2010 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)
IPCSIT vol. 53 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V53.No.1.80

Answering Tag-Term Keyword Queries over XML Documents in DHT Networks

Weimin He[1+] and Teng Lv[2]

[1]Department of Computing and New Media Technologies, University of Wisconsin-Stevens Point, Stevens Point, WI 54481, USA

[2]Teaching and Research Section of Computer, Artillery Academy, Hefei 230031, P.R.China

Abstract. The emergence of Peer-to-Peer (P2P) computing model and the popularity of Extensible Markup Language (XML) as the web data format have fueled the extensive research on retrieving XML data in P2P networks. In this paper, we developed an efficient and effective keyword search framework that can support tag-term keyword queries in Distributed Hash Table (DHT) networks. We employed a concise Bloom-Filter data structure to index XML meta-data in the DHT repository. We also developed an effective algorithm that supports tag-term keyword queries over our Bloom-Filter encoded XML meta-data in the DHT network. We conducted extensive experiments to demonstrate the efficiency of indexing scheme, the effectiveness of our keyword query algorithm and the system scalability of our framework.

Keywords: Database, XML, Information Retrieval, Distributed Hash Table

1. Introduction

The emergence of Peer-to-Peer (P2P) computing model and the popularity of Extensible Markup Language (XML) as the web data format have fueled the extensive research on retrieving XML data in P2P networks. Searching distributed XML documents in P2P systems, especially in DHT networks, have attracted much attention in the literature. Since XML data are hierarchical trees instead of plain text documents, the keyword-based search over XML documents in DHT networks poses much more challenges than the keyword-based search over plain text documents in the traditional client-server models. The evaluation of keyword queries over these data are more complex and if not optimized, may involve routing and processing massive data in the DHT networks. The key issue to be addressed here is how to design efficient meta-data indexing scheme to summarize the original XML documents, develop effective keyword search algorithm to evaluate keyword queries over those meta-data, and finally return the list of XML documents as query answers to the user.

In this paper, we present a novel framework for indexing and searching distributed XML data in DHT-based structured P2P networks. First, we propose an efficient Bloom Filter-based meta-data indexing scheme. Then, we develop an effective keyword search algorithm over our indexed meta-data to locate XML document hits which can satisfy the user query in the DHT network. Finally, we experimentally validate the efficiency of our indexing scheme, the effectiveness of our keyword search algorithm, and the scalability of our system.

The rest of the paper is organized as follows. We present our Bloom Filter-based meta-data indexing scheme in section II. In section III-A, we present the specification of our keyword query and the query evaluation algorithm. We report experimental results in section IV. Related work is given in section V. Finally, we conclude the paper and present our future work in section VI.

[+] Corresponding author.
E-mail address: weimin.he@uwsp.edu.

2. Data Indexing Scheme

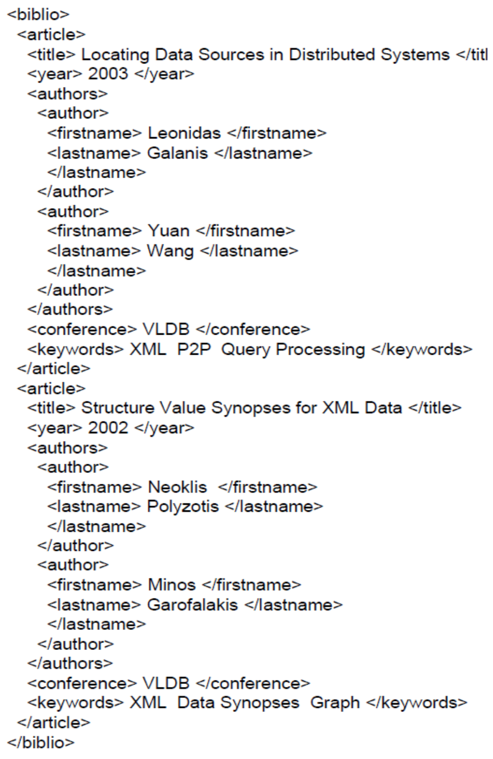
2.1 Structural summary

To enable efficient publishing and indexing of XML documents in the DHT network, we extract a concise metadata termed **Structural summary**(SS), which is a structural markup that captures all the unique paths in an XML document. Figure 1 gives an example XML document that represents the excerpted article information from DBLP. The structural summary of the example XML document is shown in Figure 2. Each node in an *SS* has a tagname and a unique Id. Note that one *SS* node may correspond to more than one element in the original XML document. For example, the node article in Figure 2 corresponds to three article elements in Figure 1. As an XML document is published by a peer, the *SS* is extracted from the document, which serves as the basis for meta-data indexing in the DHT network. We exploit Java SAX API to extract the structural summary from an XML document. Due to the space limitation, the structural summary extraction algorithm is omitted here.

2.2 Data Publishing and Indexing in DHT

Since an XML document may contain a large number of elements with the same names and each element may contain a large number of terms, fully indexing XML data in the DHT network is not feasible in terms of both data publishing and query response time. In addition, due to the dynamic characteristics of P2P networks, full-indexing scheme may also lead to the overload of data update. In order to address the above issues, we propose a Bloom Filter-based meta-data indexing scheme. A Bloom Filter [5] is a compact probabilistic data structure which is used to test whether an element might be a member of a set. A Bloom Filter is implemented using a bit vector with *m* bits [6].

More specifically, as an XML document *D* is published, for each leaf node *e* in the structural summary of *D*, we construct a triple <*bf; peerId; docId*> to summarize the textual data contained in the element *e* in the document *D*. In the triple above, *bf* is a Bloom Filter that summarizes the textual data enclosed by *e*. Each term enclosed by *e* in *D* is hashed to a bucket in *bf* and the corresponding bit is set to 1. And *docId* is the document ID of *D* and *peerId* is the peer ID of the peer that owns the document *D*. We use *tage*, which is the tag name of *e*, as the DHT key to route the triple to the peer *Ptage* , who is responsible for storing the data related to the key *tage*. In the local index of *Ptage* , we use *tage* as the key to store the triple < *bf; peerId; docId* >.



```
<biblio>
 <article>
  <title> Locating Data Sources in Distributed Systems </titl
  <year> 2003 </year>
  <authors>
   <author>
    <firstname> Leonidas </firstname>
    <lastname> Galanis </lastname>
    </lastname>
   </author>
   <author>
    <firstname> Yuan </firstname>
    <lastname> Wang </lastname>
    </lastname>
   </author>
  </authors>
  <conference> VLDB </conference>
  <keywords> XML  P2P  Query Processing </keywords>
 </article>
 <article>
  <title> Structure Value Synopses for XML Data </title>
  <year> 2002 </year>
  <authors>
   <author>
    <firstname> Neoklis  </firstname>
    <lastname> Polyzotis </lastname>
    </lastname>
   </author>
   <author>
    <firstname> Minos </firstname>
    <lastname> Garofalakis </lastname>
    </lastname>
   </author>
  </authors>
  <conference> VLDB </conference>
  <keywords> XML  Data Synopses  Graph </keywords>
 </article>
</biblio>
```

Fig.1 Example XML Document

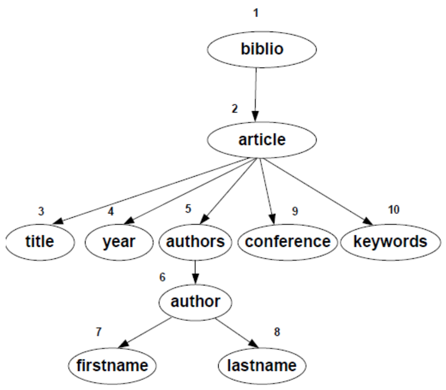


Fig.2 Strutural Summary of Example XML Document

3. Keyword Query Evaluation

3.1 Specification of tag-term keyword queries

Since XML data is hierarchical and recursive semistructured data instead of flat and plain textual data, we believe that it would be better to consider the structure features when designing the keyword search queries. We extend the traditional simple keyword search query syntax with a construct named tag:term pair, where tag is an element in an XML document and term is a text term that may appear in the element tag. In another word, A keyword in our framework is a tag:term pair instead of a single term. In consequence, our keyword query takes the following form: $tag1$: $term1$ AND $tag2$: $term2$ AND $:::$ AND $tagn$: $termn$. As a running example used throughout the paper, the following keyword search query, Q: title:XML AND author:David AND year:2003 searches for XML documents that have a title element containing the term “XML”, an author element containing the term “David” and a year element containing the term “2003”.

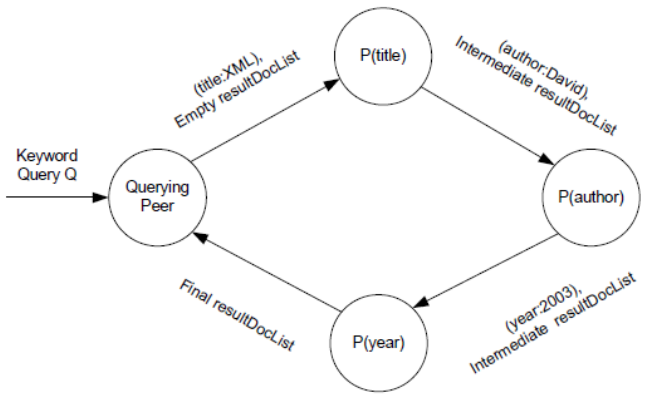
Note that the goal of the proposed framework in this paper is to locate relevant XML data sources in DHT networks. Thus the query answers of a keyword search query in our framework is a list of document hits that may satisfy the query instead of the actual XML fragments.

3.2 Keyword query evaluation algorithm

In order to evaluate a keyword user query, we have developed an efficient keyword query evaluation algorithm that can route the user query among relevant peers, collect qualified document hits along the way, and finally return the resulting document hits to the user. Our keyword query evaluation algorithm is shown in Algorithm 1. We illustrate Algorithm 1 using our running example query Q. Figure 3 demonstrates how the keyword query Q is evaluated in the DHT network. First, the user poses the query Q on the Querying Peer. Then the pair (title:XML) and the initial empty resultDocList are routed to the peer P(title). On the peer P(title), “title” is used as the key to access the local index to obtain a list of triples (bf, peerId, documentId). “XML” is then mapped to a bit in bf and the pair (peerId, documentId) will be added in the resultDocList if the bit is set to one. After that, the intermediate resultDocList and the pair (author:David) is routed to the peer P(author). On the peer P(author), “author” is used as the key to access the local index to obtain a list of triples (bf, peerId, documentId). “David” is then mapped to a bit in bf and the pair (peerId, documentId) will be added in the resultDocList if the bit is set to one. After that, the updated intermediate resultDocList and the pair (year:2003) is routed to the peer P(year). On the peer P(year), “year” is used as the key to access the local index to obtain a list of triples (bf, peerId, documentId). “2003” is then mapped to a bit in bf and the pair (peerId, documentId) will be added in the resultDocList if the bit is set to one. Finally, the resultDocList is routed back to the Querying Peer and the query answer is returned to the user.

Algorithm 1: Keyword Query Evaluation in the DHT Network

```
Input:  Q    /* User query that contains a list of tag-term pairs */
        id   /* Pastry node ID of query client */
        docList   /* Current list of document hits */
        currentKeywordIndex   /* index of current keyword being processed in the query Q */
Output: resultDocList   /* the list of document hits that satisfy the keyword query */
if ( currentKeywordIndex < Q.size )
  tag := Q.get(currentKeywordIndex).tag;
  term := Q.get(currentKeywordIndex).term;
  localList := the BFSynopsis list returned by accessing the local index using tag as the key;
  for i := 1 to localList.size do
    synopsis := localList.get(i);
    Hash term to a bit b in the Bloom Filter bf in synopsis
    if ( b=1 )
      hit := < synopsis.peerId, synopsis.docId >;
      if ( currentKeywordIndex = 0)
          docList.add(hit);
       else
          found : = false;
          for  i := 0 to docList.size do
             h = docList.get(i);
             if ( h.peerId = hit.peerId AND h.docId = hit.docId )
                found = true;
                break;
             end if;
          end for;
          if ( found = false )
             docList.remove(hit);
          end if;
      end if;
    end if;
  end for;
  if ( currentIndex + 1 = Q.size )
        resultDocList := docList;
        Route resultDocList to the query client whose Pastry NodeID is id;
    else
        Route docList to the next peer for further processing;
  end if;
end if;
```



Keyword Query Q
Querying Peer
(title:XML), Empty resultDocList
P(title)
(author:David), Intermediate resultDocList
P(author)
(year:2003), Intermediate resultDocList
P(year)
Final resultDocList

Fig. 3 Keyword Qury Routing of the Running Qury Q

4. Experimental Results

4.1 Experimental setup

We have built a prototype system based on our framework using Java (J2SE 6.0). Our DHT-based P2P back end is FreePastry 2.0 [23], which is responsible for routing various messages associated with data publishing and query evaluation to specific peers. On each peer, Berkeley DB Java Edition 3.2.13 [4] is employed as the local indexing DB. Our experiments were carried out on a laptop running WindowsXP with 2.26GHz CPU and 3G memory. The maximum heap size for Java Virtual Machine was set to 512MB. We conducted several sets of experiments to measure the efficiency of our meta-data indexing scheme, the effectiveness of our keyword search algorithm, and the scalability of our framework. Our experimental data are synthetically generated from XMark benchmark [30], from which data synopses are extracted and populated over the DHT network. For all experiments except those measuring system scalability, the size of the data set is 55.8MB and the number of documents is 11500. In our experiments we create 1150 Pastry nodes and let each node publishes 10 XML documents over DHT network. Since scalability experiments require large data volumes, we multiply the above data collection by copying when larger volumes are needed. Each peer owns its database to index the meta-data. We designed 10 queries over XMark data to conduct query-related experiments and the queries are shown in Table 1. Note that the logical "AND" operator is omitted in each query in Table I.

Table 1 :Experimental XMark Queries

Query	Query Expression
Q_1	payment:creditcard
Q_2	location:United
Q_3	payment:creditcard location:United
Q_4	payment:cash location:United
Q_5	payment:cash payment:creditcard
Q_6	text:earth type:regular
Q_7	payment:cash location:United text:earth
Q_8	payment:check location:States text:gold
Q_9	payment:creditcard payment:cash location:States
Q_{10}	text:heat type:regular text:gold

4.2 Efficiency measurement of data indexing scheme

In this experiment, we measure the efficiency of our Bloom Filter-based meta-data indexing scheme(BFI). More specifically, we compared the efficiency of our indexing scheme BFI with that of traditional full indexing scheme(FI). We first compared the data publishing time between BFI and FI. The result is shown in Figure 4, from which we can see that the average peer publishing time of BFI is only about 35% of that of FI. We then compared the index size between BFI and FI. The result is shown in Figure 5, from which we can see that the average peer index size of BFI is only about 36% of that of FI. Finally, we compared the query evaluation time between two schemes for all the queries in Table I. From Figure 6, we can see that for all the queries, BFI is much faster than FI in terms of query response time. The above experiments demonstrate our Bloom Filter-based indexing scheme is efficient in terms of data publishing time, index size and query response time.

4.3 Accuracy measurement of bloom filter-based data synopses

To demonstrate the accuracy of Bloom Filter-based Data Synopses, we measured the false positive rate for all the queries list in Table I. First, we converted each query into a corresponding XPath query and exploited an existing XQuery engine Qizx/open [32] to evaluate each query over the dataset

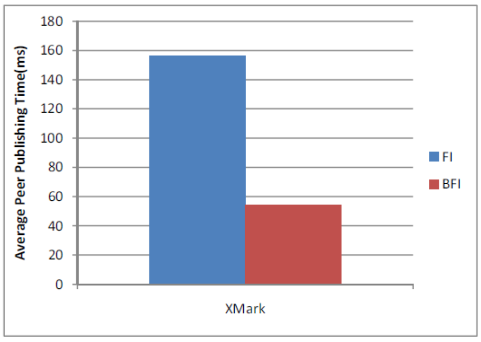


Fig.4 Comparison of peer Publishing Time

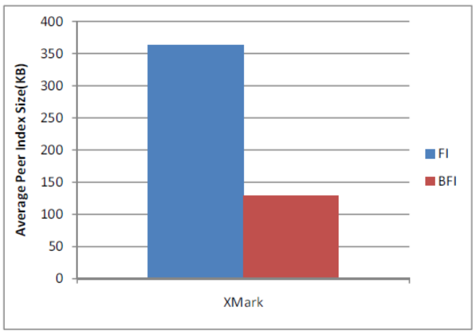


Fig.5 Comparison of Index Size

to obtain the accurate relevant set for the query. Then we run each query over the meta-data in DHT and measured the false positive rate for the query. The results are shown in Table 2. From Table II, we can see that for the queries with single tag-term pair, such as queries *Q*1 and *Q*2, the false positive rate is a little higher. However, for the queries with multiple tag-term pairs, especially queries with 3 tag-term pairs, such as query *Q*8, the answer set size is very close to that of relevant set size. This result indicates that our Bloom Filter-based Data indexing scheme is more suitable to process keyword queries with multiple search predicates. Notice that since XMark data are homogeneous, the number of false positives for some queries is high. For example, the number of false positives for *Q*10 is 124, which is much higher than the relevant set size 21. That is because among 11500 documents, although only 21 documents exactly match the query (satisfy both structural and search predicates), a lot more documents match only the structural part in the query, thus the answer set size is much larger than the relevant set size and the false positive rate is high for that query. However, under the heterogeneous environment in a real world application, the answer set size would be much smaller and thus the false positive rate should also be much lower.

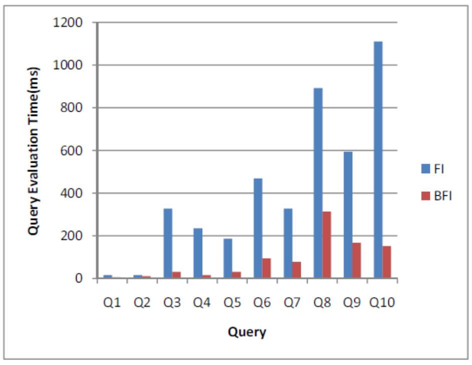


Fig.6 Comparison of Query Timr

Table 2 Accuracy Mrasurement

Query	Relevant Set Size	Answer Set Size	# of False Positives	False Positive Rate(%)
Q_1	3142	5124	1982	39
Q_2	3574	5623	2049	36
Q_3	2717	2816	99	4
Q_4	2321	2356	35	1
Q_5	533	612	79	13
Q_6	94	137	43	31
Q_7	43	62	19	31
Q_8	65	74	9	12
Q_9	245	346	101	29
Q_{10}	21	145	124	86

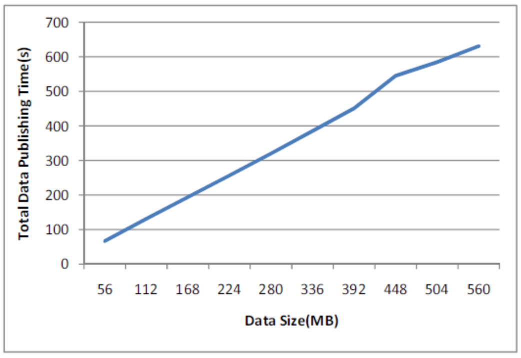


Fig. 7 Varying Data Size

4.4 Measurement of system scalability

In this experiment, we measure the scalability of our system. We first varied the data size from 56MB to 560MB to measure the total data publishing time. As we can see from Figure 7, as the data size increases, the data publishing time scales almost linearly in terms of the size of published data. We then varied the number of peers in our system to measure the total data publishing time. As we can see from Figure 8, as the network size increases, the total data publishing time increases smoothly with the increasing number of peers. The above results indicate that our system scales gracefully in terms of both data size and network size.

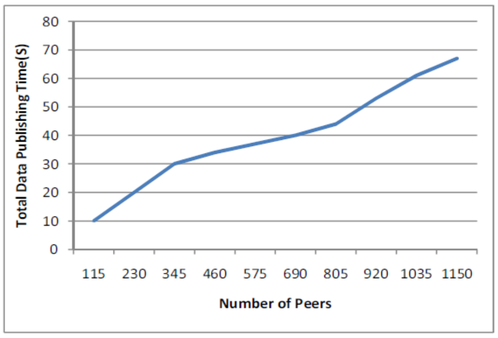


Fig. 8 Varying Network Size

5. Related Work

Extensive research work has been done on proposing new metadata indexing schemes and efficient query evaluation algorithms to query distributed XML data in P2P networks [7], [11], [18], [28]. Some work focuses on indexing and querying XML data in unstructured P2P networks. In [18], the authors proposed two multilevel Bloom filters, termed Breath Bloom Filter and Depth Bloom Filter, to summarize the structure of an XML document for efficient path query routing in unstructured P2P networks. Kd-synopsis [28] is another routing synopsis based on length-constrained FBsimulation relationship, which allows the balancing of the precision and size of the synopsis according to different space constraints on peers with heterogeneous capacity. Other researchers are interested in query processing of XML data in structured P2P networks. In [7], the authors proposed XP2P, which employs concrete paths in an XML document to index XML fragments. XP2P can answer simple linear XPath [31] queries efficiently, but does not support complex XPath queries with conditions or search predicates. In [11] Galanis et al. proposed an meta-data indexing scheme and query evaluation strategies for distributed XML data in structured P2P networks. In their framework, a distributed catalog service is distributed across the data sources themselves and is responsible for routing user queries to the relevant peers. In [19], Li et al. presented a scheme for searching XML documents by keywords in DHT-based structured P2P networks. They constructed an inverted index for XML documents in an DHT-based P2P network. To achieve that goal, they proposed to decompose XML documents and store the resulting fragments in the DHT network. Given a query in terms of keywords, their framework can find respective XML fragments that subsume given keywords. They addressed this problem by finding SLCAs (Smallest Lowest Common Ancestors) for the keywords with the help of the DHTbased inverted index.

6. Conclusions

We presented a novel framework for answering tag-term keyword queries in DHT networks. We designed efficient Bloom Filter-based meta-data indexing scheme to summarize XML documents and populate those meta-data in the DHT network. We also developed an effective keyword query evaluation algorithm over the distributed XML data in the DHT network. Finally, we conducted extensive experiments to demonstrate the efficiency of our indexing scheme, the effectiveness of our keyword search algorithm, and the scalability of our system. As our future work, we plan to adapt the classical *tf* ¤*idf* scoring in IR to develop an effective ranking scheme for our proposed framework in this paper.

7. Acknowledgment

This work is supported by the Faculty Professional Development Grant from the University of Wisconsin-Stevens Point, USA, 2010.

8. References

[1] S. Abiteboul, I. Manolescu, and N. Preda. Constructing and Querying Peer-to-Peer Warehouses of XML Resources. ICDE 2005.

[2] S. Al-Khalifa, C. Yu, and H. V. Jagadish. Querying Structured Text in an XML Database. SIGMOD 2003.

[3] W.T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive Distributed Top k Retrieval in Peer-to-Peer Networks. ICDE 2005.

[4] Berkeley DB. http://www.oracle.com/database/berkeley-db.

[5] B. Bloom, Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*. 13(7) pp. 422-426, 1970.

[6] C. Grothoff A Quick Introduction to Bloom Filters. *Technical Report*. Department of Computer Sciences. Purdue University

[7] A. Bonifati, U. Matrangolo, A. Cuzzocrea, and M. Jain. XPath Lookup Queries in P2P Networks. WIDM 2004.

[8] J.M. Bremer and M. Gertz. On Distributing XML Repositories. WebDB 2003.

[9] S. Cho, N. Koudas, and D. Srivastava, Meta-data Indexing for XPath Location Steps. in *Proceedings of SIGMOD'06*, 2006.

[10] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. VLDB 2003.

[11] L. Galanis, Y. Wang, S. R. Jeffery, and D. J. DeWitt. Locating Data Sources in Large Distributed Systems. VLDB 2003.

[12] L. Fegaras, W. He, G. Das, and D. Levine. XML Query Routing in Structured P2P Systems. DBISP2P 2006.

[13] W. He, and L. Fegaras. Approximate XML Query Answers in DHTBased P2P Networks. DASFAA 2008.

[14] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. SIGMOD 2003.

[15] R. Huebsch, *et al*. Querying the Internet with PIER. VLDB 2003.

[16] R. Huebsch, *et al*. The Architecture of PIER: an Internet-Scale Query Processor. CIDR 2005.

[17] R. Kaushik, P. Bohannon, J.F. Naughton, and H. Korth. Covering indexes for branching path queries. SIGMOD 2002.

[18] G. Koloniari, and E. Pitoura. Content-Based Routing of Path Queries in Peer-to-Peer Systems. EDBT 2004.

[19] X. Li, T. Amagasa, and H. Kitagawa. Searching XML Documents by Keywords in Structured P2P Networks. DEXA 2008.

[20] G. Koloniari and E. Pitoura. Peer-to-peer management of XML data: issues and research challenges. SIGMOD Record. 34(2): 6-17 2005.

[21] N. Polyzotis and M. Garofalakis, Structure and Value Synopses for XML Data Graphs. in *Proceedings of VLDB'02*, 2002.

[22] N. Polyzotis, and M. Garofalakis. XCluster Synopses for Structured XML Content. ICDE 2006.

[23] Pastry. http://freepastry.rice.edu.

[24] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A Scalable Content-Addressable Network. SIGCOMM 2001.

[25] I. Stoica, et al. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. IEEE/ACM Trans. on Networking, Vol. 11 (2003)17-32

[26] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. ACM SIGCOMM 2003.

[27] D. Tsoumakos and N. Roussopoulos, A Comparison of Peer-to-Peer Search Methods. WebDB 2003.

[28] Q. Wang, A.K. Jha, and M.T. Ozsu. An XML Routing Synopsis for Unstructured P2P Networks. WAIMW 2006.

[29] W. Wang, H. Jiang, H. Lu and J. X. Yu, Bloom Histogram: Path Selectivity Estimation for XML Data with Updates. in *Proceedings of VLDB'04*, 2004.

[30] XMark. http://www.xml-benchmark.org/

[31] XML Path Language (XPath) 2.0. http://www.w3.org/TR/xpath20/

[32] Qizx/open. http://www.axyana.com/qizxopen/.