Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2012, Article ID 107041, 11 pages
doi:10.1155/2012/107041

Research Article

A DHT-Based Discovery Service for the Internet of Things

Federica Paganelli and David Parlanti

CNIT, Research Unit at the University of Firenze, Via S. Marta 3, 50139 Firenze, Italy

Correspondence should be addressed to Federica Paganelli, federica.paganelli@unifi.it

Received 31 March 2012; Accepted 17 September 2012

Academic Editor: Jae-Ho Choi

Current trends towards the Future Internet are envisaging the conception of novel services endowed with context-aware and autonomic capabilities to improve end users' quality of life. The Internet of Things paradigm is expected to contribute towards this ambitious vision by proposing models and mechanisms enabling the creation of networks of "smart things" on a large scale. It is widely recognized that efficient mechanisms for discovering available resources and capabilities are required to realize such vision. The contribution of this work consists in a novel discovery service for the Internet of Things. The proposed solution adopts a peer-to-peer approach for guaranteeing scalability, robustness, and easy maintenance of the overall system. While most existing peer-to-peer discovery services proposed for the IoT support solely exact match queries on a single attribute (i.e., the object identifier), our solution can handle multiattribute and range queries. We defined a layered approach by distinguishing three main aspects: multiattribute indexing, range query support, peer-to-peer routing. We chose to adopt an over-DHT indexing scheme to guarantee ease of design and implementation principles. We report on the implementation of a Proof of Concept in a dangerous goods monitoring scenario, and, finally, we discuss test results for structural properties and query performance evaluation.

1. Introduction

The research roadmap towards the Future Internet is envisaging novel services endowed with context-aware and autonomic capabilities to support end users in daily living activities (e.g., work, leisure time, travel). In such a perspective, the technological landscape is expected to be populated by a wide range of functional capabilities offered by heterogeneous types of devices (PCs, mobile phones, household appliances, smart textiles, etc.). Several research fields are expected to contribute towards this ambitious vision, including the Internet of Things, the Internet of Services, and Cloud Computing.

The "Internet of Things" paradigm aims at providing models and mechanisms enabling the creation of networks of "smart things" on a large scale by means of RFID, wireless sensor and actuator networks, and embedded devices distributed in the physical environment [1]. This paradigm will open up the possibility to create novel value-added services by dynamically assembling different types of capabilities (sensing, communication, information processing, and actuation on physical resources, just to mention a few examples). Nonetheless, it is also well-known that the sustainable and efficient realization of IoT solutions requires the conception and development of dynamic adaptation and autonomic capabilities. Hence, the availability of mechanisms for discovering available resources and capabilities is of primary importance in the realization of the above-mentioned vision.

In the Internet of Things, "*data of real world objects and events will be available globally and in vast amounts. These data will be stored in widely distributed, heterogeneous information systems, and will also be in high demand by business and end user applications*." [2]. Discovery mechanisms are thus required to enable a client application to obtain the location and addressing information needed to access these information repositories.

The contribution of this work consists in a distributed Discovery Service for Internet of Things scenarios. The proposed solution adopts a peer-to-peer approach for guaranteeing scalability, robustness, and maintainability of the overall system. While most peer-to-peer discovery services recently proposed for the IoT support exact match queries on a single attribute (i.e., the object identifier) [2], our solution can handle also multiattribute and range queries.

Indeed, while a one-attribute exact match query works well when a client application already knows the identifiers of the target objects (e.g., by means of readers that detect objects tagged with an RFID), the capability of handling multiattribute range query allows client applications to discover information and functional capabilities of objects that have not been detected yet and are not necessarily in physical proximity. The proposed solution is based on a layered architectural design that distinguishes three main aspects: multiattribute indexing, range query support, peer-to-peer routing.

The paper is organized as follows: in Section 2 we discuss functional requirements for discovery services in the IoT. Section 3 surveys related work. In Section 4 we describe the proposed distributed discovery service based on a P2P overlay. In Section 5 we present a Proof of Concept implementation in a reference application scenario and show testing results obtained by means of computational simulations. In Section 6 we draw the conclusions by discussing achieved results and future research directions.

2. Discovery Services in IoT Scenarios

The Internet of Things (IoTs) paradigm implies the perspective of objects and devices endowed with computational, sensing, and communication capabilities and capable of producing and disseminating a large amount of events. IoT applications should adapt their behaviour to changes in an extremely dynamic environment: users enter and exit from "smart environments" (e.g., smart homes [3–5], smart hospitals [6–8], smart offices [9, 10], and tourism locations [11, 12]), objects change their position according to specific application purposes (e.g., mobility of persons [13, 14], multimodal transport of goods [15], and maritime surveillance [16]). Discovery mechanisms are thus required in order to obtain up-to-date information about functional capabilities offered by devices distributed in the environment or information repositories storing information about a given object.

As argued by Atzori et al. [17], "key components of the IoT will be RFID systems." Radio frequency identification technologies (RFID) are typically composed by two types of hardware components: RFID tags and readers [18]. Tags store a unique identifier and, optionally, additional information. They can be applied to objects, animals, or persons (e.g., goods loaded on pallets or containers, smart bracelets worn by patients in hospitals [19]) to gather information about their status or surrounding environment (e.g., measurements gathered from sensors monitoring the status of goods or biomedical parameters of a target patient) and infer knowledge about the context (e.g., alarm and critical events) [20]. RFID and sensing technologies can thus be exploited in the development of context-aware applications in a wide range of application domains: logistic, e-health, security, smart cities [17].

As envisaged in a study promoted by the European Commission [21], it is foreseeable that any "thing" will have at least one unique way of identification (via a "Unique Identifier" or indirectly by some "Virtual Identifier" techniques). Endowed with addressing and communication capabilities, these things will be capable of connecting each other and exchange information. Information and services about objects will be fragmented and handled across many entities (ranging from the creator/owner of the object to the entities that have interacted with it at some stage in its lifecycle). Available information could be provided either at the level of the single object instance (e.g., a goods item) or a group or class of objects (e.g., items of dangerous goods that are transported by land, air, sea, or a combination of these).

Although several low-level requirements could emerge in the manifold application areas of IoT, we elicited the following high-level functional requirements for a discovery service in IoT scenarios [2].

2.1. Flexible Identification Scheme. The discovery mechanism should be transparent with respect to the adopted identification scheme. For instance, in RFID applications for logistics, the Electronic Product Code (EPC) [22] is a widely adopted identification scheme. Other available identification schemes include URIs, IPv6 addresses, Universal Product Code, just to mention a few examples.

2.2. Multiattribute Query. The discovery mechanism should be capable of handling a query for an exact match of a given identifier as well as queries possibly containing other qualifying attributes (e.g., location and category).

2.3. Range Query. In addition to exact match queries, the system should support queries specifying lower and upper boundaries on a single or multiple attributes.

2.4. Multiple Publishers. Depending on the application purposes, several entities may be called to produce and publish information about a given object, besides the object's owner.

2.5. Management APIs. Authorized entities should be able to add, update and delete information associated with a given object.

3. Related Work

RFID systems are considered key components of the IoT [17]. A significant standardization effort has been performed by the EPCglobal consortium to establish principles and guidelines for supporting the use of RFID in trading and enterprise contexts. More specifically, the EPCglobal Network is a set of emerging standard specifications for a global RFID data sharing infrastructure built around the Electronic Product Code (EPC), an unambiguous numbering scheme for the designation of physical goods [22]. It aims at facilitating the handling, storage, and retrieval of information related to EPC-identified items.

The EPC is a universal identifier used for physical objects. It can take the form of a Uniform Resource Identifier (URI), thus enabling information systems to refer to physical objects. The EPC code is typically stored on an RFID attached to the referred object. Main components of the EPC Global

Architecture include the RFID Tags, the Readers, the EPC Middleware, the EPC Information Services (EPCISs), the Object Naming Service (ONS), and Discovery Services. The specifications define how Readers interrogate an RFID tag. The Middleware filters and processes data that are gathered by Reader components. Data are then stored in EPCIS repositories and made available to external clients via the EPCIS Query Interface. The ONS offers a name resolution service that translates an EPC code into the URLs pointing to the EPCIS repositories storing data about that EPC. More specifically, static and dynamic data about physical objects are stored in databases that can be handled by different actors (e.g., manufacturers, logistic providers, retailers, or third parties) and can be accessed via the EPCIS standard interface [23].

The EPC Object Name Service (ONS) is the service that provides clients with the EPCIS URL for a given EPC. The ONS is based on the Internet Domain Name System, which is characterized by a hierarchical architecture [24]. The ONS points to the manufacturer's EPCIS resources. Moreover, discovery services enable discovery of third parties' EPCIS repositories. The EPCIS Discovery Service is the lookup service providing clients with the URIs of EPCIS repositories storing information about a given EPC. The EPCIS Discovery specifications have not been published yet at the time of writing [25]. Several research contributions have thus attempted to fill this gap by proposing original solutions for discovery services.

The BRIDGE Project, funded by the European Commission, had the objective of investigating several issues related to the implementation and adoption of RFIDs in Europe. In the framework of the BRIDGE project, a prototype discovery service was implemented based on LDAP directories specifications [26]. The authors of [27] proposed an implementation of an EPC discovery service based on the IETF specifications of the Extensible Supply-chain Discovery Service ESDS [28].

More recently, some works investigated the use of peer-to-peer (P2P) systems to implement scalable and robust distributed discovery services. Schoenemann et al. [29] proposed a P2P-based architecture for enabling the information exchange among participants of a supply chain. Analogously, Shrestha et al. [30] proposed a peer-to-peer network, where each participant of the supply chain runs a node of the network. These nodes form a structured P2P overlay network with each node having a partial view of the other nodes. Manzanares-Lopez et al. [31] proposed a distributed discovery service for the EPCglobal network based on a P2P architecture, which offers item-level track and trace capabilities along the whole supply chain, also when items are not directly visible, since they are loaded within different storage systems (i.e., packages, boxes, containers, etc.). These peer-to-peer approaches typically adopt Distributed Hash Table (DHT) techniques. Distributed Hash Tables are distributed data structures where the information objects are placed deterministically, at the peer whose identifier corresponds to the information object's unique key according to a given distance metric [32].

In a DHT, each node is identified by means of a key (node-key), usually the MAC or IP address of the node. Analogously, content items handled by the network are identified by a key (content key). Both types of keys are mapped to an identifier of a given bit length by applying a hash function (e.g., SHA-1 or MD5). DHT overlay networks implementations differ in how nodes and contents are associated and how routing to the node responsible for a given identifier is defined. Well-known routing algorithms are Chord [33], Pastry [34], Tapestry [35], and Kademlia [36].

The study carried out by Evdokimov et al. [2] discusses some of the above-mentioned works [23, 24, 26, 29] by comparing strengths and weaknesses of these works and highlighting how P2P approaches better fulfil fault tolerance and communication scalability requirements. As discussed in that study, DHT peer-to-peer networks exhibit properties of scalability, efficiency, robustness, and load-balancing thanks to the adoption of consistent hashing. Typically, in a DHT of N nodes a lookup operation requires O (log N) hops. These networks show properties of self-organization and self-healing, since they are capable of handling joining and leaving events of participating members. For this reason, they also guarantee resilience to node failures and network malfunctioning. Load balancing is achieved through uniform hashing.

As highlighted in the state of the art analysis made by Evdokimov et al. [2], the above-mentioned works support queries providing an object identifier (typically the EPC code) as input, but they do not support more flexible query schemes based on object attributes, though these types of information queries could become very important in the future IoT.

4. A Discovery Service for the IoT Based on a Peer-to-Peer Network

This section describes our discovery service architecture based on a peer-to-peer overlay network. In particular, our approach aims to cope with the following functional requirements: flexible identification scheme, multiattribute query, range query, multiple publishers, management APIs. As discussed in the Related Work Section, most existing works do not support multiattribute and range queries. These contributions are able to solve an exact match query and thus they assume that the client knows the identifier of the target object. This assumption can be easily satisfied if the client application can acquire the identifier from an RFID reader that is close to the RFID-tagged object. However, such a physical proximity constraint would inevitably limit the scope of IoT applications. For instance, a good monitoring application might include the following desired features: (a) displaying on the map the current and past positions of a goods item with identifier I during a multimodal transport route, (b) displaying on the map the current and past positions of the items travelling from location X to location Y. In this example, feature (b) might be easily realized by relying on a discovery service capable of handling complex queries, in addition to exact match queries.

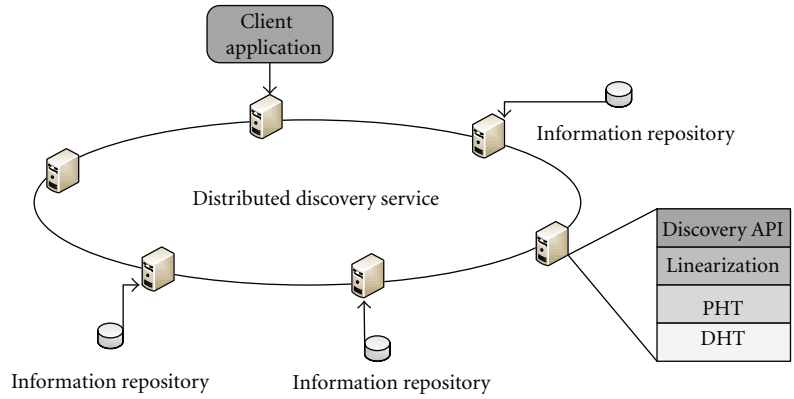

Figure 1: Functional view of the proposed Distributed Discovery Service.

Our work aims to address these limitations of related work by proposing a distributed discovery service capable of handling multiattribute and range queries. Hereafter, we describe the design of our DHT-based discovery service and our choice to adopt a layered functional architecture to promote modularity, ease of design and implementation, and maintainability of the system.

4.1. Existing DHT Implementations for Handling Complex Queries. Most widely adopted DHT implementations, such as Chord [33] and Kademlia [36], support one-attribute exact match queries. Several solutions have been proposed in order to handle range queries. The approach proposed in [21] is based on the adoption of the Prefix Hash Tree (PHT), a distributed data structure that can be built on top of a DHT implementation. The PHT overlay is based on a trie-based structure. A trie is a special tree in which each node represents a prefix of the target data domain. Interestingly, PHT relies merely on the DHT lookup() operation, and it is hence agnostic to the underlying DHT algorithm and implementation. The PHT solution presented in [37] supports only single-attribute range queries.

Mercury [38] supports multiattribute range queries by handling multiple simple overlays, one for each attribute, mapped onto a set of physical nodes. MAAN [39] extends Chord with locality preserving hashing and a recursive multidimensional query resolution mechanism. MAAN relies on a locality-preserving hashing function for each attribute, which has to be constructed using the attribute's values distribution (to be known in advance).

Squid [40] is a peer-to-peer information discovery system implementing a DHT-based structured keyword search. Each data element is associated with a sequence of keywords, the keywords form a multidimensional keyword space where data elements are points in the space and the keywords are the coordinates. In Squid, Space Filling Curves are used to map this multidimensional keyword space to a one-dimensional index space. Space Filling Curves can be defined as a continuous mapping from a d-dimensional space to a one-dimensional space, that is, $f : N^d \rightarrow N$. Examples of SFCs are the Morton curve (z-curve), the Gray code curve, and the Hilbert curve. Depending on the adopted mapping rule, SFCs show different locality-preserving properties. SFCs are locality preserving in that points that are close in the one-dimensional space are mapped from close points in the d-dimensional space. In [41] a SFC-based technique is applied on an indexing scheme built on top of a generic DHT implementation to resolve multiattribute range queries.

4.2. Layered Architecture. We chose to use the Prefix Hash Tree (PHT) distributed data structure. Since a PHT data structure can be built on top of a generic DHT implementation, major advantages of this approach are the promotion of a layered design, and thus, modularity, ease of design, implementation, and maintenance.

Our design approach distinguishes the following layers: (a) an SFC linearization technique for mapping a multi-dimensional domain into a one-dimensional one, (b) a PHT search structure leveraging on the generic DHT get/put interface, (c) a DHT implementation based on the Kademlia algorithm [20]. By exploiting the distributed data management capabilities offered by this peer-to-peer overlay network, application-specific APIs for search and management of discovery-related information can be built.

Figure 1 shows the high-level architecture of the proposed system: each peer exposes a set of discovery APIs that can be invoked by client applications to search for some objects. The Discovery services returns the URIs of the repositories that handle information and services about those objects.

Below we describe our choices for the design of the three layers (a), (b), and (c), while Discovery APIs for a reference application scenario are described in Section 5.

4.2.1. Linearization of the Multiattribute Domain. We suppose that objects can be characterized by d attributes, including the object identifier (e.g., EPC, URI identification schemes). Consequently, our target data domain is a d-dimensional space. We applied a linearization technique based on Space Filling Curves to map a d-dimensional data domain onto a 1-dimensional data domain. The derived 1-dimensional data domain can thus be easily indexed in the PHT structure, as described below.

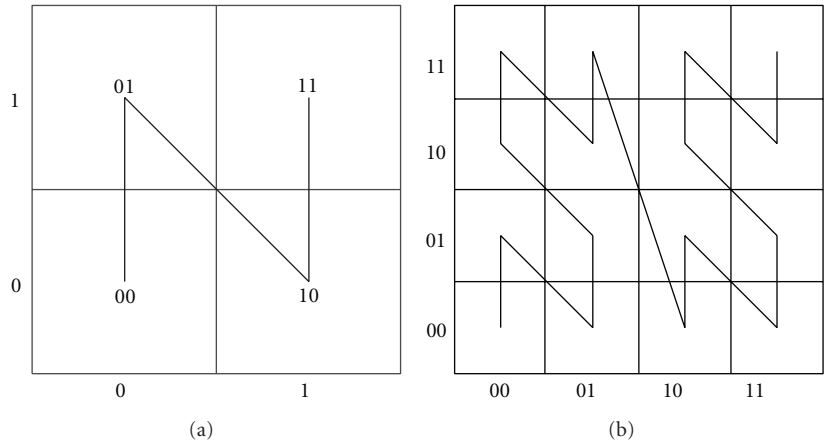

Figure 2: z-order curve approximations: (a) first-order z-curve; (b) second-order z-curve.

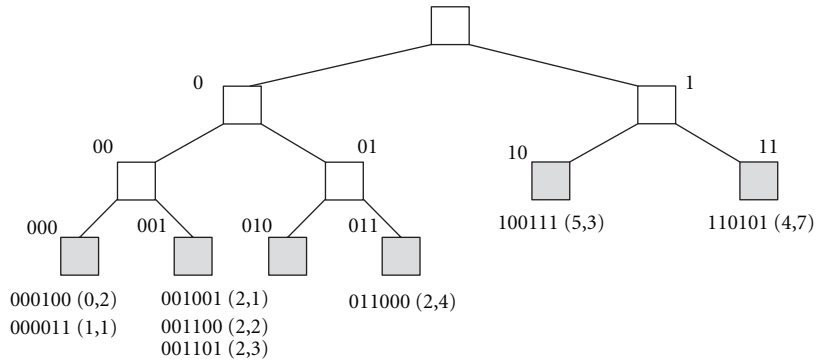

Figure 3: Example of a PHT for a two-dimensional data domain.

Among the existing types of SFC, we chose to adopt the Morton curve, known also as z-curve, since it is computationally simpler to generate than other known SFCs. A z-order-derived key is assembled by cyclically taking a bit from each coordinate of a point in d-dimensional space and appending it to those taken previously ("bit interleaving") [42]. For instance, the 2-dimensional point $P\ (x_1x_2x_3x_4 \cdots x_d, y_1y_2y_3y_4 \cdots y_d)$ is mapped onto the derived key $x_1y_1x_2y_2x_3y_3x_4y_4 \cdots x_dy_d$.

A Space Filling Curve is constructed recursively and can be characterized by two parameters: the order of multidimensionality d and the order of approximation k. The first approximation (first-order curve) is obtained by partitioning a d-dimensional cube into n^d subcubes and connecting the subcubes to form a continuous curve. The kth approximation is obtained by connecting n^{kd} subcubes. Figure 2 shows an example of a SFC built in a 2-dimensional space over the binary domain {0, 1}: Figure 2(a) shows the first-order approximation and Figure 2(b) shows the second-order approximation.

4.2.2. PHT Data Structure. We assume that the data set to be indexed is some number N of D-bit binary keys. A PHT data structure is a binary trie, in which each node of the trie is labelled with a prefix. The prefix is defined recursively: a node with label l has either zero or two children, and the right and left child nodes are labelled with $l0$ and $l1$, respectively. A key K is stored at the leaf node whose label is a prefix of K. Each leaf node can store at most B keys. Consequently, the shape of the PHT depends on the distribution of keys: the trie depth is greater in regions of the domain that are densely populated and lower in sparsely populated domain regions. Figure 3 shows an example of a 2-dimensional data set indexed in a trie-based structure: first, 2-dimensional data objects are mapped into one-dimensional binary keys by means of the z-order curve linearization technique; then, the data objects are stored in the PHT leaf nodes whose label is a prefix of their one-dimensional binary key.

The PHT structure is an indexing data structure built on top of the DHT overlay network. As described in [21], the logical PHT structure is distributed across the nodes in a DHT. PHT prefixes are hashed and assigned to the proper DHT node, that is, a PHT vertex with label l is assigned to the node whose identifier is closest to HASH(l). The PHT can be built by relying on two basic DHT operations: put(key, value) and get(key) primitives. It does not require knowledge of the DHT routing algorithm and implementation details. Consequently, it can be built on top of any DHT implementation.

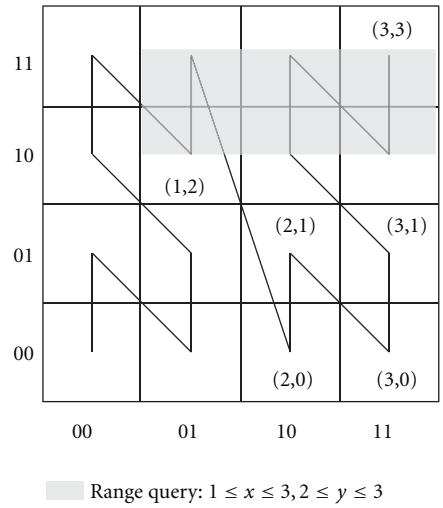

Figure 4: Results for a range query over a one-dimensional z-curve.

PHT Lookup Operation. Given a key K of D bits, a PHT lookup operation shall return the unique leaf node whose label is a prefix of K. According to the algorithm proposed in [37], we adopted a binary search approach. The first step is to determine the prefix from which to start the search, so we consider the first $D/2$ bits of key K. Then, the get() operation is invoked on the DHT overlay with the $D/2$ key prefix provided as input parameter. The result returned by the DHT can be of three types: null, internal node, or leaf node. If the retrieved node is a leaf node, the search algorithm returns all the keys, along with the desired key, stored in the node; if the node does not exist (null), the prefix length is reduced. If the prefix is associated to an internal node, the search algorithm tries longer prefixes, repeating this step recursively until a leaf node is found. This kind of search has been preferred to a linear approach since it allows to considerably reduce the workload on the root and it requires log D DHT gets, thus making the lookup operation efficient [37].

PHT Insert/Delete Operation. The insertion or deletion of a key K in the PHT relies on the lookup operation to find the proper leaf node. In case of insertion, if the constraint on the number of maximum allowed children is violated, the node has to be split into two nodes, and the keys stored in the original node are distributed across the two novel leaf nodes.

PHT Range Query. In a one-dimensional data domain, a range query can be expressed as follows: given two keys L and H, where $L < H$, a range query shall return all the keys K that satisfy the condition $L \leq K \leq H$.

The PHT search algorithm first defines the smallest prefix that covers the whole specified range. If this prefix is the label of an internal node, then the query is forwarded recursively to the child nodes whose prefix falls within the specified range, until leaf nodes are reached [37].

An analogous approach is adopted for a range query in a d-dimensional domain. Minimal and maximal values for each coordinates are linearized to obtain minimum and maximum derived keys. Then, the maximum common prefix of these derived keys can be obtained. The maximum common prefix thus allows locating the node the search is started from. The search is forwarded recursively to child nodes, selecting those nodes whose prefix falls within the monodimensional range delimited by the minimum and maximum derived keys, until leaf nodes are found.

The above-mentioned search algorithm can return leaf nodes that contain keys whose delinearized d-dimensional value does not fall within the original range query. This fact depends on the locality-preserving property of the adopted SFC curve. Points that are close on a linearized monodimensional space can be distant in the originating d-dimensional one. As shown in Figure 4, a range query in a 2-dimensional domain ($1 \leq x \leq 3$, $1 \leq y \leq 3$) can be mapped into a range query on a 1-dimensional binary domain ($0111 \leq z \leq 1111$) by applying the z-order curve linearization technique, but some results of the 1-dimensional query fall outside the original 2-dimensional query.

As searches of the left and right subtrees can be performed in parallel, the complexity of the algorithm is linearly proportional to the depth of the trie region that has been analysed.

If the query range is increased, the length of the maximal common prefix where the search starts from decreases. Consequently, the cost of the search increases since the trie region to be analysed increases. A typical example is when a range query is specified for a subset of attributes (e.g., x attributes), while for the remaining $d - x$ attributes any value is allowed, or a wild card query is performed. In order to reduce the cost of search, we introduced a mechanism that allows starting the search in parallel branches at deeper levels in the trie, by exploiting a simple pattern matching technique. First, we defined a "relaxed common prefix". Given the minimum and maximum derived keys, L ($l_1 l_2 l_3 \cdots l_k$) and H ($h_1 h_2 h_3 \cdots h_k$), respectively, the relaxed maximal common prefix P ($p_1 p_2 p_3 \cdots p_k$) is built according to Pseudocode 1.

```
Input:
minimum derived key L (l1 l2 l3 · · · lk)
maximum derived key H (h1 h2 h3 · · · hk),
Output:
relaxed maximal common prefix P (p1 p2 p3 . . . pm) with m <= k
counter i ← 0;
counter j ← 0;
WHILE (hi ≥ li AND i < n) DO
{
IF hi = li THEN pj = hi;
IF hi = 1 AND li = 0 THEN pj = . (where. is the wildcard bit);
i + 1;
j + 1;
}
```

Pseudocode 1

For instance, in a bidimensional data domain, the range query $2 \leq x \leq 4$, $2 \leq y \leq 4$ originates the minimum and maximum derived keys 001100 and 110000, respectively. While the maximum common prefix would be null, the "relaxed common prefix" is "..", where the symbol "." represents a wildcard bit (i.e., it matches any value {0, 1}). The search can thus be performed in parallel by starting from the prefixes matching the relaxed common prefix.

4.2.3. DHT. As mentioned above, the PHT indexing scheme is independent from the underlying DHT implementation, since a PHT relies on the basic put() and get() primitives offered by any DHT. The DHT that has been used for this work has been design and developed from scratch based upon the Kademlia specifications [20]. In addition, our DHT implementation supports data replication and versioning. Replicas of each data item are maintained in N nodes of the DHT network, where N can be configured. Replication is needed to improve availability and fault tolerance, while it introduces threats to the consistency of stored data. Our system implements an eventual consistency approach by handling multiple versions of a given data item. Conflicts arising from the presence of multiple versions of a data item in the system at the same time are handled and resolved by using vector clocks [43]. A vector clock is a list of tuples (node address, counter), and each version of an object has a vector clock assigned. Vector clocks can be analysed to infer causality between two events (i.e., different versions of the same object) and thus to decide if two versions of the same object are on parallel versioning branches (i.e., they are conflicting resources). When a versioning conflict is detected, the two resources are reconciled. In the current implementation, a basic reconciliation mechanism is implemented by merging the conflicting versions.

5. Experimentation

We performed a set of experimentation activities in a reference scenario for goods tracing and tracking in a multimodal transport chain.

In this reference scenario, several types of actors are involved, as shown in Figure 5: multimodal transport operators, road transport operators, shipping companies, intermodal terminal operators, and customers (the sender and addressee). These actors typically interact with tracking and tracing services to query for or to insert new information about the status and location of monitored good items. Moreover, also institutional actors (e.g., Port Authorities, Customs systems, etc.) and third-party actors (e.g., banks, insurance companies) can query for information about goods on transit [44].

In this scenario, we defined a set of attributes, in addition to the goods item identifier (i.e., the EPC number), that can usefully characterize the information acquired and stored during the steps of the transport route and that can be used to ease information retrieval tasks for different application purposes.

(i) departure latitude;

(ii) departure longitude;

(iii) arrival latitude;

(iv) arrival longitude;

(v) type of goods (e.g., dangerous goods classification codes).

Our solution does not mandate any specific Object Identifier schema. For this experimentation activity we chose the EPC-64 numbering scheme (an EPC code encoded as a sequence of 64 bits). Latitude and longitude coordinates can be encoded into a sequence of 40 bits each [45]. For the type of goods attribute, we adopted the UN classification schema for dangerous goods, consisting in a 4-digit number, encoded in 16 bits. As the linearization technique has to be applied to sequences of bits of equal length, the shorter sequences are padded with zero bits on the left, so that they have the same length as the longer sequence (i.e., the 64-bit EPC code). The resulting PHT key is thus 384-bit long.

The information record stored for each key is a list of tuples (URL, timestamp), where the URL refers to an information repository storing the information about the target

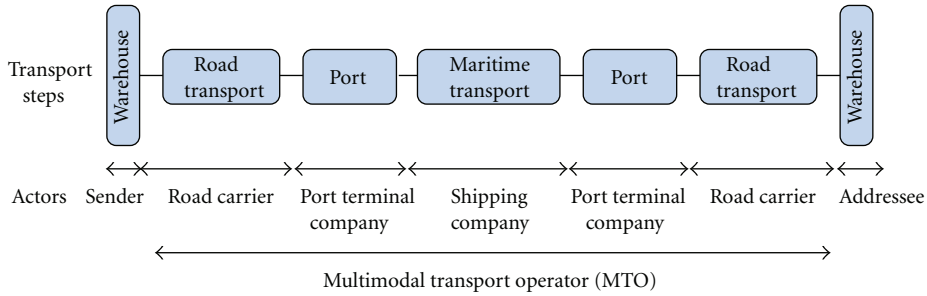

Figure 5: Actors involved in a multimodal transport chain.

object, and the timestamp marks the time when the record has been inserted in the system. As a matter of fact, during an object lifecycle, several information repositories handled by different actors could be associated to the object's identifier.

The API exposed to client applications can thus support the following operations.

(i) Inserting a new object: the client has to provide the values of the selected six attributes. These values are processed by the system to generate the corresponding PHT key. The system inserts the PHT key in the trie-based structure and the associated information (URL and timestamp) in the underlying DHT nodes.

(ii) Adding/deleting an information record associated to an object: the system uses the PHT key associated with the object to locate the information records stored in the DHT and update them by adding/deleting the given record.

(iii) Retrieving the information records for a given object: the system uses the PHT key associated with the provided query input parameters to locate the DHT nodes and retrieve stored information records.

(iv) Retrieving the information records for a set of objects that satisfy a range query over (a set of) attributes: the system exploits the trie-based structure to retrieve the PHT keys that are within the query range and to gather the information objects associated with the retrieved objects.

We performed a set of testing activities in order to analyse the structural properties and performance of the PHT. The object of our testing activities was the PHT overlay and not the underlying DHT implementation, therefore we adopted the testing methodology proposed in [41]. We used a DHT network of 20 nodes running on two physical hosts connected on a LAN environment. Since performance is measured in terms of DHT operations, we can abstract from the network characteristics.

To analyse the structural properties of the PHT indexing scheme, we conducted a set of experiments through computational simulations. We adopted two metrics: the average leaf depth of the trie and the average block utilization, which is calculated as the ratio of the number of elements stored in the leaf node to the value of block B, that is, the maximum number of keys that can be stored in a node [41]. We measured these properties on a data set of progressively increasing size (up to 20,000 keys) populated with randomly generated keys.

Figure 6 shows how the average depth of the leaf nodes varies with the block size. The average depth of the nodes decreases logarithmically with the increase of the block size. This is due to the fact that increasing values of B (block size) results in leaves containing more keys and thus a less deep trie structure. The average depth of leaf nodes increases with the data set size. Figure 6 shows the average depth measured for data sets of size 90 kB, 400 kB, 600 kB, and 1 MB. This behaviour is analogous to the one observed for other over-DHT indexing approaches [41, 46].

As shown in Figure 7, the block utilization exhibits a fluctuating behaviour as the block size increases. The bucket utilization shows how full the leaf nodes are with respect to the maximum allowed block size. Especially for smaller data sets (90 kB and 600 kB), the block utilization value fluctuates as the block size increases. As the data set size increases, the block utilization tends to increase with the block size. These results can be hardly compared with other approaches in the literature, since the behaviour of this structural property may vary with the type of the data set distribution [46].

We measured the performance of insert and lookup operations in the PHT by analysing the number of accesses to the underlying DHT for invoking get() and put() operations. We populated the PHT with 200,000 derived keys, and we measured the number of DHT operations that were performed at each level of the PHT trie for inserting a set of 1,000 uniformly distributed keys. As depicted in Figure 8, results show that higher levels in the trie are seldom accessed, thanks to the adopted binary search approach. Then, we performed a set of range query operations over a population of 200,000 keys stored in the PHT. For each iteration, we varied the range span of the queries (from wider ranges with a common prefix of zero length to an exact match query). Again, the results show that the workload in terms of accesses to the DHT seldom affects the root of the trie (Figure 9).

According to these testing results, our system behaves analogously to other over-DHT indexing approaches [41, 46]. Overlay-dependent indexing schemes can implement more efficient mechanisms for handling complex queries, since they can also modify the underlying routing mechanism, such as Mercury [38]. However, they are usually more

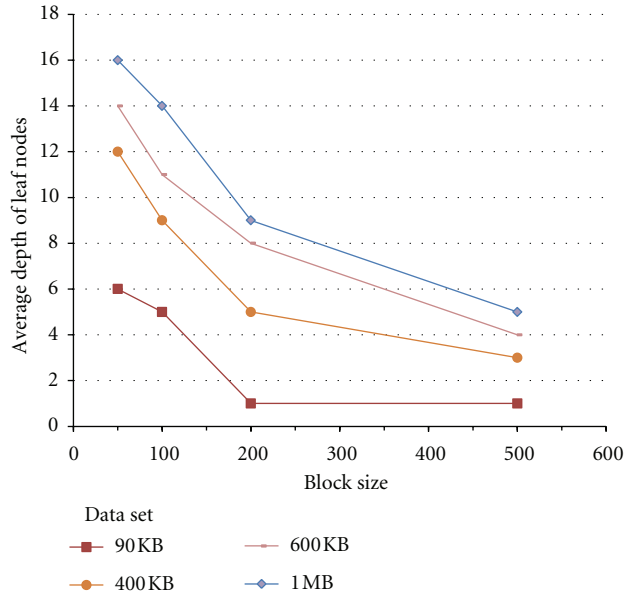

Figure 6: PHT structural properties: average depth of leaf nodes against block size for increasing data sets.

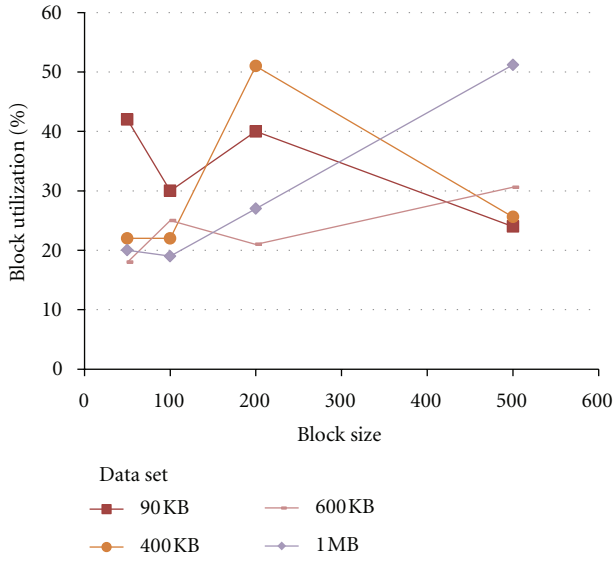

Figure 7: PHT structural properties: block utilization.

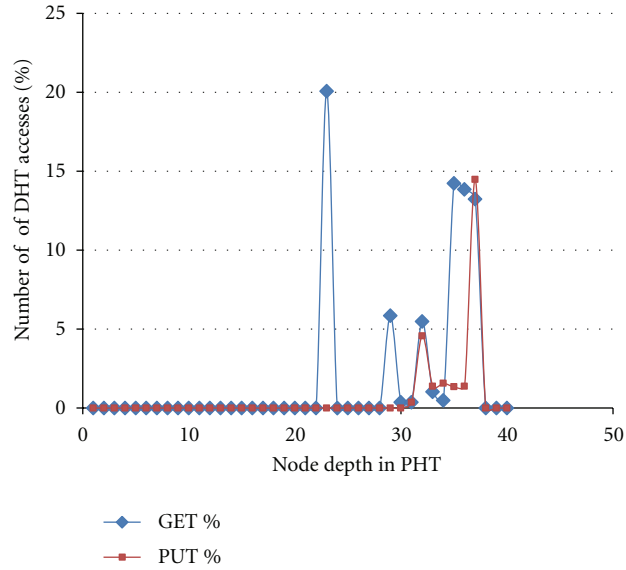

Figure 8: Number of accesses to the DHT against node depth in the PHT for the insert operation.

demanding in terms of complexity of design, development, and maintenance [46].

6. Conclusions

In this paper, we proposed a distributed Discovery Service based on a peer-to-peer overlay network for IoT scenarios. With respect to existing approaches implementing a DHT-based discovery service for the IoT and, in particular, for RFID-based scenarios, our original contribution consists in supporting more complex queries, that is, multiattribute and range queries.

Our design approach was based on the adoption of an over-DHT indexing scheme, thus easing the design of a layered functional architecture. More specifically, our discovery system design is made of the following layers: (a) an SFC linearization technique for mapping a multidimensional domain into a one-dimensional one, (b) a PHT search structure leveraging on a generic DHT get/put interface, (c) a DHT implementation based on the Kademlia algorithm. Although overlay-dependent indexing schemes can be more efficient in handling complex queries, they are usually more demanding in terms of complexity of design, development,

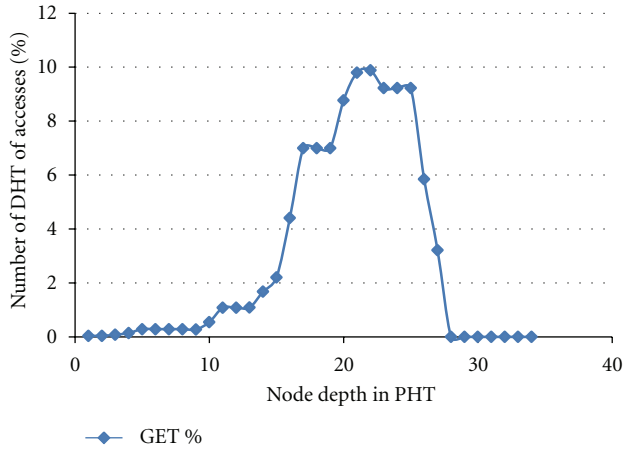

Figure 9: Number of accesses to the DHT against node depth in the PHT for the range query operation.

and maintenance [46]. Instead, our layered solution privileges ease of design and implementation.

We implemented a Proof of Concept in a reference application scenario for dangerous goods monitoring. We thus reported results achieved through experimentation activities regarding structural properties and query performance in an in-laboratory testing configuration. More extensive testing activities are planned in the near future. To this purpose, we are evaluating the possibility of exploiting the capabilities offered by PlanetLab, which is a large-scale distributed testbed [47].

Future research activities will also be devoted to carry out a case study on dangerous goods monitoring within a research project (SITMAR research project), funded by the Italian Ministry for Economic Development. To this purpose, we will also delve into security issues, ranging from secure communication to trust between interacting parties.

Acknowledgment

The authors thank Mr. Luca Capannesi from the University of Florence for his technical support.

References

[1] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Proceedings of the 2nd International Internet of Things Conference (IoT '10)*, pp. 9–129, December 2010.

[2] S. Evdokimov, B. Fabian, S. Kunz, and N. Schoenemann, "Comparison of Discovery Service architectures for the Internet of Things," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC '10)*, pp. 237–244, June 2010.

[3] G. D. Abowd, I. Bobick, I. Essa, E. Mynatt, and W. Rogers, "The aware home: developing technologies for successful aging," in *Proceedings of the American Association of Artificial Intelligence Conference*, 2002.

[4] F. Paganelli and D. Giuli, "An ontology-based system for context-aware and configurable services to support home-based continuous care," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 2, pp. 324–333, 2011.

[5] F. Paganelli and D. Giuli, "A context-aware service platform to support continuous care networks," in *Proceedings of the 4th International Conference on Universal Access in Human-Computer Interaction (UAHCI '07)*, vol. 4555, part 2 of *Lecture Notes in Computer*, pp. 168–177, 2007.

[6] J. E. Bardram, "Applications of context-aware computing in hospital work—examples and design principles," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 1574–1579, March 2004.

[7] F. Paganelli, E. Spinicci, A. Mamelli, R. Bernazzani, and P. Barone, "ERMHAN: a multi-channel context-aware platform to support mobile caregivers in continuous care networks," in *Proceedings of the IEEE International Conference on Pervasive Services (ICPS '07)*, pp. 355–360, July 2007.

[8] F. Paganelli and D. Giuli, "An ontology-based context model for home health monitoring and alerting in chronic patient care networks," in *Proceedings of the 21st International Conference on Advanced Information Networking and ApplicationsWorkshops/Symposia (AINAW '07)*, pp. 838–845, May 2007.

[9] F. Paganelli, M. C. Pettenati, and D. Giuli, "A metadata-based approach for unstructured document management in organizations," *Information Resources Management Journal*, vol. 19, no. 1, pp. 1–22, 2006.

[10] A. Kocurova, S. Oussena, P. Komisarczuk, and T. Clark, "Context-aware content-centric collaborative workflow management for mobile devices," in *Proceedings of the 2nd International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA '12)*, pp. 54–57, IARIA.

[11] A. García-Crespo, J. Chamizo, I. Rivera, M. Mencke, R. Colomo-Palacios, and J. M. Gómez-Berbís, "SPETA: social pervasive e-Tourism advisor," *Telematics and Informatics*, vol. 26, no. 3, pp. 306–315, 2009.

[12] F. Paganelli, G. Bianchi, and D. Giuli, "A context model for context-aware system design towards the ambient intelligence vision: experiences in the eTourism domain," in *Proceedings of the 9th ERCIM Workshop "User Interfaces For All", Special Theme: 'Universal Access in Ambient Intelligence Environments'*, Lecture Notes in Computer Science, Springer, Königswinter, Germany, September 2006.

[13] D. Giuli, F. Paganelli, S. Cuomo, and P. Cianchi, "A systemic and cooperative approach towards an integrated infomobility system at regional scale," in *Proceedings of the IEEE International Conference on ITS Telecommunications (ITST '11)*, pp. 547–553.

[14] J. Santa and A. F. Gómez-Skarmeta, "Sharing context-aware road and safety information," *IEEE Pervasive Computing*, vol. 8, no. 3, pp. 58–65, 2009.

[15] S. Turchi, L. Ciofi, F. Paganelli, F. Pirri, and D. Giuli, "Designing EPCIS through linked data and REST principles," in *Proceedings of the International Conference on Software, Telecommuniccations and Computer Networks (SoftCOM '12)*, Split, Croatia, September 2012.

[16] D. Parlanti, F. Paganelli, and D. Giuli, "A service-oriented approach for network-centric data integration and its application to maritime surveillance," *IEEE Systems Journal*, vol. 5, no. 2, pp. 164–175, 2011.

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

[18] K. Finkenzeller, *RFID Handbook*, Wiley, 2003.

[19] W. Yao, C. H. Chu, and Z. Li, "Leveraging complex event processing for smart hospitals using RFID," *Journal of Network and Computer Applications*, vol. 34, no. 3, pp. 799–810, 2011.

[20] I. Zappia, F. Paganelli, and D. Parlanti, "A lightweight and extensible Complex Event Processing system for sense and respond applications," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10408–10419, 2012.

[21] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, *Vision and Challenges for Realising the Internet of Things*, Cerp-IoT Cluster of European Research Projects on the Internet of Things, European Commission, 2010.

[22] F. Thiesse, C. Floerkemeier, M. Harrison, F. Michahelles, and C. Roduner, "Technology, standards, and real-world deployments of the EPC network," *IEEE Internet Computing*, vol. 13, no. 2, pp. 36–43, 2009.

[23] B. Fabian and O. Günther, "Security challenges of the EPCglobal network," *Communications of the ACM*, vol. 52, no. 7, pp. 121–125, 2009.

[24] EPCglobal, Object Name Service (ONS) 1.0.1, Ratified Standard Specification with Approved, Fixed Errata, 2008.

[25] EPCGlobal, http://www.gs1.org/gsmp/kc/epcglobal.

[26] BRIDGE Project, "Working prototype of serial-level lookup service," 2008, http://www.bridge-project.eu/data/File/BRIDGE_WP02_Prototype_Serial_level_lookup_service.pdf.

[27] U. Barchetti, A. Bucciero, M. De Blasi, L. Mainetti, and L. Patrono, "Implementation and testing of an EPCglobal-aware discovery service for item-level traceability," in *Proceedings of the International Conference on Ultra Modern Telecommunications and Workshops (ICUMT '09)*, pp. 1–8, October 2009.

[28] M. Young, "Extensible supply-chain discovery service concepts (Draft 04)," Internet Draft, IETF, 2008.

[29] N. Schoenemann, K. Fischbach, and D. Schoder, "P2P architecture for ubiquitous supply chain systems," in *Proceedings of the 17th European Conference on Information Systems*, pp. 2255–2266, 2009.

[30] S. Shrestha, D. S. Kim, S. Lee, and J. S. Park, "A peer-to-peer RFID resolution framework for supply chain network," in *Proceedings of the 2nd International Conference on Future Networks (ICFN '10)*, pp. 318–322, January 2010.

[31] P. Manzanares-Lopez, J. P. Muoz-Gea, J. Malgosa-Sanahuja, and J. C. Sanchez-Aarnoutse, "An efficient distributed discovery service for EPCglobal network in nested package scenarios," *Journal of Network and Computer Applications*, vol. 34, no. 3, pp. 925–937, 2011.

[32] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[33] I. Stoica, R. Morris, D. Liben-Nowell et al., "Chord: a scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[34] A. Rowstron and P. Druschel, "Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01)*, pp. 329–335, Springer, London, UK, 2001.

[35] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.

[36] P. Maymounkov and D. Mazieres, "Kademlia: a peer-to-peer information system based on the XOR metric," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '01)*, pp. 53–65, Springer, London, UK, 2002.

[37] S. Ramabhadran, S. Ratnasamy, J. M. Hellerstein, and S. Shenker, "Brief announcement: prefix hash tree," in *Proceedings of the 23rd annual ACM symposium on Principles of Distributed Computing (PODC '04)*, pp. 368–368, ACM, New York, NY, USA.

[38] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," in *Proceedings of the Conference on Computer Communications (ACM SIGCOMM '04)*, pp. 353–366, September 2004.

[39] M. Cai, M. Frank, J. Chen, and P. Szekely, "MAAN: a multi-attribute addressable network for grid information services," *Journal of Grid Computing*, vol. 1, pp. 3–14, 2004.

[40] C. Schmidt and M. Parashar, "Squid: enabling search in DHT-based systems," *Journal of Parallel and Distributed Computing*, vol. 68, no. 7, pp. 962–975, 2008.

[41] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, S. Shenker, and J. Hellerstein, "A case study in building layered DHT applications," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '05)*, pp. 97–108, ACM, New York, NY, USA.

[42] J. K. Lawder and P. J. H. King, "Using space-filling curves for multi-dimensional Indexing," in *Proceedings of the 17th British National Conferenc on Databases: Advances in Databases (BNCOD '00)*, Springer, London, UK, 2000.

[43] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[44] Bollen et al., Sea and Air Container Track and Trace Technologies: Analysis and Case Studies, Project NO. TPTT01/2002T, APEC, July 2004, http://www.apec-tptwg.org.cn/new/archives/tpt-wg24/safe/its/itf-track-trace.pdf.

[45] IETF Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information, Request for Comments: 3825, http://www.ietf.org/rfc/rfc3825.txt.

[46] Y. Tang, S. Zhou, and J. Xu, "LIGHT: a query-efficient yet low-maintenance indexing scheme over DHTs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 1, pp. 59–75, 2010.

[47] E. Jaffe and J. Albrecht, "PlanetLab—P2P testing in the wild," in *Proceedings of the 9th International Conference on Peer-to-Peer Computing (IEEE P2P '09)*, pp. 83–84, September 2009.