Difficulty-Aware Hybrid Search in Peer-to-Peer Networks

Hanhua Chen, *Member*, *IEEE*, Hai Jin, *Senior Member*, *IEEE*,
Yunhao Liu, *Senior Member*, *IEEE*, and Lionel M. Ni, *Fellow*, *IEEE*

Abstract—By combining an unstructured protocol with a DHT-based global index, hybrid peer-to-peer (P2P) improves search efficiency in terms of query recall and response time. The major challenge in hybrid search is how to estimate the number of peers that can answer a given query. Existing approaches assume that such a number can be directly obtained by computing item popularity. In this work, we show that such an assumption is not always valid, and previous designs cannot distinguish whether items related to a query are distributed in many peers or are in a few peers. To address this issue, we propose QRank, a difficulty-aware hybrid search, which ranks queries by weighting keywords based on term frequency. Using rank values, QRank selects proper search strategies for queries. We conduct comprehensive trace-driven simulations to evaluate this design. Results show that QRank significantly improves the search quality as well as reducing system traffic cost compared with existing approaches.

Index Terms—Peer-to-peer, hybrid search, flooding, DHT, difficulty awareness.

1 Introduction

Since the emergence of peer-to-peer (P2P) [28] file sharing applications, such as Gnutella [5] and BitTorrent [18], millions of users have started using P2P tools to search for desired files. P2P networks have shown a great potential to become a network tool for harnessing the information stored on desktops.

There are three different architectures for P2P file sharing systems: *centralized*, *decentralized structured*, and *decentralized unstructured*. The centralized architecture maintains a central global file index for searching, and it is commonly believed that the central index is a single point of failure for the system and is vulnerable to denial-of-service attacks. In decentralized structured models, the shared data placement and topology characteristics of the network are tightly controlled based on distributed hash functions. Decentralized unstructured P2P networks are organized in an ad hoc fashion and require no centralized directories and no precise control over the network topology or the data placement.

Decentralized P2P networks utilize different search techniques. Unstructured P2P networks mainly employ flooding-based search mechanisms to locate items (in the rest of this paper, we use the terms "file" and "item" interchangeably). Each query is tagged with a maximum *Time-To-Live* (TTL) to limit the number of hops it travels. In structured P2Ps, items are located through distributed hash table (DHT) interfaces. Recent studies show that flooding is effective for locating highly replicated items but less effective for rare items. In [15], Loo et al.'s experiments show that queries for rare items in Gnutella have very low recall rate. Around 18 percent of all Gnutella queries return no results, despite the fact that for at least two thirds of these queries, the desired results are available in the system. In addition, such queries often suffer long response time. On the contrary, DHTs guarantee perfect recalls and good response time for rare items, while incur significant bandwidth cost for popular item publishing and multiple keyword search [7].

Hybrid P2P networks have recently attracted much attention [14], [15], [29]. A hybrid P2P network combines an unstructured flooding-based network with a structured DHT-based global index [14], [15]. To make this concept clear, let us consider a simple example in our daily life. When we need an answer for an easy question, such as "what is the date today?" probably we can get it from most of the people around us. For a difficult question, such as "partial derivative," randomly asking people around us may not work, and we would achieve better (or quicker) results by looking it up in books in the library or consulting with experts. This scenario is similar to the selection of flooding and DHT in P2P networks. By combining an unstructured P2P approach with a structured DHT-based index, a hybrid P2P utilizes selective techniques that publish only rare items into the DHTs. Searches are performed either by flooding the unstructured network, or by looking up in the DHTs, according to the popularity of desired items. Hybrid search improves the recall rate and response time of queries for rare items with low bandwidth overhead, as well as maintaining good recall and response time for highly replicated items.

The key issue in improving hybrid search is how to better estimate the number of peers that can answer a question (query), so that we can determine the best search operation. Existing approaches for differentiating queries

- *H. Chen and H. Jin are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, P.R. China. E-mail: {chenhanhua, hjin}@hust.edu.cn.*
- *Y. Liu and L.M. Ni are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: {liu, ni}@cse.ust.hk.*

Manuscript received 22 June 2007; revised 24 Dec. 2007; accepted 16 Apr. 2008; published online 6 May 2008.
Recommended for acceptance by M. Parashar.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2007-06-0204.
Digital Object Identifier no. 10.1109/TPDS.2008.72.

fall into two categories. The first type is detection-based, for example the *simple hybrid strategy* [14], in which a search is first performed via flooding. If not enough results are being returned within a predefined time, the query is reissued as a DHT query. Although both popular and rare items can be ultimately located, locating rare items has a worse response time than pure DHTs, and extra cost is incurred by the preflooding. The other type of approaches use aggregation information gathered through a gossiping method to estimate the popularity of the items [29]. By using a threshold on the number of copies of related items to determine whether flooding or DHT, such approaches outperform simple hybrid techniques.

Nevertheless, differentiating among queries by estimating the number of items containing all the query terms has a fatal flaw. As identified by Sripanidkulchai et al. [22], the interest locality is a common and powerful principle of P2P file sharing systems. If a peer has a particular piece of content that a user is interested in, it is likely the peer has other items that user has interest as well. Hence, for a given query, when it is estimated that there are many related items in the network, it is not necessary that many peers have the desired files. Current approaches cannot distinguish whether the many items related to a query are distributed in many peers or are in a few peers, thus fail to provide a proper selection of flooding or DHT.

To address this issue, we propose QRank, a difficulty-aware hybrid search scheme, which employs a push synopsis based random gossiping algorithm [12], [16] to gather statistical information and term frequency. Using the rank values, QRank selects proper search strategies for queries. Trace-driven simulation results show that, compared with the copy popularity based technique, QRank significantly improves the query recall rate, as well as reduces the response time and traffic overhead.

The rest of this paper is organized as follows: Section 2 discusses the related work. Section 3 introduces the basic idea of QRank. Detailed design of QRank scheme is presented in Section 4. Section 5 describes how we collect a real trace as well as the simulation methodology. Performance evaluation of this design is presented in Section 6. We conclude the work in Section 7.

2 Related Work

Hybrid P2P model is first introduced by Loo et al. [15]. Having observed that the Gnutella network is inefficient for locating rare items, Loo et al. propose the SimpleHybrid architecture, combining Gnutella and DHT-based PIER [14]. SimpleHybrid search first performs flooding techniques with limited TTL. Queries that return no results within 30 seconds are reissued using the PIER search engine. Using this method, the recall rate for rare items is improved. The experiment results also show that PIER returns the first result within 10-12 seconds. Hence, SimpleHybrid approach reduces the average latency for rare items to 40-42 seconds, which is 65 seconds by flooding only. In addition, SimpleHybrid reduces the number of queries that receive no results in Gnutella by 18 percent. The drawback of SimpleHybrid is twofold. First, due to the preflooding operation, the response time for rare items (40-42 seconds) is much longer than that (10-12 seconds) directly using DHT. Second, the method incurs unnecessary bandwidth cost for rare items.

To better identify rare items, Zaharia and Keshav propose a gossip-based scheme called GAB [29]. GAB uses pushing synopsis based random gossip scheme to get global statistics on the number of copies. Search selection is based on the popularity of items. Every super peer maintains a synopsis for global item titles. In the synopsis, every distinct title uses a bit vector as a counter. The synopses are disseminated among super peers using a randomized gossip algorithm. When a query arrives, GAB sums up all the counts of titles that contain all the terms in the query. If the sum is above a specified threshold, the query will be flooded; otherwise, it will be looked up in the DHT. Compared with SimpleHybrid, GAB achieves a 10-20 percent higher recall rate, a 20-25 percent smaller response time, and a 45 percent reduction in the mean volume of query traffic. GAB, however, has a limited accuracy due to the fact that many titles containing the desired terms does not always mean that many peers can answer the query (we will explain in more detail in Section 3).

It is well known that weighting of terms provides improved retrieval performance because it differentiates useful terms from less useful ones, for example $TF \times IDF$ scheme [20]. To weight a term in a specific query, TF quantifies the fact that terms more frequently used in a query are likely important to describe the meaning of the query [20]. It is defined by

$$TF_t = 1 + \log_e f_{q,t}, \tag{1}$$

where $f_{q,t}$ denotes the frequency of term t in query q. Generally speaking, a verbose descriptive query can provide an indication of term importance [20].

Inverse document frequency (*IDF*) quantifies the fact that terms appearing in many documents in a collection are less important for differentiating the query:

$$IDF_t = \log_e\left(1 + \frac{N}{f_t}\right), \tag{2}$$

where N is the number of documents in the collection, and f_t denotes the number of documents in which term t appears.

The following equation is a standard form of $TF \times IDF$ to weight a term with the specified query:

$$\omega_t = TF_t \times IDF_t = (1 + \log_e f_{q,t}) \times \log_e\left(1 + \frac{N}{f_t}\right). \tag{3}$$

Such methods however are not directly applicable to P2P systems. First, in a P2P system, it is often difficult, if not impossible, to maintain a central repository or index, and large numbers of items are distributed over the entire network where nodes are joining and leaving in a dynamic and ad hoc manner. The absence of global statistical information hinders the use of *IDF* to differentiate terms in a query. Second, different from queries in traditional information retrieval application, P2P queries are typically short and when they are issued, all terms are usually used only once, offering less help. In order to address this issue, we propose QRank for hybrid P2P systems.

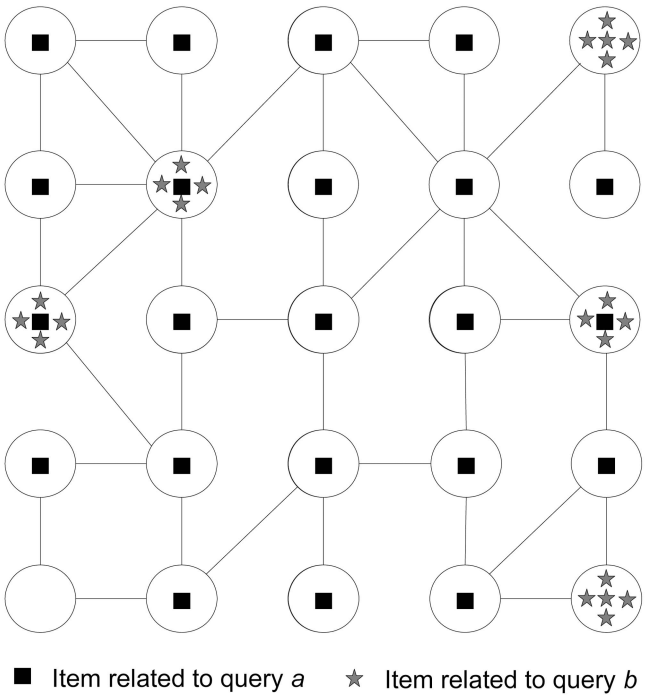

Fig. 1. "Many items" $\neq$ "many peers."

3 QRank Design

3.1 "Many Items" $\neq$ "Many Peers"

Current hybrid search selections are based on the assumption that when a query has many related items, these items are distributed among many peers. It is demonstrated in diverse traces of popular P2P systems [22]; nevertheless, P2P users prefer to store similar items of interest on their local disks. As shown in the example in Fig. 1, both queries a and b have a similar number of related items. However, since items related to query a are widely distributed, flooding will be more effective than a DHT for query a. On the other hand, a DHT is better for query b, as its results are available on only a few nodes in the network. Selecting search strategies on the number of related items often leads to degradation of query recall and an increase in search traffic.

QRank employs a ranking model by weighting the difficulty of queries and selects the best search strategy based on rank values. It is known that keywords in a query might be differently discriminative for searching. Considering the query "peer-to-peer network," more items and peers contain the keyword "network" than the keyword "peer-to-peer" because "network" is a less specific term than "peer-to-peer." But clearly, "peer-to-peer" is more important in this query.

3.2 Weighting Query Terms

In our query term weighting model, documents are identified by titles. Each title T contains a set of keywords: $T = (k_1, k_2, \ldots, k_n)$, and a query q is a sequence of terms, $q = (t_1, t_2, \ldots, t_m)$. QRank weighting model has two parameters, *Inverse Peer Frequency (IPF)* and *Term Frequency (TF)*. *IPF* is given by $IPF_t = \log_e(1 + \frac{N}{f_{t,p}})$, where N is the number of peers in the P2P network, and $f_{t,p}$ denotes the number of peers that have items containing term t. Terms that appear in more peers are less important and vice versa.

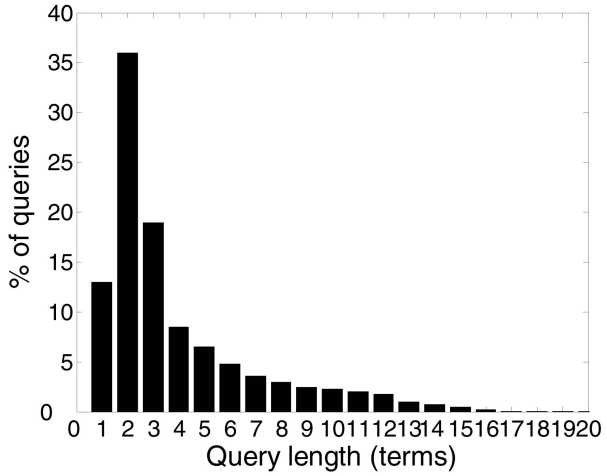

Fig. 2. Gnutella query length popularity.

IPF quantifies the fact that terms appearing in many peers in a P2P network are common terms and flooding is more feasible. We change *IDF* into *IPF* based on the observation that many documents containing t does not mean that many peers can answer the queries containing t. The value $f_{t,p}$ reflects the difficulty of a term t.

The challenging issue here is the parameter TF. The standard query term weighting method in traditional information retrieval field, $TF_t = 1 + \log_e f_{q,t}$, is not applicable here. We analyze the Gnutella query trace provided by Zeinalipour-Yazti [30]. We observe that the average length of Gnutella queries is 3.54 terms. Fig. 2 plots the query length distribution showing that 83 percent queries have no more than five terms. Hence, the variable $f_{q,t}$, the frequency that a keyword occurs in a query, is always 1 in the equation of TF_t, leading the TF_t constant.

To address this problem, QRank replaces TF_t with $apTF_t$, where $apTF_t = 1 + \alpha \log_e \frac{f_t}{f_{t,p}}$, f_t is the number of items whose titles contain t, and $f_t/f_{t,p}$ denotes the average frequency that the term t appears in a peer, while $\alpha > 0$ is the parameter scaling the contribution of $apTF_t$.

The formula of $apTF$ quantifies the degree of interest-based locality for terms. Given fixed value of f_t, the lower the value of $f_{t,p}$ is, the higher the degree of interest locality of the term t is. The terms highly replicated in few peers are difficult ones, and flooding is not appropriate for them.

Then, the QRank weight is given by

$$\omega_t = apTF_t \times IPF_t = \left(1 + \alpha \log_e \frac{f_t}{f_{t,p}}\right) \times \log_e\left(1 + \frac{N}{f_{t,p}}\right). \tag{4}$$

3.3 Search Selection

We define the difficulty of queries, ϖ_q, by

$$\varpi_q = \frac{\sum_{t \in q} \omega_t}{|q|}, \tag{5}$$

where $|q|$ denotes the length of query q.

We use the following σ_q to quantify by how far the weights of terms differ from ϖ_q:

$$\sigma_q = \frac{\sum_{t \in q} (\omega_t - \varpi_q)^2}{|q|}. \quad (6)$$

The lower the value of σ_q is, the slighter the difference of the terms is.

Intuitively, queries with popular keywords tend to be easier to be searched using flooding, while DHT search is more feasible for the queries consisting of rare terms due to the perfect recall and low bandwidth cost. If a query contains popular terms, DHT search may incur unacceptable bandwidth cost and long average result latency due to the distributed intersection operation [7]. On the other hand, interest-based locality of terms also affects search efficiency. Flooding is less suitable for the terms with higher degree of interest-based locality. Indeed, if the values of ϖ_q and σ_q are both low, the terms in the query are all easy and flooding technique is more feasible for such queries. If the value of ϖ_q is high and the value of σ_q is low, the terms in the query are all difficult and DHT search may be more feasible. When a query is issued, QRank selects flooding or DHT according to the rank values (ϖ_q, σ_q) of the query. As it is difficult for a system administrator to provide simple thresholds for both ϖ_q and σ_q, QRank uses a *Support Vector Machine* (SVM) [25] to classify a query into different types. In the following, we introduce how we obtain the classifier.

Based on the traces collected, a training set is formed and is denoted as $D = \{\overrightarrow{x_i}, y_i\}$, where $\overrightarrow{x_i}$ denotes the sample vector, and $y_i \in \{+1, -1\}$ is a class label. The input to the training algorithm is a set of examples x_i with labels y_i:

$$\begin{aligned} &\{(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)\}, \\ \text{where } &\begin{cases} y_i = +1, & \text{if } x_i \in class\ A, \\ y_i = -1, & \text{if } x_i \in class\ B. \end{cases} \end{aligned} \quad (7)$$

Considering the linearly separable data space, the aim of SVM is to find a decision function, an optimal separating hyperplane $w \cdot x + b = 0$ that separates the positive and negative data with the maximum margin.

The classifier can be written as

$$f(x) = \begin{cases} +1, & \text{if } w \cdot x + b \geq 0, \\ -1, & \text{if } w \cdot x + b < 0. \end{cases} \quad (8)$$

In the case of nonlinear space, SVM maps the input space to a high-dimensional space by kernels [25].

In order to label each example with best search type in the data set {(ϖ_q, σ_q, BestSearchType)} for training the search type classifier, where the label BestSearchType is either "Flooding" or "DHT," we define the metric vector $M = (m_1, m_2, m_3, \ldots, m_n)$. The metrics have two kinds. Some of the metrics such as recall could be positive, i.e., the higher the value, the higher the quality. Other metrics are negative, i.e., the higher the value, the lower the quality. To normalize the metric values for queries, we scale positive metrics according to (9). For negative metrics, their values are scaled according to (10):

TABLE 1
Classification of Experimental Results

Search Type	Precision	Recall	F-Measure
Flooding	0.97	0.891	0.929
DHT	0.868	0.963	0.913

$$V_{i,j} = \begin{cases} \frac{m_{i,j} - m_i^{\min}}{m_i^{\max} - m_i^{\min}}, & \text{if } m_i^{\max} - m_i^{\min} \neq 0, \\ 1, & \text{if } m_i^{\max} - m_i^{\min} = 0, \end{cases} \quad (9)$$

$$V_{i,j} = \begin{cases} \frac{m_i^{\max} - m_{i,j}}{m_i^{\max} - m_i^{\min}}, & \text{if } m_i^{\max} - m_i^{\min} \neq 0, \\ 1, & \text{if } m_i^{\max} - m_i^{\min} = 0, \end{cases} \quad (10)$$

where $m_{i,j}$ denotes the value of the ith metric of query q_j, and $V_{i,j}$ denotes the normalized value of $m_{i,j}$. Then, we define the utility function as

$$Utility(q_j) = \sum\nolimits_i (V_{i,j} \times W_i), \quad (11)$$

where $W_i \in [0, 1]$ and $\Sigma_i W_i = 1$. W_i is a normalized weight given by users to represent the importance of metric m_i.

Although this method can be easily extended to more metrics, we consider three metrics (m_1, m_2, m_3) as follows:

- *Recall*: percentage of relevant items returned as results for a query. It is a widely accepted standard metric in the information retrieval research area [10]:

$$Recall = \frac{\#\text{ of relevant items returned}}{\text{total }\#\text{ of relevant items in the network}}. \quad (12)$$

- *Latency*: average latency of query results:

$$Latency = \frac{\text{sum of latencies of all the results for a query}}{\text{total }\#\text{ of results for a query}}. \quad (13)$$

- *Traffic*: bandwidth cost for searching a query.

Thus, the utility function for query q_j is given by

$$Utility(q_j) = \frac{m_{1,j} - m_1^{\min}}{m_1^{\max} - m_1^{\min}} \times W_1 + \sum_{i=2}^{3} \left(\frac{m_i^{\max} - m_{i,j}}{m_i^{\max} - m_i^{\min}} \times Wi \right). \quad (14)$$

Using the set of Gnutella query logs, we randomly extracted 6,759 examples as a training data set to train a classifier. Each example is represented as (ϖ_q, σ_q, BestSearchType), where the label BestSearchType is either "Flooding" or "DHT," whichever has the better value of utility function. The accuracy of an SVM classifier is quite critical for the efficiency of search strategy selection. In order to get a good classifier, QRank varies α, the parameter scaling the contribution of $apTF_t$ in (4), and tests several SVM kernels, including linear, polynomial, and Gaussian. We use cross-validation technique [25] for training and testing. Table 1 summarizes the experimental results of the trained classifier.

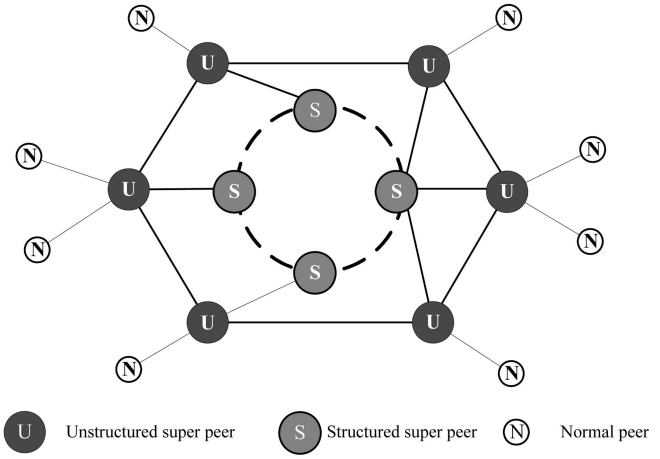

Fig. 3. QRank hybrid P2P network.

The results show that the trained classifier has good classification Precision and Recall [25] for both search types. Here, recall is the fraction of the correctly classified results to the total results in the set, while precision is the fraction of the correctly classified results to the total number of returned results. We use the F-Measure [25], which was computed by $F\text{-Measure} = (2 \times \text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})$, to measure the overall performance of the trained classifier. The result shows that the trained classifier performed well. The trained classifier is employed for QRank search type selection.

For a real-world system design, we need to conduct experiments with large-scale data sets, for example, the TREC WT100G [3], a 100-Gigabyte text collection on an open platform, PlanetLab. By using different network metrics obtained from the PlanetLab, such as end-to-end delay, end-to-end bandwidth, and so forth, we are able to successfully collect the necessary training data set. As shown in (4), QRank's weighting model is not dependent on the scale of network size and the absolute value of term frequency. Later in Section 6, we will demonstrate this point in the different simulation runs. Thus, the training data collected from PlanetLab should be applicable for a real-word system design.

4 Hybrid Search with QRank

We first give an overview of QRank hybrid search and then describe the method of collecting global information in distributed P2P networks in detail. We will also discuss how to improve the search performance by using adaptive methods.

4.1 QRank Hybrid Search

QRank defines four types of nodes [29]: unstructured super peers, structured super peers, normal peers, and bootstrap peers, as illustrated in Fig. 3. A normal peer connects to at least one unstructured super peer via bootstrap peers. An unstructured super peer maintains a local index of items from the normal peers connected to it and performs query search on their behalf. Every QRank unstructured super peer connects to one or more structured super peer in order to enable both flooding and DHT search in each unstructured super peer. To reduce the overhead of DHT maintenance, only a small fraction of server-like nodes are selected to act as structured super peers and participate in the global DHT in the P2P network. The DHT allows every individual keyword to be mapped to documents across the network. Using this single-term based global index, a list of entries for each keyword in a query can be retrieved. To decrease the overhead associated with item publishing, QRank only publishes items with rare keywords into the global index. Titles are decomposed into multiple keywords and separately published into the global single-term based inverted index.

A query issued by a normal peer is first sent to a connected unstructured super peer. The unstructured super peer computes ϖ_q, σ_q as the input of the embedded QRank classifier using the global statistical information of f_t, $f_{t,p}$, and N. The query is then checked by QRank classifier to select flooding or DHT. If flooding is selected, the query is inserted into the unstructured network with a given TTL. Otherwise, the keywords of the query are looked up in the global DHT separately via the connected DHT nodes, with a consequent intersection operation for the retrieved posting lists. The SVM classifier can effectively identify the queries with high ϖ_q and low σ_q and select DHT search technique for them. For such queries, DHT nodes need to transmit short posting list across wide area networks. Thus, the QRank difficult-aware hybrid search scheme can efficiently reduce bandwidth cost for multikeyword search. Based on the DHT global index, we have further optimized the bandwidth cost of multikeyword search using techniques such as bloom filter in our previous work [7].

4.2 Global Information Collection

QRank gathers the global statistics in the network by using a variation of the pushing synopsis based gossip algorithm first proposed in [16]. It is a combination of the gossip-style computation [12] and duplicate-insensitive probabilistic counting [9], [29]. Such gossip scheme causes the computation of aggregation information to converge exponentially [12]. After $\log(n)$ rounds of gossip, where n is the maximum number of all the nodes involved in the gossip, all the nodes will get the global statistics. Within the structure of hybrid P2P network, the algorithm enables every super peer to quickly collect the statistics, including f_t, $f_{t,p}$, and N in the query ranking model, as described in Section 3.2.

4.3 Adaptive Hybrid Search

Due to the dynamic property of P2P networks, the accuracy of QRank is influenced by the uptime of a peer. In adaptive search, when a new query is issued to an unstructured super peer, the peer first asks its neighboring super peers. If all the neighbors agree on the same search type, the agreed search type will be performed. Otherwise, the decision of the peer with longest uptime will be used. The maximum uptime $Uptime_{max}$ of neighbors will be tagged in the query and forwarded together with the query.

During flooding, when a query $(q, Uptime_{max})$ is forwarded to a super peer, the peer will compare $Uptime_{max}$ tagged with the query and its own uptime

$Uptime_{local}$. If $Uptime_{local} > Uptime_{max}$ and the local classifier suggests the query should be looked in DHT, the flooding based forward will be stopped by setting the TTL value of this message to "0" and a consequent DHT lookup will be performed. In order to prevent looking up DHT multiple times, this DHT lookup operation will be discussed with the source peer. By this adaptive hybrid search strategy, more accurate decisions can be made to further improve search performance.

5 Simulation Methodology

5.1 Gnutella Trace Collection

The query traces from *Zeinalipour-Yazti* are quite representative for real-world P2P systems. However, we found that the traces are not complete enough for obtaining complete topology information. So, we collect the topology information of Gnutella using the crawler we developed.

As mentioned in the Gnutella protocol, individual nodes, also called servants, perform tasks normally associated with both clients and servers. They provide client-side interfaces through which users can issue queries and view search results. At the same time, they also accept queries from other servants and respond with applicable results.

Originally, all Gnutella peers are connected with each other randomly, which caused the scalability problem in the Gnutella network. Current Gnutella categorizes peers as leaves and super peers. A leaf keeps only a small number of connections to super peers. A super peer behaves as a proxy for the leaves connected to it.

According to the Gnutella protocol, a ping message with $\mathrm{TTL} = 2$ and $\mathrm{HOP} = 0$ is regarded as a crawler ping, and peers, upon receiving a crawler ping, should respond with appropriate pong messages. Based on this mechanism, our crawler discovers the Gnutella topology by performing a breadth first search on the network. From our experience and observations, we find that some clients such as Gnucleus and Morpheus (based on GnucDNA) do not respond to the crawler ping appropriately. Fortunately, these clients send an information page summarizing servants' status to any web browser trying to connect to it. Motivated by this, we also developed a web spider as a means of collecting topology information from these clients. We then integrated the web spider into the crawler, which accelerated the crawling process remarkably. The crawlers are written in Java based on Limewire's [2] open source client and run in parallel using 40 threads. Our crawler can explore more than 50,000 peers within half an hour.

Using the crawler, we crawled seven topologies with sizes of 31,747, 34,206, 45,650, 48,134, 57,926, 68,737, and 75,643 nodes. In Fig. 4, we plot the distribution of the node degree corresponding to the collected Gnutella topology traces. The plots of all data sets are in good agreement with previous results [19].

5.2 Simulator Design

We wrote a custom simulator in Java to compare QRank with the GAB search technique. We first generate the underlying network topology. Based on generated network, we simulate hybrid P2P overlay and the search techniques.

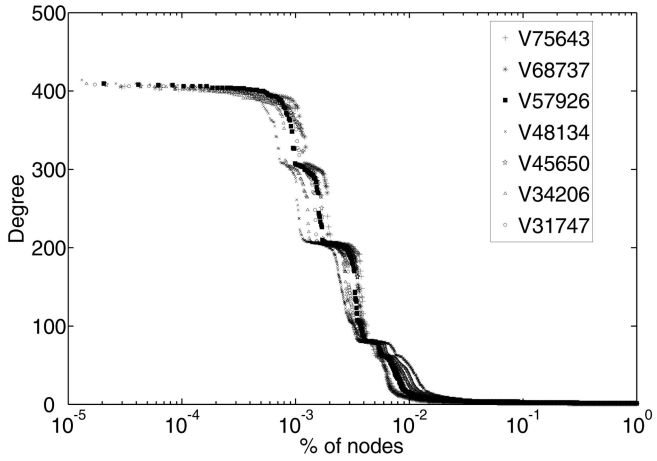

Fig. 4. Gnutella trace.

5.2.1 Hybrid P2P Overlay

In order to better represent real-world systems, we consider both the underlying physical topology and the P2P overlay.

The physical topology should represent the real topology with Internet characteristics. Previous studies have shown that large-scale Internet physical topology follows the small world and power law properties [24]. Power law describes the node degree, while the small world describes characteristics of path length and clustering coefficient. The study in [4] found that the topologies generated using the AS Model have the properties of the small world and power law. BRITE [1] is a topology generation tool that provides the option of generating topologies based on the AS Model. Using BRITE, we generate a physical topology with 100,000 nodes.

We use the Gnutella traces we collected to simulate the P2P overlay, where all P2P nodes are mapped into the underlying physical topology. The communication cost between two logical neighbors is calculated based on the physical shortest path between the pair of nodes. Using BRITE, we can simulate the underlying Internet with rich configuration information, including bandwidth configuration, latency, and so forth. The ultra peers of the Gnutella trace are nominated as unstructured super peers in our simulator. The uptime of peers follows the distribution of Gnutella P2P systems reported in [21]. About 10 percent ultra peers have an average uptime longer than 80 minutes and we nominate such peers as DHT nodes. (In practice, QRank can classify the structured super peers according to the characteristics such as uptime history age (how long the peer used to be online on average per minute), history role the peer used to play, bandwidth configuration, and how long the peer has been alive [29].) Each peer is assigned a lifetime in seconds. We simulate the joining and leaving behavior of peers via turning on/off peers. The lifetime is decreased by one after each passing second. A peer leaves in the next second when its lifetime reaches zero. During each second, there are a number of peers leaving the system. We then randomly pick up (turn on) the same number of peers from the physical network to join the overlay.

We simulate the flooding search technique among the unstructured super peers and Chord [23] DHT in structured super peers. The TTL for the flooding search is set to 7, as it is found in [19] that 95 percent of any two Gnutella node pairs could exchange messages in fewer than seven hops. A DHT search is performed by searching each keyword with a consequent join operation. In the simulation, the distributed intersection for multikeyword searching is performed by transmitting the posting list for a rarer keyword to the DHT nodes that are responsible for other more popular keywords.

5.2.2 Queries and Files

In order to compute the query popularity, we use the Gnutella query trace, which we analyzed in Section 3. The trace is quite representative for real-world P2P systems.

We simulate files with the file names generated from exact queries. The file distribution is computed according to the study result in [8] that files in P2P network exhibits popularity characteristics that fit a log-quadratic distribution $y = 10^{-2.98x^2-0.68-0.07}$, where x is the percentage of unique files ranked in log scale, and y is the percentage of all files in log scale. All data sets were processed with Porter stemming algorithm to reduce morphological variants for a given term to the common base form of it. Common stop words such as "the," "and," and so forth were removed from the data set [10].

5.2.3 Gathering Global Statistics

To gather global statistics, we simulate the push synopsis based randomized gossip method. We also implement the $apTF \times IPF$ ranking model. QRank leverages the fact that the global statistics are slowly changing in the large-scale P2P networks. Hence, infrequent computation of these statistics is sufficient for good performance. The communication cost for the gossiping algorithm is quite acceptable for a real-world system.

5.2.4 Classifier Training

To obtain the training query set, we search each query using both flooding and DHT strategies and compute the utility of both strategies for the query. Thus, each sample of a query can be represented as $(\varpi_q, \sigma_q, label)$, where the label is either "Flooding" or "DHT," whichever has the better value for the utility function defined as (14) in Section 3. We trained a classifier using Weka 3 [27] and use the classifier to predict which strategy should be used for any given query in the simulation.

In Section 6, we will show that the performance of the trained SVM classifier is quite stable when we change the distribution of the underlying data set. Generally, the overall situation including the system scale, the principle of interest-based locality, and even the global statistics changes slowly, so using SVM classifier is quite efficient for search strategy selection. Although there is overhead associated with training a classifier, we believe such a cost is acceptable compared with the overhead associated with adaptively adjusting a threshold like GAB or predetecting the P2P network using flooding like SimpleHybrid.

For each simulation, we take 50 runs and report the average or the representatives.

6 Performance Evaluation

QRank considers both search quality and efficiency. Quality focuses on user-perceived qualities, such as the number of returned results, recall, and search latency, while efficiency focuses on resource utilization, such as bandwidth cost.

For simulating the GAB and QRank algorithms, we use the utility function defined in Section 3.3, which mainly considers three normalized metrics: *recall*, *latency*, and *traffic*. In the utility function, we set the weight W_i to 1/3 for $1 \leq i \leq 3$ to show the same importance of each metric m_i. The different metrics are all normalized into the space [0, 1] by (9) and (10). GAB's utility function shows that its performance is largely determined by the value of the specified R_{max}. Choosing proper R_{max} value for different network sizes and document collection sizes is not trivial because the same R_{max} value may not work in different systems. For example, in GAB's simulation, R_{max} is set to 25, while due to the partial matching it is likely that in a larger scale network with much more documents, a user may not obtain what it really desires although the system returns 25 items with titles containing all the keywords in the query (this is why a real-world search engine needs to rank the results). Thus, in our testbed, we simply use the same parameter setting of R_{max} as GAB's simulation could be not feasible. GAB paper has not discussed about how to better choose R_{max} in different network sizes and different scales of document collection. Thus, in the utility function proposed in Section 3.3, we do not separately consider R_{max}. Instead, we use *recall*, a more widely accepted standard metric in the information retrieval research area, which reflects the fraction of relevant returned items. We believe this comparison is fair since both algorithms are simulated based on the same utility function.

GAB's adaptive thresholding algorithm can help a real-world system achieve the optimal value for t, where flooding and DHT search provide equal utility. In our preliminary research [6], we found that the optimal value for t is crucial to search performance. In this extended simulation, in order to bound the performance improvement of our scheme over GAB, we use much stricter methodology to obtain the optimal values of t for GAB. We first let the synopsis-based gossip algorithm to converge to the statistics. Then, we randomly choose a set of unstructured super peers and let them issue a certain number of queries. Each query is performed by both flooding and DHT lookup. We compute the t values for the queries that have equal utility. In the simulation, we use an average value of all such t values as a threshold for search selection for GAB. Our simulator periodically performs the above steps to achieve new optimal thresholds and finds that the values change very slightly during the simulation. With this simulation scheme, we can successfully get the optimal threshold that GAB's adaptive algorithm aims to achieve. Thus, our simulation can obtain almost the upper bound performance of GAB. We believe such comparison is quite fair.

Fig. 5 plots the number of returned results for all queries, in which 77 percent GAB queries return 100 or

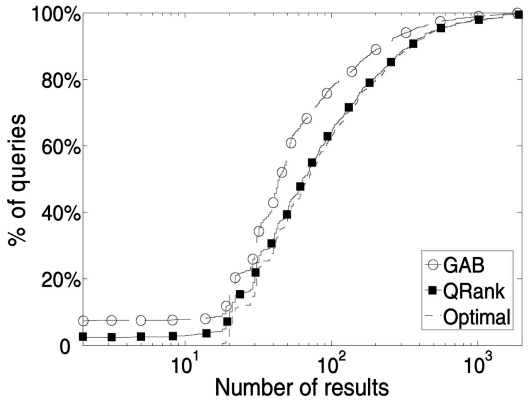

Fig. 5. Number of results.

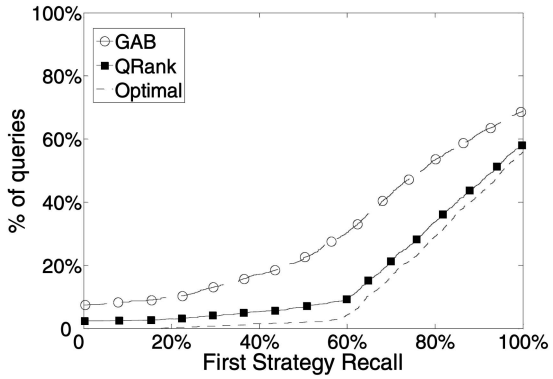

Fig. 7. FSR.

fewer results, while 62 percent QRank queries return less than 100 results, showing that QRank outperforms GAB for highly replicated items. The reason for the increase is that the queries for searching the items with strongly interest-based locality distribution can be identified as difficult queries in our ranking model. The SVM classifier may choose DHT-based search instead of flooding.

Fig. 5 also shows that about 7 percent GAB queries return no results, while only 2.5 percent QRank queries return nothing. It shows that both GAB and QRank are efficient for the queries with rare related items. The reason why more GAB queries return nothing is because using flooding for the queries with many related items may be inefficient. When these many items may be distributed only on a few peers in the network, GAB may have lower hit rate for such queries by flooding.

Fig. 6 shows the average number of query results in all the 50 simulation runs. Statistically, the average number of query results is improved by 48 percent.

We also conduct experiments for the optimal search policy. To achieve the optimal search policy, each query is performed with both DHT and flooding. We use the strategy with better utility as an optimal strategy. Figs. 5 and 6 also plot how well the optimal search policy can do. We can see that the performance of QRank is very close to the optimal selection.

Non-detection-based hybrid search schemes aim at improving the performance of the first strategy selected for a query. In this simulation, both QRank and GAB search a query either by flooding or DHT, without combining the detection-based strategy (such as SimpleHybrid).

We evaluated the *First Strategy Recall (FSR)* of queries. The *FSR* denotes the percentage of returned relevant items out of all relevant items in the network as defined in (12) using the first strategy selected for a query.

Fig. 7 plots the *FSR* of all queries where 53 percent queries using GAB have 80 percent or lower recall. By using QRank, less than 32 percent queries have a recall rate less than 80 percent. Since both GAB and QRank use DHT for the queries with rare related items, they are both efficient for such queries. The lower recall for GAB is because using flooding for the queries with many related items may be inefficient when these many items may not be widely distributed in the network. QRank achieves a better recall by using DHT searches for such queries.

Fig. 8 plots the statistical average query *FSR* in each run. The statistical average query *FSR* is increased by 21 percent, which means that 82 percent of the potential improvement can be achieved by the optimal search selection.

Short search latency is always desirable in P2P systems. In Figs. 9 and 10, we evaluate the average latency of all the results of a query as defined in (13). Fig. 9 shows that about 45 percent QRank queries have short average result latency less than 7 seconds, while only about 29 percent GAB

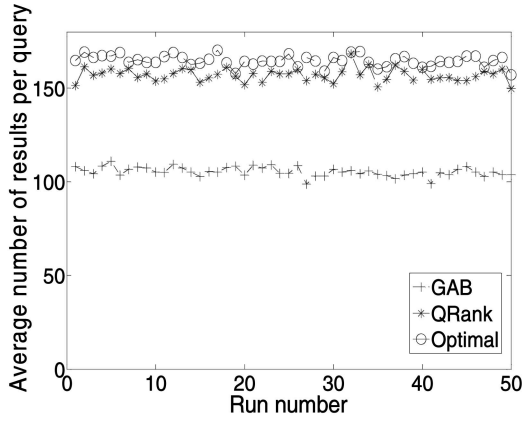

Fig. 6. Average number of results.

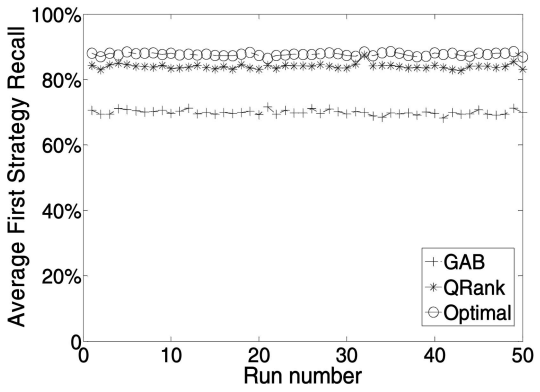

Fig. 8. Average recall rate.

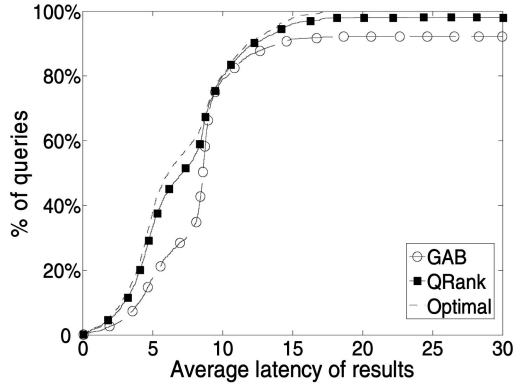

Fig. 9. Average result latency.

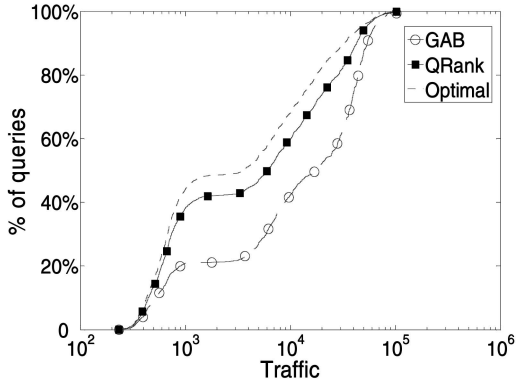

Fig. 11. Traffic cost.

queries achieve such short latency. In the simulation, we set a maximal *Timeout* value of 30 seconds for the queries. A query returns failure with a latency of 30 seconds. Fig. 9 shows that GAB has more queries that return no results before *Timeout*.

Fig. 10 plots the statistical average query result latency in each run. The statistical average latency of results is significantly reduced by 26 percent, which is 76 percent of the potential improvement that can be achieved with an optimal search selection. We also noticed in Fig. 9 that a number of queries have average result response times with only slight differences. This is because the latency of DHT is decided by the Chord latency, $\frac{1}{2}\log N$, where N is the total number of the structured super peers in the hybrid P2P network.

P2P traffic has significant impact on the underlying network. Heavy network traffic limits the scalability of P2P networks.

We define the traffic cost as network resource used in a search process of P2P systems, which is mainly a function of consumed network bandwidth and other related expenses. Specifically, we assume that all the messages have the same length. When messages traverse an overlay connection during a given time period, the traffic is the summed traffic cost of all the hops. The traffic cost of a hop is given by $Tc = M\sum_i L_i/B_i$ [13], where M is the size of the message, and L_i and B_i, respectively, represent the length and the bandwidth of the physical links that this message traverses on the underlying physical network during this hop in the overlay.

Fig. 11 plots *traffic cost* of all the queries. About 73 percent of QRank queries have a traffic cost less than 2×10^4, while only about 52 percent GAB queries achieve 2×10^4 or less traffic cost. Fig. 12 plots the statistics. The average query traffic is decreased by 40 percent, which is 75 percent of the potential improvement that can be achieved with an optimal search selection.

To better evaluate the overall performance of QRank, we also define two comprehensive metrics: *Search Efficiency* and *Search Utility*. *Search Efficiency* is defined as the ratio of *FSR* to search *traffic cost*, $Efficiency = FSR/traffic\ cost$. It is a more fair comparison than looking at *FSR* and search *traffic cost* separately [26]. In P2P search, we often desire to achieve higher recall with lower traffic cost.

Figs. 13 and 14 show the contrast of *Search Efficiency* among QRank, GAB, and the optimal. We can see that QRank greatly outperforms GAB. The statistical average search efficiency is increased by 61 percent, which is 73 percent of the potential improvement that can be achieved with an optimal search selection.

Search Utility, as defined in Section 3, considers multiple metrics, including *average latency of results*, *FSR*, and *traffic cost*. As shown in Figs. 15 and 16, QRank improves *Search Utility* compared with GAB.

In the simulation, we find it difficult to model the interest-based locality. We obtain some observation from the SemreX system [11], which is a P2P system for computer

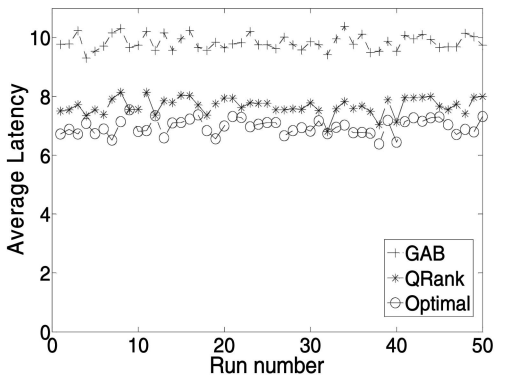

Fig. 10. Statistical average latency.

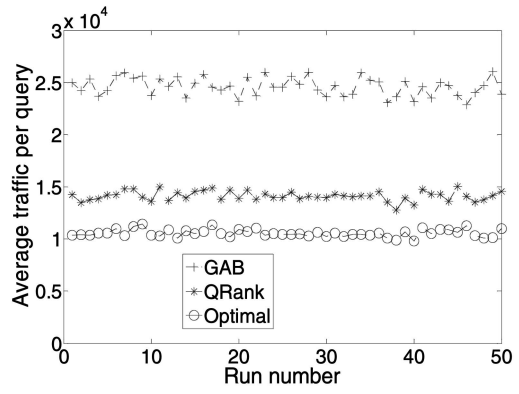

Fig. 12. Average traffic cost.

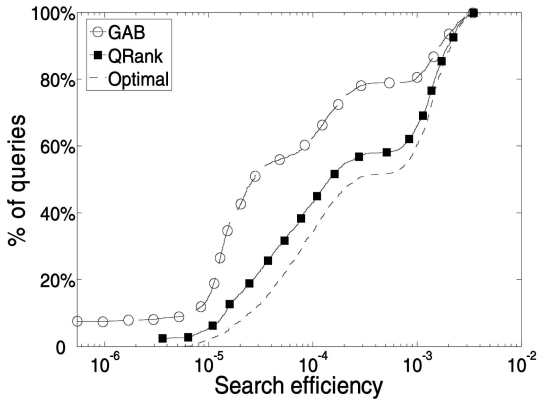

Fig. 13. Search efficiency.

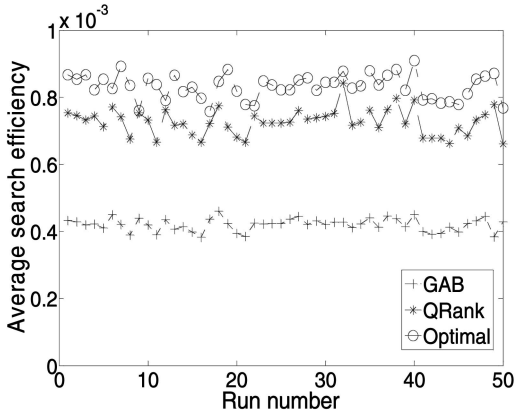

Fig. 14. Average search efficiency.

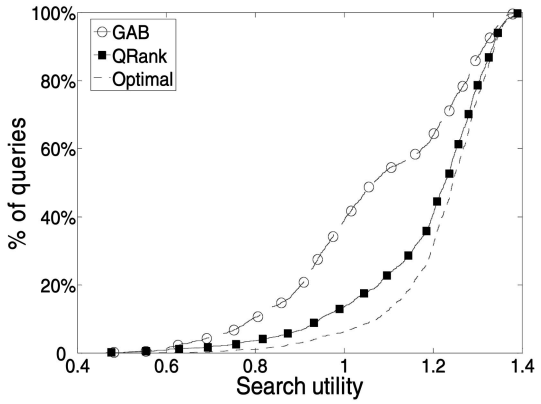

Fig. 15. Utility.

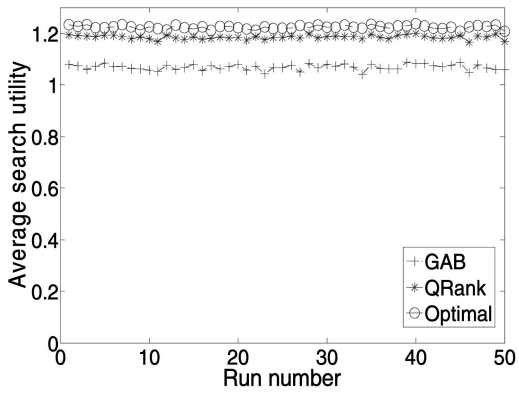

Fig. 16. Average utility.

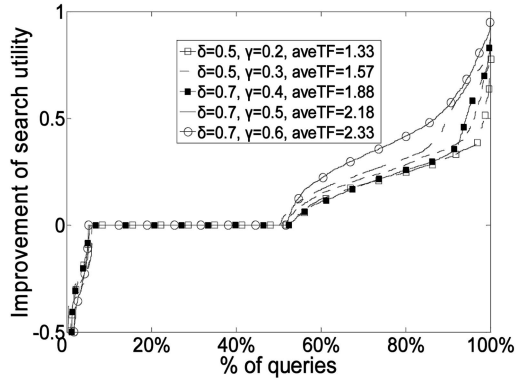

Fig. 17. Utility improvement versus change of term frequency.

science researchers to share literatures. The system is developed by the Cluster and Grid Computing Laboratory, Huazhong University of Science and Technology in the summer of 2005. It shows that a large fraction of researchers tend to store documents of topics about their research interest on their hard disks. In such peers, a large fraction of pdf files belong to only a small number of topics and it is very common that the files belonging to the same research topic may have titles containing common keywords. Based on the observation, we simulate the distribution of the files in our design.

To better examine the effect of interest locality, in the simulation for this journal version, we change the degree of interest-based locality by varying δ, the fraction of the peers that follow the locality principle, and the parameter γ, the fraction of documents that have similar titles in such peers. The parameter δ ranges from 50 percent to 70 percent, and γ ranges from 20 percent to 60 percent. It is difficult to simulate partial match, so we use the same title for simulating different documents. That is to say, even there are two same titles, they are different items. We believe such a method can simulate the interest-based locality to some extent.

As aforementioned, we can quantify the degree of interest-based locality using $apTF$. We compute $aveTF = \frac{1}{n}\sum_t apTF_t$, where n is the number of all the terms in the synopsis to quantify the overall degree of interest-based locality in the P2P network. Figs. 17 and 18 analyze the impact of the interest locality on the improvement of search performance. We compute QRank's improvement over GAB for each query by $\Delta U = Utility_{QRank} - Utility_{GAB}$ and plot the results in Fig. 17. We can see that only about 5 percent of queries with QRank have a worse search utility than using GAB, while about 56 percent of queries have better utility. It is also shown that ΔU is greatly influenced by the average $apTF$ value. When $apTF$ increases, QRank achieves a better ΔU. This shows that QRank can achieve more improvement when the degree of interest-based locality increases.

Similarly, in Fig. 18, QRank's improvement of search efficiency is influenced by the $apTF$ of terms. About 40 percent of QRank queries have better search efficiency than that of GAB.

As GAB's paper discussed [29], we estimate the synopsis size for the gossip algorithm in our system as follows: The bit

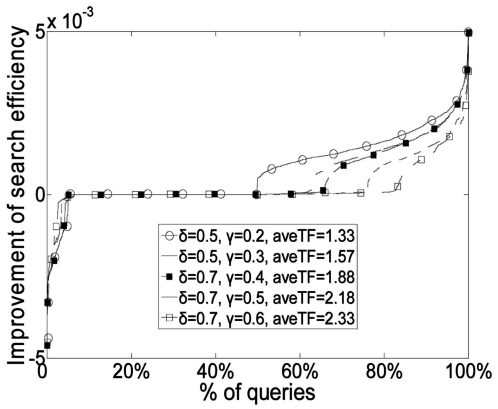

Fig. 18. Efficiency improvement versus change of term frequency.

vector for each unique keyword takes 4 bytes. The average English word length is five. Thus, each term will take an average number of 9 bytes in the synopsis. There are about 615,100 distinct terms in the Oxford dictionary. Hence, if each peer maintains an original full word list, it needs a storage size of 5.28 Mbytes for gathering the complete global statistics. The Burrows-Wheeler Transform can reduce the storage size to 1.78 Mbytes, which is an upper bound of the storage. According to Zipf's law of words, which has been demonstrated in NLP research [17], we can reduce a large fraction of uncommon words and significantly reduce the size of the synopsis much lower than the upper bound. On the other hand, due to the fact that the global statistics in a large-scale network are slowly changing, infrequent computation of those statistics is sufficient. Therefore, the communication cost for the gossiping algorithm is acceptable for a real-world system.

7 Conclusions and Future Work

Hybrid search provides better efficiency for P2P systems. To further improve its performance, we need to better estimate how many peers can answer a given query so as to determine a proper search strategy for the query. To address this problem, SimpleHybrid uses a detection-based algorithm to estimate the item popularity. GAB improves the search performance by using a non-detection-based scheme by employing a synopsis gossip algorithm to estimate the number of items that match a given query. We find that many items matching a query does not always mean that flooding is more efficient. To address the issue, we have proposed QRank, a query difficulty-aware scheme, which ranks the query difficulty based on global statistical information and term frequency. QRank uses SVM classifier to select a proper search strategy. We collect real P2P traces and design a trace-driven simulator to evaluate this design. Results show that QRank outperforms GAB in terms of search efficiency and quality.

In the future, we will implement our difficulty-aware hybrid search strategy on some open platforms for a real-world system design. We will also address the transfer learning issue of SVM-based algorithm for QRank.

Acknowledgments

This work was supported by the National Science Foundation of China (NSFC) under Grant 60433040, NSFC/RGC Joint Research Foundation under Grant 60731160630, the National 973 Key Basic Research Program under Grant 2003CB317003, and Hong Kong RGC Grants N_HKUST614/07 and HKUST6152/06E. The preliminary result of this paper was published in Proceedings of the International Conference on Parallel Processing (ICPP) 2007, where it was nominated as best paper award candidate.

References

[1] *BRITE,* http://www.cs.bu.edu/brite, 2008.
[2] *Limewire,* http://www.limewire.com, 2008.
[3] *WT100G Test Collection,* http://ir.dcs.gla.ac.uk/test_collections/, 2008.
[4] T. Bu and D. Towsley, "On Distinguishing between Internet Power Law Topology Generators," *Proc. IEEE INFOCOM,* 2002.
[5] Y. Chawathe, S. Ratnasamy, and L. Breslau, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM,* 2003.
[6] H. Chen, H. Jin, Y. Liu, and L.M. Ni, "Difficulty-Aware Hybrid Search in Peer-to-Peer Networks," *Proc. Int'l Conf. Parallel Processing (ICPP),* 2007.
[7] H. Chen, H. Jin, J. Wang, L. Chen, Y. Liu, and L.M. Ni, "Efficient Multi-Keyword Search over P2P Web," *Proc. 17th Int'l World Wide Web Conf. (WWW),* 2008.
[8] J. Chu, K. Labonte, and B.N. Levine, "Availability and Locality Measurements of Peer-to-Peer File Systems," *Proc. ITCom: Scalability and Traffic Control in IP Networks,* 2002.
[9] P. Flajolet and G.N. Martin, "Probabilistic Counting Algorithms for Data Base Applications," *J. Computer and System Sciences,* vol. 31, pp. 182-209, 1985.
[10] D. Hawking, N. Craswell, P. Bailey, and K. Griffihs, "Measuring Search Engine Quality," *Information Retrieval,* vol. 4, no. 1, pp. 33-59, 2001.
[11] H. Jin and H. Chen, "SemreX: Efficient Search in Semantic Overlay for Literature Retrieval," *Future Generation Computer Systems,* vol. 24, no. 6, pp. 475-488, 2008.
[12] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregation Information," *Proc. 44th Ann. IEEE Symp. Foundations of Computer Science (FOCS),* 2003.
[13] Y. Liu, X. Liu, L. Xiao, L.M. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems," *Proc. IEEE INFOCOM,* 2004.
[14] B.T. Loo, J.M. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica, "Enhancing P2P File-Sharing with an Internet-Scale Query Processor," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB),* 2004.
[15] B.T. Loo, R. Huebsch, I. Stoica, and J.M. Hellerstein, "The Case for a Hybrid P2P Search Infrastructure," *Proc. Third Int'l Workshop Peer-to-Peer Systems (IPTPS),* 2004.
[16] S. Nath, P.B. Gibbons, S. Seshan, and Z.R. Anderson, "Synopsis Diffusion for Robust Aggregation in Sensor Networks," *Proc. Second ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys),* 2004.
[17] A. Parker-Rhodes and T. Joyce, "A Theory of Word-Distribution Frequency," *Nature,* vol. 178, p. 1308, 1956.
[18] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," *Proc. ACM SIGCOMM,* 2004.
[19] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing,* vol. 6, no. 1, pp. 50-57, 2002.
[20] G. Salton and C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management,* vol. 24, pp. 513-523, 1988.
[21] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking (MMCN),* 2002.
[22] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," *Proc. IEEE INFOCOM,* 2003.

[23] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM,* 2001.
[24] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based versus Structural," *Proc. ACM SIGCOMM,* 2002.
[25] V. Vapnik, *The Nature of Statistical Learning Theory.* Springer, 1999.
[26] H. Wang and T. Lin, "On Efficiency in Searching Networks," *Proc. IEEE INFOCOM,* 2005.
[27] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques,* second ed. Morgan Kaufmann, 2005.
[28] L. Xiao, Z. Xu, and X. Zhang, "Low-Cost and Reliable Mutual Anonymity Protocols in Peer-to-Peer Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 9, pp. 829-840, 2003.
[29] M. Zaharia and S. Keshav, "Gossip-Based Search Selection in Hybrid Peer-to-Peer Networks," *Proc. Fifth Int'l Workshop Peer-to-Peer Systems (IPTPS),* 2006.
[30] D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos, "Exploiting Locality for Scalable Information Retrieval in Peer-to-Peer Networks," *Information Systems,* vol. 30, pp. 277-298, 2005.

Hanhua Chen is a PhD candidate at Huazhong University of Science and Technology (HUST), Wuhan, China. His research interests include peer-to-peer computing, grid computing, information retrieval, and wireless sensor network. He was awarded the Microsoft fellowship in 2005. He is a member of the IEEE and the IEEE Computer Society.

Hai Jin received the PhD degree in computer engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1994. He is a professor of computer science and engineering and the dean of the School of Computer Science and Technology, HUST. In 1996, he was awarded a German Academic Exchange Service Fellowship to visit the Technical University of Chemnitz, Chemnitz, Germany. He worked at the University of Hong Kong between 1998 and 2000 and was a visiting scholar at the University of Southern California between 1999 and 2000. He is the chief scientist of ChinaGrid, the largest grid computing project in China. His research interests include computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security. He has coauthored 15 books and published more than 300 research papers. He is the steering committee chair of the International Conference on Grid and Pervasive Computing (GPC) and Asia-Pacific Services Computing Conference (APSCC). He is a member of the steering committee of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), the IFIP International Conference on Network and Parallel Computing (NPC), the International Conference on Grid and Cooperative Computing (GCC), the International Conference on Autonomic and Trusted Computing (ATC), and the International Conference on Ubiquitous Intelligence and Computing (UIC). He was awarded an Excellent Youth Award from the National Science Foundation of China in 2001. He is the member of Grid Forum Steering Group (GFSG). He is a senior member of the IEEE and a member of the ACM.

Yunhao Liu received the BS degree in automation from Tsinghua University, Beijing, in 1995, the MA degree from Beijing Foreign Studies University, Beijing, in 1997, and the MS and PhD degrees in computer science and engineering from Michigan State University in 2003 and 2004, respectively. He is currently with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He is also an adjunct professor of Xi'an Jiaotong University and the Ocean University of China. His research interests include peer-to-peer computing, wireless sensor network, and pervasive computing. Together with his student Li Mo, he received the Hong Kong ICT Best Innovation and Research Grand Award in 2007. He is a senior member of the IEEE and a member of the ACM.

Lionel M. Ni received the PhD degree in electrical and computer engineering from Purdue University, West Lafayette, Indiana, in 1980. He is a chair professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST). He is also the director of the HKUST China Ministry of Education/Microsoft Research Asia IT Key Laboratory and the director of the HKUST Digital Life Research Center. His research interests include parallel architectures, distributed systems, high-speed networks, and pervasive computing. He has chaired many professional conferences and has received a number of awards for authoring outstanding papers. He is a fellow of IEEE.