Content-based search in peer-to-peer networks

Yun Zhou
yzhou@cs.umass.edu

W. Bruce Croft
croft@cs.umass.edu

Brian Neil Levine
brian@cs.umass.edu

Dept. of Computer Science , University of Massachusetts, Amherst, MA 01003

ABSTRACT

Gnutella, a well-known P2P system, uses resources inefficiently when directly applied to information retrieval problems. In this paper we propose an efficient search mechanism that extends the standard Gnutella protocol to support content-based retrieval in P2P networks. The idea is to estimate locally the relevance of peers when they receive query messages. Only those peers estimated as relevant will retrieve the query and send response messages back to the source. Based on a large real testbed evaluation, we show that our method improves the tradeoff between the quality of retrieval and resources consumed while preserving most advantages of standard Gnutella.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Search process;H.3.4[**Systems and Software**]:Distributed Systems, Performance Evaluation

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

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

1. INTRODUCTION

Peer-to-Peer (P2P) networks are a powerful architecture for sharing computing resources and data. In the strictest definition, each peer has the functionality of both server and client, and accordingly, can both provide and request information. The decentralized nature of P2P systems can be an advantage over client-server architectures. First, they tend to be more fault-tolerant as there is no single point-of-failure. Second, processing, network traffic, and data storage can be balanced over all peers, which enables the network to scale well with the number of peers. These advantages come with the cost of requiring a more complicated strategy to locate specific resources.

Because they may support a large number of peers and have no central index, efficient search in a P2P system can be a challenge. In general, peers can locate resources or content by propagating queries through the network and then waiting for results from peers with relevant results. Many specific strategies have been proposed. The simplest approach is taken by the Gnutella [1] protocol, which broadcasts query messages to each neighbor, hop-by-hop across the network within some distance from the source. Although it is not efficient in terms of network bandwidth, this technique is simple, robust, and has a minimum requirement on cooperation and consistency among peers. For example, it allows arbitrary network topologies, and each peer stores no information regarding the state of others.

Distributed Hash Tables (DHT) (e.g., [2]) are more efficient in terms of network bandwidth, but scale poorly with the number of terms (i.e., *keys*) indexed per document. If the entire filename is one key, then the cost of indexing content is cheap; indexing documents based on terms that appear in the content itself is impossible with a DHT. Therefore, Internet-based digital libraries of text documents cannot be supported with DHTs or Gnutella, though it is still desirable to support a sophisticated semantic-based search with a P2P architecture.

Some techniques, such as SETS [3], try to reorganize the topology of network so that topic-related peers are close to each other. By taking advantage of a rigid topology, network traffic can be reduced. In other techniques, such as the localized search mechanism proposed by Kalogeraki, *et al* [4], each node maintains an index or a profile of its neighbors' content that is used rank its neighbors. Search is then restricted to what are believed to neighbors with relevant results.

The cost of those approaches, like in DHTs is increased coordination among peers, which must be sustained as peers join and leave, and change their content. Moreover, each peer has an increase storage cost to participate in the system.

In this paper we propose an efficient search mechanism that extends standard Gnutella protocol to support content-

based retrieval in P2P networks. Our approach is for peers themselves to estimate their relevance to each query they receive. If they judge themselves to have a relevant response, peers send a response messages directly back to the source. Our approach eliminates coordination that other protocols incur during membership or content changes, and it reduces the number of responses that a source receives to a tunable amount. Although our approach does not directly lower the number of queries received by each peer, our use of a random topology allows us to localize queries so that they do not reach every peer in the system.

We evaluated our approach on a large testbed with thousands of peers. We found that the tradeoff between the quality of retrieval and resources consumed is greatly improved while most advantages of standard Gnutella are preserved. For example, according to our experiments, Gnutella consumes more than three times the network bandwidth required by our approach for the same level of recall.

The rest of the paper is organized as follows: Section 2 describes related work. Section 3 introduces our resource-efficient algorithm for content-based search in P2P networks. We present our testbed and evaluation methodologies in Sections 4 and 5. Experimental results are shown in Section 6. Section 7 summarizes the conclusions of this paper and discusses future work.

2. RELATED WORK

Search in peer-to-peer networks is a problem rich in previous work.

As we stated, Gnutella [5] takes the simplest possible approach to search. A peer forwards query messages to all of its neighbors until some distance from the source is reached. Since the query routing policy in Gnutella treats each peer equally regardless of queries, Gnutella is very inefficient in network bandwidth.

Other approaches improve the efficiency of routing queries by storing information about other peers' content. For example, Kalogeraki, *et al* [4] proposed storing profiles of the past query behavior of each neighbor to improve the future search efficiency. This approach is not robust since it assumes that users would submit similar queries. Yang, *et al* [6] proposed a technique where each peer maintains an index of other peers' resources who are within some number of hops. Maintaining such an index is costly if topology or membership changes are frequent. One advantage of our method is that each node is fully decentralized and does not need to store other peers' information.

Another idea to improve search efficiency in P2P networks is to reorganize the topology of networks [7] [3]. For instance, Mayank, *et al* [3] propose maintaining a topology where peers grouped together if they have libraries of similar topics; queries are then routed only to the closest clusters. Although the search algorithm can take advantage of the reorganized topology to improve efficiency, the approach suggests some trade-offs. First, placing peers together based on topic may degrade network performance if juxtapose peers have poor bandwidth between them. Second, maintenance costs does not scale well with membership changes. And finally, in the quality of the clusters, which depends on the characteristics of collections of peers, has a large affect on the success of the reorganized topology [3].

Lu, *et al* [8] addresses the problem of content-based retrieval in hybrid P2P architectures. In contrast to pure P2P systems where all nodes are equal and no functionality is centralized, hybrid P2P architectures introduce directory nodes responsible for regionally centralized directory services. In [8] a directory node, also known as *super peer*, keeps pruned indices of its neighboring leaf nodes and use a KL-divergence-based similarity between query and collection to rank those leaf nodes. Query messages are only forwarded to top ranked leaf nodes. In our approach each peer locally decides its relevance to the query by calculating the same KL-divergence-based similarity. The advantage of our approach is that there are no directory nodes that are costly to maintain.

Distributed IR research assumes a central sever-client architecture where the central server has direct access to the indices of all collections in clients. One of distributed IR problems most related to this paper is resource selection, that is, how to pick the most relevant collections.

The CORI resource selection algorithm [9] uses a Bayesian inference network model in the INQUERY system to rank collections. Although it is stable and effective, it is hard to integrate this method to search engines other than INQUERY. Xu et al[10] proposed a method where collections are ranked by Kullback-Leibler divergence between query language model and collection model. In [11] Luo Si et al used the very similar approach for resource selection. Both methods estimate language models by word frequency. The only differences are in details on how to estimate the language models. In our experiments we use the same approach to estimate the relevance of a collection given a query.

3. SEARCH IN P2P NETWORKS

In this section, we first define our search problem and detail our assumptions. Then, we introduce our efficient search algorithm, which is an extension of the Gnutella approach.

3.1 Problem definitions and assumptions

In a P2P network each peer has the same role and the communication between any two nodes is symmetric. Such a network can be viewed as an undirected graph. Each node represents one peer in the network. If peer A directly connects to peer B then there is a logical edge between the two corresponding nodes. In that case, A is called the neighbor of B and vice versa. In this paper, we also assume that the network graph can be arbitrary as long as it is fully connected and we do not modify the topology.

Each peer is assigned a local document collection on which some IR search engine runs. Peers return top-ranked documents as the result for a given query. For simplicity, we assume each search engine is optimal, that is, it returns all of relevant documents it has. The queries considered here are in natural language style such as, "information about what manatees eat".

The peer who submits a query to the network is called the *source* of the query. The source can only send the query message to its neighbors and the query message is propagated over the network by their sending to their neighbors. The routing protocol decides to which neighbor the query message is sent and when to stop sending it. The peer receiving the query message, if it decides to answer the query, will run the search engine and send back the response message containing retrieved documents for the query. Here we assume that the response message is sent back directly to the source.

3.2 The efficient search mechanism: extending the Gnutella protocol

A naive application of Gnutella to content-based retrieval would flood query messages within some predefined search depth limit and all peers receiving the query message do retrieval and reply to the source. By doing so, a lot of network bandwidth is wasted since only a few peers have documents relevant to a given query.

Our goal is to achieve a desired level of recall as efficiently as possible while preserving the advantages of Gnutella, which includes the lack of coordination among nodes. In a random graph, peers that have relevant documents are randomly distributed across the network. Thus, on average, the more peers that are visited by a query, the more relevant documents the source peer may receive; and consequently, the more bandwidth that is consumed. We cannot achieve a sufficient recall level by only visiting a small number of nodes in our defined problem. There is an unavoidable trade-off between the quality of retrieval and the network bandwidth. We are interested in how to minimize this trade-off. Since a peer is visited only when it receives the query message, it is difficult to reduce the number of query messages if a reasonable recall level is required. However, the response message, which is much larger than the query message, can be used more efficiently.

We modify Gnutella such that, when receiving query messages, only peers estimated as relevant reply to the source. No matter estimated as relevant or not, all peers receiving query messages still forward query messages to all of their neighbors until a distance from the source is reached. At each peer receiving the query message, we calculate $P(Q|C)$, the probability of generating query, Q, from the collection, C, of a peer. Then we set a threshold. Peers with $P(Q|C)$ above the threshold are regarded as relevant. Only relevant peers will retrieve the query against their collections and send back the source retrieval results.

Even though our approach floods query messages, it is still resource-efficient. The reasons for this are two-fold. First, the size of query message is much smaller than the size of response message that contains retrieval results. Response messages are well utilized in our algorithm. Secondly, in traditional server-client architecture or hybrid P2P architecture, a central server or a super peer is responsible for calculating the relevance of a larger number of neighboring peers in order to rank them. In our approach such a resource-consuming computation is divided over individual peers, which is desirable because of the decentralized nature of P2P networks.

This probability, P(Q|C), is computed as follows [11]

$$P(Q \mid C) = \prod_{q \in Q} \{\lambda P(q \mid C) + (1 - \lambda) P(q \mid G)\}$$

$$where \quad P(q \mid C) = \frac{\#(q, C)}{\| C \|},$$

$$P(q \mid G) = \frac{\#(q, G)}{\| G \|}$$

where C is the local collection of a peer and G is the global collection that can be obtained from some large general-purpose English collection. $\#(q,C)$ and $\#(q,G)$ denote the total counts of the term q in the collection C and G respectively. $||C||$ and $||G||$ denote the total counts of all terms in the collection C and G, respectively. $P(q|G)$ can be seen as a global information and can remain stable for different collections as long as they are large enough and general-purpose. λ is the smoothing parameter and in section 6 we vary λ to investigate its effect on performance.

$P(Q|C)$ was proposed by Xu and Croft [10] for the resource selection problem in distributed IR. $P(Q|C)$ measures the similarity between the query model and the collection model. So the larger the $P(Q|C)$ is, the more likely the collection is relevant.

Given $P(Q|C)$, we need to set a threshold to make a binary decision: relevant or not. Here we define the original threshold as

$$t = \prod_{q \in Q} P(q \mid G)$$

for a query Q. It represents the likelihood for a query Q to be generated from a general and large English corpus. In section 6 we vary the threshold to see its effect on performance.

In addition to the local collection frequency $P(q|C)$, which is already in the local index , each peer also needs to store the global collection frequency $P(q|G)$. Since usually the query language only covers a quite small part of the vocabulary of the global collection, the vocabulary of global collection can be greatly pruned to save space on each peer. We also notice that $P(Q|C)$ can be cheaply calculated by simply looking up query terms in the two collection frequency tables. All of these factors make the computation of P(Q|C) efficient and feasible.

4. EXPERIMENTAL SETTINGS

We use TREC web corpus WT10G [12] as our test collection. This corpus has many properties of real Web data, such as a realistic distribution of the number of documents per web site. There are about 16.9 million documents in WT10G. In this section, we describe how our testbed was set up, the network topology, and the query set we used.

4.1 Testbed Setup

We group documents in WT10G into more than ten thousand collections according to their IP addresses. Each IP address corresponds to one peer. Table 1 shows the number of peers, and the number of documents per peer, where the number of documents is binned into four ranges. For example, there are 6,188 peers with a collection size of between 5 and 29 documents.

N(the # of docs)	[5,29]	[30,59]	[60,99]	More than 100
The # of peers	6188	1951	1068	2305

Table 1. document-peer distribution in WT10G

Many peers have libraries with only a few documents; we select only those peers that have at least 30 documents. Thus we use 5,324 peers, and a total of 16.1 million documents in all. Each peer has one subset where all documents are from the same IP address. Our testbed covers most of documents in WT10G and has a large number of nodes.

4.2 Network Topology

The connections between peers were generated randomly. Since recent studies have shown that Internet graphs follow power laws [13], to make our experiments more real, we adopt the Power-Law Out-Degree algorithm in [14] to generate such a random graph. Peers are assigned randomly to nodes in a one-to-one correspondence and there are 5,324 nodes.

4.3 Query Set

We use 50 title queries from TREC 2001 web *ad hoc* topics that have relevance judgments from the TREC corpus [16]. According to our statistics, 4,603 peers do not have any relevant documents for any query in the query set. So relevant documents are distributed over a very small number of peers, which makes locating them difficult in a random topology.

5. EVALUATION METHODOLOGY

In P2P networks, we want to minimize resources consumed for a fixed level of retrieval quality. In this paper, resources we are interested in are network bandwidth and query processing cost. We measure retrieval quality with two metrics: recall and a modified version of *Mean Reciprocal Rank* (MRR). We evaluate our approach in the following three ways: query-processing efficiency, network efficiency, and a modified version of MRR.

5.1 Query-processing efficiency

Here the query-processing cost is referred to as the cost consumed by the search engine on individual peers. Assuming each peer uses the same search engine, query-processing cost can be measured by the number of peers required to do retrieval given a query. We are interested in this measure because many effective retrieval algorithms are both time and space consuming and peers may receive a lot of queries at the same time. On the other hand, in a real P2P system, given a query, many peers do not contain any relevant documents. In order to utilize the computing resources of each peer as efficiently as possible, we introduce query-processing efficiency. Given a query, if we assume N peers are searched and M out of N peers need to perform retrieval on their collections, then the recall level, R, is the number of relevant documents in those M peers over the total number of relevant documents in the network. The query processing efficiency is measured by the average ratio of (R/M).

5.2 Network bandwidth efficiency

The efficiency of network bandwidth is measured by the average bandwidth consumed at a certain recall level. Given a query, we assume the cumulative recall after N peers are searched is R, and M out of N peers need to reply the source, the bandwidth consumed at recall level R is calculated as:

$$\text{Bandwidth} = N*S1+M*S2,$$

where $S1$ denotes the size of the query message and is set to 100 bytes in our experiments; $S2$ denotes the size of the response message. We assume each response message contains 10 documents with the size of 1,000 bytes and the response message header with the size of 100 bytes. So $S2$ is set to 10,100 bytes in our experiments.

5.3 MRR

Above, we use recall to measure retrieval performance. Mean reciprocal rank (MRR) is another metric to measure retrieval performance when high retrieval accuracy is the primary concern. In information retrieval systems, typically the user receives a ranked list and in some cases the user only needs one relevant document, which he hopes is ranked as high as possible. *Question Answering* (QA) systems only require one correct answer and are given no credit for multiple correct answers [15]. In these cases, the rank of the first relevant document is more important than how many documents are relevant in the ranked list. So MRR, which is defined as the inverse of the rank of the first relevant document, is a better metric for evaluating high accuracy retrieval.

MRR as defined in information retrieval cannot be directly used in our problem since we do not rank peers. Accordingly, we introduce a modified version of MRR. The search process in our approach described in Section 3.2 can be viewed as a breadth-first search and furthermore can be simulated by a queue. In such a queue, those peers replying the source consist of a returned list. MRR in this paper is defined as the inverse of the position of the first relevant node in a returned list. For example, suppose the search queue is *P1, P2, P3, P4, P5* and only *P2, P4* and *P5* reply to the source. We also assume *P1, P4* and *P5* are relevant peers; that is, they are peers that have one or more relevant documents given a query. The returned list is *P2, P4, P5* and the MRR is ½ since *P4* is the first relevant peer. If there is no relevant peer in a returned list, the MRR will be set to 0. MRR is bounded between 0 and 1. In the optimal case where only relevant peers reply to the source, the MRR should always be 1.

6. EXPERIMENTAL RESULTS

In this section, we show results for our three evaluation methodologies. Since every peer can post a query, for each query we randomly pick 100 nodes from the network as the sources and average them. In our experiments, we are only interested in cases where only a part of the network is searched, which is common in real P2P systems.

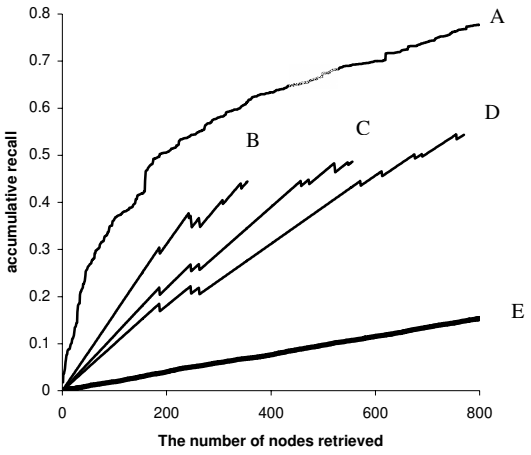
6.1 Results for query-processing efficiency

We first investigate the effect the smoothing parameter lambda has on the performance. We use centralized architecture and standard Gnutella as our baseline. In Figure 1, the line labeled "Central mode" denotes the accumulative recall when all peers are directly connected to the central server and the server ranks peers according to $P(Q|C)$ defined in Section 3.2 and retrieves peers in that order. "Gnutella" denotes the naive application of Gnutella to content-based retrieval where each peer receiving the query message will reply the source. Gnutella can be seen as the special case of our approach when every node is considered as relevant. In Figure 1 our algorithm uses the original threshold values defined in section 3.2, with lambda set to 0.2, 0.5, and 0.8, respectively. These three lambda settings represent the three cases: heavy smoothing, medium smoothing, and light smoothing with the global collection model.

In Figure 1 and 2 each point is actually the average number over a set of queries and we require that there should be at least 90% of queries (45 queries) for each point. If some points are not averaged over all queries, the corresponding curve may not be continuous.

In our experiments we use title queries. The query terms are distributed over only a few peers, the higher the lambda is, the more a peer relies on the occurrence of exact query terms to be judged as relevant. In that case, the number of peers above threshold is decreased when lambda is increased. That's why the curve with "lambda=0.8" has a fewer number of retrieved node than does the curve with "lambda=0.2".

Figure 1: query-processing efficiency



A: Central mode

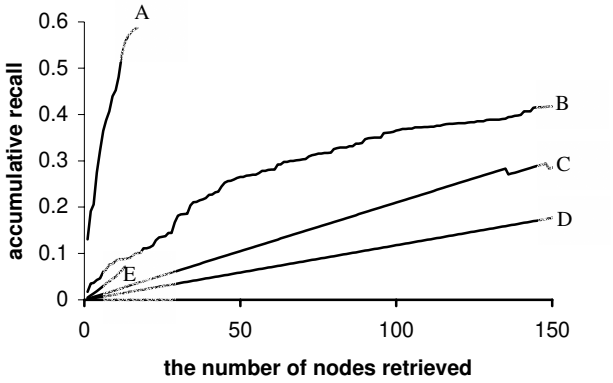
B: Lambda=0.8

C: Lambda=0.5

D: Lambda=0.2

E: Gnutella

Figure 2:query-processing efficiency cont.



A: Optimal mode

B: Central mode

C: Threshold_1

D: Original threshold

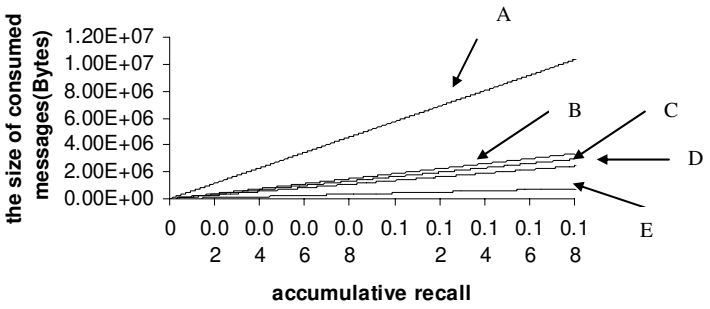
E: Threshold_3

Figure 1 shows that our approach is significantly better than Gnutella no matter how lambda is set. This is because our approach can intelligently estimate the relevance of each peer. Secondly, a higher lambda is helpful for improving query-processing efficiency. Lastly, a higher lambda has the risk that some of the queries may not have a high recall even when the whole network is searched.

In Figure 2, "Optimal mode" denotes evaluation of Gnutella where only peers who really have relevant documents are retrieved and reply to the source. "Optimal mode" shows us the case when prediction of the relevance of peers is 100% accurate, so it can provide us with a theoretical upper bound on recall when a certain number of nodes are retrieved. "Central mode" is the same as in Figure 1 with only the first 150 nodes plotted. "Original threshold" is the same as "Lambda=0.5" in figure 1 with only the first 150 nodes plotted. "Threshold_1"and"Threshold_3" are different thresholds in our algorithm while lambda is kept to 0.5. In "Threshold_1" the threshold is set to $e^{1.0}$ (that is 2.72) times the original threshold. Similarly, "Threshold_3" is $e^{3.0}$ times the original threshold .

From figure 2 we can see that a high threshold is very helpful for improving the ratio of the number of retrieved node over recall. However, there is some trade-off. The higher the threshold is, the fewer the number of peers above the threshold. So the highest recall could be very low, as in experiments for "Threshold_3", where the threshold is set too high.

Figure 3:network-bandwidth efficiency



A: Gnutella

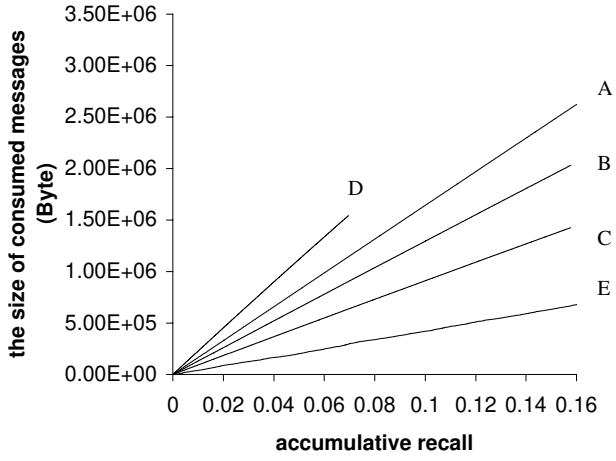
B: Lambda=0.2

C: Lambda=0.5

D: Lambda=0.8

E: Optimal mode

Figure 4:network-bandwidth efficiency cont.



A: Original threshold

B: Threshold_1

C: Threshold_2

D: Threshold_3

E: Optimal mode

Table 2 MRR results: vary lambda

lambda	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
MRR	0.074	0.076	0.078	0.081	0.085	0.092	0.098	0.10	0.12	0.14
Mess. consumed (M bytes)	1.62	1.54	1.50	1.42	1.35	1.28	1.19	1.11	1.02	0.96

Table 3 MRR results: increase threshold

Multiple of orig.thre.(Log scale)	1.0	2.0	3.0	4.0	5.0	5.5	6.0	7.0
MRR	0.15	0.16	0.18	0.21	0.24	0.25	0.19	0.15
Mess.size (M bytes)	1.0	1.2	1.5	1.9	2.2	2.4	2.6	2.8

6.2 Results for network-bandwidth efficiency

Network bandwidth is a valuable resource in networks. We always want to locate as many relevant peers as possible within bounded network bandwidth

First we show what effect the smoothing parameter lambda has on network bandwidth efficiency. In figure 3 we adopt "Gnutella" and "Optimal mode" as our baselines. "Gnutella" and "optimal mode" are defined as in Section 6.1. We keep the original threshold for "lambda=0.2", "lambda=0.5" and "lambda=0.8".

From figure 3 we can gain the following insights: (1) our approach, no matter how the lambda is set, can greatly improve network bandwidth efficiency compared to Gnutella. (2) High lambda is helpful for improving network bandwidth efficiency. However, we noticed that a high lambda decreases the recall as discussed in section 6.1.

Figure 4 shows how increasing the threshold affects network bandwidth efficiency. In figure 4, "Original threshold" is the same as "lambda=0.5" in figure 3. Lambda is kept to 0.5 for "Threshold_1","threshold_2"and"threshold_3", where"Threshold_1","threshold_2"and"threshold_3"are $e^{1.0}$, $e^{2.0}$ and $e^{3.0}$ times the original threshold respectively.

From figure 4 we can see that an appropriate threshold is important for performance. Network bandwidth efficiency is improved when the threshold is increased within some range. However, when it is set too high, as in "threshold_3", it hurts efficiency. When the threshold is very high, few or none of the peers will be estimated as relevant by our approach. In this case, network-bandwidth is wasted since we need to search a larger part of the network to reach the same level of recall.

6.3 MRR Results

Table 2 shows the results as lambda increases and the threshold is kept to original one as defined in 3.2. In table 2 the first row is lambda, the second row is the MRR defined in 5.3, the third row is the size of messages consumed when a certain MRR is reached. To calculate the size of consumed messages, we use the same assumption described in 5.2 and assume that the search process will stop if one relevant peer is found or some pre-defined search depth is reached. We can see that the case with lambda=1.0 leads to the best MRR and the lowest size of consumed messages. Lambda=1.0 means no smoothing at all and given a query only peers including all query terms will be estimated as relevant by our algorithm. No smoothing helps

here because we use title queries that typically consist of only a few words.

Table 3 shows the results when threshold increases and lambda is still kept to 1.0. The first row shows multiplies of the original threshold in log scale. For example, 5.0 in the first row means $e^{5.0}$ times the original threshold. The second and the third row have the same meaning as in Table 2.

From Table 3 we can see that MRR increases with the increasing of threshold until $e^{5.5}$ times original threshold which gives the best MRR value. The size of the consumed message also increases when threshold increases because more peers will be visited.

7. CONCLUSIONS AND FUTURE WORK

We proposed an efficient search algorithm by extending the Gnutella protocol. The idea is to estimate the relevance of peers locally when receiving query messages. Only those peers deemed as relevant will retrieve the query and send response messages back to source. Based on evaluation of a large real testbed, we show that the tradeoff between the quality of retrieval and the resources consumed is greatly improved over Gnutella while simplicity, robustness and the autonomy of peers are maintained.

We also notice that the collection language model P(Q|C) is far from being the perfect indicator of the relevance of peers. For the future work, P(Q|C) may be improved by trying different smoothing techniques such as Dirichlet smoothing. We are also interested in other methods that may help to predict the relevance well in P2P systems.

8. ACKNOWLEDGMENTS

We thank Steve Cronen-Townsend, Vanessa Murdock and Xiaoyong Liu for their comments on this work. This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903 . Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

9. REFERENCES

[1]The Gnutella protocol specification v4.0 http://www9.limewire.com/developer/gnutella_protocol_v4.0.pdf

[2]S. Ratnasamy,et al. A scalable content-addressable network In *ACM SIGCOMM '01,Auguest 2001*

[3] M. Bawa, G. S. Manku, P. Raghavan. SETS: Search enhanced by topic segmentation. In proceedings of the 26th annual international ACM SIGIR conference, 2003

[4] V. Kalogeraki, D.Gunopulos,D.Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *proceedings of the ACM CIKM 02 conference ,2002*

[5] http://www.gnutella.com/

[6] B. Yang, H. Garcia-Molina. Efficient search in Peer-to-Peer Networks. In *Proc. Intl. Conf. On Distributed Computing Systems (ICDCS), Vienna, Austria, 2002*.

[7]] H. Zhang, W.B.Croft and B.N. Levine. Efficient topologies and search algorithms for peer-to-peer content sharing. In *Univ. of Massachusetts,Amherst, CIIR technical report IR-314,August,2003*

[8] J. Lu, J. Callan. Content-based retrieval in hybrid peer-to-peer networks" In *proceedings of the ACM CIKM 03 conference ,2003*

[9]J. Callan "Distributed information retrieval" In *W.B.Croft ,editor,Advances in information retrieval . Kluwer Academic Publishers. (pp.127-150)*

[10]J. Xu, W.B. Croft. Cluster-based language models for distributed retrieval In *proceedings of the 22th annual international ACM SIGIR conference, 1999*

[11] L. Si, R. Jin, J. Callan, P. Ogilvie. Language modeling framework for resource selection and results merging In *proceedings of the ACM CIKM 02 conference ,2002*

[12] http://es.csiro.au/TRECWeb/wt10g.html

[13] M. Faloutos, P.Faloutsos, C. Faloutos On power-law relationships of the internet topology. In *FIGCOMM,1999*

[14] C. Palmer and J. Steffan Generating network topologies that obey power law. In *proceedings of GLOBECOM '2000.*

[15] J. Lin and B. Katz. Question answering techniques for the world wide web. In *Tutorial presentation at EACL,2003*

[16] http://trec.nist.gov/data/topics_eng/topics.501-550.txt