UbiQuest: A P2P-based Resource Discovery System for Ubiquitous Computing*

Seungjae Lee[1], Jeongkeun Lee[1], Jaeyoung Choi[1], Taekyung Kwon[1], Yanghee Choi[1], and YongWan Ju[2]

[1] School of Computer Science and Engineering
Seoul National University, Seoul, Korea
{sjlee, jklee, jychoi, tk, yhchoi}@mmlab.snu.ac.kr

[2] National Internet
Development Agency of Korea
ywju@nida.or.kr

Abstract. Users in ubiquitous computing environments need to find and utilize ambient resources (ubiquitous services or devices which provide ubiquitous services). In ubiquitous computing, users and resources are highly mobile, and furthermore, the attributes of ubiquitous resources may change dynamically; the static/centralized resource management systems such as conventional DNS are inappropriate for ubiquitous computing. The new resource discovery system should be independent of specific protocols, service providers and types of devices, and should be capable of finding resources only by using the attribute information of the wanted resource. This paper presents a novel resource discovery system, UbiQuest. UbiQuest has a simple and flexible architecture and provides a general-purpose/semantic-free resource discovery service. To support distributed resource management and discovery, UbiQuest adopts a DHT-based overlay network architecture. Despite the inherent rigidness of DHT, UbiQuest supports range queries and flexible search by using logical operators as well as the discovery of dynamically changing resources. The performance of the UbiQuest system is evaluated by simulation experiments, which reveal that UbiQuest can handle dynamically changing resources more efficiently.

Keywords: Ubiquitous Computing, Resource Search, DHT

1 Introduction

In ubiquitous computing environments, a user or an application program should be able to find nearby services or devices without a need to recognize a network or a computer. For example, if a laptop user has to manually find and install a printer driver to print a document wherever the user visits a new place, it will make the user uncomfortable.

To address this issue, we focus on a general-purpose resource discovery system, in which a user can discover various services published by ubiquitous devices. For example, a printing program in a laptop can automatically send a document to the

* This work was supported by NIDA (2005) and KOSEF project (2006).
The ICT at Seoul National University provides research facilities for this study.

nearest available network printer, or to the most appropriate printer depending on application's requirements or user preference. An MP3 player should be able to find the nearest stereo speaker when its user wants to hear music with his friends. Similarly, a traveler can acquire information about menu and price of nearby restaurants by means of the resource discovery system. Consequently, we need a resource discovery system, based on properties rather than static location information, to find dynamically changing resources in ubiquitous environments.

In such a resource discovery system, a common language that describes properties of a resource is necessary. For example, the Semantic Web technology (which is designed to offer an intelligent service based on the Internet) enables machines to understand and infer the meaning of information by Ontology, Resource Description Framework (RDF), Ontology Web Language (OWL) and etc. Not only vocabulary but also semantics, a classification system, the relationship between properties have to be dealt with Ontology. Besides, Ontology should have been standardized and shared among information providers, information consumers and the Web systems.

But it appears not easy to define and share a unified Ontology in ubiquitous networks. While powerful computers are interconnected with high-speed networks in the Internet, cheap and simple devices should communicate using wireless channels in ubiquitous networks. A vast and almighty ontology is too heavy to be handled by such devices with limited capability. The ontology compatibility problem among heterogeneous devices and information providers also can be an obstacle. Therefore, to provide a resource discovery system for an open ubiquitous service in the near future, a simple protocol is considered to be a more practical solution rather than a rich ontology-based protocol.

The following is a brief summary of requirements that a ubiquitous resource discovery system has to satisfy.

- It should be a semantic-free system, which allows a user to perform discovery without a need to know a complex metadata format.
- It should operate in a distributed manner. A centralized server is a single point of failure and too costly to deploy in ubiquitous environments.
- Various type of search queries has to be supported to pinpoint a discovery target, including inequality sign and logical operators (AND, OR).
- It should cope with a dynamic change of resource properties. A ubiquitous resource or a user may join or leave a network frequently, because of their mobility. Moreover, even a fixed resource may have a dynamically changing property. In a sensor network, for instance, temperature measured by a sensor changes over time. A resource discovery system should efficiently deal with such a case.

In this paper, we propose a resource discovery system that satisfies the above requirements, dubbed *UbiQuest*, and evaluate its feasibility by simulation. It offers distributed resource publication and discovery service based on a dynamic hash table (DHT) overlay network. In UbiQuest, ontology is not needed to be shared among resource providers and clients because a resource is described in a plain and semantic-free format.

The rest of this paper is organized as follows. The related work is presented in Section 2. In Section 3 we describe a general purpose and semantic-free resource description that enables resource discovery without ontology. We illustrate system architecture, resource publication and discovery procedure of UbiQuest in Section 4. The performance of UbiQuest is evaluated in Section 5, which is followed by the conclusion section.

2 Related Work

Studies about ubiquitous resource discovery can be classified into two categories: one for internet-oriented next generation resolution system and the other for property-based publication and discovery system. The former is represented by general-purpose distributed resolution systems. Its goal is to replace the existing resolution system like DNS to support new ubiquitous services and to enable convergence of various services and networks [3] [4] [6]. Especially, Intentional Naming System [6] suggested an identification system and a resolution architecture based on the properties of resources, not their locations. It laid the foundation of the next generation resolution system in the Internet.

The studies stated above are about resources in the Internet, different from ubiquitous resource discovery service. Balakrishnan et al. [5] proposed to use a flat identification system for services and endpoints. He argued that DNS is not suitable for a general resolution architecture and suggested a semantic-free referencing (SFR) architecture based on DHT. SFR provides a basis of an identification system for general purpose ubiquitous resource discovery.

INS/Twine offers a publication and search service for the ubiquitous resources based on INS. The INS/Twine system achieves scalability by scattering resource information to multiple resolvers. But it doesn't provide any mechanism to handle the circumstances that the properties or values of resource are changed dynamically. [8] introduced a resource publication system which utilizes the context information to improve search speed and efficiency. However, since it necessitates a well pre-defined ontology, it is very hard to realize. [10] suggested a group-based resource search protocol that relies on a hierarchical ontology. Though it is more efficient than a broadcast-based search, it needs the ontology, too.

Studies about range queries are carried out by [1] [2] [11], and they focus on searching a key among a specific range in a DHT-based p2p system. DST [11] which is the most recent study among these supports not only a range query but also a cover query that inquires the extent of a specific key. Although they provide a better performance in range queries for static properties by using the special overlay architecture or data structure, they cannot support searching dynamic properties.

3 Resource and Query Description

In this section, we introduce the format of resource description and search queries, which lay the foundation of our UbiQuest resource discovery system.

In Figure 1 (a), a resource description consists of a set of *property entries*. Each property entry is a pair of fields; <*attribute*, *value*>. Participants of the system define and use their own attributes in a proprietary format instead of conforming to common ontology.

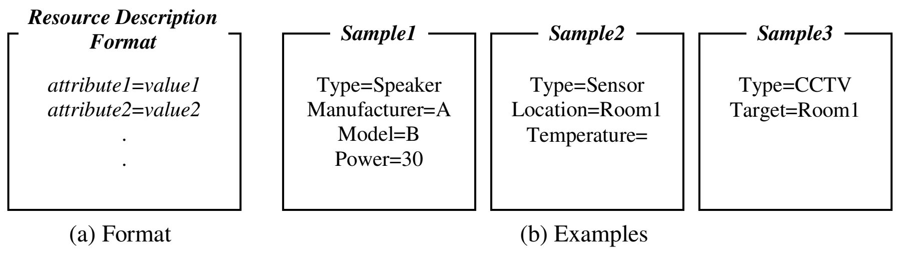


Fig. 1. Resource Description Format and Examples.

Figure 1 (b) shows some examples of resource description. A resource provider may leave a value field empty when publishing the resource into the system. For example, the second property entry of the ‘Temperature Sensor’ resource description in Figure 1 (b) has only an attribute field while the value field is empty. By not specifying the value, we avoid the overhead of republishing resources whose value is changing dynamically such as ‘temperature’.

```
<rel_op> ::= < | > | <= | >= | = | !=
<log_op> ::= AND | OR
<search_term> ::= attribute <rel_op> value | attribute | value
<search_query> ::= (<search_query> <log_op> <search_query>) | <search_term>
e.g. Type=Speaker AND (Model=M1 OR Model=M2)
     Sensor AND (Temperature>15 AND Location=Room1>
     Room1
```

Fig. 2. BNF of Resource Search Query Format.

When a user in the UbiQuest system wants to search a resource, he/she should describe the properties of the desired resource in the format of Figure 2. UbiQuest client parses the search query and converts it to search keywords which will be used by UbiQuest’s DHT to find corresponding resources. As the resource description does, the search query may have an empty field in attribute-value pairs.

4 Architecture of the UbiQuest System

The architecture and components of the UbiQuest system will be described in this section as well as the detailed procedures of resource publication and discovery.

4.1 Components

Figure 3 illustrates the architecture of the UbiQuest system. The UbiQuest system is composed of a number of *Resource Discovery Agents (RDAs).* RDAs receive resource publication requests, manage resource descriptions in the UbiQuest system, and respond to search queries in a distributed and cooperative manner. Each RDA has information about the subset of other RDAs participating in the same peer-to-peer overlay network to form a DHT lookup system. RDAs may either exist in ubiquitous computing nodes that require resource publication/discovery functionality, or be deployed as standalone service entities.

A ubiquitous user or application, that uses UbiQuest service, is called as a *client*. It is connected to one or more RDAs and publishes and/or searches for resources on the UbiQuest system through RDAs. For the sake of convenience, we will call a client that provides and publishes resources through RDAs as an *R-client* and a client that makes a request for searching as a *Q-client*. But there is no physical distinction between R-clients and Q-clients in actual UbiQuest system.

Likewise, we distinguish RDAs by their tentative roles they play at a specific point of time; *R-RDA*: An RDA that is delegated by an *R-Client* to publish and provide resource description. *Q-RDA*: An RDA that receives a search query and performs a search procedure on behalf of a Q-client. *K-RDA*: An RDA that stores and manages some portion of entire resource descriptions in the system and responds to search queries which falls into the responsible DHT key range.

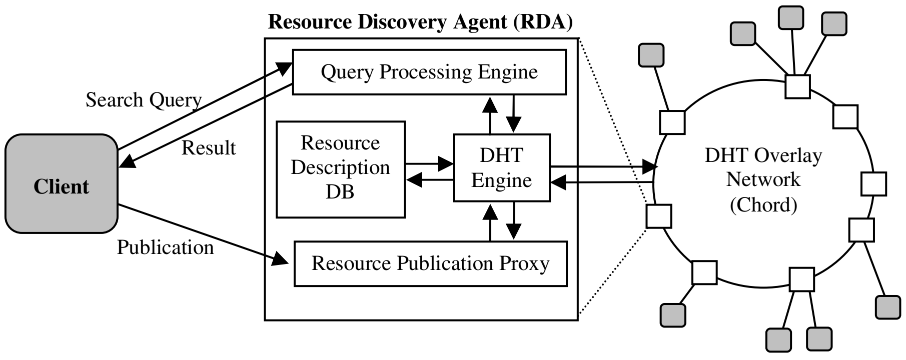


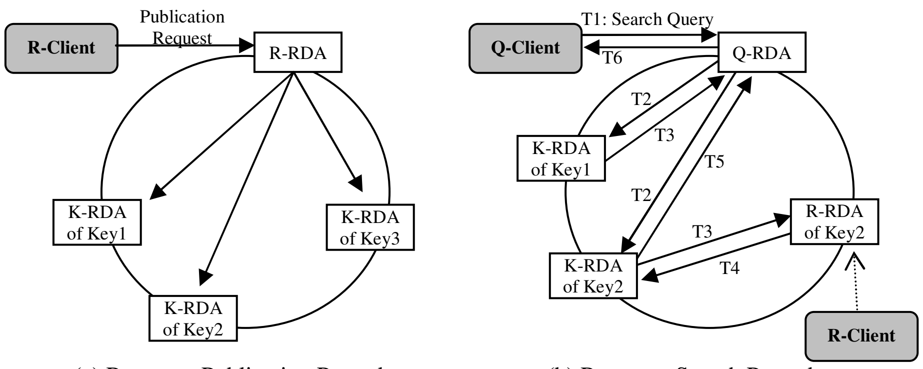
Fig. 3. UbiQuest Architecture.

4.2 Resource Publication Procedure

When an R-RDA receives a resource publication request from an R-client, the R-RDA makes three DHT keys using the same hash function with three inputs: "<attribute>", "<value>", "<attribute>=<value>" for each property entry. In the second example in Figure 1 (b), seven DHT keys can be created using the following keywords: "Type", "Sensor", "Type=Sensor", "Location", "Room1", "Location=Room1" and "Temperature". But a keyword which appears too frequently, like "Type" in this case, is not used to publish the resource because it will be associated with too many resources. For a dynamically changing property, an R-RDA creates only one DHT key using "<attribute>" because an R-client does not specify the value field. Dynamic values, which are not specified to K-RDAs at publication time, will be asked later when a Q-client initiates a search query that checks the dynamic property.

Because an R-RDA makes three DHT keys from the <attribute, value> pair to publish a resource, a user has a good chance to find resources even without knowing about resource description format for the attribute field. In Figure 1 (b), a user can find all information only with "Room1" keyword even if he/she knows nothing about the "Location" attribute field. But knowing that the attribute of the resource property entry is "Location" the user can search with "Location=Room1" keyword, which results in more efficient search in the sense that there will be fewer associated resources.

After generating DHT keys, the R-RDA sends resource publication messages that contain a full description of the resource and a network ID (typically IP address) of the R-client into the DHT network using each DHT key.



(a) Resource Publication Procedure (b) Resource Search Procedure

Fig.4. Examples of detailed publication/search procedure. In (b), the search query is "Location=Room1 OR Temperature>15", and Temperature is a dynamically changing attribute.

The message would arrive at the K-RDA in charge in charge of the key by DHT routing mechanism (e.g. Chord [9]). The K-RDA stores all information in the message in its resource description DB. As shown in Figure 4, a resource description will be duplicated in all K-RDAs which are responsible for DHT keys generated from the resource description. This feature is an overhead of storage and bandwidth in publication-time, but it allows efficient and scalable search.

4.3 Resource Searching Procedure

When a Q-RDA receives a search query message from a Q-client, it makes one or more appropriate DHT keys from the query and forwards the query to the corresponding K-RDAs through the DHT overlay network. And then, on receipt of the query, K-RDAs make a list of resources that satisfy the query and send back to the Q-RDA. Finally, the Q-RDA refines and returns the result of the search query message from K-RDAs to the Q-client.

In the case that a Q-RDA receives a query which consists of terms (*search_term* in Figure 2) that are linked with AND operators, the Q-RDA picks a representative term that is expected to minimize overhead and delay. The query is forwarded to a K-RDA that services a hash value of the selected representative term in the DHT network. For a case of “Speaker AND (Model=M1 OR Model=M2)”, in which “Speaker” is chosen as a representative term, a K-RDA which is responsible for hash(“Speaker”) will receive the query (which is same as the query Q-RDA has received) and return the list of resources that satisfy all terms of the query to the Q-RDA. If the “Speaker” keyword is considered to be too frequent, the Q-RDA will choose one of the other terms as a representative term.

In the case that a Q-RDA receives a query which consists of terms that are linked with OR operator, the Q-RDA generates keys for each term and forwards them to corresponding K-RDAs simultaneously. Responses from multiple K-RDAs are merged by the Q-RDA and returned to the Q-client. In the last example in Figure 1 (b), both “Model=M1” and “Model=M2” are selected as representative terms when “Speaker” isn’t selected. A Q-RDA will generate two keys (hash(“Model=M1”) and hash(“Model=M2”)), and forward the query to the two corresponding K-RDAs.

A search term that has a relational operator except an equal sign, like “Power>30”, cannot be used as a representative term, because resources are published only with equal signs. That is, a term which contains an attribute field or a value field only (like “<attribute>”) or an equal sign (like “<attribute>=<value>”) is selected as a representative term.

For dynamically changing resource, a resource property will be published as a form of “<attribute>”, not a form of “<attribute>=<value>” or “<value>”. Therefore, as a K-RDA receives a query for a dynamically changing resource, it asks all the corresponding R-RDAs of the current value of the specific resource.

A resource search procedure including a query about dynamic property is illustrated in Figure 4 (b). T1~T6 tags on messages represent the timestamp values. Key1 is generated from “Location=Room1” term, and Key2 is from “Temperature” term. The query is forwarded to two K-RDAs using these two keys (T2). To find resources satisfying the query, a K-RDA that serves Key2 sends a request to R-RDAs that are delegated by R-clients providing “Temperature” attribute (T3-T5).

5 Evaluation

We implemented the UbiQuest on a discrete event simulator written in C++ and performed comprehensive simulations to evaluate performance. For the underlying DHT engine, we adopted Chord which is the most widely studied DHT scheme. The number of RDAs that participate in UbiQuest is fixed at 1,000. To remove the influence of the DHT initiation on the simulation results, we start evaluating after the RDAs have finished forming an overlay network.

For comparison purposes, we compare UbiQuest with INS/Twine [7], which offers a ubiquitous resource publication and search service based on DHT. In INS/Twine, a resource is described hierarchically based on some kind of ontology like an XML document. Because an attribute-value pair of UbiQuest can be converted to a tree of depth 2 in INS/Twine, we limited height of resource description tree to 2 in simulation with INS/Twine. We assumed that a resource is republished immediately when it is changed in INS/Twine, because it has no consideration for handling dynamically changing resources.

MP3 tag data and artificially generated data are used as input of simulation. For experiments about dynamically changing resources or range query, we generated artificial data to control characteristic of resource properties. Otherwise we used MP3 tag data of ID3v1 [12] format extracted from 15GB music files. (e.g. {title=Feel So Good, artist=Chuck Mangione, album=…})

The network model is as follows:

a) Every node is connected to a backbone network with a full-duplex dedicated line. The line can send/receive 100 messages per second with delay of 10ms.
b) A backbone network has infinite bandwidth, and a message takes 100ms to pass through the backbone network.
c) Every message is assumed to have identical size.

Most of messages used in UbiQuest are small because they include only a resource description or a query statement. But when a client sends a very general query that produces a very large response set, a response message may be too long. Such a case is assumed to be blocked at resource publication time, because a property that occurs too frequently is too meaningless to be published.

Figure 5 shows the distribution of the number of messages per publication and per search for non-dynamic resources, respectively. While most of the resources generate 60-80 messages per publication in INS/Twine, 100-120 messages are generated in UbiQuest. This is because each entry of the MP3 data is published twice in INS/Twine ("<attribute>" and "<attribute>=<value>") and three times in UbiQuest. The additional message overhead ("<value>") of UbiQuest at publication time allows clients to search without knowing the attribute format at discovery time. On the other hand, message overhead at search time is almost same in INS/Twine and UbiQuest.

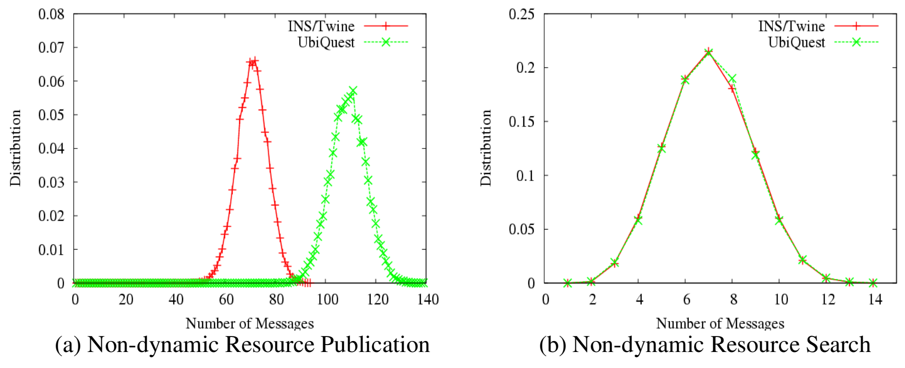
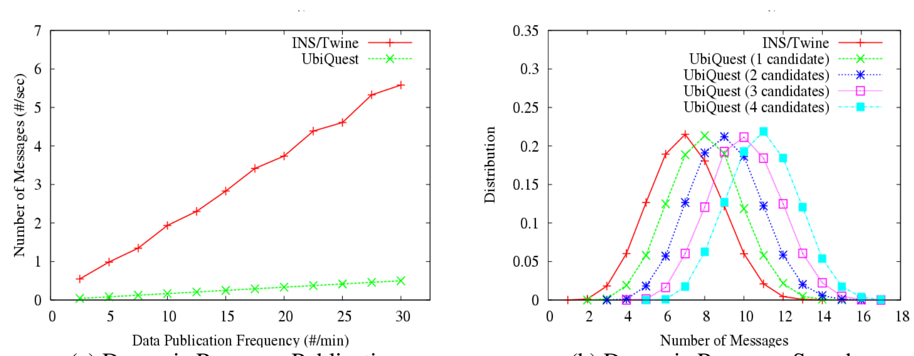


Fig. 5. Message overhead for non-dynamic resources. MP3 data is used (5-7 property entries per resource). 'Number of Messages' means the total number of messages transmitted in the system in the application layer (the number of forwarded messages from intermediate nodes in the DHT network is included).



(a) Dynamic Resource Publication (b) Dynamic Resource Search

Fig. 6. Message overhead for dynamic resources. Artificial data is used (1 property entry per resource). "Candidate" means an R-RDA which published the attribute specified in the search query.

Table 1. Performance of Complicated Logical Search Queries

Query Structure	Number of Messages	Delay (msec)
A	7.01	720.72
A&B	7.09	730.44
A&B&C	7.02	722.64
A&B&C&D	7.00	719.40
A | B	13.98	826.82
A | (B&C)	13.99	826.05
(A&B) | (C&D)	13.97	822.95
A | B | C	21.16	891.69
A | B | C | D	27.70	908.85

Figure 6 (a) represents the distribution of the number of messages per publication for a dynamic resource. The message overhead linearly increases with the data publication frequency in both INS/Twine and UbiQuest. However, the overhead of INS/Twine increases much steeper than that of UbiQuest, because a DHT lookup is needed whenever a changed property is published in INS/Twine, while only one message from the R-client to the R-RDA is sufficient for a changed property in UbiQuest.

Figure 6 (b) shows the number of messages generated to process a query that specifies a value for dynamic resources. In UbiQuest, there is additional message overhead in proportion to the number of R-RDAs that published the attribute, but still most of the message overhead is due to DHT lookup. If there are many highly dynamic resources and queries are infrequent, UbiQuest will outperform INS/Twine.

Table 1 shows the performance of complicated logical search queries in the UbiQuest system. We can see that the number of terms connected by AND operators does not affect the overall performance. This is because the Q-RDA selects only one representative term among the terms and the K-RDA checks the rest of them. On the other hand, both the number of messages and delay increase with the number of terms connected by OR operators, because every term should be forwarded to corresponding K-RDAs. However, delay is less affected than the number of messages, since the terms connected by OR operators are concurrently processed by multiple K-RDAs.

6 Conclusion

Many previous studies have tried to reduce the overhead of a resource discovery procedure using complicated ontology and ontology-based algorithms. But it is hard to define, share, and update ontology in actual ubiquitous environment. UbiQuest, a distributed, semantic-free and ontology-free discovery system based on DHT, is proposed in this paper. Range queries, flexible search by using logical operators and efficient dynamic resource handling are also supported in UbiQuest. Extensive simulation results show that UbiQuest performs efficient attribute-based resource discovery although UbiQuest does not rely on rich ontology-based protocols.

References

1. A. R. Bharambe, M. Agrawal, and S. Seshan, “Mercury: Supporting scalable multiattribute range queries,” in Proceedings of the ACM SIGCOMM 2004, Portland, USA, Sept. 2004.
2. Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, J. Hellerstein, and S. Shenker, “A case study in building layered DHT applications,” in Proceedings of the ACM SIGCOMM 2005, Philadelphia, USA, Aug. 2005.
3. V. Ramasubramanian and E. G. Sirer, “The Design and Implementation of a Next Generation Name Service for the Internet,” in Proceedings of the ACM SIGCOMM 2004, Portland, USA, Sep. 2004.
4. H. Balakrishnan, et al., “A Layered Naming Architecture for the Internet,” in Proceedings of the ACM SIGCOMM 2004, Portland, USA, Aug. 2004.
5. H. Balakrishnan, S. Shenker, and M. Walfish, “Semantic-Free Referencing in Linked Distributed Systems,” in Proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, Feb. 2003.
6. W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," in Proceedings of 17th ACM Symposium on Operating Systems Principles, pp. 186-201, Charleston, USA, Dec. 1999.
7. M. Balazinska, H. Balakrishnan, and D. Karger, “INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery,” in Proceedings of Pervasive 2002 - International Conference on Pervasive Computing, Zurich, Switzerland, Aug. 2002.
8. T. Gu, et al., “A Peer-to-Peer Architecture for Context Lookup,” in Proceedings of IEEE Mobiquitous 2005, San Diego, USA, Jul. 2005.
9. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan. “Chord: A scalable P2P lookup service for internet applications,” In Proceedings of ACM SIGCOMM 2001, San Diego, USA, Aug. 2001.
10. D. Chakraborty et al., “Toward Distributed Service Discovery in Pervasive Computing Environments,” IEEE Transactions of Mobile Computing, Vol.5, No.2, Feb. 2006.
11. C. Zheng, G. Shen, S. Li and S. Shenker, “Distributed Segment Tree: Support of Range Query and Cover Query over DHT,” 5th International Workshop on Peer-to-Peer Systems (IPTPS '06), Berkeley, USA, Feb. 2003.
12. http://www.id3.org/id3v1.html