Exploiting Locality for Scalable 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

An important problem in unstructured peer-to-peer (P2P) networks is the efficient content-based retrieval of documents shared by other peers. However, existing searching mechanisms 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.

We propose the Intelligent Search Mechanism (ISM) which is an efficient, scalable yet simple mechanism for improving the information retrieval problem in P2P systems. Our mechanism is efficient since it is bounded by the number of neighbors and scalable because no global knowledge is required to be maintained.

ISM consists of four components: A *Profiling Structure* which logs queryhit messages coming from neighbors, a *Query Similarity* function which calculates the similarity queries to a new query, *RelevanceRank* which is an online neighbor ranking function and a *Search Mechanism* which forwards queries to selected neighbors.

We deploy and compare ISM with a number of other distributed search techniques over static and dynamic environments. Our experiments are performed with real data over *Peerware*, our middleware simulation infrastructure which is deployed on 75 workstations. Our results indicate that ISM outperforms its competitors and that in some cases it manages to achieve 100% recall rate while using only half of the network resources required by its competitors. Further, its performance is also superior with respect to the total query response time and our algorithm exhibits a learning behavior as nodes acquire more knowledge. Finally ISM works well in dynamic network topologies and in environments with replicated data sources.

Key words: Peer-to-Peer Networks, Distributed Information Retrieval

* *Contact author: vana@cs.ucr.edu tel: 909-787-2556, fax: 909-787-4643*
Email addresses: csyiazti@cs.ucr.edu (D. Zeinalipour-Yazti), vana@cs.ucr.edu (Vana Kalogeraki), dg@cs.ucr.edu (Dimitrios Gunopulos).

1 Introduction

Peer-to-peer (P2P) networks are increasingly becoming popular because they offer opportunities for real-time communication, ad-hoc collaboration [12] and information sharing [10,16,22] in large-scale distributed environments. Recently the P2P model has also been proposed in [3,14,28] as an alternative model to WWW-crawling based systems to cope with information that changes frequently. Peer-to-peer computing is defined as the sharing of computer resources and information through direct exchange. The most distinct characteristic of P2P computing is that there is symmetric communication between the peers; each peer has both a client and a server role. The advantages of P2P systems are multi-dimensional; they improve scalability by enabling direct and real-time sharing of services and information; enable knowledge sharing by aggregating information and resources from nodes that are located on geographically distributed and potentially heterogeneous platforms; and, provide high availability by eliminating the need for a centralized component.

In this paper we consider the *information retrieval* problem in P2P networks. Assume that each peer has a database (or collection) of documents (see figure 1) which represents the knowledge of the peer. The documents can be collections of text, audio, video or other semi-structured documents. Each peer shares its information with the rest of the network through its neighbors. A node searches for information by sending `query` messages to its peers. Without loss of generality we assume that the queries are collections of keywords and that a querying peer is interested in finding all the documents that contain a set of keywords. A peer receiving a query message evaluates the constraint locally against its collections of documents. If the evaluation is successful, the peer generates a reply message to the querying peer which includes the identifier of all the documents that correspond to the constraint. Once a querying peer receives responses from its peers it afterwards decides which documents to download. Our goal is to decrease the number of messages sent per query while at the same time maintain a high recall rate.

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 [5,10,16]. 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, the advances in public networks along with the deployment of powerful personal computing units by end users; we believe that the P2P computing model may dominate the traditional Client-Server model in the

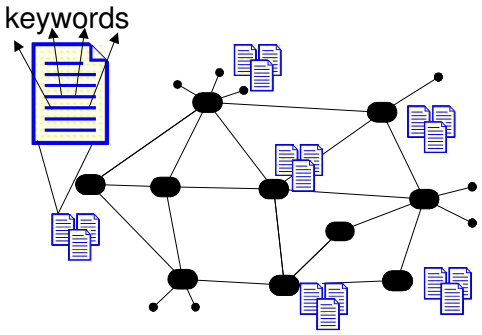

Fig. 1. **Information Retrieval in P2P systems**. Each node possesses a collection of files (text, audio or video) and nodes want to perform content-based searches over the document collection of other nodes.

coming years, and that P2P systems are going to become applicable to a wide set of applications.

Note that searching based on the file contents is not possible in most current P2P systems [5,10,22]. In those systems searching is done using file identifiers instead (such as the name of the file or the documentId). Although this allows deployment of efficient search and indexing techniques it restricts the ability of P2P users to perform content-based searches.

There are generally three type of search techniques: i) *centralized* (e.g. napster [22]), ii) *hybrid* (e.g. kazaa [16] and gnutella v0.6) and iii) *purely decentralized* (e.g. gnutella v0.4 [10]).

In *centralized* search techniques, each peer uploads an index of its files to the index server as it joins the network. These are centralized processes that exploit large databases and parallel approaches to process queries, and work extremely well. In the P2P information retrieval context however, they have several disadvantages. The biggest disadvantage is that the index needs to be an inverted index over all the documents in the network. This means that the index node has to have sufficient resources to setup and maintain such settings. Although hardware performance and costs have improved, such centralized repositories are still expensive and prohibitive in dynamic environments where nodes are joining and leaving.

Hybrid search techniques improve scalability, by employing one (or possibly more) peers with long-time connectivity and high bandwidth connection and capacity. These have additional functionality in that they become indexes for the contents of other peers or form a backbone infrastructure which can be utilized by other less powerful nodes.

Finally in *purely decentralized* search techniques each node acts independently and the actual searches are performed using brute force techniques (e.g. by broadcasting the query). However, query broadcasting becomes an expensive

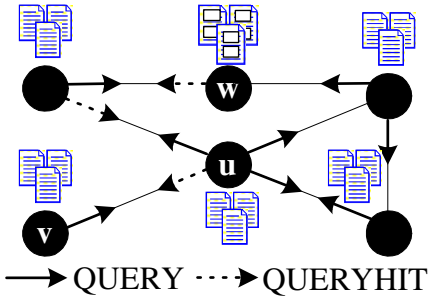

Fig. 2. Searching in a peer-to-peer network with Breadth First Search (BFS): Each peer forwards the query to all its neighbors.

operation given that such networks are extremely large. Therefore efficient information retrieval in such settings becomes a challenging task.

2 Information Retrieval in Pure P2P Networks

In this section we provide a brief overview of techniques and algorithms that can be used to perform content-based searches in P2P system. We consider a network of n nodes (peers), with average degree r (with $r << n$), that is, each peer is directly connected to around r other peers. For a given peer u, the peers in $N(u)$ are those nodes in the network that have a direct connection to u. Formally, for a given peer u, let D_u be the set of documents that are stored in u. Without loss of generality, assume that each document d is a sequence of keywords, and let $s(d)$ be the (unordered) set of keywords in d. Given a query q, itself a set of keywords, the result of the query should be the *answer set* $\{(d, u)|$ where $q \subset s(d)$ and $d \in D_u\}$, that is, the documents that include the keywords in q that are stored in peer u.

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

BFS is a technique widely used in P2P file sharing applications, such as Gnutella [10]. BFS sacrifices performance and network utilization in the sake of its simplicity. The BFS search protocol in a peer-to-peer network (see figure 2) works as follows. A node v generates a `Query` message q when it wants to search for contents located on other peers. v propagates q to all of the peers in $N(v)$. When a peer u receives a `Query` request, it first propagates q further by again using the neighbors in $N(u)$ (except the sender), and then searches its local repository for relevant matches. If some node w has a match, w generates a `QueryHit` message to transmit the result. The `QueryHit` message includes information such as the number of corresponding documents and the network connectivity of the answering peer. If, for example, node v receives a `QueryHit`

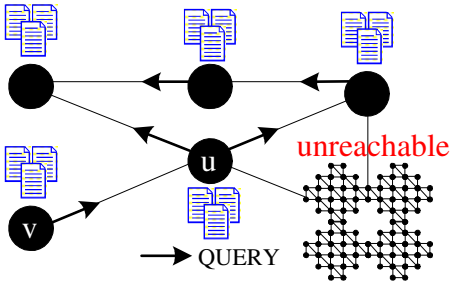

Fig. 3. Searching in a peer-to-peer network with Random Breadth First Search (RBFS): Each peer forwards the query to a random subset of its neighbors.

from more than one peer, it may choose to do the actual download from the peer with the best network connectivity. `QueryHit` messages are sent along the same path that carried the incoming `Query` messages.

The disadvantage of BFS is that a query is consuming excessive network and processing resources because a query is propagated along all links (including nodes with high latencies). Therefore the network can easily become a bottleneck. One technique to avoid flooding the whole network with messages for a single query is to associate each query with a time-to-live (TTL) field. The TTL field determines the maximum number of hops that a given query should be forwarded. In a typical Gnutella search the initial value for the TTL is usually 7, which decreases each time the query is forwarded. When the TTL becomes 0, the message is no longer forwarded. We will show in section 6 that this technique is not adequate for reducing messaging and that we can further improve on that.

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

In [15] we propose and evaluate the *Random Breadth-First-Search (BFS)* technique that can dramatically improve over the naive BFS approach. In RBFS (see figure 3) each peer u forwards a search message to only a fraction of its peers. Node u randomly selects a subset of $N(u)$ to propagate the search request. The fraction of peers that are selected is a parameter [1] to the mechanism. The advantage of this technique is that it does not require any global knowledge. Every node is able to make local decisions in a fast manner since it only needs to select some small percentage of its incoming and outgoing connections. On the other hand, this algorithm is probabilistic. Therefore some large segments of the network may become unreachable if some node does not understand that a particular link would lead the query to such segments.

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

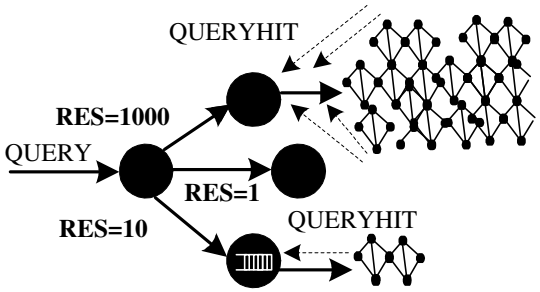

Fig. 4. The >RES heuristic is able to identify stable neighbors and neighbors connected with many others. However it fails to explore nodes which contain content related to a query.

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

In [34], Yang et al., present a technique where each node forwards a query to some of its peers based on some aggregated statistics. The authors compare a number of query routing heuristics and mention that the *The Most Results in Past (>RES)* heuristic has the best satisfaction performance. A query is defined to be *satisfied* if Z, for some constant Z, or more results are returned. In >RES a peer u forwards a search message to the k peers which returned the most results for the last 10 queries. In their experiments they chose $k = 1$ turning in that way their approach from a *Directed BFS* into a *Depth-First-Search* approach.

The technique is similar to the Intelligent Search Mechanism we propose in section 3, but uses simpler information about the peers, and is optimized to find Z documents efficiently (for a fixed Z) rather than finding as many documents as possible. The nature of >RES (see figure 4) allows it to explore the larger network segments (which usually also contain the most results) and the most stable neighbors (the peers that have routed back many queryhits), but 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.

2.4 Other Related Techniques and Algorithms

Routing Indices: In [6], Crespo et al., present a hybrid technique where each peer builds indices using aggregate information on the contents of the documents of its peers. This technique is essentially a push update technique where each peer sends to its peers information about its documents (along with updates every time a local update happens), thus it is complementary to our approach where the profiles get updated when a peer answers a query.

Random Walkers and Probabilistic Searches: In the Random Walker model, which is presented in [20], each node forwards a query message by selecting a random neighbor and the query message is called a *walker*. This model however doesn't use any explicit technique to guide the query to the most relevant content. Another similar technique to Random Walkers is the *Adaptive Probabilistic Search (APS)* [30] 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.

Randomized Gossiping and Semantic Spaces: In the *PlanetP* [7] system, participating nodes build a global inverted index which is partially constructed by each node. The framework is based on bloom filters, which capture the index of some node, and which are randomly gossiped across the community. In a different approach, the pSearch [29] system explores semantic spaces by using advanced techniques from the Information Retrieval field. It uses the Vector Space Model (VSM) and Latent Semantic Indexing (LSI) to generate a semantic space which is then distributed on top of a CAN [24] structured P2P overlay. pSearch is not designated for unstructured networks making it therefore not applicable in the context of purely distributed overlays.

Distributed IR: A number of algorithms have been proposed in [4,8,11,31,32] for distributed information retrieval in traditional database settings. In these, assuming that we want to submit a query to a subset of the databases available, the problem is to decide which databases are more likely to contain the most relevant documents. 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 have a global view of the system. In addition, most techniques assume an always-on environment. Recent work [18,23] shows that the performance can be improved, if the collections are conceptually separated.

Exploiting the P2P Network Structure: In a different approach, local search strategies that take advantage of the structure of power-law networks can be employed [1]. The algorithm explores nodes with high connectivity first. Essentially this is a directed depth-first-search mechanism.

Centralized Approaches: In centralized systems there is 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, Inktomi) as well as P2P models that provide centralized indexes [21,22,33]. 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 content and are less costly.

Searching Using Object Identifiers: Distributed file indexing systems such as Oceanstore[17], CAN[24] and Chord[27] allow peers to perform efficient searches using object identifiers rather than keywords. These systems, usually referred as *Structured Overlays* or *Distributed Hash Tables (DHT)*, use a specific structure with some hashing scheme that allows peers to perform object lookup operations getting in return the address 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, as opposed to *Unstructured Overlays* (e.g. Gnutella [10]), in which the network structure doesn't provide any lookup guaranties.

A disadvantage of DHTs is that they consider only the problem of searching for keys, and thus cannot perform content-based retrieval. Recent work in [13] shows that content-based query resolution is feasible in DHT systems if these are using *Rendezvous Points (RP)*. More specifically the framework proposes the registration of the content (i.e. attribute-value pairs that describe the content) at *RP*s. Queries might then be routed, using Chord, to a predefined set of *RPs* which consequently resolve the query.

Freenet [5] is another distributed information storage and retrieval system that uses instead an intelligent *Depth-First-Search (DFS)* mechanism to locate the object keys in the system. The advantage of DFS search is that a small set of peers can be queried quickly and efficiently; however by its nature it can take a long time if we want to find all the results to a query.

2.5 Our Contribution

In this paper we consider a fully distributed technique for addressing the information retrieval problem in pure P2P networks. More specifically:

- We propose the Intelligent Search Mechanism (ISM), which is an efficient, scalable yet simple mechanism for improving the information retrieval problem in P2P systems. ISM is entirely distributed, requires only local knowledge and therefore scales well with the size of the network.

- We provide an extensive experimental study, using our distributed newspaper infrastructure, which shows that our algorithm scales to large datasets, works well in dynamic environments, achieves high recall rates and reduces

networking costs. We compared the performance of ISM with other popular techniques and show that ISM outperforms its competitors and that it exhibits a learning behavior.

The remainder of the paper is organized as follows: In section 3 we present the Intelligent Search Mechanism. In section 4 we make an analytical study of the characteristics of the compared techniques. Section 5 describes our simulation methodology and our middleware infrastructure. In section 6 we present our experimental evaluation over static and dynamic environments by using different datasets and topologies and section 7 concludes the paper.

3 The Intelligent Search Mechanism (ISM)

The Intelligent Search Mechanism (ISM) is a new mechanism for information retrieval in P2P networks. The objective of the ISM algorithm [2], is to help the querying peer to find the most relevant answers to its query quickly and efficiently rather than finding the larger number of answers.

Our algorithm exploits the locality of past queries by using well established techniques from the Information Retrieval field. Our technique is entirely distributed and a node can make local and autonomous decisions without coordinating with any other peers, which therefore leads to reduced networking and processing costs. 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 (see figure 5).

The Intelligent Search mechanism for distributed information retrieval consists of four components:

(1) A *Profiling Structure* that a peer u uses to keep a profile for each of its peers in $N(u)$. The profile keeps the most recent past replies of each peer.
(2) A *Query Similarity* function that a peer uses locally to find the similarity between different search queries.
(3) *RelevanceRank*, which is a peer ranking mechanism that a peer u runs locally using the profiles of its peers and some query. The mechanism ranks the peers in $N(u)$, so that u can send the search query to the peers that will most likely have an answer.
(4) A *Search Mechanism* to send the query to the peers. This is the only

[2] An earlier version of the algorithm was presented in [15].

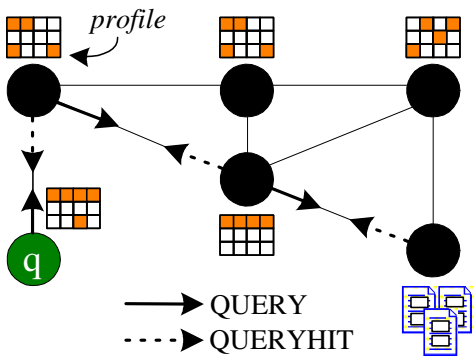

Fig. 5. Searching in a peer-to-peer network with the Intelligent Search Mechanism (ISM): Each peer uses the knowledge it obtains from monitoring the past queries to propagate the query messages to only a subset of the peers.

mechanism used by a node to communicate with its peers. It is the same mechanism employed by the Gnutella protocol for communications between peers.

3.1 **Profiling Structure**

To decide to which peers a query will be sent, a node ranks all its peers with respect to a given query. The number of peers that a query will be sent is a parameter that is defined by the user. To rank its peers, each node maintains a profile for each of its peers. The profile contains the list of the most recent past queries, which peers provided an answer for a particular query as well as the number of results that a particular peer returned. Although logically we consider each profile to be a distinct list of queries, in the implementation we use a single *Queries* table which records the described information.

Some node accumulates the list of past queries by continuously monitoring and recording the `Query` and the corresponding `QueryHit` messages it receives. For each node this list is incomplete, because each node can only record information about those queries that were routed through it. In order to limit the number of queries in each profile a node uses a size limit T. Once the repository is full, the node uses a Least Recently Used (LRU) policy to keep the most recent queries in the repository. Since the node keeps profiles for its d neighbors only, the total size of the repository is $O(Td)$.

The Profiling Structure of a host P_l is shown in table 1. As we can see, each query q_j that was routed through P_l is logged along with the peers $\{P_1, P_2, ..., P_d\}$ from where a queryhit came from. If no queryhits where routed back for a particular query, such as the query `"industrial waste disposal"`, then the Connections and Hits column contains a `null`, otherwise it contains

Table 1
Profiling Structure Snapshot. It shows from which neighbors (i.e. {P1,P2...}) each queryhit came from and the corresponding arrival time (timestamp).

Query Keywords	**GUID**	**Connections & Hits**	**Timestamp**
amazon rain forest	G568FS	(P_1,50),(P_4,80),...,(P_5,10)	10000000
industrial waste disposal	OF34QA	*NULL*	10001000
...	...	....	...
new fuel sources	LQI65D	(P_2,20), (P_3,30)	10012300

the $S(P_i, q_j)$ pair which shows the number of results that came from peer P_i for query q_j. The ranking function's performance is bounded by the number of entries and therefore yields good performance when the number of entries in the table is limited.

3.2 Query Similarity Function: The Cosine Similarity

In order to find the most likely peers to answer a given query we need a function $Qsim : Q^2 \rightarrow [0, 1]$ (where Q is the query space), to compute the similarity between different queries. Since the queries are sets of keywords, we can use query similarity metrics, such as the cosine similarity, which are deployed in the information retrieval field. We make the assumption that a peer that has a document relevant to a given query is also likely to have other documents that are relevant to other similar queries. This is a reasonable assumption if each peer concentrates on a set of topics.

The *cosine similarity* metric [2,26] (formula 1) between 2 vectors ($\vec{q}$ and $\vec{q_i}$) has been used extensively in information retrieval for nearest neighbor searches, and we use this similarity function in our setting as queries consists of keywords. Let L be the set of all words that have appeared in queries. We define an $|L|$-dimensional space where each query is a vector. For example, if the set L consists of the words $\{A, B, C, D\}$ and we have a query A, B, then the vector that corresponds to this query is (1,1,0,0). Similarly, the vector that corresponds to query B, C is (0,1,1,0). In the cosine similarity model, the similarity (sim) of the two queries is simply the cosine of the angle between the two vectors.

$$sim(q, q_i) = cos(q, q_i) = \frac{\sum(\vec{q} * \vec{q_i})}{\sqrt{\sum(\vec{q})^2} * \sqrt{\sum(\vec{q_i})^2}} \quad (1)$$

Although we use the cosine similarity, the choice of the similarity function

between two queries is orthogonal to the rest of the technique. We could therefore plug into our framework any other numeric similarity metric for documents which can be computed locally. Such metrics, which are listed in [26], are the Jaccard coefficient, the dice coefficient and the inner product.

3.3 **Peer Ranking and *RelevanceRank***

For each query received by a node P_l, P_l uses the profiles of its peers to find which ones are more likely to have documents that are relevant to the query. To compute the ranking, P_l compares the query to previously seen queries and finds the most similar ones. We formalize this procedure by defining the *RelevanceRank (RR)* function (formula 2), which is used by a node P_l to perform an online ranking of its neighbors and which will allow P_l to determine the most relevant neighbors to forward a query to. 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:

$$RR_{P_l}(P_i, q) = \sum_{\text{j="Queries answered by } P_i\text{"}} Qsim(q_j, q)^{\alpha} * S(P_i, q_j) \qquad (2)$$

where the similarity metric $Qsim$ is the cosine similarity, which was described in the previous subsection, 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 where 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 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).

In order to understand how the RR function works consider the following example. Peer P_l wants to send a query q to only two of its three peers P_1, P_2 and P_3. For simplicity assume that $\alpha = 1$ and that $\forall i, j\ S(P_i, q_j) = 2$. Let q_1, q_2, q_3, q_4, q_5 be the queries that P_l has information about (in its profiling structure), with $Qsim(q, q_1) = 0.8$, $Qsim(q, q_2) = 0.6$, $Qsim(q, q_3) = 0.5$, $Qsim(q, q_4) = 0.4$, and $Qsim(q, q_5) = 0.4$. Now if peer P_1 answered q_1, peer P_2 answered queries q_2 and q_3, and peer P_3 answered queries q_4 and q_5, then we compute the aggregate similarities of the three peers to the query q as

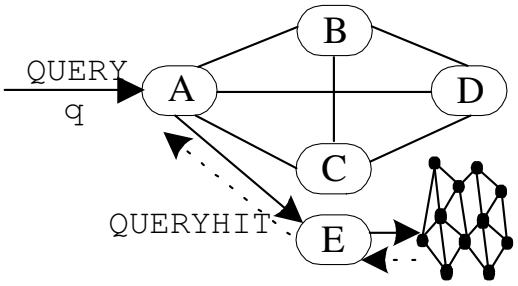

Fig. 6. With Random Perturbation we give node A the opportunity to break the cycle (A,B,C,D) in which queries may get locked and therefore allow it to explore a larger part of the network and find the correct answers.

follows: $RR_{P_l}(P_1, q) = 0.8^1 * 2 = 1.6$, $RR_{P_l}(P_2, q) = (0.6^1 + 0.5^1) * 2 = 2.2$, and $RR_{P_l}(P_3, q) = (0.4^1 + 0.3^1) * 2 = 1.4$. Therefore P_l chooses to send the query to only peers P_1 and P_2.

3.4 The Search Mechanism

The Search Mechanism deployed by ISM is very similar to the mechanism used in the Gnutella Network. A node utilizes its pre-established, socket connections, to its peers, to forward the query messages. The main difference is that ISM utilizes the profiling structure along with the RR function, to evaluate a neighbor's relevance prior to forwarding a query message.

When a peer u receives a `Query` request, it first performs an online ranking of its neighbors by invoking RR. The RR function evaluation that a peer performs is cheap, given that the profiling structure is kept small and in-memory. After node u finds the k peers with the highest relevance rank, for a given threshold k, it forwards the `Query` to these peers only. It then searches its local repository for relevant matches. As with the Gnutella protocol if some node w has a match, w generates a `QueryHit` message and transmits the results along the same path that carried the incoming `Query` message.

3.5 Random Perturbation

One problem of the technique we outline above is that it is possible for search messages to get locked into a cycle. The problem is that the search will fail then to explore other parts of the network and may not discover many results.

Consider for example figure 6 and the following scenario: Peer A receives a query q which has no answer from nodes A,B,C or D. Suppose that A chooses

to forward q to B,C and D because these nodes have successfully answered a similar query in the past. Therefore A doesn't choose node E which, this time, could lead him to the correct results. Consequently, q gets locked in a cycle (i.e. A,B,C,D) and fails to explore other segments of the network which may contain documents in its answer set.

To solve this problem, we pick a small random subset of peers [3] and add it to the set of best peers for each query. As a result, even if the best peers form a cycle, our mechanism will explore a larger part of the network that can potentially lead us to additional results that would otherwise remain unexplored.

3.6 Extend ISM to Different Environments

Although we propose the ISM mechanism for keyword-based searches, the basic mechanism can be used for general content-based retrieval and general datasets in peer-to-peer systems as long as a similarity function between the queries can be provided. For example in the context of content-based image retrieval a query could be of the form: q =*"Find the images which contain approximately x% Gray, y% Red and z% Black"*. To use the ISM mechanism, we need an appropriate similarity measure between such queries that exploits the color feature. A simple solution would be to consider the 3-dimensional space defined by taking the Gray, Red and Black axes. In this space, each query defines a 3-dimensional point and the distance between the queries is simply the Euclidian distance [25] of the points.

The ISM mechanism described in the previous subsections could easily become the query routing protocol for some hybrid P2P environment. In such an environment some nodes, known as *SuperPeers or UltraPeers*, form a backbone infrastructure which can be utilized by other less "powerful" nodes. In this context "powerful" SuperPeer nodes are those that exhibit long-time network connectivity, have high bandwidth connections and capacity. The rest nodes, which we denote as *RegularPeers*, are usually less stable and powerful. Such a model has been successfully incorporated in several systems, such as Kazaa[16] and Gnutella[10] v0.6, mainly because it allows the network size to grow to millions of users and because it differentiates short-time connection and modem users, which would otherwise become the bottleneck in query routing, from other more powerful users (e.g. ADSL, cable modem users).

ISM could be deployed in a hybrid P2P environment (see figure 7) in the following way: First some RegularPeer v obtains a list of active SuperPeers (through some out-of-band mechanism). It then connects to one (or more)

[3] In our experiments we additionally select one random peer.

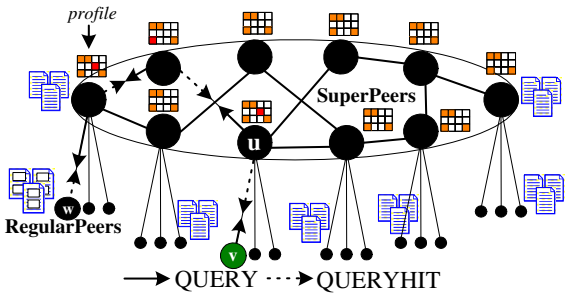

Fig. 7. Searching in a Hybrid P2P Network using ISM: Each RegularPeer posts its queries to its first immediate SuperPeer. The SuperPeer then forwards the query to only a subset of its neighboring SuperPeers using the ISM Mechanism.

SuperPeers and posts its queries when it wants to search for data. The SuperPeer u will then utilize the ISM mechanism and forward the query to a selective subset of its SuperPeer neighbors, rather than broadcasting it to all. Therefore the number of messages used at the SuperPeer level is expected to be reduced significantly. The SuperPeer u might also forward the query to all its dependent RegularPeer nodes although by keeping full indexes or bloom filters of their data, it is expected to be much more efficient. Once the query reaches some node which has an answer to the query, either SuperPeer or RegularPeer, it then sends a queryhit back along the same path the query message was received.

4 Analysis of the Proposed Techniques

In this section we describe an analysis of the proposed techniques, in comparison with the Gnutella protocol, which is a BFS Algorithm with some TTL (Time-To-Live) parameter that limits the depth a query travels. We concentrate on the *recall rate*, that is, the fraction of documents our search mechanism retrieves, compared to the other mechanisms, and the *efficiency* of the technique, that is, the ratio of number of messages that the different techniques use in order to retrieve a similar set of results for the same search.

4.1 Performance of the Random BFS Algorithm

We first consider the performance of the Random BFS technique where each peer selects a random subset of its peers to propagate a request. In a P2P network with a random graph topology, this mechanism searches only a subset of the nodes in the graph. This will result in fewer messages compared with the BFS algorithm.

Consider a random graph G with n nodes and e edges, which has average degree d. For a given node u, let $N_k(u)$ be the set of nodes at distance at most k from u. When a node u starts a Gnutella search with a TTL = k (Time To Live, as per the Gnutella search protocol), u sends approximately d messages to its neighbors, each being propagated k times. Since the BFS mechanism explores all the edges in the graph, the number of messages send by the Gnutella protocol is at least $|N_k(u)|\frac{|N_k(u)|}{n}d$.

Assume on the other hand that each node only propagates the message to a randomly chosen subset of its neighbors, of size $\frac{d}{m}$ (for a suitably chosen m). Using the same TTL (k), if $|N_k(u)|$ is smaller than $n/2$, the expected total number of messages sent is $(\frac{d}{m})^k$, and the expected number of vertices that this random BFS process visits is at least $\frac{1}{2}(\frac{d}{m})^k$. This is because if $|N_k(u)|$ is smaller than $n/2$, then most of the nodes visited in each iteration are new nodes. Consider a node v of distance i ($i < k$) from u. If $|N_k(u)| < n/2$, with high probability each edge of v is connected to a node not in $N_i(u)$. Setting $\frac{1}{2}(\frac{d}{m})^k = |N_k(u)|$, we have that, if $|N_k(u)| \approx n/2$, the random BFS needs at most a fraction of $\frac{4}{d}$ of the number of messages used by the Gnutella protocol to visit approximately the same number of vertices.

4.2 Performance of the Intelligent Search Mechanism

The previous discussion indicates that propagating a query to only a subset of neighbors in a random topology is more efficient than using the Gnutella protocol. However this approach is approximate, and cannot guarantee that all nodes in $N_k(u)$ are found. Consider for example a case where two large sub-graphs are connected by one edge. If the node attached to that edge does not choose this edge, the other sub-graph will never be explored.

The Intelligent Search technique we outlined in the previous section attempts to identify edges that are likely to have good information. Nevertheless, the accuracy of the mechanism clearly depends on how accurately a peer can compute which of its peers is likely to answer a given query. Work on distributed information retrieval has shown that current techniques for database selection can give good performance. Recent work [4] shows that even incomplete knowledge is sufficient to achieve good results. Their presented experiments show that requesting a random set of documents from a collection is sufficient to obtain accurate estimates on the word frequencies in this collection. These results are directly applicable only for the case that each peer has full statistical information for its peers. Our setting is different because the information we collect is incomplete; we keep only the queries that peers reply to, rather than all the documents in the actual replies. Nevertheless, this information is useful because the union of the queries a peer has replied to is a subset of

Table 2
Top 20 Queries on the Gnutella network in June 2002 (offensive queries marked with '_'). The total set includes 15 million query messages.

#	Query	Occurr.	%	#	Query	Occurr.	%
1	divx avi	588,146	3,88%	**6**	s__ mpg	27,895	0,18%
2	spiderman avi	50,175	0,33%	**7**	Eminem	27,440	0,18%
3	p___ mpg	39,168	0,25%	**8**	eminem mp3	25,693	0,16%
4	star wars avi	38,473	0,25%	**9**	dvd avi	25,105	0,16%
5	avi	29,911	0,19%	**10**	b_____	24,753	0,16%

the union of the documents stored in this peer. In addition this information is certainly very useful when very similar queries repeat. We also note that the more efficient search allows us to use a larger TTL compared with the Gnutella protocol, while still having a smaller number of messages overall.

Our proposed algorithm is designed to work well in environments where there is high locality of similar queries. In order to see what the real trends are, we made an extensive analysis of the network traffic found in a real P2P network [35]. In June 2002 we crawled the Gnutella network with 17 workstations for 5 hours and gathered 15 million query messages. Table 2 presents the ranking of the top 10 queries. We can clearly see that most queries are submitted in large numbers and hence there exist a high locality of specific queries. This observation is exploited by our proposed Intelligent Search Mechanism.

5 Experimental Evaluation Methodology

Our experimental evaluation focuses on: (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 as well as the required time to locate the results. As the baseline of comparison we used the results retrieved by the BFS algorithm, which almost provides us with the results we would get if we queried the collection as a corpus of documents. We chose to implement the algorithms that require only local knowledge (i.e. BFS, RBFS, >RES and ISM) which makes them directly comparable.

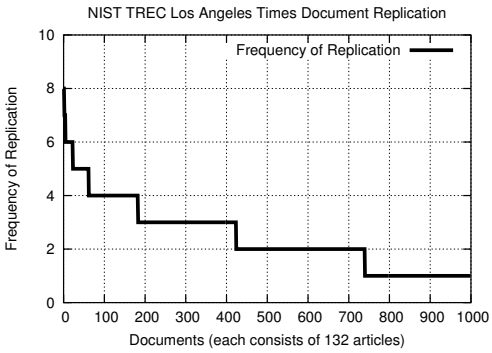

Fig. 8. Data Replication scheme for the TREC-LATimes dataset

5.1 Description of Datasets

We use three series of experiments which are based on the *Reuters-21578* and the *TREC-LATimes* datasets.

(1) **Reuters-21578**. This document collection consists of 21,578 articles that appeared on the Reuters newswire in 1987. In order to generate some specialized knowledge for each node we categorized the documents by their country attribute. This process resulted in a 31MB dataset of 104 country files, each of which had at least 5 articles, with a total of 22,769 articles. Each document is stored and queried using the GMD-IPSI XQL engine [9], which allows efficient querying of XML documents but which doesn't provide any additional IR functionality (e.g. stemming, stop-words, etc.). We used this dataset to evaluate the algorithms over a random topology of 104 peers where each node has an average degree [4] of 8. We will refer to these peers as the *Reuters-21578 Peerware*.

(2) **TREC-LATimes**. This document collection consists of randomly selected articles that appeared on the LA Times newswire from 1989 to 1990. The size of this dataset is 470MB and it contains approximately 132,000 articles. These articles were horizontally partitioned into 1000 documents each of which was subsequently indexed using the Lucene [19] IR API. These indexes, which are disk-based, allow the efficient querying of text-based sources using many IR features. We then generate a random topology of 1000 peers in which each peer shares one or more of the 1000 documents (see figure 8). We use this scheme in order to provide some degree of article replication. We don't use the *"qrels"* relevance judgments, since the compared algorithms don't attempt to address the issue of precise document retrieval. We will refer to these peers as the *TREC-LATimes Peerware*.

[4] random graphs with more than $\log n$ average degree are almost certainly connected.

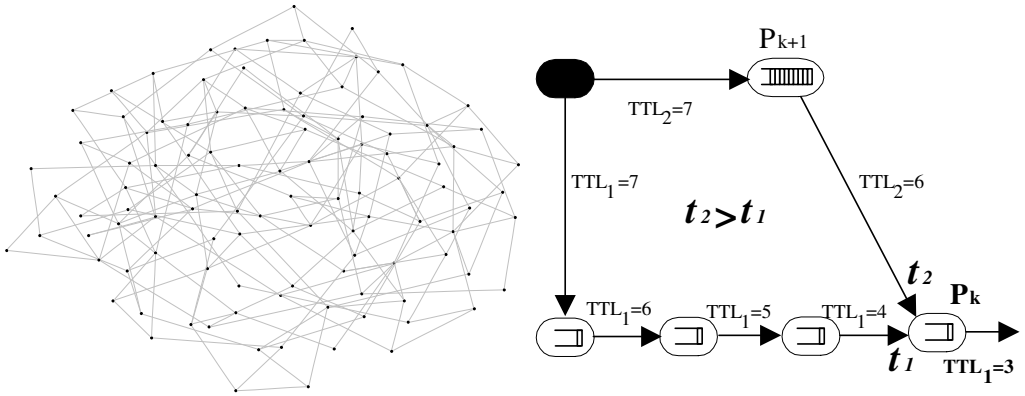

Fig. 9. a) graphgen's visualization of a random graph of 104 peers (degree=4) and b) The Discarded Message Problem: Node P_k does not forward query q with TTL_2=6 since it has already forwarded q with TTL_1=4.

5.2 **Peerware Simulation Infrastructure**

In order to benchmark the efficiency of the information retrieval algorithms, we have implemented *Peerware*[5], a distributed middleware infrastructure which allows us to benchmark different query routing algorithms over large-scale P2P systems. We use Peerware to build a decentralized newspaper network which is organized as a network of nodes. Our experiments are performed on a network of 75 workstations (each hosting a number of nodes), each of which has an AMD Athlon 800MHz-1.4GHz processor with memories varying from 256MB-1GB RAM running Mandrake Linux 8.0 (kernel 2.4.3-20) all interconnected with a 10/100 LAN. Peerware is written entirely in Java and comes along with an extensive set of UNIX shell scripts that allow the easy deployment and administration of the system.

Peerware consists of three components: (i) *graphGen* which generates network topologies (see figure 9a) and configuration files for the various nodes participating in a given experiment, (ii) *dataPeer* which is a P2P client that is able to answer to boolean queries from its local xml repository using either the GMD-IPSI XQL Engine [9] or Lucene [19], and (iii) *searchPeer* which is a P2P client that performs queries and harvests answers back from a Peerware network. Launching a Peerware of 1000 nodes can be done in approximately 30 seconds while querying the same network can be performed in around 250ms-1500ms.

[5] Details about the Peerware infrastructure can be found in [36].

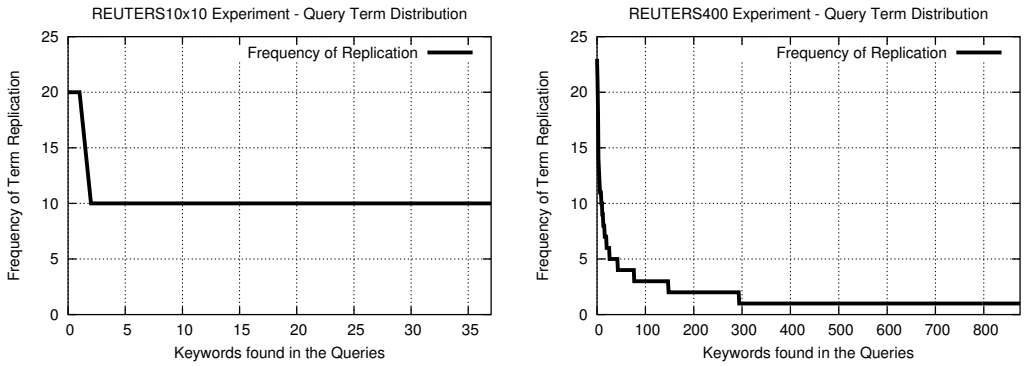

Fig. 10. **Query Term Frequency** distributions for the REUTERS10x10 and REUTERS400 querysets.

5.3 The Discarded Message Problem

One problem that affected the evaluation of the compared search techniques was the *Discarded Message Problem (DMP)* (see figure 9b). We define the DMP problem in the following way: Node P_k receives some query q with TTL_1 at time t_1. P_k first checks if it has forwarded the same query (identified by GUID) in the past. If yes, it will immediately discard the message in order to avoid forwarding the message several times. If not, it will decrease TTL_1=TTL_1-1 and forward q to some of P_k's peers. Now what happens if node P_k receives the same query q with some TTL_2, where $TTL_2 > TTL_1$ at some time t_2, where $t_2 > t_1$? Most of the commercial P2P clients will discard q. The result of the DMP problem is that a query reaches fewer nodes than estimated. We fix the DMP problem by allowing the TTL_2 message to proceed, since this may allow q to reach more peers that its predecessor TTL_1. Of course there is some redundancy which will add up in the "number of messages" graph. Unfortunately without this fix the BFS behavior is not predictable. The experimental results presented in this work don't suffer from DMP which is the reason why the number of messages is slightly higher ($\approx 30\%$) than the expected number of messages.

6 Experimental Results

In this section we describe a series of experiments that attempt to investigate the efficiency of the ISM algorithm over its competitors BFS, RBFS and >RES. We use five different querysets and evaluate the described algorithms in static and dynamic versions of the Reuters-21578 and TREC-LATimes Peerwares. A *static* version of a given topology is one in which nodes are not leaving or joining during the experiment, while a *dynamic* version is one in which nodes leave and join back after some predefined interval. We also vali-

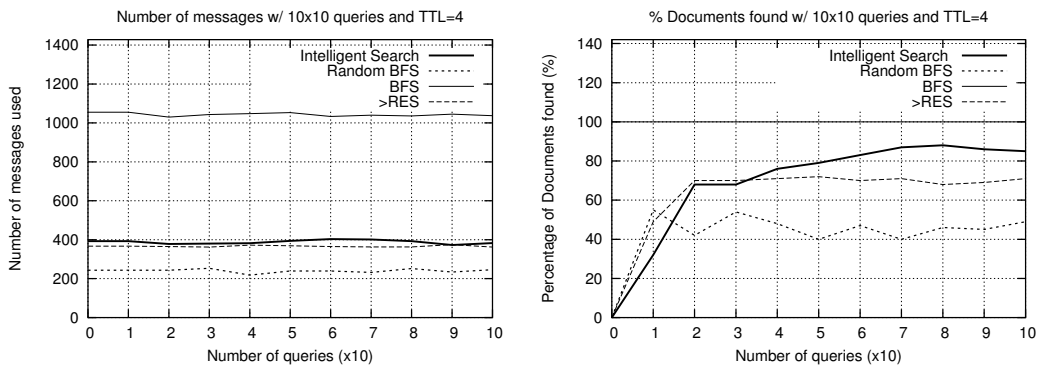

Fig. 11. **Messages** (left) and **Recall Rate** (right) used by the 4 Algorithms and the Reuters10x10 queries **(TTL=4)** over a static *Reuters-21578* Peerware.

dated the efficiency of our proposed technique when data is replicated across the P2P network. Finally we investigated how well our technique scales when the network topology grows.

6.1 Maximizing *Recall Rate* by minimizing Messaging and Time

For the first experiment we used a static version of the *Reuters-21578* Peerware, which is the network of 104 nodes described in 5.1. Since there is no standardized query set for this corpus, we generated two synthetic sets which are based on term sampling (see figure 10). More specifically we generated the following querysets:

(1) **Reuters10x10**, which is a set of 10 random queries which are repeated 10 consecutive times. Each random query consists of 4 query terms and each term has a length larger than 4 characters.
(2) **Reuters400**, which is a set of 400 random queries which are uniformly sampled from the initial 104 country files. With this assumption we make sure that the queries will refer to all the nodes rather than only a subset of them. We didn't choose to use real queries from the Gnutella network, such as `"mpeg avi superman"`, since they were related to audio/video resources and were largely irrelevant to the Reuters-21578 dataset.

Reducing Query Messages. Our objective in this experiment was to measure the number of messages used and the percentage of documents found in the case where the query messages have a TTL of 4. Figure 11 (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)*

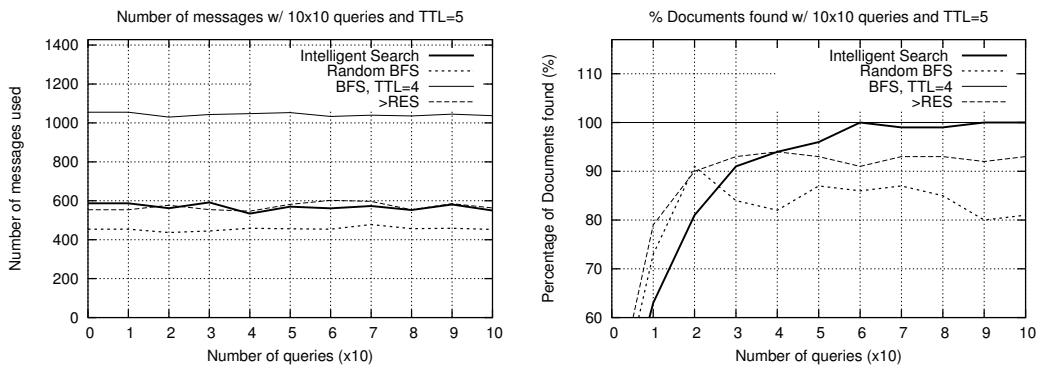

Fig. 12. **Messages** (left) and **Recall Rate** (right) used by the 4 Algorithms and the Reuters10x10 queries **(TTL=5)** over a static *Reuters-21578* Peerware.

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 BFS's messages. On the same figure 11 (right) we can see 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 is comparable to that of RBFS. In all figures presented in this paper the values shown are the averages of 10 consecutive requests.

Digging Deeper by Increasing the TTL. In the previous experiment we showed that by using RBFS, >RES or ISM we can significantly reduce the number of messages but we suffer from low recall. In the second experiment we are taking advantage of the reduced messaging of RBFS, >RES and ISM and increase the TTL parameter, which will allow us to reach more nodes deeper in the graph. Figure 12 shows that by increasing the value of the TTL field of the search requests (TTL=5), ISM discovers almost the same documents with what BFS finds for TTL=4. More specifically, ISM achieves 100% recall rate while using only 57% of the number of messages used in BFS. Another important observation is that the results for both RBFS and ISM are consistent with our analysis, and show that it is possible to search the majority of the P2P network with significantly fewer messages than the brute force algorithm.

Reducing the Query Response Time. We define the *Query Response Time (QRT)* as the interval which elapses between t_1 which is the time a node q sends out a query, until t_2 which is the time that q receives the last result from the network. Figure 13 shows the Query Response Time (QRT), as a percentage of the time taken for the BFS algorithm, for the three algorithms ISM, >RES and RBFS. BFS's QRT is in the order of 4-6 seconds while the others use only ≈30-60% for TTL=4 and ≈60-80% for TTL=5 of that time.

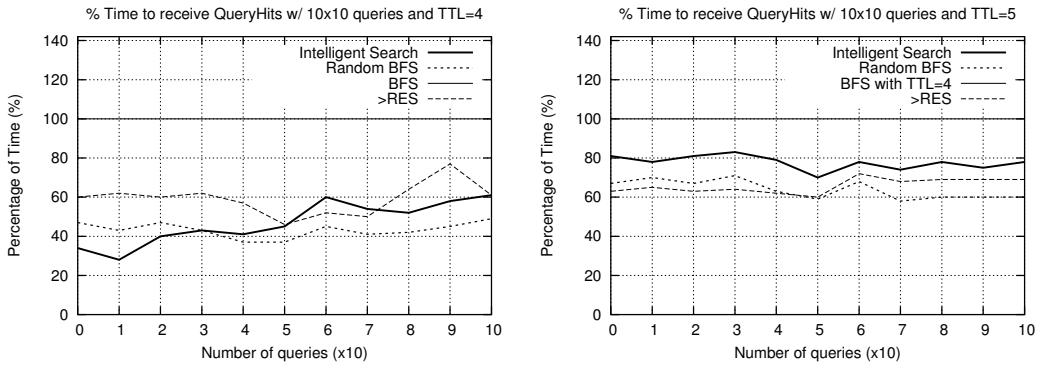

Fig. 13. **Time** used as a fraction of the time used in BFS, for the 4 Algorithms and the REUTERS10x10 queries over a static *Reuters-21578* Peerware when the queries have **TTL=4 (left)** and **TTL=5 (right)**.

From the graphs we can clearly see that the average QRT for BFS is quite large. This happens because BFS uses more messages which subsequently congest the network and which finally increase the total querying time. From the graphs we can also see that by increasing the TTL from 4 to 5 the average QRT increases. This happens because in the latter case we are exploring larger segments of the graph which consequently also produces more messages. A final point to denote is that although ISM and >RES use about the same amount of messages, ISM requires slightly more time than >RES because its decision involves some computation over the past queries.

Our study of the experimental results shows that QRT is directly affected by the actual response time r_n of a node n. r_n which is the interval that elapses between the time a query q is dequeued from n's queue until q is sent to all required k neighbors. In the Reuters experiments we found that r_n is usually 1-5 ms but it might be in some cases as high as 2500ms. This happens because in our setting several Peerware nodes run on the same physical host, which introduces large delays generated by context switching between the many hundreds of concurrent threads, and because of the Network File System (NFS), which is used by all nodes for query lookups and logging operations. In the TREC experiments, which are presented in 6.2, we change our architecture and had each peer to read and write from local disk only. This allowed us to query the 1000-node network in 250ms-1500ms. Therefore QRT doesn't provide a framework for measuring the absolute responsiveness of the various algorithms but is rather only used to relatively compare them.

Improving the Recall Rate over Time. In the previous section we used the *Reuters10x10* queryset which suits well the ISM algorithm since the query terms are repeated many times. In this experiment we use the *Reuters400* queryset, in which the query term repetition is very low (see figure 10b). In figure 14 we can see that the ISM mechanism improves its recall rate over time approaching nearly 95% recall rate while using again ≈38% of BFS's

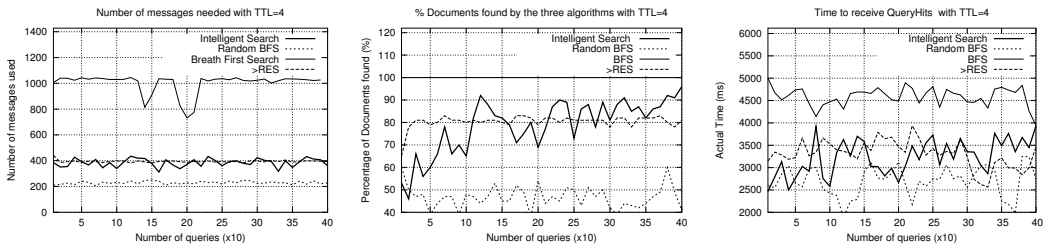

Fig. 14. **Messages** (left), **Recall Rate** (middle) and **Actual Time** (right) for the 4 algorithms using REUTERS400 queries over a static *Reuters-21578* Peerware.

messages. From the messages graph we can also see that during queries 150-200 two major outbreaks occur in BFS. This is basically an indication that some connections (i.e. sockets) broke down and that some query messages were lost. This network instability is incurred by the overwhelming amount of messages propagated by the BFS algorithm. The graph with the actual time is finally presented on the right of the same figure. It shows that BFS again requires significantly more time than the other three approaches. The QRT for RBFS, >RES and ISM is also high, as in the previous subsection, because the actual response time is affected by the architecture of the experiments.

An interesting point in this set of experimental results is that ISM requires a learning period of about 100 queries before it starts competing the performance of the >RES heuristic. This is attributed to the fact that the query term frequency is very low for the Reuters400 queryset (see figure 10b). ISM starts exploiting the locality of past queries only after its profiling structure gets populated adequately. The >RES heuristic on the other hand, is able to immediately identify the network segments that return the most results. Therefore the learning period of the >RES heuristic is significantly smaller (i.e. within the first 25 queries). We expect that ISM's learning period is acceptable given that nodes won't disconnect right after they get connected. The learning period of ISM will only become a problem, as we will see in section 6.3, if the network topology is highly unstable (>20% node failures).

6.2 Scalability and Data Replication

Thus far we have seen that with ISM, >RES and RBFS it is possible to achieve reduced messaging and time while retaining high recall rates. In this section we investigate two other important issues which are scalability and data replication. The second experimental series is performed over a static *TREC-LATimes* Peerware (described in section 5.1), which is a network of 1000 nodes in which each node possesses replicated news articles. We did this experiment in order to test if the ISM algorithm is able to scale well if the network grows. Furthermore, the incorporated data replication scheme is more realistic than the horizontal partitioning scheme that we have seen so far.

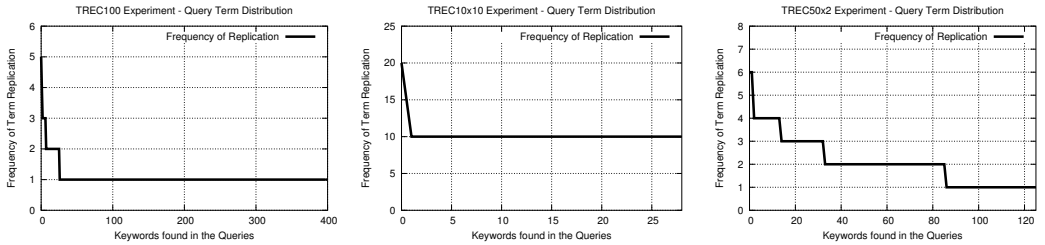

Fig. 15. **Query Term Frequency** distributions for the TREC100, TREC10x10 and TREC50x2 querysets.

For the evaluation of the TREC-LATimes corpus we will use, as indicated by NIST, the TREC topics 300-450. One problem with the provided 150 queries is that the query term frequency is very low and most terms are presented only once. This is not a realistic assumption since studies on real P2P networks (e.g. [35]) indicate that there is a high locality of query terms. Therefore we used the 150 queries to derive three new querysets. More specifically we generated the:

(1) **TREC100**, which is a set of 100 queries, out of the initial set of 150 topics, which return at least one 50% relevant article [6].
(2) **TREC10x10**, which is a list of 10 randomly sampled queries, out of the initial 150 topics, which are repeated 10 consecutive times.
(3) **TREC50x2**, for which we first generated a set $a =$"50 randomly sampled queries out of the initial 150 topics". We then generated a list b of another 50 queries which are randomly sampled out of a. *TREC50x2* is then the queries in a and b randomly shuffled.

We will use the TREC100, TREC10x10 and TREC50x2 querysets to evaluate the ISM algorithm. These querysets also attempt to address the fact that the performance of the presented algorithms varies under query sets of different term distribution. The query term frequency distribution for the three querysets is presented on figures 15.

Searching in a large-scale network topology. In our first experiment we present the number of messages and the recall rate using the TREC100 queryset, shown in figure 16. We again choose a smaller TTL parameter for BFS, (i.e. TTL=5), with which we reach 859 out of the 1000 nodes. By choosing a TTL parameter of 6 and 7 we are able to reach 998 and 1000 nodes at a cost of $8,500$ messages/query and $10,500$ messages/query respectively. The average QRT for the BFS case is in the order of 1.5 seconds but results start streaming back to the query node within the first few milliseconds.

Similar to our previous results (figures 11,12), the figures indicate that BFS requires again almost 2.5 times more messages than the other techniques. In

[6] The deployed relevance metric is Lucene's[19] default metric.

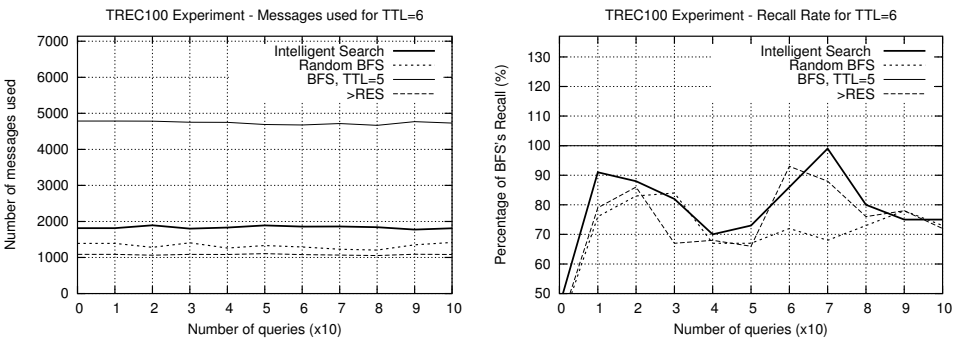

Fig. 16. **Messages** (left) and **Recall Rate** (right), for the TREC100 queryset and TTL=6 over a static *TREC-LATimes* Peerware.

addition, the recall rate validates our hypothesis that ISM won't exhibit any learning behavior if the frequency of terms is very low. The lack of repetitive query terms, in the queries, neutralizes ISM's profiling structure which accumulates information that it is never going to be utilized by *RelevanceRank*.

The effect of high term frequency. Our second experiment uses the TREC10x10 and TREC50x2 querysets and attempts to validate our claim that the recall rate will improve dramatically if the frequency of terms is high. The TREC10x10 experiment (see figure 17 top) reveals that ISM is able to achieve an even higher recall rate than what BFS finds for TTL=5. After the learning phase of the first 20 to 30 queries it scores 120% of BFS's recall by using four times less messages. This shows how prohibitive BFS becomes as the TTL ring expands.

The TREC50x2 queryset reflects a more realistic set, since a few terms occur many times in queries and most terms occur less frequently. For the TREC50x2 queryset (see figure 17 bottom), we observe that ISM monotonically improves its recall and at the 90^{th} query it again exceeds BFS performance. On the other hand both RBFS's and >RES's recall fluctuate, which indicates that >RES may behave as bad as RBFS if the queries don't follow some constant pattern (such as the one in the 10x10 queries).

6.3 Searching in Dynamic Network Topologies

Network failures in P2P systems are commonplace because of (i) the misusage exhibited at the application layer (e.g. users shut down their PCs without disconnecting), (ii) the overwhelming amount of generated network traffic or even (iii) because of some poorly written P2P clients. Such failures generate a dynamic environment in which peers are leaving or joining the network in an ad-hoc manner. In this section we aim to experimentally find the recall rate of the BFS, ISM and >RES algorithms under different levels of network

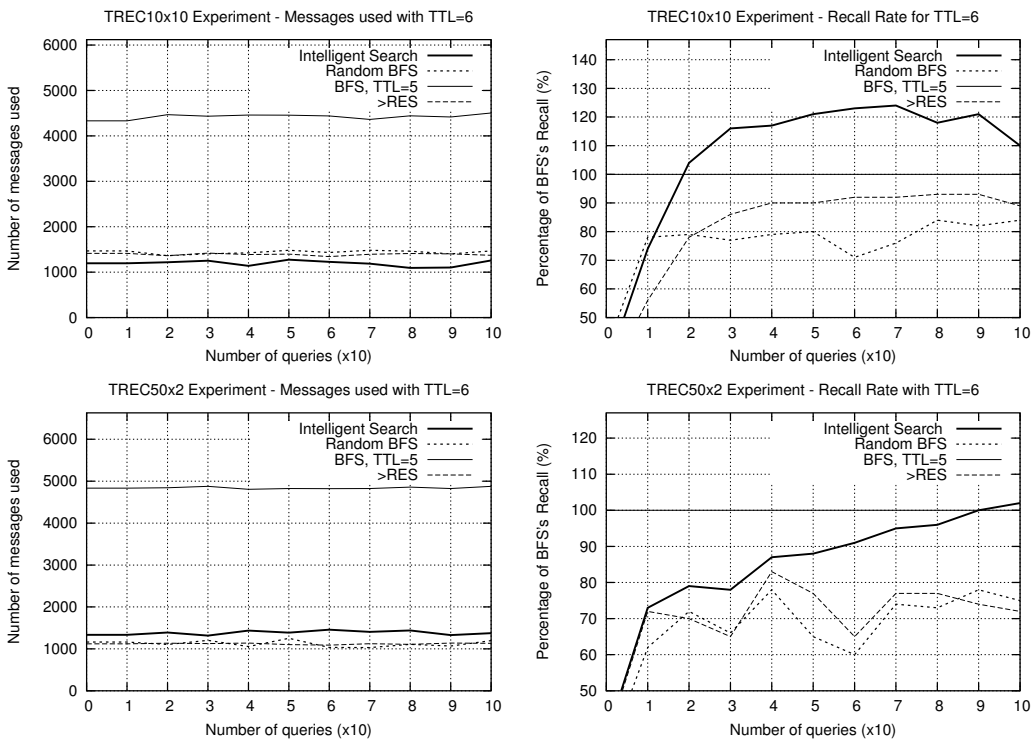

Fig. 17. **Messages** (left) and **Recall Rate** (right), for the TREC10x10 (top) and TREC50x2 (down) querysets respectively over a static *TREC-LATimes* Peerware.

failure. The observations will allow us to empirically define the maximum failure threshold to which the different techniques are tolerant to.

In order to simulate a dynamic environment we generated a scenario in which nodes are leaving or joining in a completely random way but the total number of suspended nodes at any given moment is statistically no more than *drop_rate*, where *drop_rate* is a given threshold. More specifically we have introduced into each peer p a *drop_rate* parameter which allows each peer to make a distributed decision on whether to suspend its operation or not. The *drop_rate* parameter is evaluated every k seconds against a random number r. If r is smaller than *drop_rate* then peer p will break all its incoming and outgoing connections and remain suspended for a total of l seconds. After l seconds p will attempt to re-establish its broken connections and continue its regular operation. In our experiments we set $k = 60,000ms$ and $l = 60,000ms$ and we use the TREC-LATimes Peerware along with the TREC10x10 queryset. We use four different levels of *drop_rate* (i.e. $0.0, 0.05, 0.1$ and 0.2) and r is a random number which is uniformly generated in [0.0..1.0).

In Figure 18 (left column) we can see the number of messages used by the BFS, >RES and ISM algorithms respectively. In all cases, the increase of the *drop_rate* decreases the number of messages since fewer messages are able to make it through to their destination. From the recall rate graphs (right

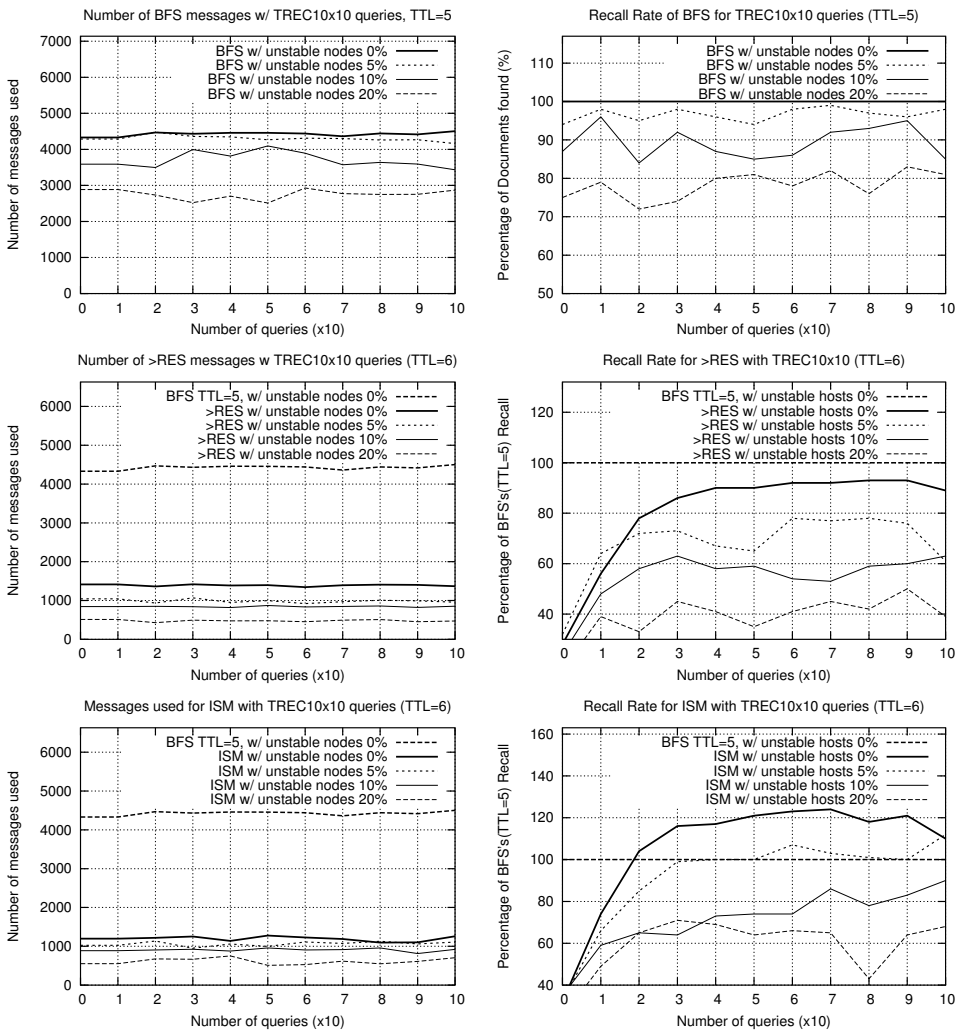

Fig. 18. **Messages** (left) and **Recall Rate** (right), when BFS, >RES and ISM perform in a dynamic *TREC-LATimes* Peerware using the TREC10x10 queryset. The basis for comparison is again the BFS algorithm (first row), which uses TTL=5, and it is compared with the >RES (second row) and ISM (third row) techniques, where each of them uses TTL=6.

column) we can observe that BFS doesn't exhibit any learning behavior at any level of *drop_rate*. Furthermore, the same graph shows that BFS is tolerable to small *drop_rates* (i.e. 5%) since the recall rate is not significantly decreased (≈3%). This happens because the BFS algorithm is highly redundant which allows nodes to receive query messages even if some edges of the graph are lost. Similarly with BFS, >RES also doesn't exhibit any learning behavior and its recall rate fluctuates at any level of *drop_rate*. On the other hand the ISM algorithm is able to perform quite well at low levels of *drop_rate*. More specifically at 5% *drop_rate* the graph line indicates that ISM locates 100% of BFS's documents. At 10% *drop_rate*, ISM is again able to take limited advantage of its profiling structure since at some point the recall rate is as high as 85%. ISM is not expected to be tolerant to large *drop_rates* (e.g.

20%) because in such case the information gathered by the profiling structure becomes obsolete before it gets the chance to be utilized. This leads ISM to several wrong query routing decisions which finally penalizes its recall rate.

7 Conclusions and Future Work

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 tradeoffs and experimentally evaluated four of the techniques that require no global knowledge.

Our work further focuses on the Intelligent Search Mechanism, which is an efficient, scalable and simple mechanism for improving the information retrieval problem in P2P networks. ISM uses the knowledge that each peer collects about its peers to improve the efficiency of the search. The scheme is fully distributed and scales well with the size of the network.

Our experimental results show that the performance of ISM improves over time because nodes learn more information about their neighbors as time elapses. ISM achieves therefore a better recall rate than its competitors, although its initial performance is similar to them. Second, we get almost as good recall rate as that of BFS while using a smaller number of messages. Thirdly ISM requires approximately the same Query Response Time (QRT) with its two competitors RBFS and >RES and far less QRT than BFS. Finally ISM scales well to large network topologies and replicated data sources. Our experimental results indicate that ISM is an attractive and promising search technique for information retrieval in Peer-to-Peer systems.

For future work we plan to probe our algorithms over new network topologies such as power-law and tree. We finally plan to make our middleware infrastructure publicly available and deploy it over a Wide Area Network of geographically dispersed hosts.

References

[1] Adamic L.A., Lukose R.M., Puniyani A.R. and Huberman B.A. "Search in power-law networks". Phys. Rev. E, 64 46135 (2001).

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

[3] Bawa M., Bayardo R.J., Rajagopalan S., Shekita E. "Make it Fresh, Make it Quick – Searching a Network of Personal Webservers". In Proc. of the 12th Int. World Wide Web Conference, WWW-2003, May 2003, Budapest, Hungary

[4] Callan J., Powell A.L., French J.C., and Connell M. "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.

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

[6] Crespo A., Garcia-Molina H. "Routing Indices For Peer-to-Peer Systems". *Proc. of Int. Conf. on Distributed Computing Systems*, Vienna, Austria, 2002.

[7] Cuenca-Acuna F.M. and Nguyen T.D. "Text-Based Content Search and Retrieval in ad hoc P2P Communities". International Workshop on Peer-to-Peer Computing, Springer-Verlag, May 2002

[8] French J.C., Powell A.L., Callan J., Viles C.L., Emmitt T., Prey K.J., and Mou Y. "Comparing the Performance of Database Selection Algorithms". *Proc. of ACM SIGIR Research and Development in Information Retrieval,* 1999.

[9] GMD-IPSI XQL Engine, `http://xml.darmstadt.gmd.de/xql/`

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

[11] Gravano L., and Garcia-Molina H. "Generalizing gloss to vector-space databases and broker hierarchies". In *Proceedings of the 21st VLDB Conference* (Zurich, Switchland, 1995).

[12] Groove Networks., `http://www.groove.net/`.

[13] Gao J., and Steenkiste P. "Design and Evaluation of a Distributed Scalable Content Discovery System." *IEEE Journal on Selected Areas in Communications*, Special Issue on Recent Advances in Service Overlay Networks, 22(1):54-66, January, 2004.

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

[15] Kalogeraki V., Gunopulos D., and Zeinalipour-Yazti D. "A Local Search Mechanism for Peer-to-Peer Networks". In *11th International Conference on Information and Knowledge Management (CIKM'2002)*, McLean, Virginia USA, November 4-9, 2002.

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

[17] Kubiatowicz J., Bindel D., Chen Y., Czerwinski S., Eaton P., Geels D., Gummadi R., Rhea S., Weatherspoon H., Weimer W., Wells C., and Zhao B., "OceanStore: An Architecture for Global-Scale Persistent Storage". *In Proc. of the 9th Int. Conf. on Architectural Support for Prog. Languages and Operating Systems (ASPLOS 2000)*, November 2000.

[18] Lu Z. and McKinley K.S. "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.

[19] Lucene, The Apache Jakarta Project. http://jakarta.apache.org/lucene/

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

[21] Melnik S., Raghavan S., Yang B., Garcia-Molina H. "Building a Distributed Full-Text Index for the Web". *10th World Wide Web Conference*, Hong Kong, 2001.

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

[23] Powell A.L., French J.C., Callan J., Connell M., Viles C.L. "The Impact of Database Selection on Distributed Searching". *Proc. of the 23rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 232–239, 2000.

[24] Ratnasamy S., Francis P., Handley M., Karp R., and Shenker S. "A Scalable Content-Addressable Network". *In Proc. ACM SIGCOMM 2001*, 2001.

[25] Rui Y., Huang T.S., and Chang S.F. "Image Retrieval: Current Techniques, Promising Directions and Open Issues" *Journal of Visual Communication and Image Representation*, Vol. 10, 39-62, March, 1999.

[26] Salton. G. "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer". Addison-Wesley, Reading, MA. 1989.

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

[28] Suel T., Mathur C., Wu J., Zhang J., Delis A., Kharrazi M., Long X., and Shanmugasundaram K. "ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval". 6th International Workshop on the Web and Databases (WebDB), June 2003.

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

[30] Tsoumakos D. and Roussopoulos N. "Adaptive Probabilistic Search for Peer-to-Peer Networks". *Proc. of the Third IEEE Int. Conf. on P2P Computing*, P2P2003, 2003.

[31] Wu Z., Meng W., Yu C. and Li Z. "Towards a Highly-Scalable and Effective Metasearch Engine". *10th World Wide Web Conference,* Hong Kong, 2001.

[32] Xu J. and Callan J. "Effective retrieval with distributed collections". *In Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 112–120, Melbourne, Australia, 1998.

[33] Yang B., and Garcia-Molina H. "Comparing hybrid peer-to-peer systems". *Proc. 27th Int. Conf. on Very Large Data Bases,* Rome, 2001.

[34] Yang B., and Garcia-Molina H. "Efficient Search in Peer-to-Peer Networks". *Proc. Int. Conf. on Distributed Computing Systems,* 2002.

[35] Zeinalipour-Yazti D. and Folias T., "Quantitative Analysis of the Gnutella Network Traffic". Dept. of Computer Science, University of California, Riverside, June 2000

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