Information Retrieval in Peer-to-Peer Networks

D. Zeinalipour-Yazti, Vana Kalogeraki, Dimitrios Gunopulos

Department of Computer Science and Engineering
University of California - Riverside
Riverside, CA 92521, USA

Abstract

Peer-to-Peer (P2P) systems are application layer networks which enable networked hosts to share resources in a distributed manner. An important problem in such networks is to be able to efficiently search the contents of other peers. In this paper we present a survey of search techniques for information retrieval in P2P networks, including recent techniques proposed by the authors. We also present a realistic experimental evaluation and comparison of these techniques, using a distributed middleware infrastructure we have designed and implemented.

Key words: Information Search and Retrieval, Distributed systems

1 Introduction

The advances in public networks and the deployment of powerful personal computing units by end users have brought a shift from the traditional Client-Server computing model to the Peer-to-Peer (P2P) computing model. HP defines P2P [16] as a way to leverage vast amounts of computing power, storage, and connectivity from personal computers distributed around the world. In the P2P model, each node (peer) has a symmetric role of both a client and a server. Large numbers of peers collaborate in a dynamic and ad-hoc manner [23] and share information [8,12] in large-scale distributed environments without any centralized coordination. Recently the P2P model has also been

Email addresses: csyiazti@cs.ucr.edu (D. Zeinalipour-Yazti), vana@cs.ucr.edu (Vana Kalogeraki), dg@cs.ucr.edu (Dimitrios Gunopulos).

proposed in [2,10,24] as an alternative model to WWW-crawling based systems to cope with information that changes frequently.

In this paper we consider the *information retrieval* problem in P2P networks. Assume that each peer has a database (or collection) of documents that it shares in the network. The documents can be collections of text, audio, video or other semi-structured documents. A node searches for information by sending `query` messages to its peers. We assume that the queries include sets of keywords. A peer receiving a query message computes the similarity of the query against its collection of documents. Typically, this involves finding the documents that contain the set of keywords in the query. If the evaluation is successful, the peer generates a reply message which contains pointers to the matching documents.

The information retrieval problem is a more complex operation than traditional search techniques based on object identifiers or filenames, currently being used in P2P systems. The Information Retrieval (IR) community has over the years developed algorithms for precise document retrieval in static data environments (such as a corpus of documents). However these methods are not directly applicable to P2P systems where there is no central repository, there are large numbers of documents and nodes are joining and leaving in a dynamic and ad-hoc manner. Given the information explosion in the last few years as well as the multi-dimensional advantages offered by the P2P model; we believe that such new capabilities are an important step for making P2P systems applicable to a wide set of applications than simply object storage.

The remainder of the paper is organized as follows. In section 2 we present a survey of search techniques for information retrieval in P2P networks including our recently proposed Intelligent Search Mechanism. In section 3 we introduce the middleware simulation infrastructure and present some of our experimental results. Finally in section 4 we conclude the paper.

2 Information Retrieval Techniques for P2P Networks

The main challenge for information retrieval in P2P networks is to be able to guide the query to the sources that contain the most relevant answers in a fast and efficient way. In particular, our objective is to decrease the number of messages sent per query while at the same time maintain a high recall rate (i.e. search result quality). In this section we present techniques for information retrieval using keyword search. For completeness we present in section 2.9 other related work that addresses search techniques using object identifiers rather than keywords. We finally present in section 2.10 some of the most significant literature from the area of distributed information retrieval.

2.1 The "naive" Breadth First Search (BFS) Technique

BFS is a technique widely used in P2P file sharing applications such as Gnutella [8]. The BFS search protocol (figure 1a) in a peer-to-peer network N works as follows. A node q generates a `Query` message which is propagated to all of its neighbors (peers). When a peer p receives a `Query` request, it first forwards the query to all the peers, other than the sender, and then searches its local repository for relevant matches. If some node r receives the query and has a match, r generates a `QueryHit` message to transmit the result. The `QueryHit` message includes information such the number of corresponding documents and the network connectivity of the answering peer. When node q receives `QueryHits` from more than one peer, it may decide to download the file from the peer with the best network connectivity. `QueryHit` messages are sent along the same path that carried the `Query` messages; therefore no path information needs to be stored in the transmitted messages.

BFS sacrifices its performance and network utilization in the sake of its simplicity. Each query consumes excessive network and processing resources because a query is propagated along all links (including nodes with high latencies). Therefore a low bandwidth node can easily become a bottleneck. One technique to avoid flooding the whole network with messages is to associate each query with a time-to-live (TTL) parameter. The TTL parameter determines the maximum number of hops that a given query should be forwarded. In a typical search the initial value for the TTL is usually 7, which is decrement each time the query is being forwarded. When the TTL becomes 0, the message is dropped. BFS guarantees high hit rates, at the expense of a large number of messages.

2.2 The Random Breadth-First-Search (RBFS) Technique

In [11] we propose and evaluate the *Random Breadth-First-Search (RBFS)* technique that can dramatically improve over the naive BFS approach. In RBFS (figure 1b) a peer q forwards a search message to only a fraction of its peers, selected at random. The fraction of peers is a parameter [1] to the mechanism. The advantage of RBFS is that it does not require global knowledge; a node is able to make local decisions in a fast manner since it only needs to select a portion of its peers. On the other hand, this algorithm is probabilistic. Therefore some large segments of the network may become unreachable because a node was not able to understand that a particular link would lead the query to a large segment of the network.

[1] In our experiments we used a fraction of 0.5 (a peer propagates the request randomly to half of its peers).

2.3 **The Intelligent Search Mechanism (ISM)**

The Intelligent Search Mechanism (ISM) is a new mechanism for information retrieval in P2P networks (figure 1c). The objective of the algorithm, which was proposed in [11], is to help the querying peer to find the most relevant answers to its query quickly and efficiently.

Keys to improving the speed and efficiency of the information retrieval mechanism is to minimize the communication costs, that is, the number of messages sent between the peers, and to minimize the number of peers that are queried for each search request. To achieve this, a peer estimates for each query, which of its peers are more likely to reply to this query, and propagates the query message to those peers only.

The Intelligent Search mechanism consists of two components:

(1) A *Profile Mechanism*, that a peer q uses to build a profile for each of its neighboring peers. The profile keeps the most recent replies of each peer.
(2) *RelevanceRank*, which is a peer ranking mechanism that uses the peer's profiles to select the neighbors that will lead a query to the most relevant answers.

The *Profile Mechanism* is used to maintain the most recent queries and the corresponding queryhits along with the number of results. Although logically we consider each profile to be a distinct list of queries, in the implementation we use a single $Queries$ table of size $O(Td)$, which keeps the last T queries for each d neighbor. Once the repository is full, the node uses the Least Recently Used (LRU) replacement policy to keep the most recent queries.

The *RelevanceRank (RR)* function is used by a node P_l to perform an online ranking of its neighbors in order to determine to which ones to forward a query q. To compute the ranking of each peer P_i, P_l compares q to all queries in the profiling structure, for which there is a queryhit, and calculates $RR_{P_l}(P_i, q)$ as follows:

$$RR_{P_l}(P_i, q) = \sum_{\text{j = Queries answered by } P_i} Qsim(q_j, q)^{\alpha} * S(P_i, q_j)$$

The deployed distance metric $Qsim$ is the cosine similarity[1] and $S(P_i, q_j)$ is the number of results returned by P_i for query q_j. RR allows us to rank higher the peers that returned more results. In addition, we use a parameter α, which allows us to add more weight to the most similar queries. For example, when α is large then the query with the largest similarity $Qsim(q_j, q)$ dominates the formula. Consider for example the situation peer P_1 has replied to queries q_1 and q_2 with similarities $Qsim(q_1, q) = 0.5$ and $Qsim(q_2, q) = 0.1$ to the query

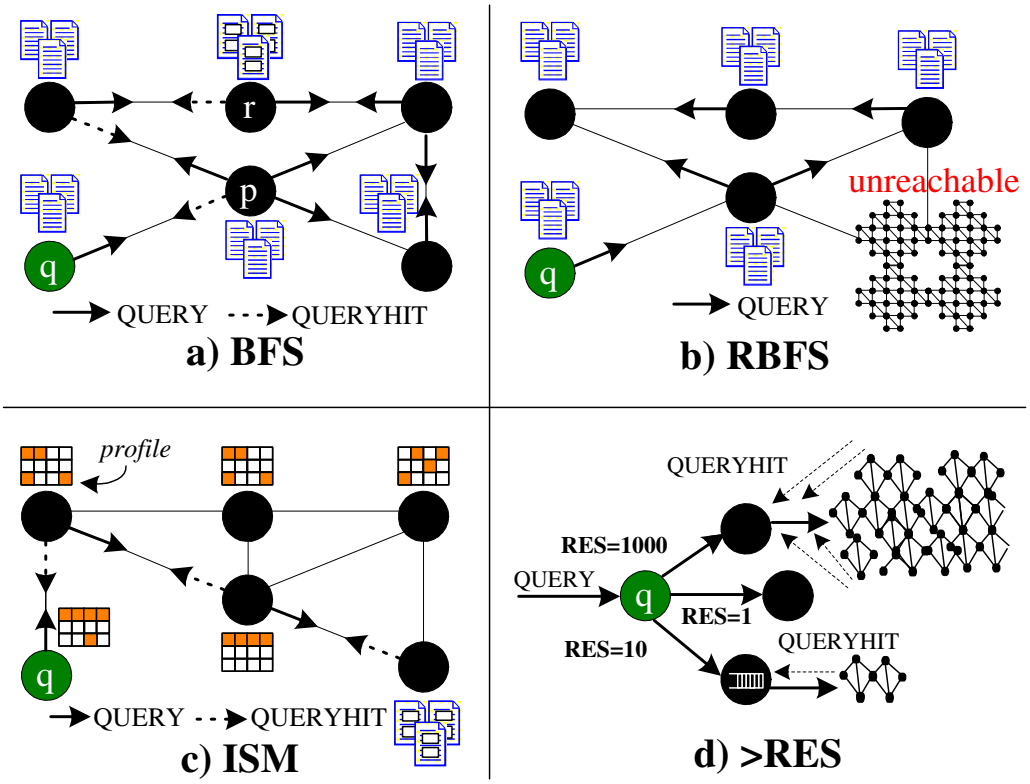

Fig. 1. *Routing Query Messages in a P2P network using:* **a) BFS** (query all neighbors), **b) RBFS** (query a random subset of neighbors), **c) ISM** (intelligently query a subset of neighbors), **d) >RES** (query the neighbors that returned the most results in the last 10 queries).

q, and peer P_2 has replied to queries q_3 and q_4 with similarities $Qsim(q_3, q) = 0.4$ and $Qsim(q_4, q) = 0.3$ respectively. If we set $a = 10$, then $Qsim(q_1, q)^{10}$ dominates, since $0.5^{10}+0.1^{10} > 0.4^{10}+0.3^{10}$. However for $\alpha = 1$ the situation is reversed because all queries are equally counted, so P_2 gets a higher relevance rank. Setting $\alpha = 0$ we count only the number of results returned by each peer (essentially, the >RES heuristic).

ISM works well in environments which exhibit strong degrees of query locality and where peers hold some specialized knowledge. Our study on the Gnutella network shows that it exhibits a strong degree of query locality. One problem of the ISM mechanism is that search messages may get locked into a cycle and consequently fail to explore other parts of the network. To solve this problem, we pick a small random subset of peers [2] and add it to the set of most relevant peers for each query. As a result, with high probability the mechanism will explore a larger part of the network and will learn about the contents of additional peers.

[2] In our experiments we additionally select 1 random peer.

2.4 Directed BFS and the Most Results in Past (>RES) Heuristic.

In [27] Yang et al., present a technique where each node forwards a query to a subset of its peers based on some aggregated statistics (figure 1d). The authors compare a number of query routing heuristics and mention that the *The Most Results in Past (>RES)* heuristic has the most satisfactory performance. A query is *satisfied* if Z, for some constant Z, or more results are returned. In >RES a peer q forwards a search message to k peers which returned the most results for the last m queries. In their experiments they chose $k = 1$ and $m = 10$ turning in that way their approach from a *Directed BFS* into a *Depth-First-Search* approach.

The >RES technique is similar to the ISM technique we propose, but uses simpler information about the peers. Its main disadvantage, with respect to ISM, is that it doesn't manage to explore the nodes which contain content related to the query. We therefore characterize >RES a *quantitative* rather than *qualitative* approach. From the experimental analysis performed in section 3 we conclude that >RES performs well because it routes queries to the larger network segments (which subsequently may also contain more relevant answers). It also captures the neighbors which are less overloaded since those neighbors usually return more results.

2.5 Searching using Random Walkers

In [15], the Random Walkers algorithm is presented. The key idea is that, each node forwards a query message, called *walker*, randomly to one of its peers. To reduce the time to receive the results the idea of the *1-walker* is extended to a *k-walker*, where *k*, instead of *one*, independent walkers are consecutively sent from the searcher. It is expected that the k-walkers after T steps will reach approximately the same number of nodes as a 1-walker after kT steps. In order to thwart duplicate messages each node may retain states. This algorithm resembles the Random Breadth First Search (RBFS) technique with the difference that in RBFS each node forwards a query message to a fraction of its neighbors. Furthermore in RBFS the incurred increase in messages is exponential while in the k-Walker model the messages used is linear. Both RBFS and k-walker do not use any explicit technique to guide the search query to the most relevant content, which is a desirable property in Information Retrieval.

Another similar technique to Random Walkers is the *Adaptive Probabilistic Search (APS)* [22] algorithm. In *APS* each node deploys a local index, which captures the relative probability of each neighbor to be chosen as the next hop for some future request. The main difference with Random Walkers is that

in APS a node utilizes feedback from previous searches to probabilistically guide future walkers, rather than forwarding the walker at random. The APS algorithm is shown to offer improved performance over the random walker model.

2.6 Using Randomized Gossiping to Replicate Global State

In *PlanetP* [6] an approach for constructing a content addressable publish/ subscribe service that uses gossiping of global state across unstructured communities is proposed. Their framework is based on a global inverted index which is partially constructed by each node n_k. More specifically n_k constructs a bloom filter b_k of its local index and propagates it to the rest of the network using gossiping. The global inverted index is then the collection of all b_i's. Bloom filters are an attractive approach for distributed environments because they achieve smaller messages which leads to huge savings in network I/O.

Given that each node n_k maintains a partially consistent list of (n_i,b_i) pairs, n_k can perform a local search to derive which nodes have the searching term and then forward the query to only those peers which have potentially some answer. Once the query reaches some node n_j, n_j can either perform an *exhaustive search* or perform a *selective search* like [21], using the vector space rank model. The main disadvantage with PlanetP is the scalability issue. Although some methods for scaling beyond communities of 10000 nodes are proposed in the paper none of them is experimentally evaluated.

2.7 Searching Using Local Routing Indices

In [5] Crespo et al., present a hybrid technique that builds and maintains local indices which contain the "direction" towards the documents. Three different techniques, namely *Compound Routing Indexes (CRI)*, *Hop-Count Routing Index (HRI)* and *Exponentially aggregated RI (ERI)* are presented and evaluated over different topologies (tree, tree with cycles and powerlaw). The ideas deployed in the routing indexes schemes can be thought as the routing tables deployed in the Bellman Ford or Distance Vector Routing Algorithm, which is used in many practical routing protocols like BGP and the original ARPAnet. The key idea is that a node knows which peers will lead to the desirable amount of documents but it doesn't know the exact path to the documents. In CRI a node q maintains for each neighbor some statistics which indicate how many documents are reachable through each neighbor. Since CRI doesn't take into account the number of hops required to reach some documents, in HRI a node q maintains a CRI for k hops. HRI has a prohibitive storage cost

for large values of k and therefore ERI is proposed. ERI addresses this issue by aggregating HRI using some cost formula.

Their experimentation reveals that ERI and HRI offer significant improvements over not using any routing index while on the same time they keep the update costs low. Since the Local Indices technique is essentially a push update, where each node sends to its peers information about its documents (along with updates every time a local update happens), thus it is complementary to the ISM approach.

2.8 Centralized Approaches

Centralized systems maintain an inverted index over all the documents in the collection of the participating hosts. These include commercial information retrieval systems such as web search engines (e.g., Google, Yahoo) and centralized P2P indexing systems [17,26]. For example, *Napster* [17] uses a central repository R to which each joining peer A uploads an index of all its shared documents. A querying node B is able to search A's documents through R. Once some desired documents are located, B can communicate with A directly (using an out-of-band protocol such as HTTP).

These techniques represent an altogether different philosophy, and they are not directly comparable. In general, one trades simplicity and robustness with improved search time and more expensive resources. Centralized approaches are faster and guarantee to find all results while the decentralized approaches allow always fresh contents and are less costly.

2.9 Searching P2P Systems Using Object Identifiers

Distributed file indexing systems such as Chord[20], Oceanstore[13] and CAN[19] allow peers to perform efficient searches using object identifiers rather than keywords. More specifically, they use a specific structure with some hashing scheme that allows peers to perform object lookup operations getting in return the address (e.g. IP) of the node storing the object. Lookups are achieved by following a path that increasingly progresses to the destination. These systems have been designed to optimize object retrieval by minimizing the number of messages and hops required to retrieve the object. The disadvantage is that they consider only the problem of searching for keys, and thus cannot capture the relevance of the documents stored in the system.

Freenet [4] is another distributed information storage and retrieval system that uses instead an intelligent *Depth-First-Search (DFS)* mechanism to locate the

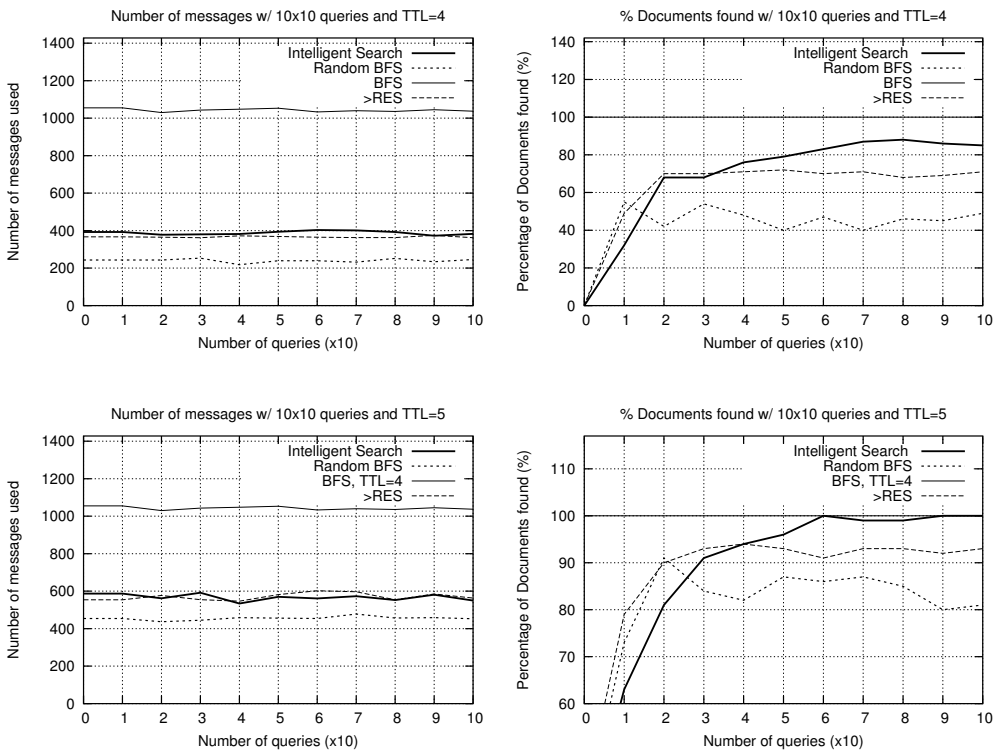

Fig. 2. **Messages** (left) and **Recall Rate** (right) used by the 4 Algorithms for 10x10 queries for **(TTL=4)** (top) and **(TTL=5)** (bottom)

object keys in the system.

2.10 Distributed Information Retrieval

The main problem in distributed information retrieval is, assuming that we want to submit a query to a subset of the databases available, decide which databases are more likely to contain the most relevant documents. A number of algorithms have been proposed and experimental results show that these algorithms achieve good results [3,7,9,25]. Recent work [14,18] shows that the performance can be improved, if the collections are conceptually separated. However, these algorithms assume that the querying party has some statistical knowledge about the contents of each database (for example, word frequencies in documents), and therefore has to have a global view of the system. In addition, most techniques assume an always-on environment.

3 PeerWare Simulation Infrastructure & Experiments

In order to benchmark the efficiency of the information retrieval algorithms, we have implemented *PeerWare*[3], a distributed middleware infrastructure which allows us to benchmark different query routing algorithms over large-scale P2P systems. For the experiments we deployed 104 nodes running on a network of 50 workstations, each of which has an AMD Athlon4 1.4 GHz processor with 1GB RAM running Mandrake Linux 8.0 (kernel 2.4.3-20) all interconnected with a 10/100 LAN.

Our evaluation metrics were: (i) the *recall* rate, that is, the fraction of documents each of the search mechanisms retrieves, and (ii) the *efficiency* of the techniques, that is, the number of messages used to find the results. To test the applicability of the information retrieval algorithms in P2P systems, we chose to implement only the algorithms that require local knowledge when searching for the documents (i.e. BFS, RBFS, >RES and ISM). RBFS, >RES and ISM all forward some query message to only half of the neighbors that would be contacted with BFS. Additionally for >RES, we set $k = 0.5 * d$ and $m = 100$, where d is the degree of a node. In this way a peer propagates the request selectively to half of its peer which returned the most results in the past 100 queries.

In our first experiment we performed 10 queries, each of which was run 10 consecutive times, and where each host has an average degree of 8 connections. The queries are keywords randomly sampled from the dataset. Figure 2 (top,left) shows the number of messages required by the four query routing techniques. The figure indicates that *Breadth-First-Search (BFS)* requires almost 2,5 times as many messages as its competitors with around 1050 messages per query. BFS's recall rate is used as the basis for comparing the recall rate of the other techniques and is therefore set to 100%. *Random Breadth-First-Search (RBFS), the Intelligent Search Mechanism (ISM)* and the *Most Results in the Past (>RES)* on the other hand use all significantly less messages but ISM is the one that finds the most documents. That is attributed to the fact that ISM improves its knowledge over time. More specifically, ISM achieves almost 90% recall rate while using only 38% of the BFS's messages. Figure 2 (top,right) illustrates that both >RES and ISM start out with a low recall rate (i.e. 40-50%) because they are initially both choosing their neighbors at random. Therefore their recall rate performance is comparable to that of RBFS. The values shown are the averages of 10 consecutive requests.

In the second experiment we investigated the effect of increasing the TTL parameter to our results. Figure 2 (bottom,left) shows that by increasing the

[3] The PeerWare infrastructure along with an extended version of the experiments presented here can be found in [28].

value of TTL to 5 the ISM mechanism discovers almost the same documents with what BFS finds for TTL=4. More specifically, our experimental results show that our mechanism achieves 100% recall rate (Figure 2 (bottom,right)) while using only 57% of the number of messages used in BFS. In [28] we have also shown that ISM has approximately the same query response time (QRT) (≈250ms) with its two competitors (RBFS and >RES) and far less QRT than BFS (which required 60% more time).

4 Conclusions

In this paper we presented a number of different query routing techniques that enable efficient information retrieval in P2P systems. Existing techniques are not scaling well because they are either based on the idea of flooding the network with queries or because they require some form of global knowledge. The main challenge for a query routing technique is to query peers that contain the most relevant content with minimum messaging. We have shown the various trade-offs and experimentally evaluated four of the techniques that require no global knowledge.

References

[1] R.A. Baeza-Yates and B.A. Ribeiro-Neto, "Modern Information Retrieval." ACM Press Series/Addison Wesley, New York, 1999.

[2] M. Bawa, R. J. Bayardo Jr., S. Rajagopalan, E. Shekita, "Make it Fresh, Make it Quick – Searching a Network of Personal Webservers." *Proc. of WWW 2003*, Budapest, Hungary, 2003.

[3] J. Callan, A. L. Powell, J. C. French, and M. Connell, "The effects of query-based sampling on automatic database selection algorithms". Technical Report IR-181, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts.

[4] I. Clarke, O. Sandberg, B. Wiley and T.W. Hong. "Freenet: A Distributed Anonymous Information Storage and Retrieval System". *Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, 2000.

[5] A. Crespo and H. Garcia-Molina, "Routing Indices For Peer-to-Peer Systems". *Proc. of ICDCS'02*, Vienna, Austria, 2002.

[6] F.M. Cuenca-Acuna and T.D. Nguyen, "Text-Based Content Search and Retrieval in ad hoc P2P Communities". *Int. Workshop on Peer-to-Peer Computing*, Pisa, Italy, 2002.

[7] J. C. French, A. L. Powell, J. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou, "Comparing the Performance of Database Selection Algorithms". *Proc. of ACM SIGIR'99*, Berkeley, CA, 1999.

[8] Gnutella, `http://gnutella.wego.com/`.

[9] L. Gravano and H. Garcia-Molina, "Generalizing gloss to vector-space databases and broker hierarchies". *Proc. of VLDB'95*, Zurich, Switzerland, 1995.

[10] JXTA Search, Project JXTA, `http://search.jxta.org/`.

[11] V. Kalogeraki, D. Gunopulos and D. Zeinalipour-Yazti, "A Local Search Mechanism for Peer-to-Peer Networks". *Proc. of CIKM'02*, McLean VA, USA, 2002.

[12] Kazaa, Sharman Networks Ltd. `http://www.kazaa.com/`

[13] J. Kubiatowicz et.al, "OceanStore: An Architecture for Global-Scale Persistent Storage". *Proc. of ASPLOS'00*, Cambridge, MA, 2000.

[14] Z. Lu and K. S. McKinley, "The Effect of Collection Organization and Query Locality on Information Retrieval System Performance and Design". Book chapter in Advances in Information Retrieval, Kluwer, New York, 2000. Bruce Croft, Editor.

[15] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks". *Proc. of ICS02*, New York, USA, June 2002.

[16] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, "Peer-to-Peer Computing" HP Laboratories Palo Alto, Technical Report HPL-2002-57, 2002.

[17] Napster, `http://www.napster.com/`.

[18] A.L. Powell, J.C. French, J. Callan, M. Connell, C.L. Viles, "The Impact of Database Selection on Distributed Searching". *Proc. of ACM SIGIR'00*, pages 232–239, Athens, Greece, 2000.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network". *Proc. of ACM SIGCOMM'01*, San Diego CA, 2001.

[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications." *Proc. of ACM SIGCOMM'01*, San Diego CA, 2001.

[21] C. Tang, Z. Xu, S. Dwarkadas, "Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks". *Proc. of ACM SIGCOMM'03*, Karlsruhe, Germany, 2003.

[22] D. Tsoumakos and N. Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks". *Proc. of P2P2003*, Linköpings, Sweden 2003.

[23] Skype, Skyper Ltd. `http://www.skype.com/`.

[24] T. Suel et. al, "ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval". *Proc. of WebDB'03*, San Diego, CA, 2003.

[25] J. Xu and J. Callan, "Effective retrieval with distributed collections". *Proc. of SIGIR'98*, pp. 112–120, Melbourne, Australia, 1998.

[26] B. Yang and H. Garcia-Molina, "Comparing hybrid peer-to-peer systems". *Proc. of VLDB'01*, Rome, Italy, 2001.

[27] B. Yang and H. Garcia-Molina, "Efficient Search in Peer-to-Peer Networks". *Proc. of ICDCS'02*, Vienna, Austria, 2002.

[28] D. Zeinalipour-Yazti, "Information Retrieval in Peer-to-Peer Systems". M.Sc Thesis, Dept. of Computer Science, University of California - Riverside, June 2003.