Accepted Manuscript

An Experimental Study on Geospatial Indexing for Sensor Service Discovery

Wei Wang, Suparna De, Gilbert Cassar, Klaus Moessner

PII: S0957-4174(14)00757-X

DOI: <http://dx.doi.org/10.1016/j.eswa.2014.11.058>

Reference: ESWA 9713

To appear in: *Expert Systems with Applications*

Please cite this article as: Wang, W., De, S., Cassar, G., Moessner, K., An Experimental Study on Geospatial Indexing for Sensor Service Discovery, *Expert Systems with Applications* (2014), doi: <http://dx.doi.org/10.1016/j.eswa.2014.11.058>

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

An Experimental Study on Geospatial Indexing for Sensor Service Discovery

Wei Wang

*Department of Computer Science and Software Engineering,
Xi'an Jiaotong Liverpool University
Email: wei.wang03@xjtlu.edu.cn*

Suparna De

*Institute for Communication Systems,
University of Surrey
Email: s.de@surrey.ac.uk*

Gilbert Cassar

*UReason, Maidenhead, UK
Email: gcassar@ureason.com*

Klaus Moessner

*Institute for Communication Systems,
University of Surrey
Email: k.moessner@surrey.ac.uk*

Abstract

The Internet of Things enables human beings to better interact with and understand their surrounding environments by extending computational capabilities to the physical world. A critical driving force behind this is the rapid development and wide deployment of wireless sensor networks, which continuously produce a large amount of real-world data for many application domains. Similar to many other large-scale distributed technologies, interoperability and scalability are the prominent and persistent challenges. The proposal of sensor-as-a-service aims to address these challenges; however, to our knowledge, there are no concrete implementations of techniques to support the idea, in particular, large-scale, distributed sensor service discovery. Based on the distinctive characteristics of the sensor services, we develop a scalable discovery architecture using geospatial indexing techniques and semantic service technologies. We per-

form extensive experimental studies to verify the performance of the proposed method and its applicability to large-scale, distributed sensor service discovery.

Keywords: Sensor Services, Sensor Service Discovery, Geospatial Indexing, R-Tree, Semantic Search

1. Introduction

The Internet of Things (IoT) is seen as the most promising multidisciplinary efforts that will help human beings live better lives by enabling many intelligent applications, e.g., smart office/home/city, remote healthcare, autonomous traffic control, emergency response, sustainable urban planning, and effective environmental monitoring and protection, to name a few. One of the driving forces behind this is that embedded devices (e.g., wireless sensors) with ever-increasing computation power and communication capabilities can be manufactured and deployed in large-scale while at low cost. The explosive development of the IoT in recent years has the potential to add millions or even billions of sensors to the future Internet. According to Cisco IBSG, there will be 25 billion devices connected to the Internet by 2015 and 50 billion by 2020 (Evans, D., 2011). Similar to large-scale deployment of many other distributed technologies, interoperability and scalability are the persistent challenges for wireless sensor networks, which inevitably make efficient sensor discovery and sensor data access difficult.

To enable interoperability, the IoT community proposes to use semantic technologies for describing and annotating sensors and entities of interest. This has the potential to enable representing, storing, interconnecting, searching and organising information related to or generated by heterogeneous things (Atzori et al., 2010) and to facilitate the creation of “Semantic Sensor Web” (Sheth et al., 2008). Some of the notable works in this line include the Semantic Sensor Network (SSN) ontology for sensor knowledge representation (Compton et al., 2012) developed by the W3C Semantic Sensor Networks Incubator Group; the ontology framework developed in (Roda & Musulin, 2014)

for supporting intelligent analysis of sensor measurement data; and the semantic service modelling for real world IoT resources (De et al., 2012). To enable scalability, researchers propose to apply service-oriented principles to the design of IoT infrastructure, which can facilitate the development of large-scale, loosely-coupled IoT based applications and services. The concept of sensor-as-a-service (Perera et al., 2014a) is important for creation of a service-oriented sensor Web (Gibbons et al., 2003), or more generally, a service-oriented IoT (Barnaghi et al., 2012). The idea is to abstract sensor functionalities and capabilities in terms of standard service interfaces and to support uniform service operations.

Following the sensor-as-a-service paradigm, if all sensors expose their functionalities as services, there will be numerous real-world services generating a large amount of streaming data continuously. This highlights the significance of efficient sensor service discovery, which aims to locating relevant and quality sensor services according to users' requirements. It should be noted that in the discovery process, the users (e.g., human users or applications) usually do not have exact knowledge on the actual sensor services (e.g., where they are located, what functionalities they provide or how they work). To this end, we share similar perspectives with Delicato *et al.*'s view on a flexible architecture with which sensor network data can be accessed by users spread all over the world (Delicato et al., 2005). The discovered services can be accessed in real-time and used for many purposes, for example, reading and aggregating sensor data (Wu et al., 2014; Stavropoulos et al., 2013), sensor data abstraction and analysis (Roda & Musulin, 2014), service composition and runtime adaptation (Coria et al., 2014).

Sensor services are significantly different from standard Web services, e.g., extremely large in number, location dependent, dynamic and unreliable. Finding a particular sensor service(s) from billions according to the search criteria can be a challenging task. Most of the existing techniques developed for standard Web service discovery are not directly applicable to sensor services, for instance, it is not possible to build a centralised index or portal for all sensor

services due to their large number and highly distributed nature; the location dependency implies that sensor services need to be organised in a way to allow efficient search based on spatial properties, e.g., region containment or overlap; the dynamic and unreliable nature of the sensor services implies that update operations may need to be performed frequently, which may introduce substantial overhead to the discovery platform.

Our work aims to overcome the identified difficulties and to design an efficient method for large-scale sensor service discovery based on geospatial indexing, semantic and service-oriented techniques. The design demonstrates the following features: (1) geospatial indexing to efficiently and effectively reduce search space; (2) resilience to dynamicity and reducing the number of expensive update operations; (3) more precise computation of the spatial properties of sensor service gateways using computational geometric techniques; and (4) distributed semantic repositories and semantic service matchmaking to provide more accurate results. The rest of the paper is organised as follows. In Section 2 we provide a review on some of the representative works for sensor service discovery. In Section 3 we first briefly present the background information relevant to sensor modelling and geospatial indexing, then we elaborate the design and implementation of the discovery platform, and how the identified problems are addressed. Section 4 presents the experiment and evaluation results, which are also compared to the ones generated by the benchmark methods. Section 5 concludes the paper and discusses the future work.

2. Related Work

Service discovery has been extensively studied in the literature, especially for services on the Web (including semantic Web) (Garofalakis et al., 2006; Coria et al., 2014; McIlraith et al., 2001) and in pervasive and mobile environments (Chakraborty et al., 2006; Nidd, 2001). The study in (Perera et al., 2013) identifies the similarities and differences between sensor search and Web service search, and argues that the scale of the IoT makes sensor discovery (as

the precursor to sensor selection and ranking) much more challenging. Recent research on extending the service-oriented principles to the IoT domain also shows that existing methods cannot be directly applied to the vast number of distributed services exposed by the networked, real-world resources on the
 90 IoT (Barnaghi et al., 2012; Wang et al., 2013; Wei & Jin, 2012; Perera et al., 2014b). Generally speaking, IoT services can be seen as a special class of Web services that have a number of distinctive characteristics, e.g., large-scale, highly distributed and dynamic, location-dependent and capability constrained.

Already there have been considerable research efforts from the IoT and
 95 service computing communities to address the identified issues. For example, the work in (Evdokimov et al., 2010; Guinard et al., 2010; Abangar et al., 2010) provides sensor based services using the Device Profile for Web Services (DPWS) (OASIS, 2009), a lightweight subset of Web service technologies that enables plug-and-play features for resource constrained devices. The work in
 100 (Guinard et al., 2010) develops a platform to facilitate discovery, selection and on-demand provisioning of real-world services for business applications. In addition to the DPWS based discovery method, the SOCRADES middleware in (Guinard et al., 2010) also implements a RESTful network discovery mechanism for devices. The work in the aWESoME middleware (Stavropoulos et al., 2013)
 105 also uses services to expose IoT devices and employs a service broker to implement the service discovery functionalities. The main limitation of the above discussed research is that with each service corresponding to a device type, the discovery methods require unique device IDs (typically MAC addresses) as input to distinguish between different devices of the same type. The authors
 110 in (Wei & Jin, 2012) highlight the resource-constrained and dynamic nature of the IoT and propose a context-aware framework for service discovery based on formal context modelling (Gu et al., 2004) and uncertain context modelling (Gu et al., 2005), in particular, their work considers the temporal dimension and applies probabilistic reasoning in dynamic Bayesian networks. However,
 115 the methods are mostly designed for environments with limited scopes (e.g., home or enterprise) and do not apply to the pervasive computing domains.

To support interoperability, recent research proposes the use of semantic Web technologies for discovery of sensor services and data in IoT environments (De & et al, 2012; De et al., 2014). The sensor data analysis framework in
 120 (Roda & Musulin, 2014) applies the Semantic Query Web Rule Language, an extension of the Semantic Web Rule Language, as the query engine language for knowledge extraction from sensor data. However, the work requires all sensor metadata to be available in a centralised registry and does not consider the location dependency characteristics of sensor deployment. The CASSARAM sensor
 125 search model (Perera et al., 2014b) utilises a weighted Euclidean distance based indexing technique to measure the similarity between the sensor description and the user requirements. A heuristic filtering and a relational expression based filtering methods are then applied to reduce the amount of metadata needed to be processed during the discovery. In contrast, our proposed method partitions
 130 the discovery space into much smaller ones based on geospatial index prior to query processing. This offline step allows the discovery to be performed in a more efficient way.

As sensor services are location dependent, geographical information plays a substantial role in discovery, especially in pervasive environments, e.g., wireless
 135 sensor networks or vehicular networks (Niforatos et al., 2012; Abrougui et al., 2012). The work in (Fredj et al., 2013) uses a hierarchy of nodes to represent indoor locations (e.g., room, building or floor). The nodes encapsulate semantic service descriptions of objects located within their geographic scope. However, the work does not consider the cost of maintaining the discovery platform which
 140 might be computationally expensive in a dynamic environment. The discovery approach in (Mayer et al., 2012) also structures nodes in a hierarchical fashion, with interactions restricted to direct communications between neighbouring nodes to support scalability. The search process can be either in the local node or by routing queries to neighbour ones. The IoT-A project proposes a resolution
 145 framework for the IoT, encompassing a number of location-based discovery mechanisms. One of these uses the R-Tree based spatial indexing technique to index sensor services (De & et al, 2012). The infrastructure consists of a

number of indexing servers that index the observation areas of individual IoT services and catalogue servers that store the scope of the indexing servers. During discovery, the top-level catalogue server is first contacted, which then uses the stored scope to identify the set of indexing servers. Another location-based approach proposes a federated architecture of geographically distributed and co-operating nodes with local reasoning and search capabilities to manage the large number of IoT devices (De et al., 2014). This approach, however, only applies to indoor environments with the support of semantic models describing logical locations. The OSIRIS sensor Web discovery framework (Jirka et al., 2009) annotates sensor instances according to the OGC Sensor Web Enablement (SWE) standards (Botts et al., 2007) and provides three different indexing mechanisms to improve the discovery performance: a spatial index, a temporal index (for temporal criteria in queries) and a full-text index (for sensor keyword search). However, how to efficiently maintain and update the services and the geospatial index in dynamic environments such as the sensor Web or the IoT, has not been adequately studied according to our investigation. Another interesting method is to use geohash¹ to describe the geospatial features of objects and conventional information retrieval indexing technique to enable efficient search, for example, in the Apache Solr project, any types of objects can be indexed based on their geohashes (Grainger & Potter, 2014). In this work, we perform a number of experiments to evaluate the performance of our method to the ones introduced in (De & et al, 2012; Grainger & Potter, 2014).

Our review shows that the existing works have recognised the importance of semantic modelling and geographical information for sensor or IoT service discovery (Stavropoulos et al., 2013; Roda & Musulin, 2014; De & et al, 2012; Liang & Huang, 2013; Fredj et al., 2013). However, they have not sufficiently addressed the challenges related to the extremely large number of services, and their distributed and dynamic nature. Furthermore, a more precise geometric descriptor for the geographical features of the objects is needed to construct the

¹<http://geohash.org>

spatial index. In the following sections, we elaborate the design and implementation of an efficient method which aims to address the identified problems.

3. Large Scale Sensor Service Discovery

180 Sensor services have a number of features that can be used for discovery, such as the sensor type (e.g., temperature or light), input, output and geographical location. The geographical feature can be used to effectively reduce the search space, while others can be exploited to perform accurate semantic search within the reduced search space. With this consideration, we build the
185 discovery architecture using geospatial indexing techniques and distributed semantic repositories. In Section 3.1, we first give an overview on the proposed sensor service discovery architecture. In Section 3.2 we present the semantic modelling for sensor services and the distributed semantic repository implementation. Then in Section 3.3 we show how to construct a geospatial index by
190 using a precise descriptor for the geographical features of the service gateways to enable accurate discovery. In particular, we discuss how the approach can address the dynamicity problem, ease the index maintenance and reduce computational complexity. In Section 3.4, we highlight the advantages of the proposed discovery method.

195 3.1. Overview

Figure 1 provides a functional view of the service discovery architecture. This work focuses on two functional components: distributed gateway and discovery server.

200 In the distributed gateway component, functionalities of the sensors are abstracted as services and annotated according to the semantic model (see Section 3.2 for more details). Besides managing a wireless sensor network and its communications with the backbone network, a gateway provides resources to host the sensor services. In addition, a semantic repository which stores service descriptions and a semantic search interface are also implemented. It should be

Figure 1: The proposed architecture for large-scale sensor service discovery

205 noted that the gateways and wireless sensor networks are distributed in different geographical locations. In the discovery server, the geospatial index helps the discovery engine reduce the search space and locate the gateway(s) that are likely to contain the services with respect to the queries. The current work only focuses on the service discovery, ranking of the sensor services is discussed in
 210 Section 5.

3.2. Semantic Modelling and Search

The use of semantic technologies as the fundamental knowledge representation framework has been motivated by their ability to create homogeneous, standardised and machine-processable data. A well developed semantic model
 215 is required to describe the features of sensor services and related entities of interest and to link them based on the semantic links. The descriptions are stored in the distributed semantic repositories.

We reuse the semantic service description model² developed in the IoT.est project³ to create the semantic repositories on the gateways. The model follows
 220 a “profile-model-grounding” design pattern based on OWL-S⁴ and RESTful (Fielding, 2000) atomic service modelling (Wang et al., 2013). The “profile” defines the non-functional service aspects such as service name, category, QoS and location; the “model” defines the functional aspects such as the operations allowed on the service; and the “grounding” describes service interaction elements
 225 (e.g., endpoint addresses and communication protocols), and captures the RESTful resources (e.g., Input/Output parameters and the relevant access URLs).

Figure 2 shows a temperature sensor service (TS500Service) annotated according to the semantic service model. Location information can be described
 230 using WGS-84⁵ latitude/longitude coordinates or concepts in existing ontologies (e.g., place names in the GeoName ontology⁶). The indoor location ontologies allows definition of sensor observation areas to be more specific, e.g., a room or building. This provides more fine-grained descriptions for features of services and captures region containment and relative positioning. The profile
 235 also describes the sensor, or more generally, IoT resources, whose functional-

²<http://artemis.ccsr.fi.net:8686/KnowledgeManagement/ontology/IoT.est-Service.v1.9.owl>

³<http://www.ict-iotest.eu/>

⁴<http://www.ai.sri.com/daml/services/owl-s/1.2/>

⁵http://www.w3.org/2003/01/geo/wgs84_pos

⁶<http://www.geonames.org/ontology/>

Figure 2: A temperature sensor service (TS500Service) annotated according to the semantic model

ties are exposed by the corresponding services, e.g., resource with resourceID “TempSensor” (see Figure 2). These resources also have location properties as a resource’s location might be different from the area being observed by its service (e.g., a camera resource). An important property in the service description is the feature being exposed. The modelled features have associated operations (e.g., `getTS500TemperatureData`) linked to the grounding part of the model (e.g., the endpoint URLs through which the features are accessible).

```
PREFIX IoT: <http://ict-iotest.eu/iotest/ontologies/v1.6/IoT.est-Service.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?s WHERE
{
  ?s IoT:supports ?grounding .
  ?grounding IoT:hasRESTResource ?rest .
  ?rest IoT:hasRESTComponent ?component .
  ?component IoT:hasRESTResponse ?fRes .
  ?fRes IoT:hasSemanticType "http://purl.oclc.org/NET/ssnx/qu/dim#Humidity"^^xsd:anyURI.
}
```

Figure 3: SPARQL query for a humidity sensor service

Queries for sensor services are represented using the SPARQL semantic query

language⁷, which specifies the user search criteria as a graph pattern. Figure
 245 3 shows an example of a SPARQL query which searches for humidity sensor
 services. During discovery, the location information provided in the query is
 first used to search the geospatial index to locate the gateways that potentially
 contain the required services. Then the queries are forwarded to the semantic
 repositories on those gateways, which subsequently perform semantic search
 250 using SPARQL. The services that provide all the required functionalities and
 semantic types (e.g., temperature or humidity) are then retrieved and returned
 to the requester. The semantic types used in the experiments are taken from
 the QU-ontology⁸.

3.3. Geospatial Indexing

255 Geospatial indexing techniques have been used in many application areas
 to accelerate and optimise query processing, such as computer vision, compu-
 tational geometry and geographical information systems (Lu & Ooi, 1993). A
 number of effective indexing structures have been proposed to search informa-
 tion consisting of geographical features based on object bounding (Lu & Ooi,
 260 1993), for instance, the R-Tree (Guttman, 1984), R*-Tree (Beckmann et al.,
 1990), packed R-Tree (Roussopoulos & Leifker, 1985), and buddy Tree (Seeger & Kriegel,
 1990). In this paper, we utilise the R-Tree to construct a spatial index of
 gateways for sensor service discovery due to its effectiveness and relatively low
 complexity.

3.3.1. The R-Tree

265 R-Tree is particularly effective for answering queries containing spatial op-
 erations such as “nearby”, “intersect” or “contain”. An R-Tree consists of two
 types of nodes: leaf and non-leaf nodes. Leaf nodes contain index entries of the
 form $(I, tuple - identifier)$, where $tuple - identifier$ is a tuple pointing to an
 270 object to be indexed and I is the minimum bounding rectangle (also known as

⁷<http://www.w3.org/TR/2013/REC-sparql11-query-20130321/>

⁸http://www.w3.org/2005/Incubator/ssn/wiki/QU_Ontology

minimum bounding box) for that object. Non-leaf nodes contain index entries of the form $(I, child_pointer)$, where *child-pointer* points to all its children in the lower level of the R-Tree and *I* is the minimum bounding rectangle covering all the bounding rectangles of its children. An R-Tree of order (m, M) needs
275 to preserve the following fundamental properties (Manolopoulos et al., 2005) at any time.

- A node in the R-Tree (except the root) can store between $[m, M]$ entries, where *M* stands for the maximum number of entries and *m* is the minimum number of entries allowed, $m \leq M/2$;
- 280 • All leaves of the R-Tree must be at the same level (height balanced tree);
- The minimum allowed number of entries for the root is 2.

Update operations (e.g., node insertion and deletion) need to preserve these properties and they tend to be computationally expensive. For example, inserting a new geometric object into the R-Tree might result in node overflow; in
285 the worst case, the overflow might propagate to the root node, which causes the tree height to increase. Deletion operations might cause node underflow which also may propagate to other levels of the tree and a condense algorithm needs to be used to reinsert a number of objects.

3.3.2. R-tree of Gateways

290 Every sensor service is associated with a minimal bounding rectangle, which represents the observation area of the underlying sensor. Every gateway is also associated with a minimal bounding rectangle, which encloses the observation areas of all the sensor services registered to that gateway (see Figure 4).

With the bounding rectangles, both sensor services and gateways become
295 spatial objects and therefore can be indexed using the R-Tree. If we choose to index each individual sensor service' bounding rectangle, then the size of the tree would be extremely large given the numerous sensing devices in the IoT. So our proposal is to create an R-Tree of gateways based on their minimal

Figure 4: Gateway Binding Boxes and R-Tree Index

bounding boxes. The most notable advantage is that this indexing approach
 300 can support efficient search while at the same time minimising the need for the
 expensive update operations. Due to the dynamic nature of the IoT environ-
 ment, geographical locations and status of the services might change frequently
 due to many factors, for example, service mobility, new sensor service regis-
 tration, sensor failure, wireless communication quality, or battery drain. If an
 305 R-Tree indexes the sensor services directly, then a large number of update op-
 erations might need to be performed frequently. However, the gateways hosting
 the sensor services are much more reliable and stable. The bounding rectan-
 gles of the gateways would remain unchanged in many situations despite the

310 frequent service changes. Changes with respect to sensor services can be conveniently captured by the gateway and handled locally by updating the semantic repositories.

3.3.3. *Geographical Descriptor for Gateway*

Calculating a gateway's minimal bounding rectangle which aligns to the axes of the geographical coordinate system is straightforward (It should be noted that only this rectangle can be directly indexed by the R-Tree.). We only need to find 315 the minimum and maximum values for the latitude and longitude, respectively, and then construct the bounding rectangle. However, using this rectangle as the geographical descriptor for the gateway might introduce high false positive rate, which means that large areas outside of the gateway's range are included in the axis-aligned minimal bounding rectangle. To alleviate this problem and thus 320 improve the accuracy, we propose a more precise descriptor for the gateway's geographical features. We first use the Graham scan algorithm (Graham, 1972) to compute a convex polygon using the geographical locations of all sensor services inside a gateway. The pseudocode⁹ for the Graham's scan algorithm is given in Algorithm 1. It starts with a point that is guaranteed to be on the 325 convex hull and then iteratively adds other points to the convex hull.

The convex polygon is the most precise descriptor for the geospatial features of the gateway; however, it is computationally expensive to test if a point falls into the polygon as there might be a large number of vertices on the polygon 330 (usually a gateway manages a few hundreds of wireless sensors). A descriptor which can enclose the convex polygon with less vertices is preferred. We further use the rotating calliper algorithm (Toussaint, 1983) to compute a minimal bounding rectangle with orientation not necessarily aligned to the geographical coordinate system (pseudocode¹⁰ is shown in Algorithm 2). As the minimal 335 bounding rectangle is the one that coincides with one of the edges on the convex

⁹<http://courses.csail.mit.edu/6.006/spring11/rec/rec24.eps>

¹⁰<http://cgm.cs.mcgill.ca/~orm/rotcal.html>

Algorithm 1 The Graham scan algorithm for computing the convex hull

Require: An empty list H ;

Ensure: H , the list of points on the convex hull;

- 1: Choose p_0 to be the point with the lowest y -coordinate;
 - 2: Add p_0 to H ;
 - 3: Let $(p_1, p_2, \dots, p_n)$ be the remaining points sorted by their polar angles relative to p_0 from smallest to largest.
 - 4: **for** For each point p_i **do**
 - 5: **if** adding p_i to the convex hull results in “left turn” **then**
 - 6: add p_i to H
 - 7: **else**
 - 8: remove p_i for consideration
 - 9: **end if**
 - 10: **end for**
-

polygon (Toussaint, 1983), the algorithm keeps rotating the lines and finds the rectangle with the minimum area.

We propose to use the overlap of the minimal bounding rectangle computed through the rotating calliper algorithm and the minimal bounding rectangle aligned to the geographical coordinate system as the descriptor for the geographical features of the gateway. The overlapped area is also convex and we refer to it as the Approximated Convex Polygon (ACP) of the gateway. ACP guarantees to enclose the original convex polygon and it is a more accurate geographical descriptor than both rectangles. Furthermore, as there are at most eight vertices on the ACP, efficient computation for point containment can be ensured. The axis-aligned minimal bounding rectangle of the gateway is used for the R-Tree indexing, and the ACP is stored on the gateway. During service discovery, when a gateway is selected by the spatial index, the search process first checks if the location specified in the query falls under the ACP. The gateway will only perform search in the semantic repository if the condition is true. The additional step can improve the discovery accuracy and potentially save search

Algorithm 2 The Rotating Calliper algorithm for computing the minimal bounding rectangle of arbitrary orientation

Require: A convex polygon;

Ensure: Minimum area bounding rectangle;

- 1: Compute all four extreme points for the polygon, i.e., $xminP$, $xmaxP$, $yminP$, $ymaxP$.
 - 2: Construct four lines of support (two sets of “calipers”) through all four points.
 - 3: **if** one or more lines coincide with an edge **then**
 - 4: compute the area of the rectangle determined by the four lines.
 - 5: **else**
 - 6: set minimum area as infinite.
 - 7: **end if**
 - 8: **repeat**
 - 9: rotate the lines clockwise until one of them coincides with an edge of the convex polygon.
 - 10: compute the area of the new rectangle.
 - 11: update the minimum if smaller.
 - 12: **until** The lines have been rotated an angle greater than 90 degrees
 - 13: Output the minimum area enclosing rectangle.
-

cost on the gateway; however, it can be skipped if a lower discovery accuracy or longer response time can be tolerated.

3.4. Geospatial Index of Gateways and Distributed Semantic Repository Search

355 The core idea behind the design is to integrate the two different while complementing techniques, Geospatial Indexing of Gateways + distributed Semantic Repositories (GIGW+SR), to allow large-scale and accurate sensor service discovery. The search starts from the root of the R-Tree and moves towards the leaf level (i.e., gateway). Once a gateway(s) is located and the gateway confirms that the area of interest falls under its ACP, GIGW+SR initiates semantic
360

search to find the services that offer the same functionalities as requested by the query.

Some of the notable advantages of this design are explained as follows. The geospatial index can efficiently and effectively reduce the search space to one or a few gateways; therefore, only a limited number of distributed semantic repositories need to be searched (semantic search is more time-consuming than index search). The design is resilient to frequent service changes caused by the sensors or the dynamic environment (e.g., service registration, failure, or sensor hardware fault). Most of the changes can be described in the semantic repository locally, which avoids the need to perform the expensive insertion and deletion operations in the spatial index. The spatial index only needs to be updated when the minimal bounding rectangle or ACP of the gateway is affected by the changes; however, this is not expected to happen frequently.

As an illustrating example, consider that a new sensor service registers with a gateway. If the service falls under the ACP or minimal bounding rectangle of the gateway, then the GIGW+SR index does not need to be updated. The gateway simply inserts the semantic description of the service into the repository, which is kept up-to-date based on the best efforts of the gateway. If the service is mobile and moves outside the territory of a gateway, the semantic description of that service is removed from the gateway's repository. Assume that the mobile sensor service eventually moves into the range of another gateway and registers with it, then the similar procedure will follow. If the sensor moves to a location outside of the ACP or minimal bounding rectangle, then the gateway recalculates it and sends request to update the spatial index accordingly. The spatial index also needs to be updated if changes directly occur at the gateway level, such as a new gateway deployed or an existing gateway removed. As these changes are rare in practice, the system is able to keep the computation cost of index maintenance significantly low.

4. Experiments and Evaluation

With the distributed implementation of the semantic repositories and the geospatial indexing structure, the proposed approach has great potential to improve discovery efficiency and scalability. In this section, we perform a series of experiments and compare the results to those obtained using the conventional geospatial indexing methods.

4.1. Experimental Settings

A dataset that consists of 100 gateways distributed over 22 different locations around the University of Surrey is created for the experiments. 100 sensor services of different types (e.g., temperature, light, and windspeed) are generated for each gateway, which results in a total number of 10,000 services. A semantic repository is constructed on each of the gateways to store the semantic descriptions of the services. The Jena API¹¹ is used to handle SPARQL queries and to perform the semantic matchmaking.

Each gateway is associated with a minimal bounding rectangle aligned with the axes (used for constructing the spatial index) and an ACP (used as a filter before semantic matchmaking). We use the JTS Topology Suite¹² and OpenCarto libraries¹³ to compute the rectangle and ACP, and the Java Spatial Index¹⁴ to create the spatial index. The maximum and minimum number of entries per node, M and m , need to be set properly to create a tree structure that can adapt easily to distributed spatial objects, while not branching out too widely at the same time (Guttman, 1984). In our experiments, we set $M = 8$ and $m = 4$, respectively.

To evaluate the performance of our method, we compare it to two alternative methods using conventional geospatial indexing methods proposed in

¹¹<http://jena.apache.org>

¹²<http://www.vividsolutions.com/jts/JTSHome.htm>

¹³<http://sourceforge.net/projects/opencarto/>

¹⁴<http://jsi.sourceforge.net>

(De & et al, 2012) and (Grainger & Potter, 2014). The first one is called *Geospatial Indexing of Sensor Services* (GISS) which creates an R-Tree index with individual sensor services as the leaf nodes. The minimal bounding rectangle defining the observation area of the sensor service is directly used for indexing. For the convenience of implementation, GISS also makes use of semantic repositories to retrieve service descriptions. When the location or observation area of a sensor service changes, both the semantic service description and the spatial index need to be updated. As can be expected, a considerable number of update requests need to be sent to the spatial index in a dynamic environment. Based on the R-Tree, GISS attempts to find up to five services nearest to the location of interest specified in the user's query. If the result is not empty, it forwards the query to all the gateways to which the services are registered. A semantic search is then initiated to find out whether the services provide the required functionalities. As GISS has some undesirable characteristics, it is anticipated that it is much more computationally expensive than GIGW+SR.

The second method we implement for the comparison study is called *Aggregated Geohash Indexing* (AGHI). Geohash encodes geographical coordinates into arbitrary strings such that the closer two geographical locations are, the longer their common geohash prefix string is. As such, geospatial searching can be enabled by leveraging the power of indexing techniques from the field of information retrieval. For example, the Apache Solr search server¹⁵ allows indexing and searching information objects with geohash values. An attractive property of this technique for sensor service discovery is that geohashes of different services within the same gateway can be aggregated and indexed (therefore, the size of the index can be reduced significantly). In our implementation, we first calculate and aggregate the geohash values for all the sensor services and then create an index of geohashes using Apache Lucene¹⁶. The discovery process first searches the aggregated geohash index, locates the gateways (if there

¹⁵<http://lucene.apache.org/solr/>

¹⁶<http://lucene.apache.org/>

is any) and then performs semantic search on the gateways.

Server applications are developed for GIGW+SR, GISS and AGHI, respectively, to compare their performance in terms of query response time, throughput, index creation time and index size. The indices and the distributed semantic repositories are stored on the same server. As such, in our experiments, the network communication time and delay between the index server and all the gateways is not considered.

4.2. Evaluation Results

The two most important metrics for evaluating performance of the discovery methods are query response time and throughput. The creation time and size of the geospatial index also have considerable impact on the performance of the discovery. We first compare GIGW+SR with GISS in terms of index creation time, size of the index and query response time. Then we compare GIGW+SR with AGHI by sending a large number of concurrent queries and measuring the throughput to assess their scalability.

4.2.1. Index Creation Time

Figure 5 shows the time taken for both GIGW+SR and GISS to create the indices as a number of services is added (from 100 to 10,000). It should be noted that the index creation time corresponds to the time for a specified number of insertion operations. In GIGW+SR, only one minimal bounding rectangle for the gateway needs to be indexed for every 100 sensor services added into the system. While in GISS, the minimal bounding rectangles for all sensor services need to be indexed. The results shown in the figure indicate that constructing an index in GIGW+SR takes much less time than constructing one in GISS.

4.2.2. Index Size

In a highly dynamic environment where changes are frequent, size of the geospatial index might have substantial impact on the update operations. Generally, the update operations (e.g., performing splitting and condense algorithms) tend to be more expensive if the size of the index becomes larger. The

Figure 5: R-Tree Index Creation Time.

GIGW+SR method keeps the index size reasonably small and constrains many update operations within the gateway by updating the semantic repositories. In many situations, the spatial index remains unchanged despite the frequent changes of the sensor services. Figure 6 shows the maximum possible R-Tree height for GIGW+SR and GISS as the size of the dataset is increased (from 100 to 10,000 services). Sizes of the R-Trees in both methods increase steadily as the number of services is increased. However, the height of the R-Tree in GISS increases more rapidly than the one in GIGW+SR, indicating that GIGW+SR has the potential to be more scalable than GISS.

4.2.3. Query Response Time

We compare the response time of the two methods by gradually increasing the dataset size. For GIGW+SR, the query response time is split into the time to find the gateways that contain or overlap with the location of interest and the time to find relevant services from the semantic repositories. For GISS, the query response time is split into the time to find the services nearest to the location of interest and the time to search the semantic repositories (to verify whether a service provides the functionalities required by the queries). Figure 7 shows the results by superimposing the index search time over the semantic search time.

The results show that GIGW+SR responds to a query much more quickly

Figure 6: Maximum R-Tree Height

Figure 7: Query Response Time.

than GISS. GISS attempts to find the sensor services nearest to the location of interest, however, the services may not fulfil the requirements of the query,

which results in the invocation of SPARQL queries on multiple gateways. This explains why semantic search for GISS takes much longer time than GIGW+SR.

495 On the contrary, GIGW+SR first narrows the search space significantly to a much smaller one near the location of interest and then uses just one or very few SPARQL queries to find the required sensor services, which results in much shorter response time. Another disadvantage of GISS is that it is likely to generate much more network traffic than GIGW+SR.

500 4.2.4. Throughput

To assess the scalability of the GIGW+SR method in a more realistic environment where there are a large number of concurrent queries, it is essential to measure the throughput. The proposed method is compared to AGHI, which exploits the power of geohashing and indexing technique from the field of information retrieval. We perform the experiments using the Apache JMeter¹⁷, a full multi-threading framework for testing the functional behaviour and performance of applications of many different kinds concurrently. Both GIGW+SR and AGHI are implemented as server applications and JMeter is used as the client.

510 Figure 8 shows the throughput for both GIGW+SR and AGHI with 50 concurrent queries. The queries are sent to the service applications continuously and repeatedly by the JMeter client. The average throughput for GIGW+SR is 689.157/minute, while the average throughput for AGHI is only 469.441/minute.

Table 1: Throughput testing for GIGW+SR and AGHI. The number of concurrent queries is from 10 to 100.

Methods	10	20	30	40	50	60	70	80	90	100
GIGW+SR	580.4	688.7	675.3	689.2	689.2	690.0	709.4	717.7	705.0	712.3
AGHI	390.7	454.8	448.1	466.7	469.4	444.2	439.2	456.4	466.7	462.5

We repeat the simulations by changing the number of concurrent queries

¹⁷<http://jmeter.apache.org/>

(a) GIGW+SR

(b) AGHI

Figure 8: Throughput testing using JMeter. The number of concurrent queries is 50.

515 (from 10 to 100) sent from JMeter. We believe that the experimental setting is close to the realistic situation, for example, the number of the sensor services within a gateway, and the number of possible concurrent queries. The average throughput values for all settings are shown in Table 1. From the figures, we can see that both methods are in fact scalable since the throughput does not de-
 520 grade as the number of concurrent queries increases (for up to 100 concurrently queries). The figures also indicate that GIGW+SR produces larger throughput than AGHI in all settings.

In AGHI, one has to decide the length of the geohash prefix during the

index search; for convenience, we define the prefix length as 7. We index all the
 525 geohashes of seven-characters long and only take the first 7 characters from the
 query for matching. This often causes multiple gateways to be selected during
 the index search, and semantic search has to be performed on all those gateways.
 This explains the performance difference between AGHI and GIGW+SR in
 terms of throughput. If we increase the length of the geohash prefix, then
 530 services that potentially fulfil the discovery criteria might be excluded from
 the discovery results because of the discrepancy between the geohash computed
 from query and the ones stored in the index. Generally, geohash index performs
 better with exact matching, however, this is difficult in sensor service discovery
 as requesters do not have any knowledge on the exact geographical locations
 535 of the sensor services beforehand. On the contrary, spatial indexing is effective
 and efficient in proximity based search, such as nearby, contain or overlap.

5. Conclusion

The distinctive characteristics of the emergent sensor services necessitate
 new discovery methods to be developed. We have identified and addressed some
 540 of the most prominent issues related to this need. The major contributions of the
 work are summarised as follows. First, we design a scalable platform that can
 support large-scale sensor service discovery. The approach exploits the synergy
 between geospatial indexing and distributed semantic search: geospatial index-
 ing reduces discovery space and ensures efficiency, while distributed semantic
 545 search and matchmaking provides enhanced search accuracy and ensures inter-
 operability. Second, the proposed method is resilient to the dynamicity associ-
 ated with sensor services. Most of the changes can be constrained locally within
 the gateway and do not necessarily affect the indexing structure. This signifi-
 cantly reduces the need for the computationally expensive update operations in
 550 the geospatial index. Third, a method for more accurate computation of the ge-
 ographical features of gateways and services is proposed based on computational
 geometric techniques. It provides potential to improve discovery precision and

to save semantic search cost. Fourth, we perform an extensive experimental study to verify the feasibility and scalability of the proposed method against the benchmark ones. Superior performance of the method are confirmed by the evaluation results in terms of throughput and response, and are consistent with the theoretical analysis.

The presented work is useful for many practical sensor Web applications in which efficient service discovery based on spatial properties is needed. For example, the discovered real world services can participate in automated business process composition (Coria et al., 2014) to create more intelligent, context-aware applications. In the context of smart university and smart city, energy saving, building control and asset health monitoring applications (Stavropoulos et al., 2013; Nati et al., 2013) can benefit from the semantic service descriptions and spatial search which enable exploration of resources in both indoor and outdoor environments. Methods for querying the semantic sensor Web data have been proposed (Calbimonte et al., 2012), augmented with data aggregation (Wu et al., 2014) and analysis with temporal abstractions (Roda & Musulin, 2014). Our approach can complement and extend these works to support service and sensor data access and analysis at the Internet scale.

In real discovery applications, the number of sensor services would be several magnitude larger than the one in our current work. We may need to extend the proposed discovery infrastructure to a hierarchy of geospatial indices to address the scalability challenge. There are also limitations related to the implementation of the experiments, for example, changes related to the sensor services and the physical environment are not simulated with the synthesised dataset; the distributed semantic repositories are not implemented on different physical gateways; and the communications between the gateways and the discovery server are not considered. As a result, the real-time quality of information for sensor networks, which is useful in real service discovery systems, cannot be used in the current work.

The outcome of the discovery is usually a list of semantically equivalent sensor services that all fulfil the query criteria. One future research direction is to

design ranking methods for the discovered sensor services according to appropriate qualitative and quantitative measures. Such ranking methods are important for a number of real world applications, for example, to select the best services for data integration, or to help a service composition engine perform automatic service adaptation in case of service failure. The current work only focuses on gateways installed at fixed locations. In the future work, we aim to extend the existing platform to support mobility of gateways. We envisage a method based on the distributed hash table, in which spatially indexed static gateways can volunteer to form an overlay peer-to-peer network and provide distributed discovery for mobile gateways, e.g., mobile phones. Another research direction is concerned with the creation of more intelligent, value-added services based on the proposed distributed framework, e.g., abstraction, pattern analysis and event extraction from sensor data. This is in line with the idea of distributing intelligent computation at the edge of the Internet, which would bring substantial benefits to many IoT and big data based applications.

Acknowledgement

The work reported in this paper is funded by the EU FP7 Programme IoT.est project: Internet of Things Environment for Service Creation and Testing (<http://ictiotest.eu/iotest/>), contract number: 288385.

References

- Abangar, H., Barnaghi, P., Moessner, K., Tafazolli, R., Nnaemego, A., & Balaskandan, K. (2010). A Service Oriented Middleware Architecture for Wireless Sensor Networks. In *Proceedings of Future Network and Mobile Summit 2010*.
- Abrougui, K., Boukerche, A., & Ramadan, H. M. S. (2012). Efficient load balancing and QoS-based location aware service discovery protocol for vehicular ad hoc networks. *EURASIP J. Wireless Comm. and Networking*, (p. 96).

- Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54, 2787–2805.
- Barnaghi, P., Wang, W., Henson, C., & Taylor, K. (2012). Semantics for the Internet of Things: Early Progress and Back to the Future. *Int. J. Semant. Web Inf. Syst.*, 8, 1–21.
- Beckmann, N., Kriegel, H.-P., Schneider, R., & Seeger, B. (1990). The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data SIGMOD '90* (pp. 322–331). New York, NY, USA: ACM.
- Botts, M., Percivall, G., Reed, C., & Davidson, J. (2007). *OGC Sensor Web Enablement: Overview And High Level Architecture..* Technical Report OGC.
- Calbimonte, J.-P., Jeung, H., Corcho, O., & Aberer, K. (2012). Enabling Query Technologies for the Semantic Sensor Web. *Int. J. Semant. Web Inf. Syst.*, 8, 43–63.
- Chakraborty, D., Joshi, A., Yesha, Y., & Finin, T. (2006). Toward Distributed Service Discovery in Pervasive Computing Environments. *IEEE Transactions on Mobile Computing*, 5, 97–112.
- Compton, M., Barnaghi, P., Bermudez, L., Castro, R. G., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W. D., Phuoc, D. L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., & Taylor, K. (2012). The SSN Ontology of the Semantic Sensor Networks Incubator Group. *Journal of Web Semantics*, 17, 2–32.
- Coria, J. A. G., Castellanos-Garzn, J. A., & Corchado, J. M. (2014). Intelligent business processes composition based on multi-agent systems. *Expert Systems with Applications*, 41, 1189 – 1205.

- De, S., & et al (2012). Concepts and Solutions for Entity-based Discovery of IoT Resources and Managing their Dynamic Associations, IoT-A Deliverable D4.3.
- De, S., Christophe, B., & Moessner, K. (2014). Semantic Enablers for Dynamic Digital-Physical Object Associations in a Federated Node Architecture for the Internet of Things. *Ad Hoc Networks*, 18, 102–120.
- De, S., Elsaleh, T., Barnaghi, P. M., & Meissner, S. (2012). An Internet of Things Platform for Real-World and Digital Objects. *Scalable Computing: Practice and Experience*, 13.
- Delicato, F. C., Pires, P. F., Pirmez, L., & Carmo, L. F. (2005). A Service Approach for Architecting Application Independent Wireless Sensor Networks. *Cluster Computing*, 8, 211–221.
- Evans, D. (2011). The Internet of Things: How the Next Evolution of the Internet Is Changing Everything, CISCO white paper.
- Evdokimov, S., Fabian, B., Kunz, S., & Schoenemann, N. (2010). Comparison of Discovery Service Architectures for the Internet of Things. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)* (pp. 237–244).
- Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. thesis.
- Fredj, S. B., Boussard, M., Kofman, D., & Noirie, L. (2013). A Scalable IoT Service Search Based on Clustering and Aggregation. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCoM), IEEE International Conference on and IEEE Cyber, Physical and Social Computing* (pp. 403–410).
- Garofalakis, J., Panagis, Y., Sakkopoulos, E., & Tsakalidis, A. (2006). Contemporary web service discovery mechanisms. *J. Web Eng.*, 5, 265–290.

- Gibbons, P. B., Karp, B., Ke, Y., Nath, S., & Seshan, S. (2003). IrisNet: An Architecture for a Worldwide Sensor Web. *IEEE Pervasive Computing*, 02, 22–33.
- Graham, R. L. (1972). An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Inf. Process. Lett.*, 1, 132–133.
- Grainger, T., & Potter, T. (2014). *Solr in Action*. Manning Publications.
- Gu, T., Pung, H. K., & Zhang, D. Q. (2004). A Bayesian Approach for Dealing with Uncertain Contexts. In *Proceedings of the 2nd International Conference on Pervasive Computing*. Austrian Computer Society.
- Gu, T., Pung, H. K., & Zhang, D. Q. (2005). A service oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28, 1 – 18.
- Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., & Savio, D. (2010). Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. *IEEE Trans. Serv. Comput.*, 3, 223–235.
- Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching. *SIGMOD Rec.*, 14, 47–57.
- Jirka, S., Brring, A., & Stasch, C. (2009). Discovery Mechanisms for the Sensor Web. *Sensors*, 9, 2661–2681.
- Liang, S., & Huang, C.-Y. (2013). GeoCENS: A Geospatial Cyberinfrastructure for the World-Wide Sensor Web. *Sensors*, 13, 13402–13424.
- Lu, H., & Ooi, B. C. (1993). Spatial Indexing: Past and Future. *IEEE Data Eng. Bull.*, 16, 16–21.
- Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A. N., & Theodoridis, Y. (2005). *R-Trees: Theory and Applications (Advanced Information and Knowledge Processing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

- 690 Mayer, S., Guinard, D., & Trifa, V. (2012). Searching in a web-based infrastructure for smart things. In *Proceedings of the 3rd International Conference on the Internet of Things (IoT 2012)*.
- McIlraith, S. A., Son, T. C., & Zeng, H. (2001). Semantic Web Services. *IEEE Intelligent Systems*, 16, 46–53.
- 695 Nati, M., Gluhak, A., Abangar, H., & Headley, W. (2013). SmartCampus: A user-centric testbed for Internet of Things experimentation. In *Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on* (pp. 1–6).
- Nidd, M. (2001). Service discovery in DEAPspace. *IEEE Personal Commu-*
700 *cations*, 8, 39–45.
- Niforatos, E., Karapanos, E., & Sioutas, S. (2012). PLBSD: a platform for proactive location-based service discovery. *J. Locat. Based Serv.*, 6, 234–249.
- OASIS (2009). Devices Profile for Web Services (DPWS) v1.1.
- Perera, C., Zaslavsky, A., Christen, P., Compton, M., & Georgakopoulos, D.
705 (2013). Context-Aware Sensor Search, Selection and Ranking Model for Internet of Things Middleware. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on* (pp. 314–322). volume 1.
- Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014a). Sensing as a service model for smart cities supported by Internet of Things. *Transac-*
710 *tions on Emerging Telecommunications Technologies*, 25, 81–93.
- Perera, C., Zaslavsky, A., Liu, C., Compton, M., Christen, P., & Georgakopoulos, D. (2014b). Sensor Search Techniques for Sensing as a Service Architecture for the Internet of Things. *Sensors Journal, IEEE*, 14, 406–420.
- Roda, F., & Musulin, E. (2014). An ontology-based framework to support intelligent data analysis of sensor measurements. *Expert Systems with Applications*,
715 41, 7914 – 7926.

- Roussopoulos, N., & Leifker, D. (1985). Direct Spatial Search on Pictorial Databases Using Packed R-Trees. In *SIGMOD Conference* (pp. 17–31). ACM Press.
- 720 Seeger, B., & Kriegel, H.-P. (1990). The Buddy Tree: An Efficient and Robust Access Method for Spatial Data Base. In *Proceedings of the Sixteenth International Conference on Very Large Databases* (pp. 590–601). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Sheth, A., Henson, C., & Sahoo, S. S. (2008). Semantic Sensor Web. *IEEE Internet Computing*, 12, 78–83.
- 725 Stavropoulos, T. G., Gottis, K., Vrakas, D., & Vlahavas, I. (2013). aWESoME: A Web service middleware for ambient intelligence. *Expert Systems with Applications*, 40, 4380 – 4392.
- Toussaint, G. T. (1983). Solving geometric problems with the rotating callipers. In *Proceedings of IEEE MELECON*.
- 730 Wang, W., De, S., Cassar, G., & Moessner, K. (2013). Knowledge Representation in the Internet of Things: Semantic Modelling and its Applications. *special issue on Knowledge Acquisition and Management in the Internet of Things, Automatika - Journal for Control, Measurement, Electronics, Computing and Communications*, 54, 388–400.
- 735 Wei, Q., & Jin, Z. (2012). Service discovery for internet of things: a context-awareness perspective. In *Proceedings of the Fourth Asia-Pacific Symposium on Internetwork* Internetwork 12.
- Wu, C.-I., Kung, H.-Y., Chen, C.-H., & Kuo, L.-C. (2014). An intelligent slope disaster prediction and monitoring system based on WSN and ANP. *Expert Systems with Applications*, 41, 4554 – 4562.
- 740

Highlights

- (1) A scalable platform supporting large-scale sensor service discovery
- (2) A discovery approach more resilient to dynamicity of sensor services and environment
- (3) Accurate computation of the geographical features of gateways and sensor services
- (4) Superior performance over existing methods in response time and throughput.