Vol.2010-MPS-79 No.5
2010/7/12

P2P 環境における ファセット検索のためのデータ配置

渡辺知恵美[†]　　斎藤真衣[†]

近年 P2P 技術の発達及び普及により，様々なデータが P2P ネットワークによって共有されるようになってきている．このような環境においてデータの検索は重要であり，近年ではキーワード検索以外にも全文検索や SQL による問合せなども提案されている．しかしながらユーザが明確な問合せの意図を持たずどのようなデータがあるかを閲覧したい場合には対応することができない．ユーザが明確な検索意図を持たない場合，データに含まれる値のリストを表示しながら対話的にユーザがほしいデータを絞り込む手法としてファセット検索が提案されているが，この検索方式が P2P に適用されている例はこれまでにない．
そこで我々は P2P 環境におけるファセット検索インタフェースを提案し，効率的にファセット検索を行うためのデータ配置戦略を設計した．

Data Assignment for Faceted Search in P2P Environment

Chiemi Watanabe[†]　and Mai Saito[†]

Because of the development of technologies for a P2P network, we can share various types of data such as relational tables and xml data and can query the data not only by using a simple keyword search but also by using an SQL-like query expression. Applications that are developed by using these technologies should have a query interface so that users can easily reach any information they want. In particular, in a P2P network, when there is no rule for adding the metadata of the objects, it is not easy to find appropriate keywords for search. In this study, we focus on ―faceted search,” which is a design pattern that helps query behaviors, and we investigate data architectures and data assignment strategies for processing the faceted search in a P2P network.

1. Introduction

Recently, sharing data in the Peer to peer (P2P) network environment has become popular. In a P2P environment, the query interface is important for finding the data that users are looking for. Many applications have a keyword search interface. A distributed hash table (DHT) mechanism is appropriate for a keyword search because the data are allocated according to the hash value of the keyword corresponding to the data. PIER [2] proposes the mechanism for sharing relational data in a structured P2P environment; further, a user can specify query statements by using SQL. In addition, many search techniques such as range query [8] and full text search [5] have been developed thus far. By using these techniques, a user can specify various types of queries to find the required data in a P2P network.

However, the existing P2P applications do not suppose that users may only have ambiguous images about what they want and that they cannot find appropriate keywords for search. There are also cases when the users may not have any target data; they may just want to look at what data are shared. In such cases, the query interfaces of applications should support the users in finding the objects that the users are looking for.

We propose an interactive query mechanism in a P2P environment for users who cannot find appropriate query keywords to find what they want. We focus on ―faceted search,” which is a technique for accessing a collection of data represented by using a faceted classification, thereby allowing the users to explore the data by filtering the available information. A faceted classification system allows the assignment of multiple classifications to an object, enabling the classifications to be ordered in multiple ways rather than in a single, pre-determined, taxonomic order.

In this paper, we propose a faceted search mechanism for a P2P environment. We apply the proposed mechanism in a structured P2P network by using DHT. The features of faceted search are as follows: (1) Faceted search is interactive. Users send queries multiple times according to the results of the previous query. (2) Aggregations are used for generating faceted values. Aggregate operations are difficult to process in a P2P network. DHT algorithms such as Chord and Pastry are designed for finding an exact match to the query keywords, and these algorithms are not good at processing aggregate operations. In a P2P environment, aggregate operations are processed by broadcasting. In a P2P environment, aggregate operations are processed by broadcasting. However, it is not practical that the system carries out multiple broadcasting through all nodes on the P2P network every time the users send their queries. Therefore, we propose a data allocation strategy that can efficiently

[†] お茶の水女子大学　大学院　人間文化創成科学研究科
Graduate School of Humanities and Sciences, Ochanomizu University, Tokyo, Japan

process multiple aggregate operations for faceted searches. Next, we propose a cashing strategy to process queries interactively and efficiently.

2. Preliminaries

2.1 Faceted Search

Faceted search interfaces have recently been used by various applications. Figure 1 shows a snapshot of a DBLP [9] faceted search interface. DBLP is a database of computer science bibliography. The left side of Figure 1 shows the result of the keyword query "author:chiemi_watanabe." The result is a list of papers written by the author. In addition, several facets are displayed on the right side of the page. In a faceted search, the term facet refers to an aspect by which the target object can be classified into several groups. In Figure 1, coauthors, conferences, and publishing years are selected as facets. In each facet column, facet elements are listed. When a user clicks a facet element, the objects are filtered by the entity in addition to the query keyword.

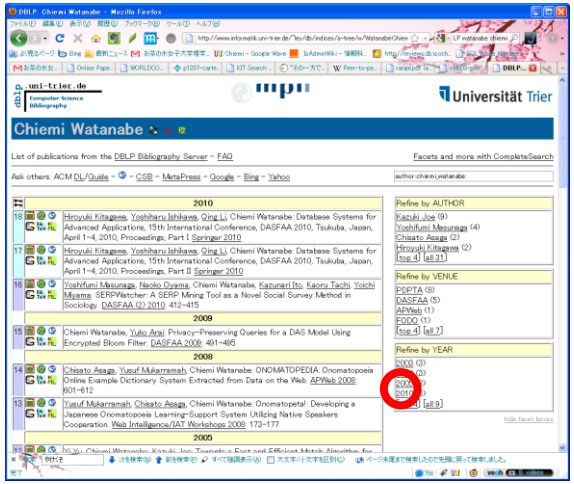


Figure 1. Screenshot of DBLP Faceted Search Interface

Users can add facet elements for narrowing down the objects according to the results, and then, users can find what they want are looking for. To each facet element name, a number is added. The number shows how many objects contain the facet element. For example, the facet element name and the number "Kazuki Joe (9)" shows that there are nine papers in which one of the coauthors is Kazuki Joe. When a user selects this facet element, the system searches for the papers whose authors are Chiemi Watanabe and Kazuki Joe; the facets and their entities are generated according to the search result. The interactive search interface can clarify the user's search purpose.

2.2 Selection of Appropriate Facets

The method of selecting appropriate facets depends on the structure of the target objects for search. In general, the information of target objects is expressed as records of a relational table. In this case, the attributes of the object are the candidates of facets. Suppose that bibliographies are stored in the following tables. Table 1 shows the example tables of bibliography that include the information of two papers.

Paper (*id*, *title, conference, year, start_page, end_page*)
Person (*id*, *name, affiliation, title*)
Author (*paper_id*, *person_id*)

where *Author.paper_id* is a foreign key for the table *Paper*, and *Author.person_id* is a foreign key for the table *Person*. The records of the table *Paper* are supposed to be the target objects for the faceted search interface. In this example, the attributes of the table *Paper*, which are *title*, *conference*, *year*, *start_page*, and *end_page*, are the candidates of the facets. In addition, the attributes of tables that refer to the table *Paper* can be the candidates of facets. In this example, the attributes of the table *Person*, namely, *name*, *affiliation*, and *title*, are also the candidates of facets. From among the candidates of facets, application designers select several attributes that are appropriate for facets according to the property of the target object or the application.

There are several researches that propose measures for finding the appropriate facets for the target objects.

Table 1. Example Tables of Bibliography

Paper

id	title	conference	year	start page	end page
P101	A Query Description Model and its Implementation as an Interactive Query Tool for Visualization System	PDPTA	2004	359	365
P102	Privacy-Preserving Queries for a DAS Model Using Encrypted Bloom Filter.	DASFAA	2009	491	495

Person

id	name	affiliation	title
H23	C. Watanabe	Ochanomizu University	Lecturer
H24	Y. Arai	Ochanomizu University	Student
H25	K. Joe	Nara Women's University	Professor
H26	A. Ishida	Nara Women's University	Student

Author

paper_id	person_id
P101	H23
P101	H25
P101	H26
P102	H23
P102	H24

3. Faceted Search in P2P Environment

3.1 Architecture of Faceted Search Application

We define the architecture of the applications that use a faceted interface in a P2P environment. Figure 2 shows the architecture. Each node manages its database. We do not specify the structure of the databases. In Figure 2, node A manages data by using a relational database system, node B manages the data by using files in the RDF format, and node C manages a set of documents. In order to realize faceted search among these databases, we require two types of metadata—*Facet data* and *Object data*.

Facet data:

These data are used for generating the contents in facet columns. Facets should be selected from facet candidate attributes manually or by using one of the selection measures proposed by previous research. For example, in the case of the bibliography described in section 2.2, the attributes *conference* and *year* of the table *Paper* and the attribute *name* of the table *Person* are selected as facets. According to the list of facets, facet names and the facet elements that correspond to each target object should be published to the P2P network as facet data.

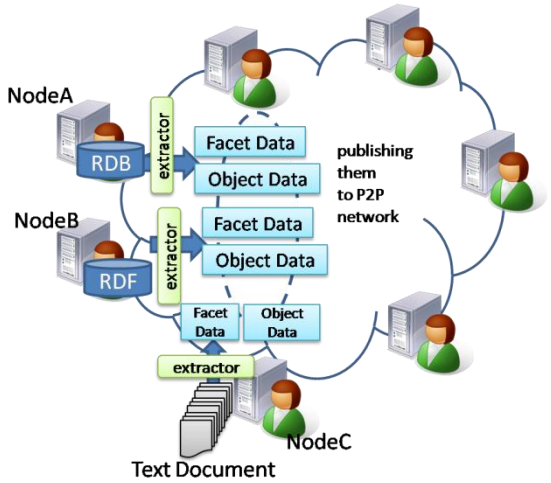


Figure 2. Architecture of Faceted Search in P2P Network

Object data:

These data are used for showing the result objects in the result column of the page. The attributes that can be used for identifying the target object should be specified as object data. For example, in the case of the bibliography described in section 2.2, the target objects are the records of the table *Paper*. Then, the values of the attribute *id* of the table *Paper* are specified as target object ids, and the values of the attribute *title* are specified as the information that can identify the object.

At each node, facet information and object data are extracted from the database and published to the P2P network. The method of extraction of such information is defined by each node according to the structure of the data and the purpose of the application.

3.2 Data Structure for Facet data

We defined the data structure of facet data *f* as the following triple:

fi = (*target_object*, *facet_name*, *facet_ entity*)

where *target_object* includes the Node ID and the ID of the target object, *facet_name* is the facet name that is selected by the data manager of the node, and *facet_entity* is the value of the *facet_name* of the *target_object*. The set of facet data *FI* is defined as follows:

FI_{nodeID} = {*fi*| *fi* is facet information that is extracted from the data on *nodeID*}

The facet data of FI_{nodeID} are generated from the data on the node and are published and shared to the P2P environment. For example, we assume that node A in Figure 2, whose node id is ND001, manages the bibliographic data by using a relational database whose relation schemas are the same as those described in section 2.2; the attributes *conference* and *year* of the table *Paper* and the attribute *name* of the table *Person* are selected as facets. The following faceted information is generated.

FI_{ND001} = {
...
(_ND001_P101‘, _conference‘, _PDPTA‘),
(_ND001_P101‘, _year‘, 2003),
(_ND001_P101‘, _author‘, _C. Watanabe‘),
(_ND001_P101‘, _author‘, _A. Ishida‘),
(_ND001_P102‘, _conference‘, _PDPTA‘),
...}

Next, we describe how to derive the data for generating the content in the facet column. We first define the query if we manage all facet data in a single relational database. We assume that all facet data are managed in the relational table *FacetInfo*(*target_object*, *facet_name*, *facet_entry*). For example, suppose that a user searches for papers whose author is ―C. Watanabe” from the bibliography database. The query, which gets the facet name, its facet elements, and the number of objects that have the facet element, is described as follows:

```
SELECT FA.facet_name, FA.facet_entry, count(FA.facet_entry)
FROM FacetInfo FQ1, FacetInfo FA
WHERE FQ1.facet_name='author'
      and FQ1.facet_entity='C. Watanabe'
      and FQ1.target_object=FA.target_object
GROUP BY FA.facet_name, FA.facet_entry
```

Figure 3. Query that Gets Information for Facet Search

The query statement shows the following features of the query operation for getting facet elements and the number of objects.

(1) The query requires the self-join operations of a table for answering with tables for query conditions. In the above query statement, a table for answers (named FA) and a table for a query condition ―author=C. Watanabe” (FQ1) are generated. If a user adds another query condition, a table for the query condition is required. We should notice that the self-join operation takes a long time for processing the data if there are considerable facet data.

The query needs to count the number of objects for each facet element. As described before, the number shows how many objects have the facet element. The counting operation is a type of aggregate operation. We should notice that it is difficult to process aggregate operations among the data that are distributed throughout the nodes in a P2P environment.

3.3 Data Structure for Object Data

We define the data structure of object data *obj* with the following facet data:

obj = (*target_object*, *ident_attribute*, *value*)

where *ident_attribute* is the attribute that can be used for identifying the target object should be specified as object data. For example, in the case of Table 1, a set of object data is defined as follows:

{(_ND001_P101‘, _title‘, _Privacy-Preserving Queries for a DAS Model Using Encrypted Bloom Filter‘), (_ND001_P101‘, _title‘, _A Query Description Model and its Implementation as an Interactive Query Tool for Visualization Systems‘)}

4. Data Allocation Strategy

From the observation described above, we consider how to allocate facet data in the P2P environment for processing operations for a facet search.

We first consider the appropriate architecture of a P2P environment. There are two types of architecture of a P2P environment—*structured network* and *unstructured network*. A structured network defines the connection of nodes and data allocation according to algorithms such as Chord and Pastry. The structured network can reduce the network traffic for queries. However, the most common type of structured network is based on a distributed hash table (DHT); the applicable operations are limited to the ones that can be processed by using a hash table. *Unstructured network*s do not use any algorithm for the organization or optimization of network connections. Queries are flooded through the network to find as many nodes as possible in the P2P network. The amount of network traffic is larger than that

in a structured network, but the user can specify queries that are more flexible than those in a structured network.

We apply structured networks as the architecture of a P2P environment for two reasons. The first reason is that we should consider the amount of network traffic because users interactively issue several queries until they find what they want. The second reason is that operations can be limited to an exact matching and keyword search that can be processed by using a hash table.

Next, we consider the allocation of facet data. Because we use DHT, we define what should be the keys of the hash table and their corresponding values for processing faceted searches. In the previous section, we described the SQL statement for the table FacetInfo to get the facets, facet elements, and the number of objects. PIER [2] defines the method of allocating tuples of a relational table to process queries that are described by the SQL statement.

However, there are two problems related to the application of PIER's method. These problems are caused by the features of queries used for obtaining the facet data described in Figure 3.

(1) Self-join operations using the attribute *object* should be processed in the query. If pairs of facet data, whose *object* value are the same, are allocated on the different nodes, the operation needs a large amount of network traffic.

(2) PIER does not support aggregate operations. Aggregate operations over DHT have been proposed [3]. However, PIER processes aggregate operations by using a broadcast algorithm. It is impractical to use a broadcast algorithm because every query needs to process a count operation, and this causes a flood of network traffic.

On the basis of these observations, we define an allocation strategy. We first define a set of self-joined facet data JFI_{nodeID} by using the attribute *object* to get all pairs of the facet data for the same object.

$$JFI_{nodeID} = \{(fi_1,fi_2) | \ ^{\exists}fi_1 \in FI_{nodeID}, \ ^{\exists}fi_2 \in FI_{nodeID}, \ fi_1.target\_object = fi_2.target\_object\}$$

If *put*(*key*,*value*) is defined as a function that adds an entry to DHT, we define a function $putFI(fi_1,fi_2)$, where (fi_1,fi_2) is a pair of facetdata. An element of JFI_{nodeID} is defined as follows:

$$putFI(fi_1,fi_2) := put(fi_1.facet\_name + \_:' + fi_1.facet\_entity, fi_2)$$

Table 2 shows pairs of the key and the value for the *put* function for publishing facet data in FI_{ND001}, which is defined in section 4.2. The *put*(*key,value*) function of DHT determines which node the *value* is stored in according to the result of the hash function *h*(*key*). $fi_1.target\_name$ and $fi_1.target\_entity$ of (fi_1,fi_2) in JFI_{nodeID} shows a query condition that is processed when a user clicks the corresponding facet element, and fi_2 shows another facet element of the object that is applied to the query condition. This implies that all information for generating a facet list, entries of each facet, and the number of objects are allocated on the same node when a user specifies a facet element for filtering an object. At each node, a set of pairs (fi_1,fi_2) can be stored in a memory database.

Table 2. Pairs of Key and Value for *put* Function for Facet Data in FI_{ND001}

key	value
conference : PDPTA	(_ND001_P101', _year',2003)
conference :PDPTA	(_ND001_P101', _author', _C. Watanabe')
conference :PDPTA	(_ND001_P101', _author', _A. Ishida')
conference :PDPTA	(_ND001_P101', _author', _K. Joe')
...	...
year :2003	(_ND001_P101', _conference', _PDPTA')
year :2003	(_ND001_P101', _author', _C. Watanabe')
...	
conference :DASFAA	(_ND001_P102', _year', 2009)
...	

The query described in Figure 3 can be processed in a P2P network by issuing the following query on the node that corresponds to the hash value of *h* (_author:C. Watanabe'):

```
SELECT FA.facet_name, FA.facet_entry, count(FA.facet_entry)
FROM FacetInfo FA
GROUP BY FA.facet_name, FA.facet_entry
```

The proposed method can solve the problems described above. The first problem can be solved because the self-join operation is already carried out. In addition, there is no problem when the user specifies an additional facet element. All facet data that have the first facet element are on the same node; the self-join operation does not require any network traffic. For example, when a user selects a facet element "PDPTA" of the facet "conference", the user can obtain the results by issuing the following query on the same node.

```
SELECT FA.facet_name, FA.facet_entry, count(FA.facet_entry)
    FROM FacetInfo FQ2, FacetInfo FA
```

```
WHERE FQ2.facet_name='conference'
        and FQ2.facet_value='PDPTA'
GROUP BY FA.facet_name, FA.facet_entry
```

The second problem can be solved because all facet data are in the same node and aggregate operations should not be processed among multiple nodes. Object data can be published by using a *put* function. We define the *putOI*(*obj*) as follows:

putOI(*obj*) := *put*(*obj.target_object*, *obj*)

5. Query Processing

In this previous section, we assumed that queries are processed on the node in which the result facet data are stored. In this section, we first describe the steps for processing operations for a faceted search by using an example case in which a user clicks the facet element "C. Watanabe" of the facet "author".

Step 1: When a user specifies a facet element $fent_1$ of the facet $facet_1$, the system accesses the node that corresponds to the value of the hash function $h(facet_1 + \_ : ' + fent_1)$.
Step 2: The system gets object data that are applied to the query conditions. Table 3 shows the set of object data as the answer of the query in the example case.
Step 3: The system issues a query that requires a set of the following facet data; they are facet name, facet element, and the number of objects that have the facet element. Table 4 shows the set of facet data as the answer of the query in the example case.
Step 4: By using the results, the system generates the result page.

Table 3. Object Data Acquired at Step 2

obj id	attribute	Value
P101	title	Privacy-Preserving Queries for a DAS Model Using Encrypted Bloom Filter.
P102	title	A Query Description Model and its Implementation as an Interactive Query Tool for Visualization System

Table 4. Facet Data Acquired at Step 3

facet name	facet element	number of objects
conference	DASFAA	2
conference	PDPTA	10
...		
Year	2010	4
Year	2009	1
...		
Author	K. Joe	9
...		

5.1 Getting Object Data

In step 2, the system gets the object data of the objects that are applied to the query conditions. The system first gets object ids by issuing the following query string:

```
SELECT DISTINCT target_object
FROM FacetInfo FA
```

Next, the system gets the object data corresponding to the object ids of the result objects. Now, we notice that only 5~10 result objects can be shown in a result page at once although there are many result objects. Then, the system selects 5~10 object ids to show in the result page, and the system gets the object data of these object ids. Object data are published to DHT by using the *putOI* function. The *putOI* function uses the id of the object as the key, and the value is the object data. Then, the system gets the object data of the object by issuing the DHT function *get*(*key*).

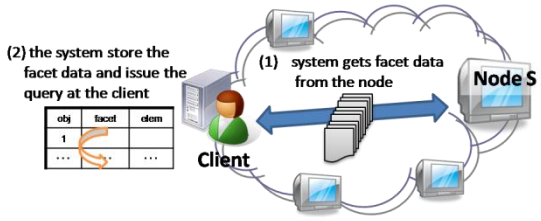
5.2 Getting Facet Data

In step 3, the system gets the facet data of the objects that are applied to the query conditions. We can obtain facet data by issuing the query described in section 5. However, we have the two alternatives for where the system should process the query.

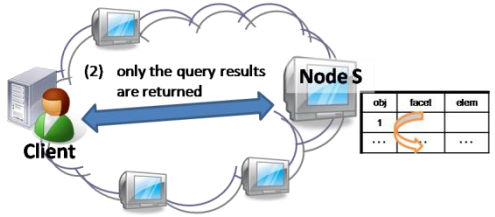
(1) Processing queries at the client (Figure 4(a))
When a user specifies a facet element, the system gets the facet data by using the DHT function *get*($facet_1 + \_ : ' + fent_1$). We store the results in the database at the client and issue the query described in section 5 to the database at the client. By getting the facet data

once, we can process the subsequent queries at the client without accessing the P2P network. On the other hand, the client may have to download a large amount of facet data, and this may take a considerable amount of processing time.

(2) Processing queries at the node (Figure 4(b))
When a user specifies a facet element, the system accesses the node that corresponds to the value of the hash function $h(facet_1 + ':' + fent_1)$ and issues the query on the node. The client can get the results of the query. The size of the results is always small. On the other hand, the system needs to access the P2P network for issuing every query, and the total processing time may be considerable if the user issues interactively queries for several times on the faceted search interface.



(a) Processing query at the client



(b) Processing query at the node

Figure 4. Two Alternative Ways of Processing Queries

We evaluate the total response time when the user interactively issues queries twice. Figure 5 shows the result of evaluation. For this evaluation, we prepare artificial data. On an average, for each facet, the number of target objects is 10000, the number of facets is 10, and the number of facet elements is 10. The number of joint facet data becomes 9,000,000. We published these joint facet data in a P2P environment that consists of 10 nodes. We implement the P2P environment by using Overlay Weaver [7] and apply Chord as an algorithm for DHT. In this evaluation, we fix the selection ratio of the second query a 50% and evaluate the response time by using various selection ratios of the first query.

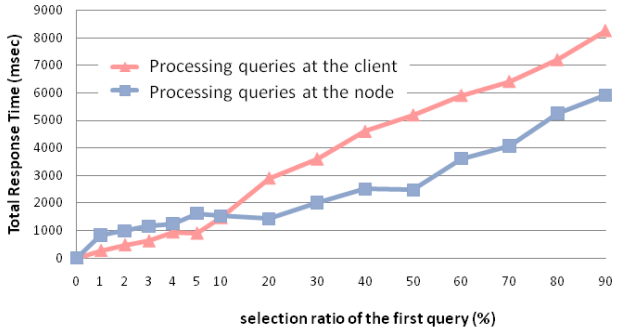


Figure 5. Total Response Time for Two Facet Search Operations

The result shows that the query needs to be processed at the node until the query result is sufficiently filtered. When the number of result objects is small, the system should get all facet data from the node to process the subsequent queries at the client.

6. Conclusion and Future Works

In this paper, we proposed a faceted search in the P2P environment and discussed how to allocate data for the faceted search over DHT. Because query operations include self-join operations and aggregate operations, we proposed an allocation strategy that does not require self-join and aggregation across the several nodes on the P2P network. In addition, we discussed which query operations should be processed at the client or the corresponding node from the evaluation of the total response time of queries for the faceted search. As a future work, we will develop and provide a framework for faceted search in the P2P environment.

References

1) D. Abadi, A. Marcus, S. Madden, and K. Hollenbach: Scalable Semantic Web Data Management Using Vertical Partitioning, In Proc. of the 33rd International Conference on Very Large Data Bases, pp.

411–422, (2007).
2) R. Huebsch, J. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica: Querying the Internet with PIER, In Proc. of the 29th International Conference on Very Large Data Bases, pp. 321–332, (2003).
3) J. Li, K. Sollions, and D. Y. Lim: Implementing Aggregation and Broadcast over Distributed Hash Table, ACM SIGCOMM Computer Communication Review, Vol. 35, No. 1, pp. 82–92 (2005).
4) E. Oren, R. Delbru, and S. Decker: Extending Faceted Navigation for RDF Data, In Proc. of the 5th International Semantic Web Conference, pp. 559–572, (2006).
5) W. Rao, A. W. Fu, L. Chen, and H. Chen: STAIRS: Towards Efficient Full-Text Filtering and Dissemination in a DHT Environment, In Proc. of the 25rd IEEE International Conference on Data Engineering (ICDE'09), pp. 198–209 (2009).
6) S. B. Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania: Minimum Effort Driven Dynamic Faceted Search in Structured Databases, In Proc. of the 17th ACM Conference on Information and Knowledge Management, pp. 13–22 (2008).
7) K. Shudo, Y. Tanaka, and S. Sekiguchi: Overlay Weaver: An Overlay Construction Toolkit, Computer Communications (Special Issue on Foundations of Peer-to-Peer Computing), Elsevier Science, Vol. 31, No. 2, pp. 402–412 (2008).
8) C. Zheng, G. Shen, S. Li, and S. Shenker: Distributed Segment Tree: Support Range Query and Cover Query over DHT, In Proc. of the Fifth International Workshop on Peer-to-Peer Systems (IPTPS) (2006).
9) DBLP Computer Science Bibliography, http://www.informatik.uni-trier.de/~ley/db/index.html.