A Local Search Mechanism for Peer-to-Peer Networks

Vana Kalogeraki
Hewlett-Packard Labs
Palo Alto, CA
vana@hpl.hp.com

Dimitrios Gunopulos
Computer Science and Eng.
Univ. of California, Riverside
dg@cs.ucr.edu

D. Zeinalipour-Yazti
Computer Science and Eng.
Univ. of California, Riverside
csyiazti@cs.ucr.edu

ABSTRACT

One important problem in peer-to-peer (P2P) networks is searching and retrieving the correct information. However, existing searching mechanisms in pure peer-to-peer networks are inefficient due to the decentralized nature of such networks. We propose two mechanisms for information retrieval in pure peer-to-peer networks. The first, the modified Breadth-First-Search (BFS) mechanism, is an extension of the current Gnuttela protocol, allows searching with keywords, and is designed to minimize the number of messages that are needed to search the network. The second, the Intelligent Search mechanism, uses the past behavior of the P2P network to further improve the scalability of the search procedure. In this algorithm, each peer autonomously decides which of its peers are most likely to answer a given query. The algorithm is entirely distributed, and therefore scales well with the size of the network. We implemented our mechanisms as middleware platforms. To show the advantages of our mechanisms we present experimental results using the middleware implementation.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search Process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed Systems*

General Terms

Algorithms, Management, Design

Keywords

Peer-to-Peer Netwroks, Distributed Information Retrieval

1. INTRODUCTION

Peer-to-peer (P2P) networks are increasingly becoming popular because they offer opportunities for real-time communication, ad-hoc collaboration and information sharing in a large-scale distributed environment. 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 the 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 single centralized component.

The P2P network creates a virtual point-to-multipoint network of many peers built on top of the physical infrastructure. The basic characteristic of the P2P network is that there is a group of nodes with the same type of interests connected over the same communication system. The P2P network is self-organized and self-administrative as the nodes autonomously discover their peers, and self-healing as the nodes automatically try to find new peers if their current peers are (temporarily or permanently) disconnected from the network. The peers are connected in an ad-hoc manner; there is no restriction on the number of peers in the network. A node connects to the network of peers, by establishing a relationship with at least one peer currently on the network. The connections are driven by the interests of the peers. For example, Clip2 [5] shows that the number of peers found in the Gnutella [10] network during a given weekday was 43,546 peers who shared 1,843,549 files. Peers can arrive and disappear dynamically, their arrival times are not known a priori. This contrasts with the web, where the majority of the web pages are statically allocated and change less frequently. The advantage is that the peers can join and leave the group dynamically without explicit knowledge of their memberships or need for group membership algorithms. Moreover, the topology of the resulting network could be arbitrary with various connectivity degrees between the peers, which makes searching and retrieving the correct information a difficult problem.

Current peer-to-peer applications are used for sharing resources, such as music files or videoclips, and consecutively support only rudimentary search mechanisms. Typically the user specifies the name of the file he/she is looking for and searches for the file using a brute force mechanism, as we described in the previous section. Clearly, to support higher level resources (such as, databases) for more sophisticated applications more efficient query mechanisms are required.

CIKM'02, November 4–9, 2002, McLean, Virginia, USA.

These mechanisms however should accommodate the distinctive characteristics that make the peer-to-peer computational model attractive: scalability, no centralized control, fluidity.

In this paper we consider the *information retrieval* problem in the P2P networks. Assume that each peer has a database (or collection) of documents, that is also made available to all peers connected in the network; this represents the knowledge of the peer. A node searches for information by sending `query` messages to its peers. We assume that the queries are collections of keywords. The querying peer is interested to find all the documents that contain all the 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 all the documents that correspond to the constraint. Once it receives responses from all the peers, the querying peer can afterwards decide which documents to download. Each document can be associated with a unique `documentId` (using for example a hash function) to uniquely identify the same documents from different peers.

Note that searching based on the file contents is not possible in most current P2P systems [8, 10, 19]. In those systems searching is done using file identifiers instead. This allows for efficient search and indexing techniques [8, 19, 24], however this limitation restricts the ability of users to share documents in a P2P network. Techniques for efficient searching P2P networks have been recently proposed in [6, 20, 21].

To solve the search problem, most current systems either rely to centralized control ([16]) or broadcast the query messages ([10]). The second approach (broadcasting the query) can be easily extended to solve the problem we consider here, searching with keywords: we have to modify the query message to include the query terms instead of the desired file identifier. This approach works best for pure peer-to-peer networks, were all functionality (including indexing and resource sharing) is decentralized. Such systems do not use peers with special functionality [20]. Gnutella is an example of such a system. In hybrid peer-to-peer networks [12, 16], one (or possibly more) peers have additional functionality in that they become partial indexes for the contents of other peers. Each peer, as it joins the network uploads a list of its files to the index server. Commercial information retrieval systems such as web search engines (*e.g.,* Google, Yahoo) are using this approach for indexing the web. They 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: first, for the index to be useful, it needs at least an inverted index of all the documents in all the peers it indexes. This means that the index node has to have sufficient resources, and have large bandwidth available to serve all queries. Although hardware performance and cost have improved, such centralized repositories are expensive to set up and hard to maintain. They require efficient human intelligence to build and keep the information they contain relevant and current. It is also difficult to capture all the information available, when this information can be updated in a daily basis; the current search engines perform periodic crawling which is typically a number of days old. Furthermore, they fail to capture information that is available in private databases since they do not have access to the private repositories maintained by the peers. For example, current search engines cannot show reservations for a given flight. Also, these systems assume that there is continuous availability (always on) of the peers and therefore the information. Yang *et al* [20] compared different hybrid P2P architectures that use centralized index nodes. The experimental results suggest that a distributed centralized index is the most efficient and scalable choice. They do not consider fully decentralized distributed systems however.

1.1 Our Contribution

In this paper we consider a fully distributed technique for addressing the information retrieval problem in pure P2P networks. We consider the pure peer-to-peer network (Gnutella network) and we propose new techniques that are more efficient than the Gnutella search.

The first, the modified Breadth-First-Search (BFS) mechanism, is an extension of the current Gnuttela protocol, allows searching with keywords, and minimizes the number of messages that are needed to search the network. The mechanism runs locally on each peer and automatically selects a random subset of the peer's neighbors to forward each query message.

The second technique, the Intelligent Search mechanism, improves on the first by having each peer in the network build a profile of its peers, and use the profile to find, for each query, which peers are likely to answer. It then forwards the query to those peers only. The profiles are aggregated and collected in real-time.

The mechanisms are distributed so we can avoid the problems with single points of failure that systems like Napster ([16]) have. At the same time, they minimize the number of messages and avoid the scalability problems of systems like Gnutella ([10]) that use broadcasts. The system has been implemented as a middleware platform that extends the Gnutella protocol. The reason is that we wanted to use existing infrastructure and investigate its behavior and its limitations.

2. INFORMATION RETRIEVAL IN A PURE PEER-TO-PEER (P2P) NETWORK

We consider a network of n nodes (peers), with average degree d (with $d << n$), that is, each peer is directly connected to around d other peers. For a given peer u, the peers of u, $N(u)$ are those nodes in the network that have a direct connection to u. Figure 1 shows an example of a peer-to-peer network. Each node in the figure represents a peer and an edge corresponds to a direct communication between the peers.

Each peer possesses and maintains a set of documents, which can be also made available to his peers. This set represents the knowledge of the peer. We assume that each document is stored in semi-structured form: for each document we have a set of attributes, such as, title, author, keywords, history, owner as well as text. Without loss of generality, we assume that the queries are in XML format. Such documents may be replicated between peers. Note though, that data possessed by the peers are not necessarily unique. They can be replicated at the edges of the network, but not all documents or files are fully replicated or replicated an equal number of times. Generally, those documents or files that

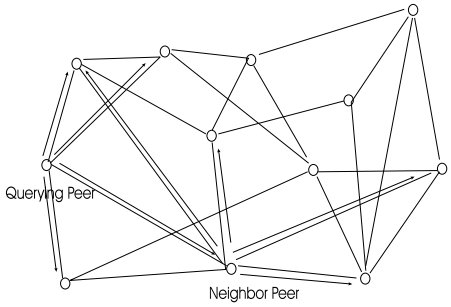

Figure 1: Searching in a peer-to-peer network: Each peer forwards the query to all its neighbors. In the figure we show the messages that the original peer sends, and the messages that one of the peers that receive the query sends.

are most requested by the users, will be available from more peers, and therefore, will be highly replicated. Allowing replication of the documents is important for two reasons. First, it distributes the workload across multiple peers and eliminates the burden from a single node. Second, it reduces the network traffic and minimizes the latency by data replication on the edges, closer to the location of the users.

Peers generate queries to search for data. The queries include attributes that characterize the documents. Examples of queries are "American Airlines" or "U.S. Banks". Each node sends messages only to its peers. If a node receives the same message more than once from different peers, it discards all the duplicated messages as it replies only to the first message it receives.

2.1 Extending the Gnutella search: searching using keywords in a fully distributed peer-to-peer network

The search protocol in the peer-to-peer network works as follows. A node issues search messages when it wants to search for data and information among its peers. The node generates a `Query` message with the search query and propagates the message to all of his peers. When a peer receives a `Query` request, it searches its local repository for relevant matches. If there is a match, the peer generates a `QueryHit` message to reply with the result. The `QueryHit` message along with the results, also includes the address of the peer and its network connectivity. For example, if the node receives the same data from more than one peers, it may choose to obtain the data from that peer with the best network connection. `QueryHit` messages are sent along the same path that carried the incoming `Query` messages. In addition, the peer will propagate the search to all of his own peers (Figure **??**).

The current search mechanisms have the disadvantage that they propagate all the queries across the network (including nodes with high latencies), therefore the network can easily become a bottleneck. To avoid flooding the network with messages, in the Gnutella protocol each search message is associated with a time_to_live (TTL) field that determines the maximum number of times the message will be propagated in the network. The TTL is decremented each time the message reaches a peer. When the TTL becomes 0, the message is no longer forwarded.

2.2 The Modified Random BFS Search mechanism

We first consider an intermediate search technique, Modified BFS Search. In this technique, each peer instead of forwarding a search message to all its peers, it randomly selects a subset of its peers to propagate the search request. The fraction of peers that are selected is a parameter to the mechanism. In our experiments we used a fraction of 0.5 (the peer propagated the request to half its peers, selected at random). The advantage of this technique is that, given a P2P network with a random graph topology, a peer can search the nodes of the graph more efficiently (that is, with a smaller number of messages overall) than the brute force Gnutella protocol. On the other hand, this algorithm is probabilistic,

3. INTELLIGENT SEARCH IN P2P NETWORKS

In this section we present a new mechanism for information retrieval in the P2P networks. The objective is to help the querying peer to find the most relevant answers to its query quickly and efficiency.

The 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 only propagates the query message to those peers.

3.1 The Intelligent Search Mechanism

The Intelligent Search mechanism for distributed information retrieval consists of 4 parts:

1. A search mechanism to send the query to the peers. This is the only mechanism used by a node to communicate with its peers. It is the same mechanism employed by the Gnutella protocol for communications between peers.

2. A profile mechanism, 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 the peer.

3. A peer ranking mechanism that a peer runs locally, using the profiles of its peers and the specific query, that ranks the peers in $N(u)$ in order to send the search query to the most likely peers.

4. A similarity function that a peer uses locally to find the similarity between different search queries.

3.1.1 The Search Mechanism

Assume that a peer initiates a search to find documents about a specific topic. Since he is initiating the search, we call him the *querying_peer*. The *querying_peer* generates a `Query` message that describes his request, finds which of his peers are most likely to provide an answer (using the profile mechanism and the peer ranking mechanism) and broadcasts the `Query` message to those peers only.

If a peer receives a `query` message we call him the *receiver_peer*. If the *receiver_peer* can provide an answer, it

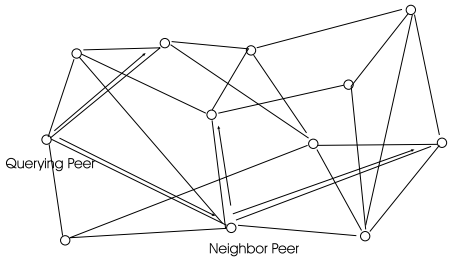

Figure 2: Intelligent searching in the peer-to-peer network: Each peer uses the knowledge it obtains from monitoring the past queries to propagate the query messages only to a subset of the peers.

returns the document to the requesting *querying_peer*, otherwise, it propagates the `Query` message only to those of his peers it considers most likely to provide the answer (Figure 2). To provide a termination condition so that the messages are not propagated indefinitely in the network, the *querying_peer* sets a bound on the depth of the recursion. When a reply is sent back to the *querying_peer*, the peers in the answer path (which is the same as the query path) record the query and the name of the peer that provided the answer in a $(query, peer)$ table. Each peer sets a bound on the number of pairs to be recorded, and uses a least recently used strategy to allow space for new queries.

3.1.2 Peer Profiles

To decide to which peers a query will be sent, a peer ranks all its peers with respect to the 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, that the specific peer that provided the answer for. (Although logically we consider each profile to be a distinct list of queries, in the implementation we use a single $Queries$ table with $(Query, Node)$ entries that keeps the most recent queries the peer has recorded).

The node accumulates the list of past queries by two different mechanisms. In the first mechanism the peer is continuously monitoring and recording the `Query` and the corresponding `QueryHit` messages it receives.

In the second, each peer, when replying to a `Query` message, broadcasts this information to its neighbor peers. This operation increases the accuracy of the system, at the expense of $O(d)$ extra messages (where d is the average degree of the network) per answering node.

The node keeps the list of queries in its local repository. For each node this list is incomplete, because each node can only record information about those queries that were routed through it. The node uses a size limit T that limits the number of queries in each profile. 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 neighbors only, the total size of the repository is $O(Td)$.

3.1.3 Peer Ranking

For each query it receives, the *receiver_peer* 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, the *receiver_peer* compares the query to previously seen queries and finds the most similar ones in the repository. To find the similarity between the queries, it uses the function provided (described below). The reason that we employ a Nearest Neighbor classification technique is that it is simple, and it has shown good accuracy in many different settings. It has also been shown that the Nearest Neighbor classification has asymptotic error rate at most twice the Bayes error rate, independent of the distance metric used [7].

Since it is likely that some peers will be associated with many similar queries, and others with some, we compute an aggregate similarity of a peer to a given query. Given the K most similar queries to q, the aggregate similarity of peer P_i to query q that peer P_k computes is:

$$Psim_{P_k}(P_i, q) = \sum_{q_j \text{ was answered by } P_i} Qsim(q_j, q)^\alpha$$

In this sum, q_j is one of the K most similar queries to q. This parameter limits the influence to the similarity to the most similar queries only. In addition, we use the parameter α, which allows us to add more weight to the most similar queries. For example, when α is very large, $Psim$ reduces to one-nearest neighbor. For $\alpha = 0$, $Psim$ reduces to K-nearest neighbor. If $\alpha = 1$, $Psim$ adds up the similarities of all queries that have been answered by the peer.

Consider the following example: Assume $K = 5$ and $\alpha = 1$. Peer P wants to send a query q to two of its peers. Let q_1, q_2, q_3, q_4, q_5 be the most similar queries to q, among the ones P has information about, with $Sim(q, q_1) = 0.8$, $Sim(q, q_2) = 0.6$, $Sim(q, q_3) = 0.5$, $Sim(q, q_4) = 0.4$, and $Sim(q, q_5) = 0.4$. 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 follows: $Psim_P(P_1, q) = 0.8^1 = 0.8$, $Psim_P(P_2, q) = 0.6^1 + 0.5^1 = 1.1$, and $Psim_P(P_3, q) = 0.4^1 + 0.3^1 = 0.7$. Therefore P chooses to send the query only to peers P_1 and P_2.

The *receiver_peer* then sends the query to the m peers (for a user defined constant $m < d$) that have the higher rank.

3.1.4 Distance Function: The cosine similarity

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

The *cosine similarity* metric has been used extensively in information retrieval, and we use this distance function in our setting. 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 is 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 of the two queries is simply the cosine of the angle between the two vectors.

3.1.5 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 issue is that the search will fail then to explore a significant part of the peer-to-peer network and may not discover many results. Consider for example the following case: Peer A sends a query to peers B, C, and D. If none of them has the answer, D considers A,C,D the best peers, C considers A,B,D the best peers and D considers A,B,C the best peers, then the query will never go to other peers. To solve this problem, we pick a small random subset of peers (1 peer in the experiments) and add it to the set of best peers for each query. As a result, even if the best peers form a cycle, with high probability the mechanism will explore a larger part of the network and will learn about the contents of additional peers.

4. PERFORMANCE OF THE PROPOSED TECHNIQUES

In this section we describe the characteristics of the proposed techniques, in comparison with the Gnutella protocol.

4.1 Performance of the modified BFS Search

We first consider the performance of the modified random BFS technique where each peer randomly selects a subset of its peers to propagate a request (that is, here a profile of its peers is not used). In a P2P network with a random graph topology, this mechanism searches the nodes of the graph more efficiently (that is, it sends fewer messages) than the Gnutella protocol.

Consider a random graph G with n nodes and e edges, that 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 modified 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 modified 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 a random subset of one's peers is more efficient in searching nodes in a P2P network with random graph topology than using the Gnutella protocol with respect to the total number of messages. 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 subgraphs are connected by one edge. If the node attached to that edge does not choose this edge, the other subgraph 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 ([3]) shows that even incomplete knowledge is sufficient to achieve good results. 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 the queries that peers reply to, rather than the documents in the actual replies. This is certainly very useful when very similar queries repeat. For a large number of queries, it also gives an approximation of a peer's collection of documents.

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. As a result, this mechanism can visit nodes that the Gnutella protocol would not visit. We explore this trade-off in the experimental evaluation of the technique.

In summary, the Intelligent Search mechanism for distributed information retrieval that we propose has the following characteristics:

1. The algorithm uses fewer messages compared to the standard Gnutella strategy, and scales better with respect to the size of the network (because it can search a larger network using the same number of messages)
2. The size of the profiles is proportional to the number of direct connections per peer. This is likely to remain small (constant) even for very large networks.
3. The scheme uses the combined knowledge about the peers, and adapts and modifies its behavior as each peer learns more information about its peers. On the other hand, peers do not have to export any information about their databases.

5. EXPERIMENTS

To illustrate the intelligent search mechanism we have built a decentralized online newspaper application. The newspaper is organized as a network of peers; each peer maintains a set of articles. The peer uses our information retrieval mechanism to efficiently search for articles in the newspaper. The user uses the graphical user interface to give keywords and the system retrieves all the articles that contain these keywords. The system has been used in an internal HP network and was implemented on top of the Gnutella P2P network in Java.

In our experimental study we compared the performance of the Intelligent Search mechanism with the Gnutella search, and with the modified BFS mechanism where each peer forwards the query to a random subset of half of its peers. Our evaluation metrics were the *recall* rate, that is, the fraction

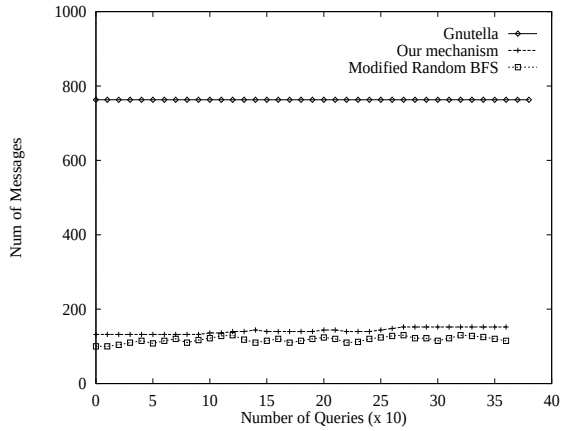

Figure 3: Number of messages for 400 queries in Gnutella, our mechanism and modified random BFS (connectionsPerPeer = 7 and TTL=4).

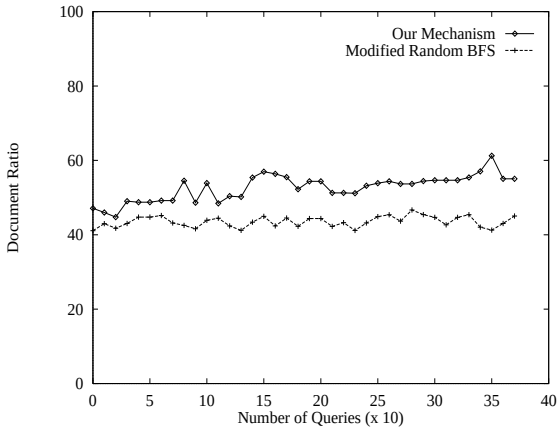

Figure 4: Ratio of documents that the Intelligent Search mechanism and the modified BFS mechanism find (connectionsPerPeer=7 and TTL=4).

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 for the same search.

We used a simulation testbed for the experiments, rather than our middleware implementation because it was easier to simulate large numbers of peers than running large numbers of Gnutella clients on a small number of machines. Nevertheless, our simulation computes the metrics we use exactly. We used the document collection from the REUTERS dataset [18] as the set of documents that are available to the various peers. We partitioned the dataset into documents that are relevant to each country (using the metadata information available in the collection). There were 84 different countries with at least 10 documents, and the total number of documents for these 84 countries was 22531 (some documents belonged to more than one country. We created a network of 100 peers. The topology was a random graph with average degree 7 (random graphs with more than $\log n$ average degree are almost certainly connected). Each peer was assigned collections from three countries (with large country collections split into groups of 50 documents) leading to an average of 135 documents per peer. Our objective was to simulate the situation where each peer has sets of documents about specific topics.

For brevity we present one set of experiments, where we ran 400 user requests arriving "sequentially" in the system. At each time interval, a peer submits a query in the network. The queries are generated automatically and include query sets (keywords) from the documents of the dataset. To generate the query sets we used a set of 100 keywords. All queries were generated by the same peer. The TTL for each query was set to 4 and 5.

In the first experiment we measured the number of messages per query that the three techniques generate when the time_to_live (TTL) field of the request messages is set to 4. In the Intelligent Search mechanism, each peer in the query path determines the 3 best peers to which to send the query requests. In addition, it sends the query request to a randomly chosen peer. Figure 3 indicates that the brute force algorithm always sends 763 messages, and the modified random BFS algorithm sends 131 messages on average, regardless of whether they are likely to have responses to the requests. In the figures, the values shown are the averages of 10 consecutive requests. The figure shows that the number of messages for our mechanisms is significantly smaller than the Gnutella algorithm.

Figure 4 shows that in this situation our mechanism discovers over half of the documents found by the brute force algorithm, and always more than what the modified BFS mechanism finds.

It is also important to note that the recall ratio improves over time, as peer profiles are learned, unlike the modified BFS mechanism.

In the second experiment we show that we can improve the results substantially by increasing the TTL parameter. Figure 5 shows that by increasing the value of the time_to_live field of the search requests (TTL = 5) the Intelligent search mechanism discovers almost the same documents that the Gnutella search finds for TTL = 4. Our experimental results (Figures 5, 6) show that our mechanism achieves 90% recall rate while using 35% of the number of messages of the Gnutella search. Again, the recall rate increases as the number of queries increases over time. The results of the modified BFS mechanism 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. This justifies our hypothesis that in the pure P2P network, a large number of peers receive unnecessary messages.

The number of messages in the Intelligent Search mechanism increases over time. The reason for this is that, as the nodes accumulate more knowledge about their peers, peers that provided answers in the past are still queried in subsequent queries. As more peers become likely to be queried, they themselves continue to explore the network by propagating the requests to their peers.

In conclusion, our preliminary experimental results show the following. First, the performance of the Intelligent Search mechanism improves over time as the peers learn more in-

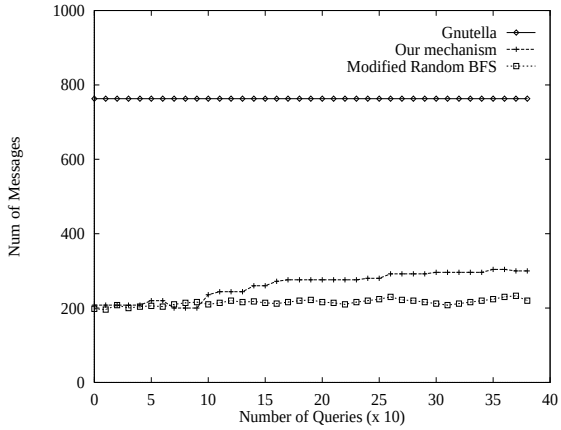

Figure 5: Number of messages for 400 queries in the Gnutella (TTL=4), our mechanism and the modified random BFS (TTL=5).

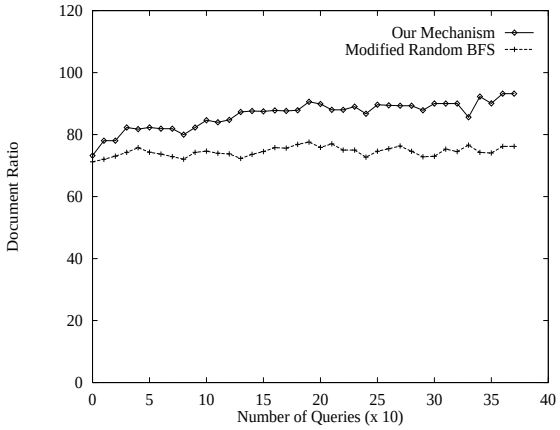

Figure 6: Ratio of documents that our distributed information retrieval mechanism and the modified random BFS find for TTL=5, and Gnutella_TTL=4).

formation about their peers and therefore becomes better than the modified BFS algorithm, although the initial performance was similar. Second, we get almost as good recall rate as that of Gnutella using a much smaller number of messages.

6. RELATED WORK

Directed DFS Search and Freenet: In [21] a technique for searching a P2P network is proposed where each node propagates the query to one of its peers based on some aggregated statistics (including which peer was the last to answer a query, a random peer). If the peer does not reply, the requesting peer selects a new peer. The technique is similar to the Intelligent Search Mechanism we propose, but uses simpler information about the peers, and is optimized to find k documents efficiently (for a fixed k) rather than finding all documents so a Depth-First-Search approach rather than a Breadth-First-Search approach is used.

Freenet [8] also uses an intelligent DFS search mechanism to find files in the P2P network. The approach, which is based on keeping in local caches pairs of document keys and the peer that contain the document. This technique was further improved by [24]. Our approach is more general because Freenet allows only searching with file identifiers, instead of the file contents. In addition, we use a Breadth-First-Search approach, where many messages are propagated in the network concurrently, rather than a Depth-First-Search approach, where each node sends a message to one peer and waits for a reply before forwarding it to another peer. 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, that happen to be distributed in many peers.

Using Local Indices: [6] 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 (and has to send updates every time an update happens), thus it is complementary to our approach where the profiles get updated when a peer answers a query.

Distributed Information Retrieval: In distributed information retrieval, the main problem is, assuming that we want submit an information retrieval query to a subset of the databases available only, 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, 9, 11, 23, 22]. Recent work [14, 17] shows that good performance can be achieved in this setting, 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. Nevertheless, the metrics we use in our technique are similar to the ones used in such techniques.

Collaborative Filtering: Our technique has similarities with collaboration filtering approaches [2]. In particular a number of approaches have been proposed for collaboration filtering algorithms that work over the web [4, 13]. In general these techniques assume that documents are public and known to all participants, and therefore are difficult to use in the peer-to-peer environment.

Exploiting the peer-to-peer network structure to improve the search performance: Recent approaches to improve search performance in peer-to-peer networks use a consistent hashing scheme to distribute the objects in the peer-to-peer network, so that an efficient location algorithm can be implemented [19]. This approach is not possible here because we are searching using keywords, rather than using a unique identifier of an object (such as a name). As a result, we want to find all documents that contain the keywords because they may contain different information. Another approach uses a gossiping protocol to propagate the searches in the peer-to-peer network. The problem with this

approach is that the rate of gossiping is slow and therefore the messages are not propagated fast to the peers. 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: These include commercial information retrieval systems such as web search engines (e.g., Google, Yahoo) that are centralized processes, as well as P2P models that provide centralized indexes [16, 20, 15]. 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, can index databases and are less costly.

7. CONCLUSIONS AND FUTURE WORK

Peer-to-peer networks offer several advantages such as simplicity of use, robustness and scalability. In this paper we propose and implement a middleware platform that extends the Gnutella protocol to support information retrieval in a peer-to-peer network. We present the Intelligent Search mechanism, that 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.

For future work we plan to address the problem of actively changing the topology of the P2P network in order to improve the performance of the search. We also plan to consider the problem of security and privacy, and how such considerations affect the quality of search in P2P networks. We plan to do deploy the mechanism and experiment on larger P2P networks. Finally we plan to consider the problem of efficiently maintaining the profiles of the peers, combining the information from different queries, and updating the profiles when the document collections of the peers change.

8. REFERENCES

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

[2] J. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Proc. of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98).* San Francisco, CA, pp. 43-52

[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] S. Chakrabarti, S. Srivastava, M. Subramanyam, M. Tiwari. Using Memex to archive and mine community Web browsing experience. *Computer Networks 33(1-6)*: pp. 669-684 (2000)

[5] Clip2. www.clip2.net

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

[7] R.O. Duda, and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, Inc., 1973.

[8] Freenet, http://freenet.sourceforge.net

[9] 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 Research and Development in Information Retrieval,* 1999.

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

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

[12] Kazaa, http://www.kazaa.com

[13] H. Lieberman, N. Van Dyke, and A. Vivacqua. Let's Browse: A Collaborative Web Browsing Agent. MIT Media Lab Tech Report.

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

[16] Napster, http://www.napster.com.

[17] Powell, A. L., French, J. C., Callan, J., Connell, M., and Viles, C. L. The Impact of Database Selection on Distributed Searching. In *SIGIR 2000: Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval,* pages 232-239, 2000.

[18] REUTERS-21578 dataset. http://www.research.att.com/ lewis

[19] 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 2001.*

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

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

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

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

[24] H. Zhang, A. Goel, R. Govindan. Using the Small-World Model to Improve Freenet Performance. *Proc. of IEEE Infocom,* 2002.