Content-Based Retrieval in Hybrid Peer-to-Peer Networks

Jie Lu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jielu@cs.cmu.edu

Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
callan@cs.cmu.edu

ABSTRACT

Hybrid peer-to-peer architectures use special nodes to provide directory services for regions of the network ("regional directory services"). Hybrid peer-to-peer architectures are a potentially powerful model for developing large-scale networks of complex digital libraries, but peer-to-peer networks have so far tended to use very simple methods of resource selection and document retrieval. In this paper, we study the application of content-based resource selection and document retrieval to hybrid peer-to-peer networks. The directory nodes that provide regional directory services construct and use the content models of neighboring nodes to determine how to route query messages through the network. The leaf nodes that provide information use content-based retrieval to decide which documents to retrieve for queries. The experimental results demonstrate that using content-based retrieval in hybrid peer-to-peer networks is both more accurate and more efficient for some digital library environments than more common alternatives such as Gnutella 0.6.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Retrieval models, Search process, Selection process.

General Terms

Algorithms, Performance, Experimentation, Design

Keywords

Peer-to-peer, Hybrid, Retrieval, Search, Content-based

1. INTRODUCTION

Peer-to-peer (P2P) computing is a relatively new approach to federated search of large networks of digital libraries. In P2P networks the nodes can send and receive information in a way that makes them both servers and clients. *Pure* P2P architectures are completely decentralized; each node can issue requests, respond to the requests that it can satisfy, or route requests to other nodes. *Hybrid* P2P architectures include two types of nodes. There are *leaf nodes* which provide information as well as post requests ("queries"). Leaf nodes can be used to model an individual with an information need or an information resource (e.g., a digital library). There are also *directory nodes* which don't have contents of their own but which provide regionally centralized directory services to the network to improve the routing of information requests. Directory nodes are also called "ultrapeers", "hubs", or "supernodes" in the research literature. Each directory node provides directory services for portions of the network and directory nodes work in a cooperative manner to cover the whole network.

Early P2P architectures provided federated search by either relying on a single centralized directory service or employing the *flooding* technique in completely decentralized manner (a node broadcasting query messages to all of its neighbors) to decide how to route query messages. The former approach suffers from a single point of failure and has limited scalability, while the latter approach is less efficient and may overload the network. Hybrid P2P architectures that use multiple decentralized directory services were developed to solve these problems. For example, the Gnutella 0.6 protocol adopts the hybrid P2P architecture [10] to overcome the weaknesses of the pure P2P architecture in the Gnutella 0.4 protocol [9].

Although research on information systems using P2P architectures is very active recently, most recent research focuses on improving the efficiency, robustness, and load-balancing of distributed information storage or file-sharing systems [1, 6, 21, 22]. Resource selection and document retrieval in P2P networks have so far mostly been limited to simple *name-based* methods: Matches between query terms and document names or identifiers are used to determine how to route query messages and which documents to be retrieved. These techniques may be sufficient for networks of small digital libraries that use well-known naming conventions and provide simple services, as is common in music file-sharing applications. Extending peer-to-peer architectures to networks of large and complex digital libraries that provide more sophisticated services requires *content-based* retrieval.

In this paper, we explore content-based retrieval in P2P networks that adopt hybrid P2P architectures ("hybrid P2P networks"). In particular, we apply content-based resource selection and document retrieval algorithms to hybrid P2P networks. Directory nodes model the contents of neighboring nodes based on their resource descriptions or responses to past queries, and use these models to route query messages ("resource selection"). Leaf nodes use a probabilistic information retrieval algorithm to determine which documents to retrieve for queries ("document retrieval"). In this paper, we show that using content-based retrieval in hybrid P2P networks can greatly reduce the average number of query messages per query, and increase Precision while causing little degradation in Recall.

The following section describes related work. Section 3 presents in more detail name-based and content-based retrieval in hybrid P2P networks. Sections 4 and 5 discuss our data resources and evaluation methodologies. Experimental settings and results are presented in Section 6 and Section 7. Section 8 concludes.

CIKM'03, November 3-8, 2003, New Orleans, Louisiana, USA.

2. RELATED WORK

Many consider Napster to be the first peer-to-peer search system; it used centralized indexes for routing queries. Centralization ensured relatively consistent coverage and speed, but also a single point of failure; few systems have used that model since. Gnutella was among the first "pure" peer-to-peer architectures. Gnutella 0.4 uses the flooding technique for query routing, which is robust, but also inefficient and not very scalable [9]. The Gnutella 0.6 protocol [10] adopts the hybrid P2P architecture to overcome the weaknesses of Gnutella 0.4. For systems using the Gnutella 0.6 [16] and similar protocols [15], leaf nodes generate descriptions of the identifiers of their own documents, and directory nodes use these descriptions to select leaf nodes for query routing. However, only simple name-based query term matching is used for resource selection and document retrieval. Directory nodes still use the flooding technique to route query messages to other directory nodes.

JXTA Search is a distributed search system designed for P2P networks [23]. It also uses the hybrid P2P architecture. Directory nodes ("hubs") in JXTA Search are dedicated nodes that efficiently route queries from the leaf nodes that post queries ("consumers") to the leaf nodes that provide information ("information providers"). The information needed by a directory node to route a query is the descriptions that the information providers register with the directory node. The descriptions describe information providers themselves as well as the kinds of queries they can handle. In theory directory nodes can cooperate with each other and queries can also be routed to other directory nodes. However, the cooperation mechanism is yet to be defined.

The problem of content-based retrieval in pure P2P architectures is addressed by [5]. Each node tries to collect compact summaries of all other nodes' inverted indexes. A node uses a TF.IDF algorithm to decide which nodes to contact for information requests based on the summaries it collects. Because no special resources are dedicated to support directory services in pure P2P architectures, it is somewhat inefficient for each node to collect and store information about the contents of all other nodes, especially in dynamic P2P networks.

There have been attempts to extend and improve the Gnutella 0.4 protocol to enable efficient search and retrieval in pure P2P architectures. For example, [14] proposes that a node use the past query responses from its neighbors to help it decide how to route new query messages. Each node in the network builds a run-time profile for each of its neighbors. The profile keeps the most recent past queries the neighboring node answered. These profiles are used to select those nodes that are most likely to have documents relevant to a new query by comparing the query with past queries in the profiles. This approach improves local query routing, but still makes it difficult to reliably find relevant information in distant parts of the network.

The problem of routing queries in peer-to-peer networks is essentially a problem of resource selection. There has been considerable prior research on resource selection for distributed information retrieval, for example CORI [3], gGlOSS [7, 8] and Kullback-Leibler (K-L) divergence-based algorithms [24]. Although algorithms for resource selection in distributed IR were developed for the case of a single directory service only, they might be extended to the case of multiple directory services.

Resource selection algorithms need a resource description of each digital library in order to decide which libraries are more likely to satisfy the user's information need given the query. Maintaining resource descriptions for many digital libraries is time and space consuming, which may be a more serious problem in P2P networks. Pruning techniques have been explored to reduce the storage costs associated with content-based retrieval [4, 19]. Index or content pruning can reduce storage costs significantly while causing only minor losses in retrieval accuracy.

Although real applications of P2P file-sharing systems have reached network sizes of hundreds of thousands of nodes sharing millions of documents, the attempts to evaluate the performance of retrieval activities in P2P networks have had far smaller scales. For example, [5] evaluated content-based retrieval on a P2P network of no more than 400 nodes. The total number of documents and queries used in the experiments were 91,775 and 407 respectively. The network size used by [14] for evaluation was 100 nodes with a total of 22,531 documents and 400 queries. Evaluation of retrieval performance in P2P networks with more realistic settings requires a testbed of larger scale.

The prior research suggests that i) hybrid P2P networks are robust and (sometimes) scalable, ii) hybrid P2P architectures can better support the complicated functionality required for content-based retrieval and efficient query routing, iii) existing solutions in distributed IR could be adapted to content-based retrieval in hybrid P2P networks, iv) pruning techniques could be used to reduce the storage costs at directory services, and v) a larger testbed with more realistic settings is needed to evaluate content-based retrieval in large-scale P2P networks.

3. RETRIEVAL IN HYBRID P2P NETWORKS

Our reason for studying content-based retrieval in hybrid P2P networks is to enhance the functionality of regional directory services (directory nodes) to improve efficiency and accuracy. To improve the efficiency of query routing a node must selectively route query messages to a subset of its neighboring nodes instead of using the flooding technique. In order to do that without sacrificing retrieval accuracy the neighboring nodes selected must be those that are most likely to respond to the query. This requires either full or partial knowledge of the contents that each node can provide (for leaf node) or cover (for directory node). The knowledge of the contents could be represented by language models, which could be obtained either by asking the neighboring nodes to provide them directly or by learning from their responses to past queries. Different approaches could be used to develop different resource selection algorithms in hybrid P2P networks.

Although hybrid P2P architectures don't prohibit leaf nodes from conducting resource selection, resource selection usually occurs only at directory nodes because they consume additional resources such as time and space. In this paper, we only explore the resource selection conducted by directory nodes.

In this section we first introduce the name-based retrieval used in the Gnutella 0.6 protocol and how to extend it to support content-based retrieval. Then the resource selection and document retrieval algorithms for content-based retrieval in hybrid P2P networks are presented.

Most of the algorithms presented here require the *query matching rule*, which defines the number of query terms that need to be

matched for a query. Depending on the algorithms, query terms are matched against document names, collection vocabularies, or document contents, which are described in more detail below.

3.1 Name-Based Retrieval

For the name-based retrieval used in the Gnutella 0.6 protocol, both resource selection and document retrieval are name-based, in which query terms are matched against document names.

3.1.1 Name-Based Resource Selection (NBRS)

Resource selection in the Gnutella 0.6 protocol is limited to directory nodes selecting neighboring leaf nodes in order to route query messages. When a leaf node issues a query, it routes the query to the directory node it connects to. The directory node looks up the query terms in the hash tables sent by neighboring leaf nodes for matches and only routes the query to those leaf nodes that satisfy the query matching rule. The hash table of a leaf node is generated by hashing all the individual terms from the names of its documents. The directory node broadcasts the query to all of its neighboring directory nodes ("flooding").

3.1.2 Name-Based Document Retrieval (NBDR)

Document retrieval at a leaf node is also name-based. Given a query, a node generates a query hit message to include those of its documents whose names satisfy the query matching rule.

3.2 Match-Based Retrieval

Match-based retrieval is a simple extension of Gnutella 0.6's name-based retrieval in order to support content-based retrieval. We implemented it as a comparison to the content-based retrieval presented in Section 3.3.

3.2.1 Match-Based Leaf Node Selection (MBLNS)

To support content-based retrieval, if we still use the simple resource selection algorithm based on query term matching, the contents instead of the names of the documents need to be checked for matching. The set of all the documents a leaf node provides in the network is referred to as the node's *collection* in this paper. The *vocabulary* of a leaf node is the set of all the unique terms that occur in this node's collection. For Match-Based Leaf Node Selection, each directory node uses the vocabularies of its neighboring leaf nodes to check query terms and route query messages to those leaf nodes that satisfy the query matching rule.

3.2.2 Random Match-Based Leaf Node Selection (RMBLNS)

If a directory node connects too many leaf nodes, it would be inefficient for the directory node to route query messages to all of the leaf nodes that satisfy the query matching rule because there might be a lot of them. To improve efficiency, the directory node can randomly select up to a threshold some neighboring nodes that satisfy the query matching rule and route query messages to them. Because the directory node using match-based resource selection cannot estimate which leaf nodes are more likely to have relevant documents for the given query, this approach is expected to degrade the retrieval accuracy.

3.3 Content-Based Retrieval

Content-based resource selection and document retrieval algorithms use the content models of nodes to select nodes and retrieve documents that are most likely to satisfy the user's information need. We use resource selection and document retrieval algorithms based on statistical language models and Kullback-Leibler (K-L) divergence [20, 24], as discussed below.

3.3.1 Content-Based Leaf Node Selection (CBLNS)

Resource selection based on Boolean term matching may lead to too few or too many query messages. If we also consider term frequency information, then it is possible to use advanced algorithms to calculate the likelihood that a node will satisfy the user's information need and select nodes based on their ranked likelihood scores. In this paper, we adapt a K-L divergence-based resource selection method [24] to leaf node selection. The likelihood that a leaf node will satisfy the user's information need given a query is calculated by the negative of the K-L divergence between query Q and the collection of documents C from the leaf node as:

$$KL(Q,C) = -\sum_{q \in Q} \mathrm{P}(q \mid Q) \log \frac{\mathrm{P}(q \mid Q)}{\lambda \mathrm{P}(q \mid C) + (1-\lambda)\mathrm{P}(q \mid G)} \qquad (1)$$

where $\mathrm{P}(q \mid Q)$ is the query language model, $\mathrm{P}(q \mid C)$ is the collection language model, and $\mathrm{P}(q \mid G)$ is the background language model used for smoothing. Because $\mathrm{P}(q \mid Q)$ is independent of any node's collection C, the ranking of leaf nodes based on $KL(Q, C)$ scores is equivalent to ranking based on $S(Q,C)$ scores calculated as:

$$S(Q,C) = \sum_{q \in Q} \log\{\lambda \mathrm{P}(q \mid C) + (1-\lambda)\mathrm{P}(q \mid G)\} \qquad (2)$$

A directory node uses $S(Q,C)$ scores to rank its neighboring leaf nodes and routes query messages to top-ranked nodes up to a threshold number.

The only information needed to construct the resource descriptions used by Equation 2 is term and term frequency information of a node's collection. The *full resource description* of a leaf node is thus defined as all the unique terms that occur in this node's collection and the corresponding collection term frequencies. For selection based on the full resource descriptions of leaf nodes, the collection language model $\mathrm{P}(q \mid C)$ is calculated using the full resource description. Content-Based Leaf Node Selection using full resource descriptions is referred to as CBLNS-F in our experiments.

When P2P networks include complex resources such as digital libraries with large collections, maintaining the full resource descriptions of leaf nodes at a directory node is space inefficient. Zipf's Law states that a term's frequency is roughly proportional to the reverse of its position in the list of all the terms ranked by term frequencies in the collection. From Zipf's Law it is not difficult to infer that the number of terms that occur only once in the collection is approximately half the size of the collection's vocabulary. Thus if we discard those single-occurrence terms, we could reduce the sizes of resource descriptions by half. The *pruned resource description* of a leaf node is defined here as all the unique terms that occur more than once in the node's collection, and the corresponding collection term frequencies. For selection based on the pruned resource descriptions of leaf nodes, the collection language model $\mathrm{P}(q \mid C)$ is calculated using the pruned resource description. Content-Based Leaf Node Selection using pruned resource descriptions is referred to as CBLNS-P in our experiments.

Table 4.1. Summary statistics for the test data.

	min	avg	max
Number of documents for a collection	8	568	26,505
Number of collections for a cluster	10	376	1,008
Number of clusters a collection belongs to	1	4	12

3.3.2 Content-Based Directory Node Selection (CBDNS)

Similar to Content-Based Leaf Node Selection, a directory node could also ask its neighboring directory nodes to provide the resource descriptions for the contents of the portions of the network that each directory node covers in order to conduct directory node selection for query routing. The resource description of a directory node is the union of all the resource descriptions of the leaf nodes it connects to. If a directory node connects to many leaf nodes, then its resource description is very large, even after pruning single-occurrence terms. Instead of obtaining the resource descriptions directly from its neighboring directory nodes, the directory node learns a content model for each neighboring directory node by recording the query terms of past queries that the neighboring node has responded to. The content model learned is restricted to a small size. When the model size reaches its limit, it deletes a third of the terms in the model in ascending order of their frequencies to make room for new terms.

Initially, directory nodes have empty content models for neighboring directory nodes, so queries are routed using the flooding technique. Directory nodes learn content models by observing which queries each neighboring directory node responds to. Given a new query, the directory node computes scores for its neighboring directory nodes using Equation 2, for the neighbors whose models match at least one query term. The query is routed to the top-ranked directory nodes, up to a threshold number; if too few directory nodes are ranked, the flooding technique fills out the set. In addition, the directory node randomly selects 1 more neighboring directory node. This random perturbation is used to make the algorithm more robust.

3.3.3 Content-Based Document Retrieval (CBDR)

When a leaf node receives a query message, it uses a K-L divergence retrieval algorithm [20] to rank the documents of its collection and generate a query hit message, returning information about the 50 top-ranked documents that satisfy the query matching rule. The query terms are matched against the contents of the documents in this node's collection.

4. TESTBED

The behavior of resource selection and document retrieval algorithms in hybrid P2P networks was evaluated using a simulator. The simulator was a version of the JavaSim network simulator [13] extended by colleagues to simulate simple peer-to-peer networks [2], and further extended by us to support name-based, match-based, and content-based retrieval in hybrid P2P networks.

There has been no standard data for evaluating the performance of content-based retrieval in P2P networks, so we developed one based on the TREC WT10g web test collection, which is a 10 gigabyte, 1.69 million document subset of the VLC2 collection [11]. We briefly describe below how we used the WT10g collection to generate the contents, topology, and queries for simulating retrieval in hybrid P2P networks. Table 4.1 summarizes some statistics for the testbed.

4.1 Contents

The WT10g data was divided into 11,485 collections based on document URLs. 2,500 collections were randomly selected for use in the experiments described in this paper. The total number of documents in these 2,500 collections was 1,421,088. The HTML title fields of the documents were used as document names during tests of name-based retrieval algorithms. Each of the 2,500 collections defined a leaf node in a hybrid P2P network [18].

4.2 Topology

Directory nodes can use many criteria to determine which leaf nodes to include in a directory; for example, a directory might cover a specific geographic region or type of content. Grouping *documents* that have similar content improves resource selection accuracy and reduces the number of resources searched [3, 24], so the research reported here focused on *directory nodes* that cover specific types of content.

A similarity-based soft clustering algorithm [17] was used to organize leaf nodes by topic; twenty-five clusters were created, and leaf nodes that covered multiple topics could appear in multiple clusters. Each cluster defined the contents of a single directory node in the hybrid P2P network (i.e., which leaf nodes were served by the directory node).

The connections between directory nodes were generated randomly. Each directory node could have no more than 7 and no less than 1 directory node neighbors. A directory node had on average 4 directory node neighbors.

4.3 Queries

The number of queries provided by NIST for the TREC WT10g web test collection is far from enough to be used in studies on content-based retrieval in P2P networks. Although web logs from search engines could provide a large amount of queries, there is no way to guarantee that there are relevant documents in the WT10g collection for these queries. One way to generate a large amount of queries in a controlled manner is to extract key terms from the documents in the WT10g collection and use them as queries. Prior research shows that 85% of the queries posted at web search engines have 3 or less query terms [12], so for most documents, we should only extract a few key terms as queries. We tried a variety of approaches to rank and extract key terms from the documents. The best approach (judged manually) was to use the combination of the unigram document language model with linear interpolation, the bigram document language model, and some heuristic rules to rank document terms or term pairs for use as query terms. We describe this approach in more detail below.

The unigram document language model with linear interpolation considers the probability $P_{emp}(t \mid d)$ that a term occurs in a document as a linear interpolation of the probability $P_{core}(t \mid d)$ that the term is generated by the unigram document language model, and the probability $P(t \mid background)$ that the term is generated by the background (general English) model:

$$P_{emp}(t \mid d) = \lambda P_{core}(t \mid d) + (1-\lambda) P(t \mid background) \quad (3)$$

where λ is the smoothing weight in this mixture model.

Table 4.2. Randomly selected sample queries of different lengths.

Length	Terms
1	sdtech
2	malignant hyperthermia
3	cardiac surgery; anesthesia
4	trade remedy; nafta law
5	drug drive collision police investigate
6	quarter company revenue increase sybase cash

$P_{emp}(t \mid d)$ is calculated by maximum likelihood estimation with simple Laplacian smoothing. $P(t \mid background)$ is calculated based on the term frequency of term t in the entire collection of WT10g. $P_{core}(t \mid d)$ is the probability we use to evaluate how important a term is to the document. It is calculated based on an algorithm described in [25].

The bigram document language model approach uses $P(t_1, t_2 \mid d)$ to measure the importance of a "phrase" (two terms occurring next to each other in the document) to the document. It is calculated as a mixture of maximum likelihood estimates:

$$P(t_1, t_2 \mid d) = 0.5P(t_1 \mid d)\frac{c(t_1, t_2 \mid d)}{c(t_1 \mid d)} + 0.5P(t_2 \mid d)\frac{c(t_1, t_2 \mid d)}{c(t_2 \mid d)} \quad (4)$$

where $c(\bullet)$ denotes count, $P(t_1 \mid d)$ and $P(t_2 \mid d)$ are smoothed maximum likelihood estimates of the probabilities that document d generates terms t_1 and t_2 respectively, and $c(t_1, t_2 \mid d) / c(t_1 \mid d)$ and $c(t_1, t_2 \mid d) / c(t_2 \mid d)$ are un-smoothed empirical estimates of $P(t_2 \mid t_1, d)$ and $P(t_1 \mid t_2, d)$ respectively.

The way to combine the unigram document language model and the bigram document language model is that any two top-ranked terms that appear to be a "phrase" in the top-ranked "phrases" of the document are replaced by this "phrase".

Other heuristic rules include:

- The k-stem stemmer is used because the stemmed terms it generates are easier for people to understand, and because stemming a term more than once does not change it further;
- Single-character terms are eliminated because it is rare to have single-character query terms;
- Terms that begin with numbers are eliminated;
- Terms that belong to a set of web-specific stopwords such as "please", "thank", "previous" and "next" are eliminated; and
- Terms occurring in the title of the document are emphasized by a weight of 1.5.

No query had more than 6 terms. Most queries had 2-3 terms. Table 4.2 shows randomly selected examples of the automatically-generated queries for different query lengths [18]. 15,000 queries were randomly selected from the automatically-generated queries to be used in our experiments.

For each query, a leaf node was randomly chosen to issue the query on the condition that the node didn't have the document used to generate that query.

5. EVALUATION METHODOLOGY

For content-based retrieval in P2P networks, both the retrieval accuracy and the efficiency of query routing are very important, so the performance of different resource selection and document retrieval algorithms in hybrid P2P networks is measured by the retrieval accuracy and the efficiency of query routing.

5.1 Measuring Retrieval Accuracy

It is expensive to obtain relevance judgments for so many automatically-generated queries. Instead, we used the retrieval results from a single large collection as the baseline, and measured how well the P2P network could reproduce this baseline. The single large collection was the subset of the WT10g used to define leaf node contents in the P2P network (Section 4.1), and agreement was measured over the top 50 documents retrieved for each query. Although this methodology is not ideal, it is not unreasonable because distributed retrieval systems are not yet better than the "single collection" baseline (e.g., [3]).

Accuracy was measured with modified forms of set-based Recall and Precision, defined as follows:

$$\text{Recall} = \frac{|r|}{|A|} \quad (5) \qquad \text{Precision} = \frac{|r|}{|R|} \quad (6)$$

where R is the set of the documents returned by retrieval in the P2P network, A is the set of (up to 50) top-ranked documents returned by retrieval using the single WT10g-subset collection, and r is the intersection of R and A. $|\bullet|$ denotes the size of the set.

Our use of set-based forms of Recall and Precision focuses attention on how well content-based retrieval in hybrid P2P networks returns the "right" documents for a query, and ignores the problem of merging the results from different information providers to create an integrated ranked list of documents ("result merging"). Result-merging is an important problem, but it was outside the scope of the research reported in this paper.

The harmonic mean of Recall and Precision (F-Score), computed as shown below, was also used to measure retrieval accuracy.

$$\text{F-Score} = 2/\ (\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}) \quad (7)$$

5.2 Measuring Query Routing Efficiency

The efficiency of query routing was measured by the average number of query messages routed for each query in the network.

6. EXPERIMENTAL SETTINGS

A series of experiments was conducted to study several combinations of resource selection and document retrieval algorithms for retrieval in hybrid P2P networks (Table 6.1).

As in the Gnutella protocol, each message had a time-to-live (TTL) field that determined the maximum number of times it could be relayed in the network. The TTL was decreased by 1 each time the message was routed to a node. When the TTL reached 0, the message was no longer routed. The initial TTL was set to 4 for query messages routed to directory nodes. With 25 directory nodes and 4 neighboring directory nodes on average for each directory node, messages with a TTL of 4 could reach almost all the directory nodes in the network using the flooding technique, which provided good coverage but could be very inefficient. For query messages routed to leaf nodes by directory nodes, the TTL was set to 1 because leaf nodes were not supposed to further route query messages.

Table 6.1. Combinations of resource selection and document retrieval algorithms tested in the experiments.

Resource Selection and Document Retrieval Algorithms	Directory Selects Leaf	Directory Selects Directory	Leaf Selects Leaf	Leaf Selects Directory	Leaf Retrieves Document
NBRS + NBDR	NBRS	Flooding	N/A	Flooding	NBDR
MBLNS + CBDR	MBLNS	Flooding	N/A	Flooding	CBDR
RMBLNS + CBDR	RMBLNS	Flooding	N/A	Flooding	CBDR
CBLNS-F + CBDR	CBLNS-F	Flooding	N/A	Flooding	CBDR
CBLNS-P + CBDR	CBLNS-P	Flooding	N/A	Flooding	CBDR
MBLNS + CBDNS + CBDR	MBLNS	CBDNS	N/A	Flooding	CBDR
CBLNS-F + CBDNS + CBDR	CBLNS-F	CBDNS	N/A	Flooding	CBDR
CBLNS-P + CBDNS + CBDR	CBLNS-P	CBDNS	N/A	Flooding	CBDR

Table 6.2 shows the parameter values used in our experiments for different algorithms. The meanings of these parameters are:

- *rm* (*ratio of matched query terms*): the percentage of the query terms that need to be matched for resource selection and document retrieval. It defines the query matching rule. For example, the value of 100% means that all the query terms of a query need to be matched.
- *rs* (*ratio of selected leaf nodes*): the maximum percentage of the neighboring leaf nodes that can be selected by a directory node for query routing, which is used by Random Match-Based Leaf Node Selection (RMBLNS) and Content-Based Leaf Node Selection (CBLNS-F and CBLNS-P).
- *ns* (*number of selected directory nodes*): the maximum number of the neighboring directory nodes that can be selected by a directory node for query routing, which is used by Content-Based Directory Node Selection (CBDNS).
- *ms* (*model size*): the maximum number of the terms that are recorded in a directory node's content model learned from past queries, which is used by Content-Based Directory Node Selection (CBDNS).
- λ (*smoothing weight*): the smoothing parameter used in Equation 2 by content-based retrieval.

The Gnutella 0.6 protocol doesn't specify how many query terms need to be matched for the query matching rule of name-based retrieval. It is left to be defined by each implementation of the Gnutella 0.6 protocol. We ran name-based retrieval with two values for parameter *rm* (*ratio of matched query terms*): 100% and 50%. Because name-based resource selection and document retrieval only use the title fields of the documents, which are usually short and hence difficult for all of the query terms to match, the value of 100% for *rm* may lead to too few query messages and retrieved documents (i.e., low Recall). However, the Precision could be higher than that using 50% for *rm*. In contrast, for match-based and content-based methods the query terms are matched against a node's vocabulary for resource selection or a document's content for document retrieval. It is much easier to match all the query terms and in fact, there may be too many of such matches. To avoid too many query messages, *rm* (*ratio of matched query terms*) for match-based and content-based methods was set to 100%.

Table 6.2. Parameter values used in the experiments.

Resource Selection and Document Retrieval Algorithms	*rm*	*rs*	*ns*	*ms*	*λ*
NBRS + NBDR 1	100%	N/A	N/A	N/A	N/A
NBRS + NBDR 2	50%	N/A	N/A	N/A	N/A
MBLNS + CBDR	100%	N/A	N/A	N/A	N/A
RMBLNS + CBDR	100%	2.5%	N/A	N/A	N/A
CBLNS-F + CBDR	100%	1.0%	N/A	N/A	0.2
CBLNS-P + CBDR	100%	1.0%	N/A	N/A	0.2
MBLNS + CBDNS + CBDR	100%	N/A	2	750	0.2
CBLNS-F + CBDNS + CBDR	100%	1.0%	2	750	0.2
CBLNS-P + CBDNS + CBDR	100%	1.0%	2	750	0.2

For parameter *rs* (*ratio of selected leaf nodes*), the values of 1.0% and 2.5% indicate that a directory node could route query messages to up to 1.0% and 2.5% of its neighboring leaf nodes for Content-Based Leaf Node Selection and Random Matched-Based Leaf Node Selection respectively. The choice of 1.0% for *rs* in Content-Based Leaf Node Selection was quite greedy. Because the maximum number of leaf nodes a directory node connected to was 1,008 (Table 4.1), a directory node could route query messages to a very small amount of leaf nodes. This would greatly reduce the number of query messages, but may lead to lower Recall. Our experiments show that Content-Based Leaf Node Selection with 1.0% for *rs* gives satisfactory retrieval performance in the hybrid P2P network. The value of 2.5% was selected for *rs* in Random Match-Based Leaf Node Selection to yield similar average number of query messages per query as Content-Based Leaf Node Selection in order to compare their Precision and Recall on the same basis.

λ (*smoothing weight*) was set to 0.2 empirically.

Since a directory node had 4 neighboring directory nodes on average, the number of neighboring directory nodes it selected for query routing should be smaller than 4 using Content-Based Directory Node Selection. Otherwise, there would be little difference compared with using the flooding technique. We set *ns* (*number of selected directory nodes*) to 2. Directory nodes were also allowed to select one additional neighboring directory node randomly, to increase robustness, thus a directory node could route query messages to up to 3 of its neighboring directory nodes using Content-Based Directory Node Selection.

ms (*model size*) was 750 terms. When the model size reached its limit, a third of the terms in the model were discarded, in ascending order of their frequencies, to make room for new terms. Some experimental results not shown in this paper indicate that retrieval accuracy doesn't improve much as the model size is increased, so we chose 750 for simulation efficiency reasons.

7. EXPERIMENTAL RESULTS

Table 7.1 shows the experimental results for retrieval accuracy and query routing efficiency using different resource selection and document retrieval algorithms for federated search of a hybrid P2P network. Set-based Precision and Recall were measured based on comparing the retrieval results from the collections in

Table 7.1. The accuracy and efficiency of different algorithms tested in the experiments.

Resource Selection and Document Retrieval Algorithms	Precision	Recall	F-Score	Avg Num of Query Messages Per Query
NBRS + NBDR 1	70.97%	4.98%	0.0931	71
NBRS + NBDR 2	21.88%	28.85%	0.2489	479
MBLNS + CBDR	62.12%	33.76%	0.4375	540
RMBLNS + CBDR	60.94%	19.59%	0.2965	122
CBLNS-F + CBDR	**71.82%**	**29.65%**	**0.4197**	**116**
CBLNS-P + CBDR	**72.21%**	**29.42%**	**0.4181**	**114**
MBLNS + CBDNS + CBDR	62.31%	32.49%	0.4271	457
CBLNS-F + CBDNS + CBDR	**71.95%**	**28.46%**	**0.4079**	**85**
CBLNS-P + CBDNS + CBDR	**72.42%**	**28.48%**	**0.4088**	**83**

the P2P network with those from the single WT10g-subset collection, as stated in Equations 5 and 6. The retrieval accuracy results (Table 7.1) are Recall averaged over all queries and Precision averaged over those queries that returned a non-zero number of retrieved documents.

The results show that content-based retrieval in the hybrid P2P network had much higher Precision and a smaller average number of query messages per query than name-based retrieval with the query matching rule that required at least half of the query terms to be matched. This means that content-based resource selection algorithms were more effective at selecting nodes that were likely to satisfy the user's information need, and the content-based document retrieval algorithm was more likely to retrieve relevant documents than name-based approaches. Although name-based retrieval with the query matching rule that required all the query terms to be matched was very efficient and had comparable Precision as content-based retrieval, the Recall was much lower. The higher F-score value of content-based retrieval confirmed that content-based retrieval had higher retrieval accuracies than name-based retrieval. Overall, content-based retrieval was more accurate and efficient than name-based retrieval in the hybrid P2P network.

Compared with Match-Based Leaf Node Selection, Random Match-Based Leaf Node Selection greatly improved the efficiency of query routing, at the cost of reducing Recall. Although the difference in Precision was negligible (1.9%), the relative performance loss for Recall was very large (42.0%). If low Recall is not a major concern, then Random Match-Based Leaf Node Selection may be an efficient alternative to content-based resource selection algorithms for retrieval in hybrid P2P networks.

Using Content-Based Leaf Node Selection based on either the full resource descriptions or the pruned resource descriptions of leaf nodes could significantly improve the efficiency of query routing without degrading the Recall much, compared with Match-Based Leaf Node Selection. The increase in Precision indicated that using term frequency information made Content-Based Leaf Node Selection more effective at restricting the query routing only to those leaf nodes that were very likely to satisfy the user's information need. The slight drop in Recall was because query messages were routed to much fewer leaf nodes and so relevant documents from leaf nodes not selected were missed. The F-score values of Content-Based Leaf Node Selection and Match-Based Leaf Node Selection were comparable, which indicated that these two resource selection algorithms gave comparable retrieval accuracies.

Compared with using the full resource descriptions of leaf nodes, using the pruned resource descriptions of leaf nodes didn't change the accuracies much. However, the storage costs at directory nodes were greatly reduced. In our experiments, the average size of the resource selection index (the index that was generated from the resource descriptions of leaf nodes' collections) at a directory node was 81.29 MB with the full resource descriptions while it was only 37.49 MB with the pruned resource descriptions. The percentage of the disk space reduced for storing the resource selection index was 53.9% on average for a directory node.

Using Content-Based Directory Node Selection in addition to Content-Based Leaf Node Selection could further improve the query routing efficiency without degrading the retrieval accuracies.

These results demonstrate that i) content-based retrieval is more accurate and more efficient for some hybrid P2P networks of digital libraries than Gnutella 0.6's name-based retrieval, and ii) content-based resource selection algorithms are more accurate and more efficient for content-based retrieval in hybrid P2P networks, compared with simple match-based resource selection algorithms.

8. CONCLUSIONS AND FUTURE WORK

This paper studies the use of content-based resource selection and document retrieval algorithms in hybrid peer-to-peer networks. Each leaf node is modeled as a digital library running an effective content-based text retrieval algorithm, and each directory node is modeled as a content-based resource selection service covering a set of digital libraries. Experimental results demonstrate that content-based resource selection and text retrieval algorithms are far more accurate and efficient than the name-based retrieval and flooding methods that are currently more common. In particular, the average number of network messages per query is reduced substantially without reducing Recall, and set-based Precision and F-Score are both substantially higher.

Research results are only as credible as the data upon which they are based. The results reported here are based upon a new peer-to-peer testbed [18] created from the TREC WT10g dataset, which is widely available to the research community. The testbed contains over a million documents, 2500 leaf nodes ("digital libraries"), 25 directory services, and tens of thousands of queries. Although it is still small when compared to "real world" peer-to-peer networks, and the digital libraries are also relatively small on average, this testbed is one of the largest to be used so far for research on peer-to-peer systems. We hope to see more research on larger-scale peer-to-peer networks in the future.

Large-scale peer-to-peer networks have emerged during the last few years as an effective method of providing federated search across very large networks of very simple digital libraries. Peer-to-peer networks are an appealing architecture upon which to provide federated search across much larger and more sophisticated digital libraries but, as shown in this paper, the simple query-routing and information retrieval methods used in current peer-to-peer networks won't suffice. In particular, name-based retrieval is not sufficient when digital libraries are large or when file-naming conventions are uncertain.

We view peer-to-peer networks as a particular type of distributed information retrieval environment, albeit one that has unique characteristics. It is likely that other techniques developed for

distributed information retrieval would also be effective in peer-to-peer networks. For example, directory nodes could use *query-based sampling* to automatically discover the contents of leaf nodes by posting queries and observing the documents returned, and *result-merging* algorithms could be used to merge the ranked lists of documents returned by different digital libraries into a single, integrated ranked list.

In our studies, resource selection occurred only at directory nodes. For future research we would like to study the performance of networks where leaf nodes also have the ability to conduct resource selection, for example for local "neighborhood" search. We would also like to develop a framework for resource selection at directory nodes that integrates selection of leaf nodes and selection of directory nodes, rather than treating them as separate problems, as is now common.

The work reported here didn't consider the communication costs associated with sending resource descriptions from leaf nodes to directory nodes, but these costs could be significant in low-bandwidth environments (e.g., nodes connected by modems). Prior research on using sampled and pruned resource descriptions in more traditional distributed information retrieval environments [3, 19] suggests that more compact resource descriptions could also be used for resource selection in peer-to-peer environments, but this remains an item for future work.

ACKNOWLEDGMENTS

This material is based on work supported by NSF grant IIS-0096139. Any opinions, findings, conclusions or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsor.

REFERENCES

[1] K. Aberer. P-Grid: A self-organizing access structure for P2P information systems, In *Proc. of the 6th International Conference on Cooperative Information Systems*, 2001.

[2] A. Asvanund, R. Krishnan, M.D. Smith, R. Telang, S. Bagla, and M. Kapadia. Intelligent club management in peer-to-peer petworks. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[3] J. Callan. Distributed information retrieval. W. B. Croft, editor, *Advances in information retrieval*, chapter 5, pages 127-150. Kluwer Academic Publishers, 2000.

[4] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. S. Maarek and A. Soffer. Static index pruning for information retrieval systems. In *Proc. of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[5] F. Cuenca-Acuna and T. Nguyen. Text-based content search and retrieval in ad hoc p2p communities. Technical Report DCS-TR-483, Rutgers University, 2002.

[6] F. Dabek, M. Kaashoek, D. Karger, R. Morris, I. Stoica. Wide-area cooperative storage with CFS. In *Proc. of the18th ACM Symposium on Operating Systems Principles*, 2001.

[7] L. Gravano and H. Garcia-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In U. Dayal, P. Gra and S. Nishio, editors, *VLDB'95*, *Proc. of 21th International Conference on Very Large Data Bases*, pages 78-89, 1995.

[8] L. Gravano, H. Garcia-Molina and A. Tomasic. The effectiveness of GlOSS for the text database discovery problem. In R. T. Snodgrass and M. Winslett, editors, *Proc. of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 126-137, 1994.

[9] The Gnutella protocol specification v0.4. http:// www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[10] The Gnutella protocol specification v0.6. http://rfc-gnutella.sourceforge.net.

[11] D. Hawking. Overview of the TREC-9 web track. In *Proc. of the 9th Text Retrieval Conference (TREC-9)*, 2000.

[12] M. Jansen, A. Spink and T. Saracevic. Real Life, real users, and real needs: A study and analysis of user queries on the web. *Information Processing and Management*, 36(2).

[13] Javasim. http://javasim.ncl.ac.uk/.

[14] V. Kalogeraki, D. Gunopulos and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *Proc. of the 11th International Conference on Information Knowledge Management*, 2002.

[15] KaZaA. http://www.kazaa.com.

[16] Limewire. http://www.limewire.com.

[17] K. Lin and R. Kondadadi. A similarity-based soft clustering algorithm for documents. In *Proc. of the 7th International Conference on Database Systems for Advanced Applications*, 2001.

[18] J. Lu and J. Callan. Peer-to-peer testbed definitions: trecwt10g-2500-bysource-v1 and trecwt10g-query-bydoc-v1. http://hartford.lti.cs.cmu.edu/callan/Data, 2003.

[19] J. Lu and J. Callan. Pruning long documents for distributed information retrieval. In *Proc. of the 11th International Conference on Information Knowledge Management*, 2002.

[20] P. Ogilvie and J. Callan. Experiments using the Lemur toolkit. In *Proc. of the 10th Text Retrieval Conference (TREC-10)*, 2001.

[21] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of the ACM SIGCOMM'01 Conference*, 2001.

[22] C. Tang, Z. Xu and M. Mahalingam. Efficient information retrieval in peer-to-peer networks. In *Proc. of HotNets-I, ACM SIGCOMM*, 2002.

[23] S. Waterhouse. JXTA Search: Distributed search for distributed networks. Technical report, Sun Microsystems Inc., 2001.

[24] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[25] Y. Zhang, W. Xu and J. Callan. Exact maximum likelihood estimation for word mixtures. In *Workshop on Text Learning of the 9th International Conference on Machine Learning (TextML' 2002)*, 2002.