Distributed Spatial Indexing for the Internet of Things Data Management

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

***Abstract*—The Internet of Things (IoT) has become a new enabler for collecting real-world observation and measurement data from the physical world. The IoT allows objects with sensing and network capabilities (i.e. Things and devices) to communicate with one another and with other resources (e.g. services) on the digital world. The heterogeneity, dynamicity and ad-hoc nature of underlying data, and services published by most of IoT resources make accessing and processing the data and services a challenging task. The IoT demands distributed, scalable, and efficient indexing solutions for large-scale distributed IoT networks. We describe a novel distributed indexing approach for IoT resources and their published data. The index structure is constructed by encoding the locations of IoT resources into geohashes and then building a quadtree on the minimum bounding box of the geohash representations. This allows to aggregate resources with similar geohashes and reduce the size of the index. We have evaluated our proposed solution on a large-scale dataset and our results show that the proposed approach can efficiently index and enable discovery of the IoT resources with 65% better response time than a centralised approach and with a high success rate (around 90% in the first few attempts).**

I. Introduction

Linear or sequential scanning for the entire connected resources in order to find the appropriate resource that might have an answer for a query is a prohibitively expensive task [1]. The current information access and retrieval methods for the Web and the Internet are often designed by exploitation of pre-defined links between different documents and resources that are archived on the Web servers. For instance, the Web search engine mainly works with textual data; it is based on keyword/text search approach [2]. However, IoT resources are distributed and often are ad-hoc in uncoordinated networks. IoT requires efficient, distributed and scalable indexing solutions that assist extracting information and gaining insights from underlying resources to respond to requests from higher-level applications and services.

Most of existing indexing approaches rely on centralised indexing or pre-defined links between resources [3, 4] which make them not scalable for a large number of connected resources in distributed environments. Other approaches are distributed such as [5, 6]. However, they are complex or do not support efficient updating for indexing structure once the indexes are constructed [7] which make them not applicable for wide-scale deployment of IoT.

The problem of indexing IoT data and resources comprises two key elements: building a distributed and efficient indexing and developing a query processing scheme. IoT indexing structure should be adaptive and dynamic because it is expected that a myriad of data streams will be processed and analysed at a higher rate than receiving user queries [8]. The processing scheme utilises the indexing structure to define the region and find the requested data given a requested query. The query is often composed of a set of key attributes such as a location (e.g. longitude and latitude) and a type (e.g. environmental: temperature and humidity). Queries can be answered using either exact or approximate search. The exact search is to find a response that matches exactly the given key attributes while approximate search gets the closest match of the requested query in case there is no exact value [9].

Our main contribution is a novel distributed and efficient indexing mechanism that allows indexing and discovery of IoT resources in distributed environments. The proposed solution provides efficient results with large dataset and is capable to be updated efficiently. It also supports type/location and approximate location-based queries.

The remainder of this paper is structured as follows: Section II discusses the background and related work. Section III details and discusses the proposed approach of spatial indexing. Evaluation and experimental results against baseline approach are discussed and analysed in Section IV. The baseline and evaluation metrics are also detailed in the same section. The conclusions, possible extensions and future work are discussed in Section V.

II. Background and Related Work

To allow efficient access and retrieval for data and services published by distributed IoT resources, a distributed indexing structure (a way of sorting and arranging) of these resources must be constructed. Answering queries utilises the constructed index to efficiently find the region and the location of the resources that should be accessed. Different types of indexing methods and solutions have been proposed such as [5, 10, 11, 12]. Ratnasamy et al. [13] propose Geographic Hash Table (GHT). GHT is a distributed indexing that is constructed by hashing types of events as keys and geographical coordinates as values. However, GHT detects only presence/absence of an event. Distributed Index for Features in Sensor Networks (DIFS) [5] extends GHT to support range queries (query key attributes are desired to be within a certain range of values). Indexes are constructed in a tree-based structure, in which each node stores information about a range of values within a geographical area. Unlike traditional tree structures (e.g. binary tree, quadtree), non-root nodes can have several parents. However, the indexing structure allows child nodes to have

parents in different geographic areas. Also, constructing and updating indexes are computationally expensive to answer complex queries of different range of values [14, 15].

Wang et al. [11] introduce R-Tree-based spatial indexing approach to index Minimum Bounding Rectangles (MBRs). Each node in the tree is associated with MBR that represents the location range of its child nodes. However, R-tree is not scalable if there are frequent changes in MBR [16]. To address this problem, the authors index gateways in which each of them is associated with MBR (the location range of its registered services). Answering queries is based on selecting gateways whose MBRs intersect with a given rectangular query. Then, a semantic matching between the description of the type of services that are registered to the selected gateways and the queried type is applied. However, the approach does not consider that the gateways might have the same type of services. Also, the matching approach between given queries and the description of the gateways is not efficient to answer all queries.

Using traditional tree structures is computationally expensive in dynamic environments where there are frequent updates. To this end, Fox et al. [12] propose spatio-temporal indexing on top of Apache Accumulo[1] which is a distributed storage for (sorted) key-value pairs. The spatial location of each record in the dataset is encoded using geohash[2] and is used as a part of Accumulo key. However, the approach allows querying data without identifying its data sources.

A Distributed Hash Table (DHT)-based network for discovery service is another approach to retrieve the service (node) that has an answer for a requested query [17]. The connected nodes are distributed on the network, and each node is represented as a (key, value) pair. DHT efficiently finds an exact match (value) for a given key. To this end, Prefix Hash Tree (PHT) [18] is proposed to enable more complex queries on top of DHT. PHT is typically a binary tree. Each node in the tree structure is identified by a prefix (label). PHT requires traversing down to the leaf nodes where data is stored. However, if the data or a reference to the data is filled in the internal nodes, there will be no need to traverse the whole tree, and this will accelerate the query processing.

There are also solutions for discovery IoT data such as Dyser and SenseMap [19]. They rely on a centralised indexing mechanism which makes them not suitable for indexing resources in ad-hoc and more dynamic networks. Generally, scalability and/or effective updating for the indexes in dynamic IoT environments are the main drawbacks of these solutions. The state-of-the-art of indexing is therefore not capable of dealing with the characteristics of IoT resources in distributed and dynamic environments.

We propose a novel distributed geohash-based indexing mechanism that allows discovery of IoT resources. The index structure is constructed by encoding the spatial features of IoT resources (i.e. locations) into geohashes. A quadtree is then built on the minimum bounding box of the geohashes. This allows shrinking the size of the index by aggregating resources with similar geohashes. It is worth noting that wireless related issues (e.g. transfer delay) are not within the scope of the current work.

[1]http://accumulo.apache.org/

[2]http://www.geohash.org/

III. Proposed Approach

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

(i) IoT resources publish their data and services.
(ii) IoT data resource has often a type (e.g. temperature, wind direction, relative humidity and wind speed).
(iii) Spatial features (i.e. longitude and latitude) of IoT resources are represented as geohash codes.
(iv) The query is often composed of type, location (spatial), and time (temporal) attributes.
(v) Published data can be archived in Information Repositories (IRs) or accessed directly from the resources.
(vi) Distributed indexing identifies the region and the location of requested resources given a requested query.
(vii) Discovery Services (DSs) receive user queries and DS either accesses the data from IR if the location of the data is known (e.g. already cached) or forwards it to the distributed indexing.

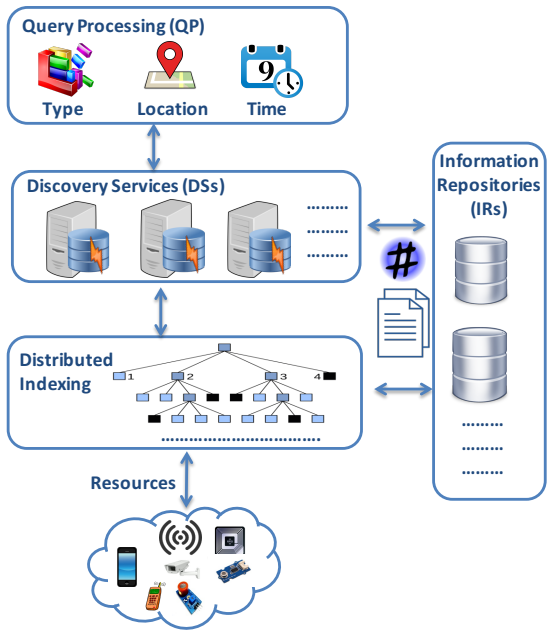

Figure 1. A higher-level architecture for distributed indexing

A. Spatial indexing

Building indexing structure for single-dimensional data is a straightforward task. However, indexing multi-dimensional data is not. Many techniques have been proposed to map multi-dimensional spatial data into single-dimensional data

where locality is preserved between data-points such as space-filling curves [20], e.g. Z-order space-filling curve (often called Morton order) [21]. Space-filling fundamentally splits a space or a plane (e.g. the globe) into (square) regions and then a line gets squiggled in a specific order (forming Z or U shape) by traversing iteratively from one square to another until it fully fills the two-dimensional space. Other approaches are based on MBR such as R-tree [22], kd-tree (k dimensional tree) [23] and quadtree [24]. In the tree structure, each non-root node in the tree structure represents MBR that covers the total range of locations (MBRs) of its child nodes, while the root node covers the entire area of the tree. R-tree and quad-tree have been used extensively in different indexing mechanisms [25]. R-tree has been recommended as indexing structure in Oracle spatial database [26]. Although R-tree is more efficient for querying data than quadtree in Oracle spatial database, quadtree has relatively low complexity in updating indexing structure [25]. In addition, R-tree might overlap between rectangles. In this paper, we utilise quadtree to construct spatial indexing based on geohash codes.

The proposed approach in this paper is based on using geohash to represent spatial coordinates due to its ability to represent the two-dimensional spatial coordinates (longitude and latitude) such as $(-118.0, 33.0)$ into a one-dimensional (encoded) string (9mgev7w7z8j7). Latitude and longitude have been commonly used as a pair of coordinates to represent a location in the globe. Longitude has a value in the range of $[-180°, 180°]$, while latitude has a value in the range of $[-90°, 90°]$. Geohashes are generated in Z (Morton order). Morton order is based on interleaving bits of the binary representation of spatial coordinate values (longitude, latitude). Interleaving bits is performed by converting the spatial coordinate (longitude, latitude) values into binary and then alternating bits from each coordinate. Interleaving the bits results in one binary string representation for a given spatial coordinates in which one coordinate goes to odd bits and another coordinate goes to even bits. The resultant binary representation is then encoded into a set of characters. Overall, geohash is a hierarchical structure that divides the globe recursively into bounding boxes until the required resolution/precision is achieved (see Figure 2 for 32-bit resolution). For example, if a geohash is represented by 32-bit precision, 16-bits for each coordinate should be interleaved. It is worth noting that longer geohash represents more precise location. The main advantage of using geohash is its locality where adjacent locations share the similar prefix. This simplifies searching for spatial locations by matching their prefix.

Geohashes can be indexed without applying any spatial indexing technique. One would think that querying all nearest neighbours of a given geohash over the worldwide longitude-latitude rectangles can be simple; it is easy to extract the neighbouring bounding boxes around a particular region (query region) in each direction (i.e. North, South, East, and West). However, if there is a large number of IoT resources that are deployed in different locations that do not cover the entire globe, getting all nearest neighbours that cover a query region

Figure 2. Earth's surface is split into 32 blocks

is a quite complex task. It is a necessity to find the smallest prefix of geohash that can cover a query region and get all the geohashes that have the same prefix (where their bounding boxes intersect with a query region). To this end, we build a quadtree [24] of geohashes that allows finding all nearest neighbour geohash ranges and can be extended to cover the entire globe.

A quadtree is a tree structure whose non-leaf nodes have exactly four child nodes (called quadrants). A quadtree is constructed using the divide and conquer strategy in which the root node represents the entire spatial range (longitude has a range of $[-180°, 180°]$, latitude has a range of $[-90°, 90°]$.). The space is recursively subdivided into four quadrants (see Figure 3). Figure 4 shows an example of how quadtree can recursively subdivide a spacial space.

We build our quadtree based on geohash codes. The original quadtree has been updated and extended for spatial indexing. Initially, the root of the tree has a bounding box that covers the entire globe $[-180, 180]$ and $[-90, 90]$ (the range of worldwide longitude and latitude). Each node in the quadtree is represented in the form of *(Identifier, bbox)*, where the identifier is a geohash and the bbox is the MBR of a given geohash. The MBR of a given geohash is obtained by reducing the precision (resolution) of the geohash. Duplicate identifiers are not allowed in the tree. To find a region query, the algorithm starts from the root and examines the MBR of each child nodes and checks if its MBR intersects with the region being queried. If it does, the algorithm examines each child node of each selected nodes whose their MBRs intersect with the query region until the leaf node is reached. The algorithm returns a sorted list of identifiers (geohashes) of each MBR that intersects with the area being queried.

As mentioned earlier, the query is often composed of location (geohash or area of interest) and type (e.g. temperature) attributes. We have explained how to get the areas of spatial coordinates that can intersect with the area being queried. Instead of examining all resources whose location is in one of the selected MBRs, another tree structure which has n child nodes, where n is the number of types is constructed

per MBR. Each child nodes contains a reference to a resource (to connect to a resource and get its real-time observation and measurement data) and a path to a repository (i.e. IR) where the data values are stored given the location and type key attributes. The main advantages of our approach comparing to using R-tree with geohash in [11] is that we identify the MBR of each geohash in a simple way taking into account that the indexing is extendible to cover the entire globe (longitude has a range of $[-180°, 180°]$ and latitude has a range of $[-90°, 90°]$). This eliminates the necessity of frequent updates for the indexing structures and allows to discover the resources based on their types and locations. It is worth mentioning that MBR's coordinates (i.e. bottom-left and top-right coordinates) for a given geohash are calculated by obtaining the maximum and minimum longitude and latitude that are corresponding to (west longitude, south latitude) and (east longitude, north latitude) coordinates.

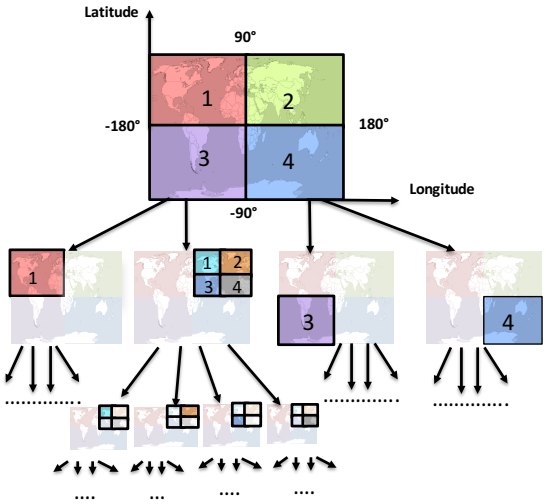

Figure 3. How quadtree can be structured

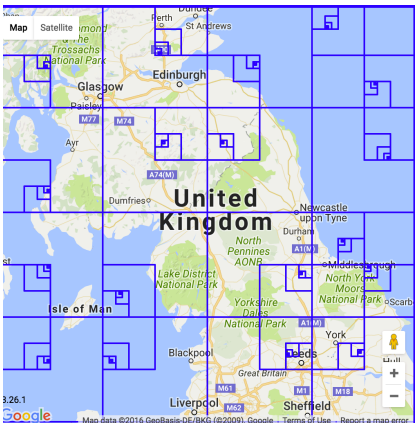

Figure 4. An example of how a spatial space is structured using quadtree

B. Real-world Dataset

We use Automated Surface Observing System (ASOS)[3] weather dataset. The weather sensory data is gathered from different weather stations that are located in different countries. We are currently using a dataset with roughly 5.5 million resources from 7 countries; United Kingdom, Belgium, Japan, Canada, Australia, Egypt, and the United States. Each resource in the dataset has the following attributes; a source identifier, geographical co-ordinates longitude (e.g. -1.32) and latitude (e.g. 51.8369), country (e.g. UK), station-name (e.g. Oxford) and time-stamp (e.g. 2014-07-02 07:50:00) on which the value is obtained. The resource has also a service type such as relative humidity (e.g. 63.82%), wind direction in degrees from north (e.g. 220.0), wind speed in knots (e.g. 4.0), and dew point temperature in Fahrenheit (e.g. 53.6). Dew point is used to measure atmospheric moisture.

We remove the invalid values based on the meta-data of the dataset and apply interpolation to fill missing values. We also convert each pair of (longitude, latitude) into a geohash code (e.g. gcpncuwgcf89).

IV. Evaluation

As mentioned earlier in Section II, most of the existing approaches and solutions are based on centralised indexing mechanisms. We use a centralised approach as our baseline. In the centralised approach, a single centralised index of all IoT resources is constructed where there no identification for the region and/or the location of requested resources given a required query. In this case, the centralised indexing has to scan all data resources to respond to user queries. The queries are randomly generated. The key element for evaluating our approach is to find successfully a list of nearby MBRs that intersect with the region being queried and that contain a resource that might have the requested attributes.

Both of the proposed and centralised solutions are tested under the same conditions and implemented in Python. We apply both solutions on the same dataset and 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. Response Time

The response time is defined as the total amount of time that an indexing scheme takes to answer queries. We measure the response time of both our proposed distributed indexing and the baseline method. Figure 5 presents the comparison between the two schemes regarding response time. It is evident that the proposed distributed indexing is more efficient than the centralised and requires less time for all sets of different queries. The distributed approach enables efficient indexing and query with nearly 65% better response time comparing to the centralised baseline approach. This demonstrates how the data resources are well distributed in our hierarchical tree structure.

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

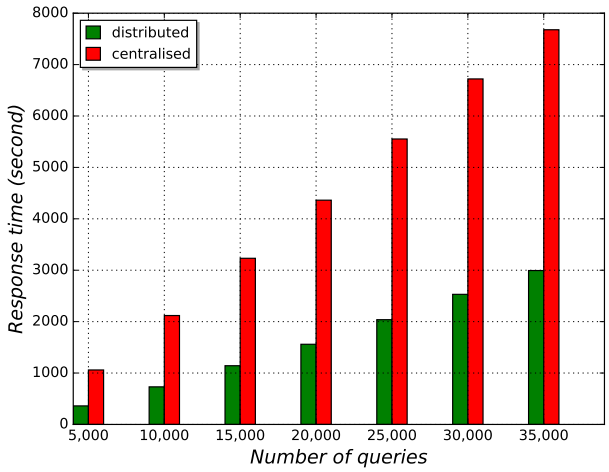

Figure 5. Comparison between response time of our proposed distributed indexing and baseline centralised schemes

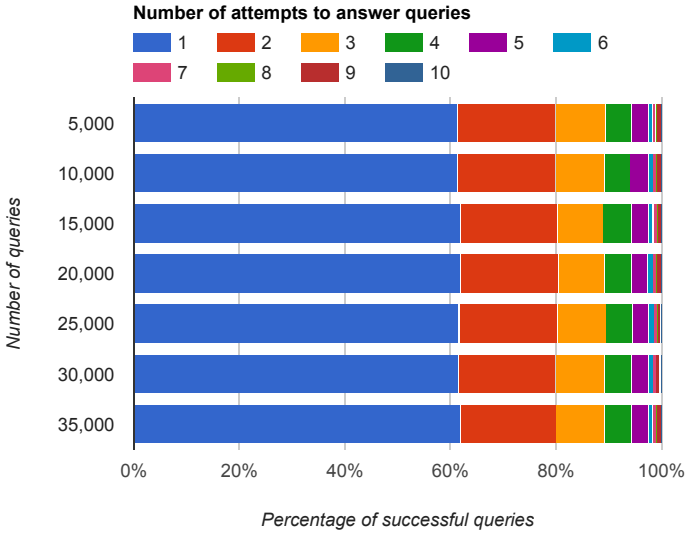

Figure 6. Ratio of success rate to answer queries

B. Success Rate

A key advantage of our proposed indexing and discovery mechanism is that it supports approximate queries (i.e. queries for approximate locations and exact type). We have evaluated the success rate by evaluating the number of attempts that is required to find a response to a query. DS receives a set of queries whose key attributes are a location (e.g. longitude, latitude) and a type (e.g. humidity). DS forwards the query to the distributed indexing (see Figure 1). Distributed indexing gets a list of nearby MBRs that intersect with the region being queried. However, there is a probability that the list does not contain the queried type (humidity). If the first attempt is not successful; the selected region does not have the queried type, a second region in the list should be examined. The process continues either by finding the region that has a resource that publishes the same (queried) type or by reaching the maximum number of (predefined) attempts. Figure 6 shows the number of attempts that are required to answer a different set of queries. The range of numbers [1, 10] represents the number of attempts (maximum number of attempts is 10) to answer a set of queries. It is evident that the 60% of queries can be answered in the first attempt, while 20% can be answered by the second attempt. Overall, the majority of queries (90%) can be answered by the first three attempts. It is worth noting that all sets of different queries can be successfully answered by reaching the maximum number of (predefined) attempts.

Although the centralised approach guarantees answering the query by sequential search for the dataset, our proposed indexing provides the same answer and requires less time to find responses for a set of queries. Our approach can successfully answer all sets of different queries comparing to the proposed solution in [11] which can not find responses to all requested queries (this work is discussed in Section II).

With regards to the other similar works in this area, recent experiments that have been conducted in [11] using R-tree with geohash can answer successfully roughly 712 queries (12%) of the total number of requested queries on a dataset with a size of $10,000$ resources. However, our approach can answer successfully roughly 60% of queries in the first attempt and overall, 90% of requested queries (see Figure 6) are answered in the first few attempts on a dataset with a size of 5.5 million resources.

V. Conclusions and Future Work

IoT applications often require fast access, and retrieval methods for very large smart connected resources and their underlying data and services in IoT distributed environments. The high- volume, velocity and variety of IoT resources and data add more constraints and complexity to discovery and access methods compared with the conventional Web access/discovery solutions. Large-scale IoT distributed environments require novel distributed, efficient and reliable access and discovery solutions and methods that are able to find the right resource at the right time that might have a response to a request from higher-level applications and services. In this paper, we propose a higher-level architecture for distributed indexing and develop and implement a distributed indexing mechanism that is capable of defining the region and finding the requested data given a requested query (based on location and type) with a high success rate and with 65% better response time than a centralised approach. We also discuss how our approach outperforms recent experiments conducted on indexing mechanism based on using other similar geohash and tree structure solutions.

Our current implementation considers a finite set of types (e.g. temperature, humidity, wind speed). A future extension will focus on how the indexing method can fold-in with new type of data (e.g. air pollution) that is not predefined and can include complex data types which can be a combination of various raw data types. We will also examine the effect of

different parameters of the tree structure (e.g. depth and a maximum number of items per quad node) on the indexing scheme. We will also consider developing crawlers and also providing open APIs for third parties to find and/or add existing IoT data resources and their links in our indexing mechanism. This will also require to build a mechanism to rank resources if there are resources with the same type of service (e.g. temperature) in a selected region.

ACKNOWLEDGMENT

This work is supported by the European Commissions Seventh Framework Programme (EU FP7) for the CityPulse project (http://www.ict-citypulse.eu/) under contract number: 609035 and the EU H2020 for the FIESTA-IoT project (http://fiesta-iot.eu/) under contract number: 643943.

REFERENCES

[1] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems Journal (KAIS)*, 3(3):263–286, 2001.

[2] Luiz André Barroso, Jeffrey Dean, and Urs Holzle. Web search for a planet: The Google cluster architecture. *IEEE Micro*, 23 (2):22–28, 2003.

[3] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, and Deborah Estrin. Data-centric storage in sensornets. *ACM SIGCOMM Computer Communication Review*, 33 (1):137–142, 2003.

[4] Yanlei Diao, Deepak Ganesan, Gaurav Mathur, and Prashant J Shenoy. Rethinking Data Management for Storage-centric Sensor Networks. In *Proceedings of the 3nd Biennial Conference on Innovative Data Systems Research (CIDR)*, volume 7, pages 22–31, 2007.

[5] Benjamin Greenstein, Sylvia Ratnasamy, Scott Shenker, Ramesh Govindan, and Deborah Estrin. DIFS: A distributed index for features in sensor networks. *Ad Hoc Networks*, 1(2): 333–349, 2003.

[6] Charith Perera, Arkady Zaslavsky, Chi Harold Liu, Michael Compton, Peter Christen, and Dimitrios Georgakopoulos. Sensor search techniques for sensing as a service architecture for the internet of things. *IEEE Sensors Journal*, 14(2):406–420, 2014.

[7] Beomseok Nam and Alan Sussman. Analyzing design choices for distributed multidimensional indexing. *The Journal of Supercomputing*, 59(3):1552–1576, 2012.

[8] Benedikt Ostermaier, Kay Romer, Friedemann Mattern, Michael Fahrmair, and Wolfgang Kellerer. A real-time search engine for the web of things. In *Internet of Things (IoT), 2010*, pages 1–8. IEEE, 2010.

[9] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining (SDM05)*, pages 506–510. SIAM, 2005.

[10] Xin Li, Young Jin Kim, Ramesh Govindan, and Wei Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 63–75. ACM, 2003.

[11] Wei Wang, Suparna De, Gilbert Cassar, and Klaus Moessner. An experimental study on geospatial indexing for sensor service discovery. *Expert Systems with Applications*, 42(7):3528–3538, 2015.

[12] Anthony Fox, Chris Eichelberger, James Hughes, and Skylar Lyon. Spatio-temporal indexing in non-relational distributed databases. In *The 2013 IEEE International Conference on Big Data*, pages 291–299. IEEE, 2013.

[13] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: a geographic hash table for data-centric storage. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 78–87. ACM, 2002.

[14] Murat Demirbas and Xuming Lu. Distributed quad-tree for spatial querying in wireless sensor networks. In *2007 IEEE International Conference on Communications*, pages 3325–3332. IEEE, 2007.

[15] Y. Fathy, P. Barnaghi, S. Enshaeifar, and R. Tafazolli. A distributed in-network indexing mechanism for the internet of things. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 585–590, 2016.

[16] Yuchao Zhou, Suparna De, Wei Wang, and Klaus Moessner. Search Techniques for the Web of Things: A Taxonomy and Survey. *Sensors*, 16(5):600, 2016.

[17] Federica Paganelli and David Parlanti. A DHT-based discovery service for the Internet of Things. *Journal of Computer Networks and Communications*, 2012, 2012.

[18] Sriram Ramabhadran, Sylvia Ratnasamy, Joseph M Hellerstein, and Scott Shenker. Prefix hash tree: An indexing data structure over distributed hash tables. In *Proceedings of the 23rd ACM symposium on principles of distributed computing*, volume 37, 2004.

[19] Deze Zeng, Song Guo, and Zixue Cheng. The web of things: A survey. *Journal of Communications*, 6(6):424–438, 2011.

[20] Jonathan K Lawder and Peter JH King. Using space-filling curves for multi-dimensional indexing. In *British National Conference on Databases*, pages 20–35. Springer, 2000.

[21] Guy M Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company New York, 1966.

[22] Antonin Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14 of *SIGMOD '84*. ACM, 1984.

[23] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9): 509–517, 1975.

[24] Raphael A. Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4 (1):1–9, 1974.

[25] Maruto Masserie Sardadi, M Rahim, Zahabidin Jupri, and DB Daman. Choosing R-tree or quadtree spatial data indexing in one oracle spatial database system to make faster showing geographical map in mobile geographical information system technology. *World Academy of Science Engineering and Technology*, 46:249–257, 2008.

[26] Ravi Kanth V Kothuri, Siva Ravada, and Daniel Abugov. Quadtree and R-tree indexes in oracle spatial: a comparison using GIS data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 546–557. ACM, 2002.