = 6378137$ are constant values for eccentricity and semi-major axis (in metres) and are WGS84 ellipsoid parameters that define the ellipsoid's shape. It is worth noting that X, Y, Z are in metres.

[4]http://mesonet.agron.iastate.edu/ASOS/

[5]http://developers.google.com/maps/documentation/elevation/

Indexing IoT resources can be a multi-dimensional (location and type). However, constructing the multi-dimensional indexing has a high complexity. To this end, a partially distributed indexing scheme is constructed that takes into account the resource locations and aggregation of resource types; this is referred to as "semi-distributed" approach. Moreover, this approach is extended to a fully distributed indexing scheme which also indexes the types within geographical areas; this is called the "fully-distributed" approach. Both schemes enable answering user queries, i.e., requests about a specific resource location and a given type.

B. Geo-location Clustering

IoT data are dynamic and often numeric and streaming data. They are also time and location dependent. To deal with their dynamicity, unsupervised statistical machine learning algorithms are used such as Gaussian Mixture Model (GMM), DBSCAN and spectral clustering. However, these techniques have parameter settings that need to be known in advance or require running the algorithm many times to determine the appropriate values for these parameters. For example, DBSCAN requires the Eps and MinPts values (MinPts is the minimum number of points in Eps-neighbourhood of each point p), and spectral clustering requires the number of clusters to be determined in advance. One promising technique that can be used is Dirichlet Process (DP)-means clustering algorithm [17]. DP-means is a variant of k-means clustering algorithm with non-parametric settings; the cluster cardinality is not known in advance, but instead it is dynamic and can be inferred from the dataset. DP-means has a cluster penalty (i.e. threshold) parameter λ that controls creating new clusters; it has a key role in deciding whether a data-point attaches/joins a cluster, or it requires creating a new cluster.

We first cluster our dataset based on the resource locations using DP-means. The output of clustering process is the number of clusters k and list of data-points (i.e. resource locations) that belong to these clusters. We have developed a modified version of DP-means that is slightly different from the original algorithm that is presented in [17]. The modified DP-means pseudo-code is shown in Algorithm 1. The main difference is that the threshold penalty parameter λ is initialised by the mean of the standard deviation of data-points (resource locations) (X, Y, Z) ($\lambda = \frac{\sigma_X+\sigma_Y+\sigma_Z}{3}$). This is because the statistical dispersion of different resource locations in $\mathbb{R}^3$ (X, Y, Z) can give an indication of how clusters could be significantly distinct from each others. We also do not initially attach all data-points to the first cluster and do not calculate the global mean as it was explained in the original DP-means. Both of the resource locations and cluster centroids are represented in Cartesian plane. We can simply get the distance between each resource location and each cluster centroid. Accordingly, we use a modified version of Euclidean distance that is used to get

the distance between two points in $\mathbb{R}^2$ to work in $\mathbb{R}^3$ (as shown in equation 3), where $p_i(x_1, y_1, z_1)$ is a data-point, $\mu_c(x_2, y_2, z_2)$ is a cluster centroid and $Dict(p_i, \mu_c)$ is the distance between a point p_i and a cluster centroid μ_c.

$$Dict(p_i, \mu_c) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (3)$$

Algorithm 1: The modified DP-means algorithm

Input : p_1, ..., p_n, data-points (resource locations)
e.g. p_1=(x_1, y_1, z_1), $x_1 \epsilon X, y_1 \epsilon Y, z_1 \epsilon Z$
▷ n : number of data-points
λ : cluster penalty parameter
▷ λ = mean of σ for (X, Y, Z)
I : maximum number of iterations

Output: Clustering l_1, ..., l_k and number of clusters k

1) Init. cluster indicators $z_i = -1$ for all data-points $p_i, i = 1, ..., n$.
2) Init. $k = 1$, k : number of clusters
3) Randomly select k_1 initial centroid (p_i), $i = 1, ..., n$.
▷ $l_1 = \{p_i\}$
4) Repeat for I iterations
   - Repeat for each point p_i, $i = 1, ...n$
     - Compute $dist_{ic} = Dict(p_i, \mu_c)$ for $c = 1, ..., k$
     - If $min_c(dist_{ic}) > \lambda$, set $k = k + 1$, $z_i = k$, and $\mu_k = p_i$
     - Otherwise, set $z_i = argmin_c(dist_{ic})$
   - Generate clusters $l_1, ..., l_k$ based on $z_1, ..., z_k$:$l_j = \{p_i | z_i = j\}$.
   - For each cluster l_j, compute $\mu_j = \frac{1}{|l_j|} \sum_{p \epsilon l_j} p$
     ▷ $j = \{1, 2, ...k\}$

C. Distributed Indexing

Indexing IoT resources is performed on a set of distributed gateways. A gateway is a physical or logical node (i.e. a machine that can act as an intermediary between the devices (resources) and DSs). The number of gateways is defined by the number of clusters obtained by the modified DP-means clustering algorithm. Each gateway represents a cluster (and its centroid) and has a direct access to the IoT resources that belong to this cluster. These clusters (gateways) are represented to the upper layer (DSs) by their cluster centroids and a set of the types (e.g. temperature, relative humidity) of their connected resources. The resource attributes and values are accessed at the resource level, and the gateway has only references to its connected resources. However, a set of different types of the resources connected to each gateway is sent along with its cluster centroid to the higher-level DS that is linked to that gateway. By this way, we make sure that we have an aggregated representation of the connected resources to the lower layer (i.e. gateways) at the upper layers (i.e. DSs) with less overhead. This is referred to as "semi-distributed" approach. Extending this approach to a fully distributed approach is performed by building a tree structure per cluster which has n children, where n is the number of types per cluster, and this is called the "fully-distributed" approach.

Each DS node has a list of all cluster centroids and types of their connected resources (devices/services). In "semi-distributed" approach, DS will receive a user query (e.g. temperature in a specific location), it will then select the cluster (gateway) with the minimum distance between its centroid and the requested query location and has type (e.g. temperature) in the set of types associated to that cluster. The gateway will then search to access a resource with the requested attributes to retrieve the requested type value (e.g. 12°C).

In "fully-distributed" approach, the gateway is not going to search sequentially to find a connected resource with the requested attributes, but instead, it will only access the resources with a given type (temperature) because each gateway has a tree structure of its types as explained earlier.

Our proposed solution considers the change in device locations and availability. However, to mitigate the overhead of frequent changes in indexes, our update mechanism is to predict simply the cluster membership of a new resource by selecting a gateway whose centroid has the minimum distance with the location of the new data item (resource). This enables on-line indexing and discovery. The computational complexity in this case (best case) is $O(k)$ (k is the number of clusters) where the new data-points are attached to existing clusters. However, if a large number of new devices connected to the network, the underlying data might change significantly and we will need to re-calculate the centroids to retain the accuracy of the indexes. In this case, the computational complexity (worst-case) is $O(i * n * k)$, where i, n, k are the number of iterations, data-points (resources) and clusters, respectively.

IV. Evaluation

We use a centralised solution as our baseline model to compare the results. This is because most of the existing approaches and solutions are based on centralised indexing as previously mentioned in Section II. In the centralised approach, all IoT data resources are connected to one gateway, and the data are archived in one Information Repository (IR). We compare the results of our proposed solution (semi-distributed and fully-distributed) to the baseline approach. Both solutions are tested under the same conditions and implemented in Python. We apply both solutions on the same dataset and have run the algorithms on an OS X machine with 16 GB memory and a 2.6 GHz Intel Core i7 processor.

The following describes the metrics that are used in our evaluations and the results are also presented and discussed.

A. Silhouette Coefficient

Silhouette coefficient is used to test the separation between the clusters independently from the number of clusters. It is a prominent quality measure of how each point in a cluster is close (i.e. similar) to other points in the same cluster when comparing to other points in other clusters. The silhouette value ranges between -1 and 1. The higher value of the coefficient means a better structure for the clusters.

The silhouette coefficient is defined as in [18].

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \quad (4)$$

Where i is a data-point that is assigned to cluster a, $a(i)$ is the average dissimilarity of i to other data-points in the same cluster (i.e. a), and $b(i)$ is the lowest average dissimilarity of i to any other clusters (i.e. neighbouring cluster). The average value of $s(i)$ for all data-points is the measure of how data-points are clustered in the entire dataset.

We select different random samples from the dataset and calculate the silhouette coefficient to show how the initial value of λ (i.e. cluster penalty) is enough to cluster the data. We compare silhouette coefficient for both k-means and modified DP-means as shown in Figure 2. For k-means, we then use the same number of cluster that is obtained using modified DP-means algorithm to have a common base for comparison (unlike DP-means, the k-means algorithm requires the number of clusters in advance).

The modified DP-means has a minimum silhouette value of 0.79 with different sample sizes which shows the closeness of assigned values into the clusters. DP-means slightly outperforms k-means algorithm in our case. Initialising the threshold penalty parameter λ value by the mean of the standard deviation of data-points (resource locations) (X, Y, Z) provides more robust results. The common approach in using k-means is to run the k-means algorithm with a different number of cluster values (i.e. k) until adequate silhouette value is obtained. Moreover, Figure 3

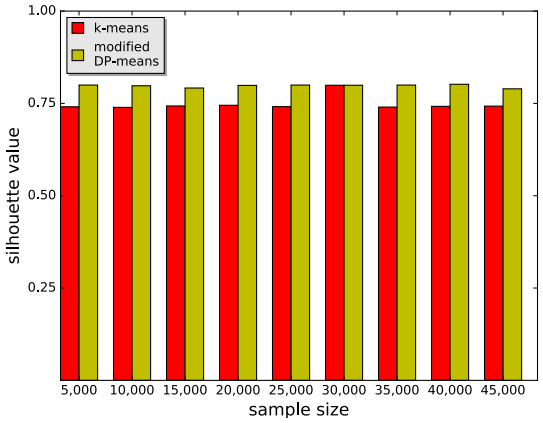

Figure 2. Silhouette analysis for k-means and modified DP-means

shows how $10,000$ data-points are clustered into 6 clusters. X, Y, Z are Cartesian co-ordinate values in million (in metres). It is worth mentioning that cluster k_1 represents resources whose locations are in [Canada, United States], cluster k_2 has resources with locations in [United Kingdom, Belgium], and cluster k_3 has all resources that are located in Australia. Also, cluster k_4 has all resources in Egypt, cluster k_5 has other resources that are located in Australia, and k_6 has all resources in Japan. It is noteworthy that all the clustering and grouping of the data-points are done automatically and without giving any prior knowledge about the countries or the number of clusters.

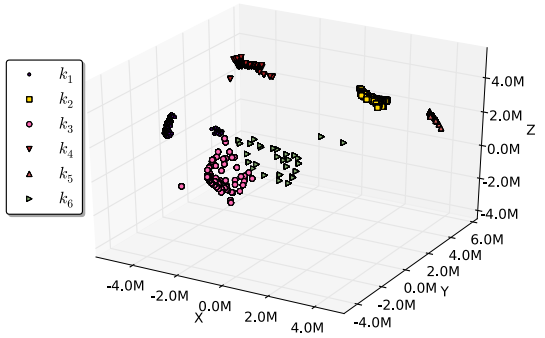

Figure 3. Clustering $10,000$ data-points by modified DP-means

B. Response Time

We define response time as the total amount of time (in seconds) that an indexing solution can take to respond to requested queries. We measure the response time of our proposed (semi-distributed, fully-distributed) and the baseline centralised mechanisms. Figure 4 shows the comparison of the response time between baseline centralised and our proposed distributed indexing. It is shown that the proposed distributed indexing is efficient and takes less time to find responses for a set of queries comparing to the centralised baseline. Our proposed approach is able to efficiently index and enable discovery of the IoT data with nearly 71% to 92% better response using semi-distributed and fully-distributed indexing scheme, respectively, compared to the centralised approach. This shows how the data are well distributed on a set of gateways that allow fast search for resources. Comparing semi-distributed and fully-distributed approaches, it is evident that indexing resource type within each cluster (gateway) in fully-distributed approach has a high impact on reducing the response time by nearly 72.4% comparing to the time the semi-distributed approach takes to answer queries.

C. Success Rate

The proposed indexing and discovery mechanism supports exact search queries (i.e. queries for exact locations and types) that already exist in our dataset. Evaluating the success rate of the proposed indexing mechanism is done by evaluating the number of attempts that is required to find a gateway that can answer the query. DS receives a set of queries and by using the aggregation mechanism it locates a set of related gateways that might have a resource that provides a response to the requested query. The gateways connected to this DS are queried according to their probability to having a resource that could respond to the query. If the first attempt is not successful, DS at the upper layer will forward the query to the second most

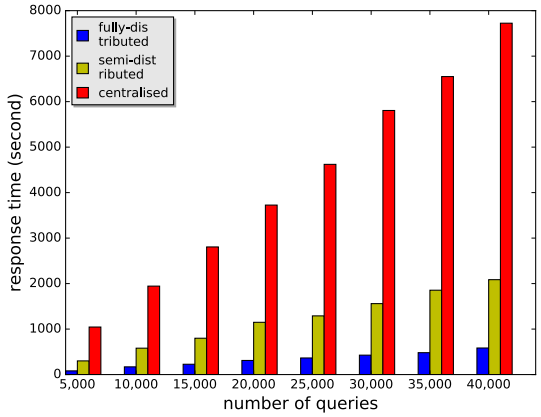

Figure 4. Comparison between response time of baseline centralised and our proposed distributed indexing schemes

probable gateway at the lower layer. This process continues until it terminates either by finding the result or reaching to a predefined value for the maximum number of attempts.

Although centralised approach guarantees answering the query by sequential search, our proposed indexing provides the same answer from the first selected gateway while minimising the response time. This confirms that selecting the gateway based on the minimum distance between requested location and cluster centroids works properly in reducing the search space to answer the queries.

V. Conclusions and Future Work

Dynamic IoT networks demand novel indexing and data access and discovery solutions for large-scale streaming data which are often numerical and multi-modal. We propose a distributed indexing mechanism that is capable of answering user queries for finding IoT data resource based on thematic and spatial attributes. The main advantage of our approach is that it provides an efficient mechanism to distribute and query the connected resources on different gateways with high success rate and less overhead. While increasing the number of queries, the proposed semi-distributed and fully-distributed schemes scale well and provide respectively 71% and 92% better response time than a centralised approach. Our current implementation mainly supports exact search queries (i.e. exact locations and types). A future extension of this work will focus on experiments to evaluate the impact of a highly dynamic network (e.g. node mobility, unreliable node connection/disconnection) on the indexing and discovery performance. The work will also focus on supporting approximated queries. However, we firstly need to check if the requested location is within our co-ordinates range. A possible solution is to check if $\lambda > d_m$, where d_m is the maximum distance between cluster centroids and the requested query location. This will allow finding approximate relevance for location based queries where there is no exact match for the queried location to obtain the data. In future work, we will develop a crawler to find the existing resources and will provide open API for third parties to find and/or add existing IoT data resources and including their links in our indexes. This work can change the way the IoT resources are currently accessed and used. It will have the same effect that Web search engines had on the extensions and updates of the Web and Web-based applications/services. Our indexing mechanism will allow IoT networks to work and operate in large ecosystems and connect closed systems. This will enable creating new data-driven applications for crowd-sourced IoT, smart cities, healthcare and in general will create data discovery infrastructure for the next generation of IoT networks.

Acknowledgment

This work is supported by the European Commissions Seventh Framework Framework Programme (EU FP7) in the CityPulse project (http://www.ict-citypulse.eu/) under contract number: 609035.

References

[1] P. Barnaghi, W. Wang, L. Dong, and C. Wang, "A Linked-Data Model for Semantic Sensor Streams," in *Green Computing and Communications (GreenCom)*. IEEE, 2013, pp. 468–475.

[2] Y. Diao, D. Ganesan, G. Mathur, and P. J. Shenoy, "Rethinking Data Management for Storage-centric Sensor Networks." in *CIDR*, vol. 7, 2007, pp. 22–31.

[3] 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," *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 223–235, 2010.

[4] 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 *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. ACM, 2002, pp. 78–87.

[5] B. Greenstein, S. Ratnasamy, S. Shenker, R. Govindan, and D. Estrin, "DIFS: A distributed index for features in sensor networks," *Ad Hoc Networks*, vol. 1, no. 2, pp. 333–349, 2003.

[6] M. Demirbas and X. Lu, "Distributed quad-tree for spatial querying in wireless sensor networks," in *IEEE International Conference on Communications, ICC*. IEEE, 2007, pp. 3325–3332.

[7] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 63–75.

[8] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4. ACM, 2004, pp. 353–366.

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

[10] P. Liu, N. Kong, Y. Tian, X. Lee, and B. Yan, "A DHT-Based Discovery Service for RFID Network," in *Internet of Things (iThings), Cyber, Physical and Social Computing (CPSCom)*. IEEE, 2014, pp. 344–347.

[11] 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)*. IEEE, 2010, pp. 237–244.

[12] E. Polytarchos, S. Eliakis, D. Bochtis, and K. Pramatari, "Evaluating discovery services architectures in the context of the internet of things," in *Unique Radio Innovation for the 21st Century*. Springer, 2011, pp. 203–227.

[13] 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 *IEEE 14th International Conference onMobile Data Management (MDM)*, vol. 1. IEEE, 2013, pp. 314–322.

[14] F. Paganelli and D. Parlanti, "A DHT-based discovery service for the Internet of Things," *Journal of Computer Networks and Communications*, vol. 2012, 2012.

[15] D. Zeng, S. Guo, and Z. Cheng, "The web of things: A survey," *Journal of Communications*, vol. 6, no. 6, pp. 424–438, 2011.

[16] B. L. Decker, "World geodetic system 1984," DTIC Document, Tech. Rep., 1986.

[17] B. Kulis and M. I. Jordan, "Revisiting k-means: New Algorithms via Bayesian Nonparametrics," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, J. Langford and J. Pineau, Eds. ACM, 2012, pp. 513–520.

[18] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.