Information Retrieval in a Peer-to-Peer Environment

Dik Lun Lee   Dyce Jing Zhao   Qiong Luo
Department of Computer Science
the Hong Kong University of Science & Technology, Hong Kong
{dlee,zhaojing,luo}@cs.ust.hk

Abstract

Due to rapid information growth, peer-to-peer (P2P) systems have become a promising alternative to centralized, client/server-based approaches for large-scale data sharing. By allowing peers to join and leave the system freely, they offer the peers autonomy to control their own resources, high scalability and high robustness against single-peer failures. P2P systems have many applications. This paper focuses on peer-to-peer information retrieval (P2PIR), which aims to retrieve textual documents based on their contents and ranks them based on some relevance measures against the query. The "open nature" of P2P systems and their lack of centralized control pose difficult challenges to the search capability and performance of P2PIR systems.

In this paper, we study the recent works on P2P information retrieval, introduce our current research, discuss a few open problems that have not been studied in great depth and propose some future directions for information retrieval research in a P2P environment.

1. Introduction

From its birth, the World Wide Web has grown with a surprisingly rapid speed. As a result, peer-to-peer (P2P) technologies have been advancing rapidly [9] since their first commercial application, Napster [12], was introduced in 1999. P2P systems exhibit three important characteristics: self-organization, symmetric communication and distributed control. Further, no global information is supposed to be available at any single peer, and there is no powerful servers serving as centralized control.

In this paper, we focus on P2P Information Retrieval (P2PIR), where autonomous computers cooperate with each other to publish and manage a large amount of data and to retrieve the result for any query submitted from any peer. Within the scope of this paper, P2PIR deals with textual documents and retrieval is based on some ranking measures computed between the query and the document texts. Since the results may come from more than one peer, result merging is needed to combine the results.

The most important benefit of P2P systems is that, by distributing the workload of data management, which includes storage, processing and network communication, they are able to index a massive amount of data with a large number of inexpensive peers. Compared to centralized search engines such as Google,[1] the peers are physically distributed and managed by their individual owners. This approach yields two major advantages. First, it avoids the system administration problem associated with large data centers, such as huge power consumption and cooling problems and the physical space needed for system expansion. Second, the owners of the peers are more motivated in maintaining the information on the peers because they put it together, compared to crawlers that gather information blindly from the web.[2] Third, unlike centralized search engines, queries in P2PIM are executed directly on the peers and thus can make use of local search facilities tailored for the information on the peers (e.g., ranking information based on local access statistics on the peers). Centralized search engines such as Google can only provide the same search interface for all kinds of web contents.

Despite the various strengths that P2P systems possess, important challenges must be overcome before P2PIR can take up practical real-world applications. For example, the peers' lack of global information about the entire system makes finding resources difficult and thus weakens the query capability and search performance of the system. As such, elaborated indexing and query routing techniques must be developed to make the query capability and retrieval quality comparable to traditional centralized search systems. Clearly, P2P systems have other problems such as their low resistance to attacks from malicious peers, but this

---

[1] Although Google has hundreds of thousands of servers distributed over a large number of data centers with load balancing, we regard it as centralized because the system are by-and-large client-server based, and the index is logically centralized.

[2] Although websites may be professionally managed, important information structures in the websites would be ignored by the crawlers.

paper will only focus on IR related issues.

The above issues have motivated a number of works with the goal to meet the challenges in P2P systems and to realize the potential of P2P technologies. In this paper, we survey the research that was done for P2P information retrieval (P2PIR). The rest of this paper is organized as follows. Section 2 presents a taxonomy of P2P IR works. Section 3 introduces some of our recent and ongoing works. The open problems and future directions are proposed in Section 4. Section 5 concludes our discussion on P2PIR.

2. Taxonomy for Peer-to-Peer Information Retrieval

P2PIR research can be classified along various dimensions. Regarding overlay structures, there are hash-based approaches, non-hash-based approaches and hybrid approaches. Considering the routing algorithm, P2P routing can be classified as structured or unstructured. In terms of query capability, the representative examples are exact match queries, partial match queries, ranked queries, range queries, and so on. The above taxonomy is by no means comprehensive. In the following, we focus on routing algorithms, studying the structured and the unstructured ones. We also discuss the query capability and four categories of query operations.

2.1 Structured and Unstructured Systems

P2P routing algorithms can be classified as structured or unstructured. One well-known example for unstructured systems is Gnutella. Gnutella [1] applies flooding for query routing, and query messages are broadcasted within the neighborhood of the query initiator before their time-to-live (TTL) values are reduced to zero. Pastry [16], CHORD [21] and the Content Addressable Network (CAN) [14] are examples for the structured systems.

Structured routing usually outperforms unstructured routing with respect to the scope (i.e., the percentage of data that a query could reach) and efficiency. However, peers in a structured network maintain information about a group of other peers, which incurs additional storage and update cost. Furthermore, since many structured systems apply strict mapping from data resources to their hosting peers, these systems do not exhibit high peer autonomy. Because the auxiliary information on peers is pre-defined, structured systems have limited query capability compared to unstructured systems. For example, Pastry, CHORD and CAN support exact match only, while almost any kind of query that a centralized system supports can be done in an unstructured system such as Gnutella.

Efforts had been made to provide ranked search in structured systems. Two recent examples are pSearch [22, 23]

and the Semantic Small World (SSW) [10]. pSearch and SSW use Latent Semantic Indexing (LSI) to map documents into a semantic vector space and perform search based on the Euclidean distance between the query point and the document points. In particular, pSearch is developed on top of CAN. In addition to LSI, pSearch employs a rolling index and registers a document to p places in CAN using p separate partial semantic spaces. This reduces the dimensionality and therefore enables CAN to manage full-text documents. In SSW, nodes form clusters, each of which manages non-overlapping regions of the semantic vector space. A cluster is split into two when the cluster size exceeds a certain threshold. Every node in a cluster knows its region and splitting history, which are used to compute a unique ID for the cluster. All the clusters form a circle with clockwise ascending cluster ID's. A query message computes a partial cluster ID using the available splitting history and hops along the circle in a greedy manner until it reaches the cluster with the complete ID. pSearch and SSW split successively the vector space into cells and position data points according to the cells that they reside in. Given a keyword based query, both approaches are able to identify the most relevant data resource while the similarity metrics can be cosine or Euclidean distance.

2.2 Query Capability

In order to be practical search solutions, P2P information retrieval systems must be able to support expressive query languages. In this section, we enumerate a list of query operations that have been extensively studied in recent years.

- **Exact Match Queries**
  Exact match is the simplest type of query. Given a query Q and the key K of a data resource D, D is a result if and only if $Q = K$. Most hash-based P2P systems [16, 21, 10] are able to support exact match. As with all hash-based methods, the hash key must be fixed at the time the resources are inserted into the system. Therefore, queries must be expressed on the chosen key. Clearly, exact match queries are useful only in applications where an "obvious" key exists (e.g., file sharing systems where the filename could be such a key).

  A generalization of the exact match query defined above is to allow multiple keys in the query. That is, a resource has multiple key attributes, such as K_1, K_2, K_3, and the query specifies a condition on each of the attributes, e.g., Q_1, Q_2, Q_3. A resource is a match if $Q_1 = K_1$, $Q_2 = K_2$, and $Q_3 = K_3$. Multiple-key exact match requires the resources to be hashed based on composite key attributes, resulting in a large index space.

- **Range Queries**

  Range queries relax exact match queries to allow range conditions to be specified on the key attributes. Approaches for range queries can be divided into those that rely on an underlying Distributed Hash Table (DHT) [17, 13] and those that do not [24]. Several DHT-based range query methods have been proposed recently. For example, Gupta, et al., relied on locality sensitive hashing to ensure that similar ranges have a high probability to be mapped to the same node [7]. They proposed one particular family of locality sensitive hash functions, called min-wise independent permutations, and showed that the hop-count distribution was very similar to that of the Chord exact match scheme. Other range query approaches avoided using DHT's since DHT was considered not suitable for range queries [3]. For example, SkipNet [8] was based on skip graphs.

- **Partial Match Queries**

  A partial match query contains a set of keywords. Likewise, data sources are indexed on a number of key attributes. Data sources which match the query in one or more key attributes form the result set. The fact that a query could specify *any number* of attributes in *any combination* may it practically impossible to use hash-based P2P systems to answer partial match queries.

  In unstructured P2P systems, routing indexes and various search heuristics [20, 6, 26] have been proposed to support partial match efficiently. However, most of them can only return partial results (i.e., not all relevant resources are explored). This limitation makes P2PIR inferior in terms of search quality when compared to centralized search services which indexes and logically searches all resources for a query.[3]

  DPTree [28] is a structured approach for partial match queries, so it supports complete search (i.e., all relevant resources are considered). It is based on a distributed index which achieves high storage and query efficiency by indexing only the frequently queried keyword combinations. See Section 3.2 for more details.

- **Ranked Queries**

  Ranking can be applied orthogonally to the query types above. That is, results returned from any of the above query types can be ranked based on some scoring or goodness measures. Ranked queries allow the system to return the top K ranked resources. This opens up the possibility of optimization if K is much smaller than the size of the entire result set.

---

[3]Centralized search services such as Google do not need to physically search through all resources for each query by utilizing a combination of indexing, caching and search heuristics.

The most popular scoring method in IR is based on the vector-space model which computes relevance scores between the query and the results. P2PIR systems had been proposed based on this model and different relevance measures had been adopted. For example, pSearch [22] adopted Euclidean distance, and DPTree [28] supported inner-product as well as Euclidean distance.

Notice that although it might sound unnatural from a traditional (relational) database point of view, even exact match queries can produce ranked results from an IR point of view because the results can be ranked by other non-query specific parameters such as the authorities of the results, which is best exemplified by Google's PageRank. In fact, P2P systems open up more ways for non-content-based ranking. For example, the availability and access time of the nodes hosting the results, which are assumed to be perfect in the web environment, are important ranking factors in a P2P environment.

The above classification includes common query types in general P2P systems. For IR specifically, exact match queries, either ranked or unranked, don't make much sense because they require the queries to specify a value for each key attributes. Likewise, range queries are not useful for textual contents unless some sort of semantic distance can be defined for the keywords. For textual contents, the most useful query type for IR is ranked partial match queries. This is the type of queries supported by arguably all IR systems and as such must be supported adequately in P2PIR systems if P2PIR systems aspire to be a competitive alternative of centralized search systems. In the rest of this paper, we examine recent research that we have done for ranked partial match queries.

3. Our Recent Research

In this section, we present our recent research on distributed information retrieval applicable to P2PIR.

3.1 Collection Ranking with Term Correlation

Collection selection is a crucial step for unstructured P2PIR. Recall that an unstructured P2P system relies on query routing to spread the query across the nodes in the P2P system in order to find the results. To reduce the number of nodes examined, queries much be routed more intelligently than flooding. One way to achieve this is for each node to maintain an index of the resources on the neighboring nodes so that it can determine which neighboring

nodes are the best candidates to receive the query. Collection ranking refers to the determination of which collections are better in matching the query based on condensed descriptors of the collections (i.e., without actually evaluating the query against each collection). When each peer represents a collection, it can also be called peer ranking. Research had been done along this direction [20, 6]. These methods, however, only maintain information (typically occurrence frequencies) about single terms in the collections. They have no information about whether terms co-occur in the same documents. Thus, they perform poorly for multi-term queries because they may mistakenly give high ranks to collections in which the query terms appear many times but very few documents match all of the query terms.

We resolve this problem by maintaining information about both the indexed terms and the correlation of the terms in each collection [29].[4] In our approach, a collection is first divided into clusters. For each cluster, we create a descriptor that consists of two components, namely, the centroid of the cluster and a co-occurrence matrix for the terms in the cluster. Collection ranking is based on both components. The co-occurrence matrix encodes the occurrence correlation values of pairs of words in the cluster. In order to reduce the storage requirement, the co-occurrence matrix only contains term co-occurrence information for the frequent terms in the cluster. We evaluated different ways of combining these two components in the server ranking formula.

We conducted experiments using the TREC-4[5] and TREC-5[6] subsets. Figures 1 and 2 compare the search precisions of our method (denoted by CL) and a collection ranking method proposed by Shen, et al. [19], under various query lengths and for uniformly distributed and Zipf distributed collection sizes. We showed that combining clustering and term correlation improves the search quality significantly. In addition, a simple but efficient clustering algorithm (e.g., the K-means algorithm was adopted as the clustering method in our implementation) and a small number of clusters are sufficient to achieve this improvement, thus alleviating concerns about the overhead of the clustering procedure.

In the above work, correlation information is captured for pairs of terms, but in reality correlation may exist across more than two terms. Encoding the correlation information for a large number of term combinations will consume a
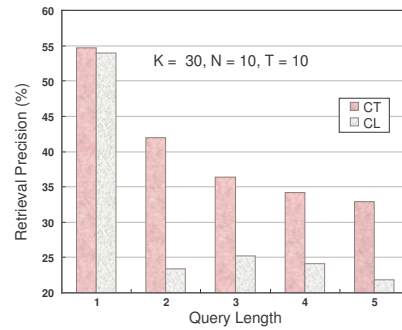

Figure 1. Uniform distribution (K: Number of clusters in each collection; N: Number of top documents retrieved from each collection; T: The number of peers that a query is forwarded to). [29]
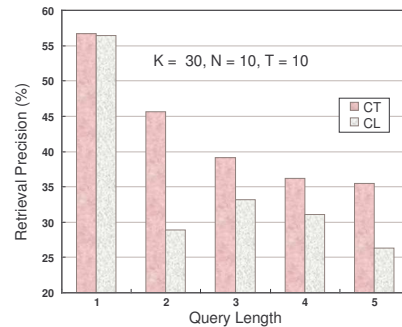

Figure 2. Zipf distribution (K: Number of clusters in each collection; N: Number of top documents retrieved from each collection; T: The number of peers that a query is forwarded to). [29]

lot of storage. To further exploit term correlation information for collection ranking, we applied latent semantic index (LSI) via singular value decomposition (SVD) to capture term correlation more comprehensively but at a comparatively low storage overhead.[7] If we represent a document collection as a matrix of term vectors, SVD decomposes this document-term matrix into three matrices: a term-concept matrix which can be used to transform a term vector into a semantic vector, a diagonal matrix and a concept-document matrix. We found that the term-concept matrix is a more effective term descriptor component than the term correlation matrix used in our previous work. To route a query, a peer computes a semantic vector for the query with respect to the term-concept matrices of its neighboring peers. We showed that semantic vectors with better sparseness and large L2-

---

[4]In [29], the term correlation method was presented in the context of a metasearch engine, but the idea of selecting search engines to send a query is the same as selecting peers to route a query.

[5]TREC Volume 4, May 1996 Collection includes material from the Financial Times Limited (1991, 1992, 1993, 1994), the Congressional Record of the 103rd Congress (1993), and the Federal Register (1994).

[6]TREC Volume 5, April 1997 Collection includes material from the Foreign Broadcast Information Service (1996) and the Los Angeles Times (1989, 1990).

[7]Ongoing research work not yet published.

norm correspond to more fruitful peers.

3.2 DPTree

Most structured P2P systems are based on DHT's and thus cannot efficiently support ranked partial matched queries that are vital for IR. In [28], we developed a distributed index called *DPTree* which efficiently supports full-text ranked partial match queries on peer-to-peer networks. The idea is that each node manages a list of relevant documents for popular queries, and organizes the document lists to be searchable within $O(\log N)$ time where N is the total number of participating nodes. Let's call the (unordered) set of keywords in a query a *pattern*. While the number of possible patterns is astronomical, we can imagine that only a relatively small number the patterns, compared to the total number of possible patterns, are frequently used in the queries. Therefore, although it is not feasible to create an index to contain all possible patterns, it is practical to do so for frequently used patterns, which can be mined from the query history. In fact, query history has been utilized successfully in other peer-to-peer search systems to improve search performance [4, 18].

The distributed pattern trees (DPTree) was developed to support the organization of patterns and pattern mining. By definition, a DPTree is a tree structure that can be implemented on one or more machines. Each DPTree node corresponds to a pattern. In particular, the root of a DPTree represents a single-word pattern, its children are responsible for 2-word patterns, and its grand children correspond to 3-word patterns, etc. Each node maintains an index to the list of documents matching the pattern maintained by the node. The roots of the DPTree's are positioned on the peers following the Chord protocol. Figure 3 displays a simple example of this DPTree-based ring structure.
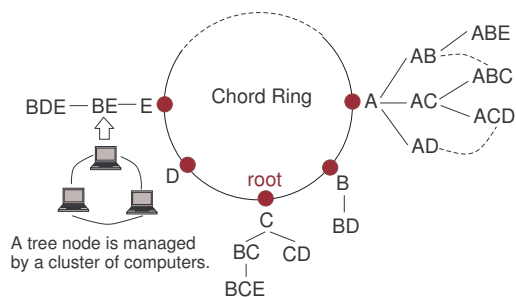

Figure 3. Distributed pattern trees positioned along a Chord ring.

A DPTree node is capable of initiating, forwarding and responding to queries. During the search procedure, a DPTree node selectively records a query history, from which

frequent patterns can be mined periodically. A DPTree starts with a single-word pattern (i.e., the root node) and is expanded and adapted dynamically based on the frequent patterns found. The roots of the DPTree's form an addressable network using distributed hash tables. By mining the query history, our approach is able to answer most queries quickly and precisely by managing a suitable number of frequent patterns. In addition, we employ *random access sequence* on patterns to establish strict mapping between a pattern and the DPTree that it resides. This eliminates redundant patterns across DPTree's without breaking the storage and network load balance among the peers. Another data structure called *sub-tree summary* enables a DPTree node to explore its entire sub-tree in an economical way, thus reducing the overlay maintenance cost.
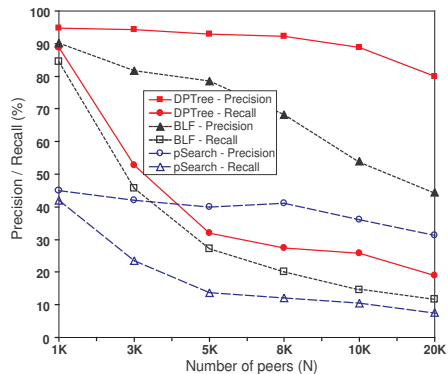

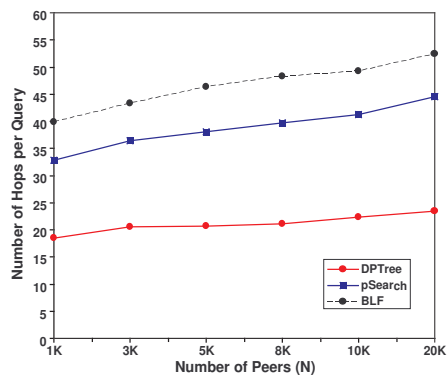**Figure 4. Precision-recall curve [28].**

Figure 5. Search Path Length [28].

In Figures 4, 5, 6, and 7, we compare our method with other two state-of-the-art systems, namely psearch [22] and Bloom Filter [15] in terms of search precision/recall, search efficiency, search cost and maintenance cost. The experimental results show that our approach achieves significant
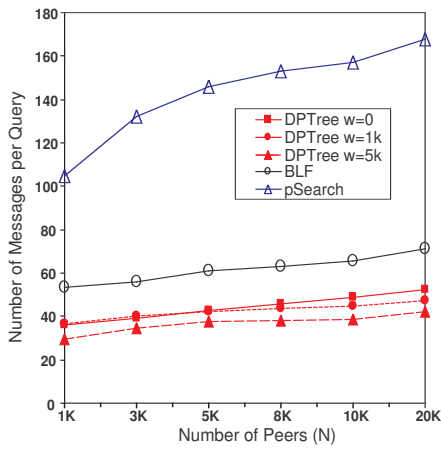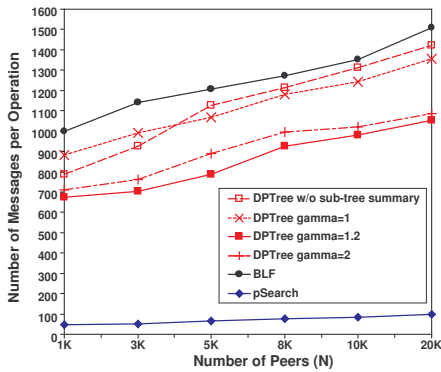
Figure 6. Search cost [28].

Figure 7. Maintenance for various number of peers [28].

gain on search effectiveness and efficiency.

4. Open Problems

In this section, we consider some open problems associated with data-sharing and search in the P2P environment that require more study. A wide variety of issues require further investigation and research such as tradeoff among autonomy, efficiency and robustness, query expressiveness, quality of service, security, anonymity, and so on. In the following, we discuss a few research issues that particularly concern IR.

4.1 Autonomy, Efficiency and Robustness Tradeoff

Autonomy, efficiency and robustness are three desirable features that a P2P system could provide. Autonomy mea-

sures a peer's independence in the network when it chooses its neighbors, indexes resources on other peers or leaves the network. Efficiency is evaluated along several lines such as storage, processing and bandwidth cost. Robustness is a network's tolerance to intensive peer joins/leaves and peer failures.

These three features, however, are often conflicting to each other. For example, structured P2P approaches such as DHT's sacrifice autonomy for efficiency by applying strict randomized hash functions on data positioning. At the same time, since these systems contain routing information so as to offer informed search, they are difficult to maintain especially when peers join and leave the network with a high frequency. Offering only local routing information, unstructured P2P systems are at the opposite extreme. Therefore, one great challenge is to seek a good tradeoff among autonomy, efficiency and robustness. For example, by restricting the topology of the P2P network and thus peer autonomy, Viceroy [11] managed to bound the maintenance overhead of a peer join/leave at a constant factor, which is better than most DHT-based method which bears logarithmic maintenance cost with respect to the network size.

The degree of autonomy and robustness needed and the sacrifice in search efficiency acceptable for IR application must be considered. Whether a P2PIR system can be tuned at a fine granularity level would be an interesting research problem.

4.2 Expressiveness

Expressiveness measures how much detail can a query language describe the desired resources. A diverse family of query languages is available in centralized systems. However, not all of them are realized in P2P networks, and here we discuss a few examples of useful query languages.

- **Ranked queries**
  The implementation difficulty of ranked keyword queries heavily depends on the ranking metric. Although quite a few works [28, 22, 10] have been done using traditional IR similarity metrics such as inner-product and Euclidean distance, these ranking methods fall short of the ranking methods employed by today's web search engines, where extensive global analysis such as PageRank and personalization methods are used. P2PIR systems will not become a main stream IR tool by replicating an older search technology. The very nature of decentralized control in P2P systems makes it difficult to implement such global ranking techniques. On the bright side, P2P systems lend themselves to ranking methods based on reputation, access frequencies and peer authority tailored for P2P systems.

- **SQL and aggregation queries**
  Although the focus of this paper is IR, IR functions are increasingly incorporated into database management systems. If P2PIR systems are to be deployed in enterprises, IR cannot exist as a separate functional module. Support of SQL requires many fundamental operations such as aggregation and join operations. Current research on supporting SQL in a P2P environment is mostly preliminary [25]. For example, RDFPeers [5] built on the Multi-Attribute Addressable Network (MAAN) which extended Chord to answer SQL-like queries over RDF triplets presented as $< subject, predicate, object >$. Although RDFPeers is a scalable approach, it only supports limited types of SQL functions and user defined schemas, which are essential for enterprise databases. Aggregation functions *Min, Max, Sum, Count* and *Avg* were studied in [2]. They are difficult to implement when the data are distributed on a huge number of autonomous peers.

- **Analytical search queries**
  Analytical search queries include both the ranked search and the SQL components. We call it analytical search because it queries about textual content as well as provides analytical functions through SQL range predicates and aggregation. It is suitable for P2P environments, where people are interested in knowing the distribution statistics of contents, comparing contents from different peers, in addition to document sharing. This type of queries is the most expressive and yet the most challenging to support due to its complexity. The ranked search and SQL components for P2P environments deserve careful study by themselves; moreover, seamless integration of the two components for P2P environments will generate great research value, given the sharp differences between database and IR processing logics and the distributed nature of P2P systems.

5. Conclusion

In this paper, we briefly surveyed P2P information retrieval (P2PIR) research and present some of our recent research in this area. We pointed out a few key research problems that must be addressed before P2PIR can be competitive to the centralized search engines. Further study is needed for P2PIR to carve out applications that are unique and complementary to existing centralized search engines.

References

[1] E. Adar, B.A. Huberman, *Free Riding on Gnutella.* Technical report. Xerox PARC, 10 Aug. 2000.

[2] M. Bawa, H. Garcia-Molina, A. Gionis, R. Motwani, *Estimating Aggregates on a Peer-to-Peer Network*, Technical Report, Computer Science Department, Stanford University, 2003.

[3] A. Bharambe, M. Agrawal, S. Seshan, *Mercury: Supporting scalable multi-attribute range queries*, Proc. SIGCOMM, 2004.

[4] H. Cai, J. Wang, *Peer-to-peer computing: Foreseer: a novel, locality-aware peer-to-peer system architecture for keyword searches*, Proc. the 5th ACM/IFIP/USENIX international conference on Middleware, Oct 2004.

[5] M. Cai, M. Frank, *RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network*, in International World Wide Web Conference (WWW), 2004.

[6] A. Crespo, H. Garcia-Molina, *Routing indices for peer-to-peer systems*, in Proc. of the 28 tn Conference on Distributed Computing Systems, July 2002.

[7] A. Gupta, D. Agrawal, A.E. Abbadi, *Approximate range selection queries in peer-to-peer systems*, Proceedings of the First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, Jan 2003.

[8] N.J. Harvey, M.B. Jones, S. Saroiu, M. Theimer, A. Wolman, *SkipNet: A scalable overlay network with practical locality properties*, in USENIX Symposium on Internet Technologies and Systems (USITS), Mar 2003.

[9] R., Konrad, *Napster Among Fastest-growing Net Technologies*, CNET news.com, October 2000.

[10] M. Li, W.C. Lee, A. Sivasubramaniam, D.L. Lee, *A Small World Overlay Network for Semantic Based Search in P2P*, 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing.

[11] D. Malkhi, M. Naor, D. Ratajczak, *Viceroy: A scalable and dynamic emulation of the Butterfly*, Proc. 21st ACM Symposium on Principles of Distributed Computing (PODC'02), July 2002.

[12] Napster, at *http://www.napster.com*.

[13] S. Ramabhadran, S. Ratnasamy, J.M. Hellerstein, S. Shenker, *Brief Announcement: Prefix Hash Tree*, in Proc 23th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), 2004.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, *A scalable content-addressable network*, In Proc. ACM SIGCOMM 2001, August 2001.

[15] P. Reynolds, A. Vahdat, *Efficient peer-to-peer keyword searching*, Proc. ACM/IFIP/USENIX Middleware Conference, volume 2672, pages 21-40, Rio de Janeiro, Brazil, June 2003.

[16] A. Rowstron, P. Druschel, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[17] C. Schmidt, M. Parashar, *Enabling Flexible Queries with Guarantees in P2P Systems*, IEEE Internet Computing, vol. 08, no. 3, pp. 19-26, 2004.

[18] Y. Shao, R.Y. Wang, *BuddyNet: History-Based P2P Search*, 23-37, ECIR, 2005.

[19] Y. Shen, Y., D.L. Lee, *A Meta-Search Method Reinforced by Descriptors of Clusters*, Proc. 2nd International Conference on Web Information Systems Engineering (WISE01), 129-136, Kyoto, Japan, Dec 2001.

[20] Y. Shen, D.L. Lee, *An MDP-based Peer-to-Peer Search Server Network*, Proc. 3rd International Conference on Web Information Systems Engineering (WISE02) Singapore, Dec, 2002, 269-278.

[21] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, In Proc. ACM SIGCOMM 2001, August 2001.

[22] C. Tang, Z. Xu, S. Dwarkadas, *Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks*, ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003.

[23] C. Tang, S. Dwarkadas, Z. Xu, *On Scaling Latent Semantic Indexing for Large Peer-to-Peer Systems*, Proc. 27th Annual International ACM SIGIR Conference, Sheffield, UK, July 2004.

[24] E. Tanin, A. Harwood, H. Samet, *Indexing Distributed Complex Data for Complex Queries*. DG.O 2004.

[25] P. Triantafillou, T. Pitoura, *Towards a Unifying Framework for Complex Query Processing over Structured Peer-to-Peer Data Networks*, VLDB '03 Workshop on Databases, Information Systems, and Peer-to-Peer Computing, September 2003.

[26] B. Yang, H. Garcia-Molina, *Improving search in Peer-to-Peer networks*, in Proc. of the 22nd International Conference on Distributed Computing Systems (ICDCS02), 2002.

[27] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, J. Kubiatowicz, *Tapestry: A resilient global-scale overlay for service deployment*, IEEE Journal on Selected Areas in Communications, 2003. 15.

[28] D.J. Zhao, D.L. Lee, Q. Luo, *DPTree: A Distributed Pattern Tree Index for Partial-Match Queries in Peer-to-Peer Networks*, in Proc. of the 10th International Conference on Extending Database Technology (EDBT), 2006.

[29] D.J. Zhao, D.L. Lee, Q. Luo, *A Meta-Search Method with Clustering and Term Correlation*, in Proc. 9th International Conference on Database Systems for Advanced Applications (DASFAA), 2004.