Efficient Multi-keyword Search over P2P Web

Hanhua Chen, Hai Jin
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430073, China
{chenhanhua,hjin}@hust.edu.cn

Jiliang Wang,Lei Chen,Yunhao Liu,Lionel Ni
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{aliang,leichen,liu,ni}@cse.ust.hk

ABSTRACT

Current search mechanisms of DHT-based P2P systems can well handle a single keyword search problem. Other than single keyword search, multi-keyword search is quite popular and useful in many real applications. Simply using the solution for single keyword search will require distributed intersection/union operations in wide area networks, leading to unacceptable traffic cost. As it is well known that Bloom Filter (BF) is effective in reducing traffic, we would like to use BF encoding to handle multi-keyword search.

Applying BF is not difficult, but how to get optimal results is not trivial. In this study we show, through mathematical proof, that the optimal setting of BF in terms of traffic cost is determined by the global statistical information of keywords, not the minimized false positive rate as claimed by previous methods. Through extensive experiments, we demonstrate how to obtain optimal settings. We further argue that the intersection order between sets is important for multi-keyword search. Thus, we design optimal order strategies based on BF for both "and" and "or" queries. To better evaluate the performance of this design, we conduct extensive simulations on TREC WT10G test collection and the query log of a commercial search engine. Results show that our design significantly reduces the search traffic of existing approach by 73%.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *distributed systems, information networks*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *distributed applications*.

General Terms: Algorithms, Design.

Keywords: P2P, DHT, Multi-keyword Search, Bloom Filter.

1. INTRODUCTION

With the emergence of peer-to-peer (P2P), file sharing applications such as Napster and Gnutella, millions of users have used P2P systems to search desired data [28, 34, 38]. P2P networks have also shown a great potential to become a popular network tool for sharing information on the Web [19, 29] based on the following observations. First, information on the Internet resides on millions of web sites in a distributed manner. P2P based systems have the ability to leave the shared, but distributed, data at their origins instead of collecting and maintaining them in a centralized repository. Second, there are significant performance, scalability, and availability benefits by distributing the indexing and querying load over a large number of collaborating peers. Third, a distributed P2P search system is more robust than a centralized search system as the failure of a single server is unlikely to paralyze the entire search system. Finally, there is growing concern about the fact that the world is dependent on a few quasi-monopolistic search engines. It is difficult to guarantee that they always bring objective results to users due to their susceptibility to commercial interests, possible biases in thematic coverage, or even censorship for different reasons [4].

While keyword search is a popular query type over the Web, how to implement keyword search mechanism efficiently on P2P systems remains a challenging task. Different from traditional web search engines, it is often difficult, if not impossible, to maintain a centralized content index in a large scale P2P network. Existing P2P retrieval mechanisms provide a scalable distributed hash table (DHT) [24, 27, 31, 36] that allows every individual keyword to be mapped to a set of documents/nodes across the network that contain the keyword. Using this single-keyword based index, a list of entries for each keyword in a query can be retrieved by using existing DHT lookups. However, compared with the single keyword search, multi-keyword search is much more popular and useful in many real world applications. For multi-keyword search, the simple solution which merges the results of each keyword search is not scalable and incurs a lot of traffic. Given an example, considering a two-keyword query "peer-to-peer network", the query is decomposed into "peer-to-peer" and "network", and then the two keywords are searched separately with a consequent intersection operation. A potentially large amount of data traffic will be raised across the wide area network.

It is well known that *Bloom Filter* (BF) [5, 9, 37] is an effective way to reduce such communication cost. A BF is a lossy but succinct and efficient data structure to represent a set *S,* which can efficiently process the membership query such as "is the element *x* in the set *S*". By transmitting the encoded sets instead of raw sets among peers for distributed intersection/union the communication cost can be effectively saved. How to get optimal results, however, is not trivial. In other words, simply using BF, such as with the goal of minimizing the false positive rate [25], will raise unacceptable high traffic cost [15].

In this work we show mathematically that the optimal setting of BF in terms of communication cost is determined by the global statistical information of keywords, not the minimized false positive rate as claimed by the previous methods [25]. We further demonstrate how to get optimal settings through numerical analysis. Indeed, the intersection order between sets is important for multi-keyword search. We design optimal order strategies based on BF for both queries with "and" and "or" operators.

We conduct comprehensive trace-driven simulations on TREC WT10G [12] test collection and the query log of a commercial search engine to evaluate the performance of this design. Results

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

show that our design significantly reduces the search traffic of existing approaches by 73%.

The main contributions of this work are as follows.

- We show mathematically that the optimal setting of BF in terms of traffic cost is determined by the global statistical information of keywords
- We derive an effective approach for a real world system to achieve BF optimal settings through numerical analysis
- We design optimal order strategies based on BF for queries with both "and" and "or" operators.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces the system design. Section 4 describes the simulation methodology. Performance evaluation is presented in Section 5. We conclude the paper in Section 6.

2. RELATED WORK

There are generally two types of P2P search engines: federated search engines over unstructured P2P networks [19], and distributed global inverted index on top of structured P2P networks [23].

In the first type, peers that maintain indexes of their local documents are organized in an ad-hoc fashion. A simple search method is flooding. Each query is tagged with a maximum *Time-To-Live* (TTL) to limit the number of hops it travels. In order to reduce the search cost, many approaches have focused on the issue of query routing. The proposed algorithms often need to perform the query search in two levels, the peer level and document level. First, a group of peers with potential answers to the query are detected. Second, the query is submitted to the identified most relevant peers to evaluate the query against their local indexes and return the matched answers. Finally, the retrieved answers are merged to produce a single answer set for the user.

PlanetP [7] proposed to replicate in every peer a global term-to-peer inverted index which contains a mapping "$t \rightarrow p$" if term t is in the local index of peer p. For each query, it ranks the peers using an IDF [26] like relevance model based on the statistical information of the replicated global index. However, in fact, it is difficult for every node in the network to collect and store such global index information. H. Zhang et al. [30] proposed a semantic-based method to link the peers with similar interest. By forwarding the queries through the interest-based shortcuts, a significant amount of unnecessary flooding is avoided. A language model-based method is designed by Jie Lu et al. [18] to locally rank the neighboring peers. Queries are forwarded to the top-ranked neighbors who are most likely to have the answers.

The other type is based on distributed indexes that partition a logically global inverted index in a physically distributed manner. Currently, there are two kinds of distributed index mechanisms: single-term-based inverted indexes and term-set-based indexes.

Searching with a single-term-based distributed index can retrieve the list of documents/nodes for each keyword in a query. In [32] frequent terms of a document are selected to be published into the global index. When such a keyword is published, the list of other terms in the document is replicated with the identifier of the document in the posting list. Multi-keyword search is performed by first locating the position of the DHT node which is responsible for a given keyword and then performing a local search in the posting list for other keywords. Finally the list of documents that contain all the keywords is returned as the results. Little is known about the performance of the full text search using selected keyword publishing, because a few selected frequent terms may not be representative for a document [26] and such replication strategy may incur unacceptable storage and communication cost. Another scheme performs a distributed intersection operation for multi-keyword search. Based on the global single term based inverted index built on DHT, the multi-keyword search looks up the sets for different keywords from multiple peers across the wide area network and returns the intersection. Although only a few nodes need to be contacted, each node has to send a potentially large amount of data across the wide area network. Reynolds et al. [25] used a BF to reduce such cost incurred by distributed intersection. In their work, they claimed that optimal results can be achieved through minimizing the false positive of a BF. However, the communication cost is still unacceptable [15].

Another way to reduce the bandwidth cost is to pre-compute the index using term-set indexing. Some preliminary experiments in [11] has shown that the term-set-based indexing is efficient for multi-keyword search across the wide area network. The major drawback of term-set-based index is that the index size may grow exponentially. To reduce the unacceptable index size, Podnar et al. [23] proposed to index only highly discriminative keyword combinations in a distributed global index in a structured P2P network. Although their method can reduce the total number of combinations to be indexed, it is difficult to guarantee that the keyword set that users are interested in are exactly the keyword set selected for indexes. Motivated by the fact that queries can reflect the real information requirements of users, Bender et al. [4] proposed to index the term sets by considering the correlation between the keywords in queries. In their design, a DHT node stores additional posting lists for term sets that are strongly correlated with the terms it is originally responsible for. Although such term-set indexing schemes reduce the scale of indexes, it is difficult to build a complete search system on top of the proposed methods.

3. SYSTEM DESIGN

In this section, we first give a brief overview of our hybrid P2P network design for P2P web search, and focus on how to optimize the communication cost of DHT-based multi-keyword search using an optimal BF. We then describe the optimization strategies for "and" queries and "or" queries. In section 3.3 we propose an optimized intersection order strategy for multi-keyword queries. We present the pushing synopsis gossip algorithm for collecting global statistical information in Section 3.4.

3.1 Solution Outline

In this design, a hybrid P2P network [17] is a combination of (1) an unstructured P2P network which can use a gossiping algorithm to gather global statistical information, and (2) a BF enabled overlay based on DHT global inverted indexes. Each peer participates in an unstructured network and acts as a structured DHT node as well (In the P2P Web search application, a peer represents a web server.) With the facility of an unstructured network, the system utilizes a push-synopsis gossip algorithm for gathering the global statistical information such as keyword popularity. For keyword search, an inverted index can be built based on existing DHT lookup services which associates a keyword to a posting list of documents containing the keyword. While our approach is general to any of these DHT techniques, for simplicity, the following discussion assumes architecture closely related to the Chord protocol [31]. In

order to reduce the communication cost, we use a BF for distributed intersection and union required by the multi-keyword search.

When a query comes, peers can minimize the communication cost by adjusting BF parameters to optimal settings according to the statistical popularities of the keywords in the query. Due to the inherent heterogeneity of web servers, randomly distributing keywords across the system runs the risk of assigning a popular keyword to a relatively under-provisioned web server in terms of memory, CPU, or network capacity. We can use the virtual host technique [8] to address this potential load balancing problem. In this approach, a DHT node can act as several logical hosts depending on its capacity. More workload will be assigned to a node that represents more virtual hosts.

3.2 Minimizing Communication Cost for Multi-keyword Search

Before we discuss the mechanism for reducing the communication cost for multi-keyword search, we introduce the following concepts.

1) Observations on user behaviors: We recently analyzed the query logs of a commercial web search engine. The query length distribution is plotted in Fig. 1, from which we can observe that 56.31% of the queries consist of at least two terms. This indicates that multi-keyword search is quite common in web content searching.

2) Bloom Filter: We review the basic of BF, following the framework of references [5, 9, 37]. A BF is essentially a bit vector *bitvec_m* with m bits, initially all set to 0, that facilitates membership test to a finite set $S = \{x_1, x_2, \ldots, x_n\}$ of n elements from a universe U. It uses a set of k uniform and independent hash functions $\{h_1, h_2, \ldots, h_k\}$ to map the universe U to the bit address space [1-m]. For each element x belonging to S, the bits $h_i(x)$ are set to 1 for $1 \le i \le k$. To check if an item y is in S, we check whether all $h_i(y)$ are set to 1. If not, y clearly is not a member of S. If all $h_i(y)$ are set to 1, we assume that y is in S.

After all the n elements of S are hashed and inserted into the BF, the probability that a specific bit of *bitvec_m* is still 0 is

$$p = (1 - \frac{1}{m})^{kn} \approx e^{-kn/m} \qquad (1)$$

The probability of a false positive after n elements inserted in the *bitvec_m* is the probability that a new element is not in S, but is separately hashed by the k hash functions to a number of k "1" bits of the *bitvec_m*.

$$f = (1 - p)^k = (1 - e^{-kn/m})^k \qquad (2)$$

3.2.1 "And" query

A common solution for a multi-keyword search needs conducting a distributed intersection operation in a wide area network. Figure 2(a) gives an example of a two-keyword (x, y) search. The query is first routed to the DHT node which is responsible for keyword x. Then X, the set of identifiers of documents that contain keyword x, is transmitted to the node which is responsible for keyword y for a consequent intersection operation to achieve $X \cap Y$, where Y is the set of document identifiers whose corresponding documents contain keyword y. The final results are returned to the client.

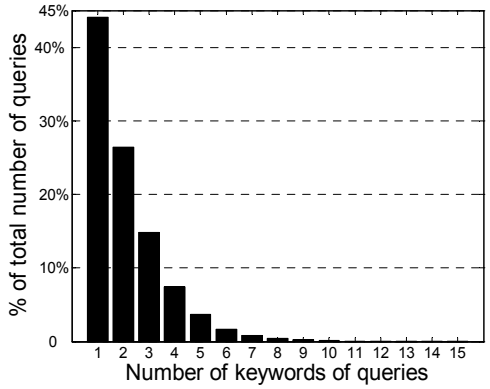

Figure 1. Term number distribution of queries from a commercial search engine.

Based on the analysis of the query logs in the WT10G data collection, we find that in most cases the minimal cardinality of set of documents that contain any single keyword in a query is several orders of magnitude above the cardinality of the intersection of all the sets. Thus, in a DHT-based P2P web search system, the straightforward distributed intersection operation only achieves a relatively small result set at the cost of sending complete sets in the wide area network. Clearly, the communication cost can be saved by transmitting the BF of sets instead of raw sets among peers for distributed intersection. Existing work [25] claimed that minimizing the false positive rate of a BF is most efficient in reducing the communication cost. In this paper, we show that this is not the case.

For the same example discussed above, our design reduces the communication cost by sending an optimal BF based on X, BF(X), instead of sending X itself, as illustrated in Fig. 2 (b). When BF(X) is transmitted to the DHT super peer which is responsible for keyword y, it determines the intersection of X and Y based on BF(X). Because the BF has no false negatives, the result set will contain all elements of the true intersection. Due to the possible false positives, the result set may contain elements that contain only keyword y but not x. Typically, a client would like to retrieve only the exact intersection of X and Y. Thus, the result set, denoted by $Y \cap$BF(X), is sent back to the DHT peer responsible for keyword x. As a result, given a fixed value of false positive, the number of the extra-transmitted elements is in proportion to $|Y|$, the popularity of keyword y. The peer responsible for keyword x removes the false positives from $Y \cap$BF(X) by calculating $X \cap (Y \cap$BF(X)), which is equivalent to $X \cap Y$.

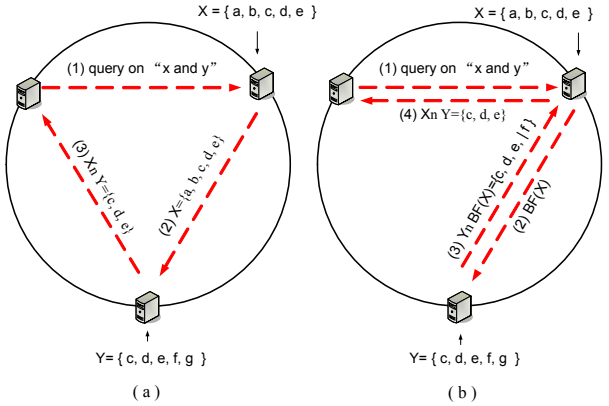

Figure 2. Straightforward distributed intersection vs. BF-based strategy.

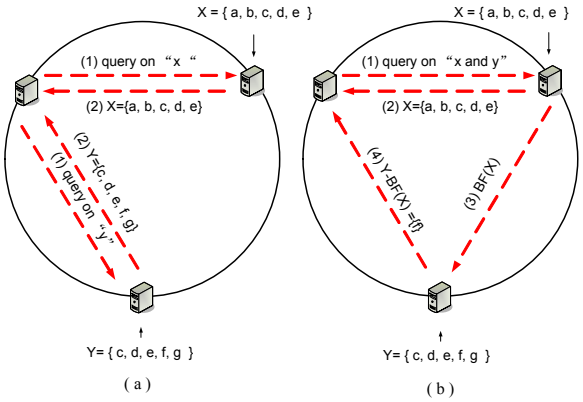

Figure 3. Straightforward distributed union vs. BF based strategy.

Thus, the communication cost of the BF-based intersection is quantified by

$$m + f \mid Y \mid r + \mid X \bigcap Y \mid r \tag{3}$$

We assume that each element in the set takes r bits. For minimizing the communication cost of BF-based distributed intersection algorithm, the communication cost for transmitting $X \cap Y$ can be ignored since it represents the final intersection result, which must be sent back regardless of what algorithms are chosen. We substitute f from Eq. (2), and the extra communication cost for distributed intersection is,

$$f(m,k) = m + (1 - e^{-k|X|/m})^k \mid Y \mid r \tag{4}$$

Equation (4) shows that the minimized communication cost can be achieved by adjusting the settings of BF. As Equation (4) shows, the optimal settings are determined by the global statistical information of keywords, not simply by the minimized false positive rate as claimed in [29].

3.2.2 "Or" query

In some applications, we need "or" queries, which desire the results containing any keyword in the query. Such query is critical for queries whose keywords are rare in the system. A search engine may combine both the "and" and "or" results for a multi-keyword query to the users. Figure 3(a) presents an example of the straightforward strategy for a two-keyword "or" query. First, the query is separately sent to the DHT nodes responsible for keywords x and y, respectively. The DHT nodes separately send back the complete list for each keyword. At last the results of both keywords are merged at the client. Thus, the total communication cost is $(|X|+|Y|)\, r$.

In our design shown in Fig. 3(b), the query is firstly routed to the DHT node which is responsible for keyword x. Then, BF(X) will be forwarded to the DHT node that is responsible for keyword y to pick out the documents which are not in X by checking elements in Y using BF(X). Only the set picked out, denoted by Y-BF(X), is returned to the client for a consequent union operation. The communication cost can be quantified as

$$\begin{aligned} & m + (\mid X \mid + \mid Y - \mathrm{BF}(X) \mid)r \\ & = m + (\mid X \mid + (1 - f)(\mid Y \mid - \mid X \bigcap Y \mid))r \end{aligned} \tag{5}$$

By avoiding repeatedly sending the intersection of X and Y, our algorithm for distributed union is promising for reducing traffic cost for some queries. Mathematically, the benefit of communication cost by BF is $M_{saved} = (|X|+|Y|)\, r - (m+|X|\, r+(1-f)\, |Y-X|\, r) = (1-f)\, |X \cap Y|\, r + f|Y|\, r - m$. Using the global statistics information of keywords, we can use a threshold to select the strategy. If $\frac{M_{saved}}{(|X|+|Y|)r} > \delta$, where δ is a threshold, we use BF for distributed union operation; otherwise, we use the straightforward strategy. Note that Y-BF(X) is slightly different from Y-X due to the false positives. Thus some results of $X \cup Y$ will be missed in the final results. Given reasonable values of $|X|$, $|Y|$, m, and k, the number of missed elements is in proportion to $|Y-X|$, equivalent to $|Y|-|X \cap Y|$. Specifically, Y-BF(X) will miss $(1 - e^{-k|X|/m})^k \mid Y - X \mid$ elements that belong to Y. Thus the recall of the final result will slightly decrease by $\frac{(1-e^{-k|X|/m})^k (|Y|-|X \cap Y|)}{|X \cap Y|} \times 100\%$.

When we choose algorithms in a real world system design, we may consider this trade-off between the search quality for the user and system resource consumption. In this design we minimize the false positive to achieve the best recall rate. Given a specific ratio of m/n, i.e., the number of bits per element, it is easy to prove that the false positive rate f is minimized when $k = \frac{m}{n} \ln 2$ and the minimal false positive rate is [20].

$$f_{\min} = 0.6185^{\frac{m}{n}} \tag{6}$$

By substituting f in Eq. (5) from Eq. (6), the communication cost for distributed union is,

$$m + (\mid X \mid + (1 - 0.6185^{\frac{m}{|X|}})(\mid Y \mid - \mid X \bigcap Y \mid))r \tag{7}$$

Given a minimized false positive f, our scheme can make a decision for the distributed union based on M_{saved}.

3.3 Intersection-order Optimization Strategy

Based on the distributed intersection and union operation, we can easily cope with the queries which have more keywords. For a query with more than two keywords, it is intuitive that there is much benefit if we first perform distributed intersection operations for the pairs of keywords that are not frequently used together in the same documents, because the intersection of these keyword pairs will be sufficiently small, and the communication cost will be reduced. However, it is difficult to estimate the size of intersection incurred by two keywords before we get the exact intersection. In our design we use BF to estimate the size of intersection between two sets for any given two keywords [6].

3.3.1 Intersection size estimation

Suppose that we have two BFs separately representing X and Y with the same number of m bits and using the same set of k hash functions. It is intuitive that the inner product of the two BFs can be used to measure their similarity [6]. Mathematically, the ith bit will be set to "1" in both BFs if it is set by using some element in $X \cap Y$, or if it is set to "1" simultaneously by some element in X-($X \cap Y$) and by another element in Y-($X \cap Y$). In total, the probability that the ith bit is set to "1" in both BFs can be quantified as

$$\left(1-\left(1-\frac{1}{m}\right)^{k|X\cap Y|}\right) + \left(1-\frac{1}{m}\right)^{k|X\cap Y|}\left(1-\left(1-\frac{1}{m}\right)^{k|X-(X\cap Y)|}\right)\left(1-\left(1-\frac{1}{m}\right)^{k|Y-(X\cap Y)|}\right) \quad (8)$$

After simplifications, the expected magnitude of the inner product of the two BFs can be quantified as

$$p = m\left(1-\left(1-\frac{1}{m}\right)^{k|X|} - \left(1-\frac{1}{m}\right)^{k|Y|} + \left(1-\frac{1}{m}\right)^{k(|X|+|Y|-|X\cap Y|)}\right) \quad (9)$$

Thus, given $|X|$, $|Y|$, k, m, and p, the value of the inner product, we can get an estimated size of $X\cap Y$ using the following equation.

$$|X\cap Y| = -\frac{\log_{1-\frac{1}{m}}\left(\frac{p}{m}+\left(1-\frac{1}{m}\right)^{k|X|}+\left(1-\frac{1}{m}\right)^{k|Y|}-1\right)}{k} + (|X|+|Y|) \quad (10)$$

3.3.2 Learning from queries

The difficulty here is that it is infeasible to exhaustively identify all the combination of term pairs and also impossible to predict all the combination of interests due to the vast communication cost. In this paper, we utilize the query history to find out near optimal pairs. Specifically, we monitor queries on the DHT nodes where the BF is transmitted to so that interesting correlations can be inferred. More specifically, in Figure 3.b the BF(X) is cached in the DHT node responsible for keyword y for calculating the cardinality of the intersection $X\cap Y$. The more frequently X and Y are searched together by users, the more correlated they are. We use a push-synopsis based gossip algorithm to propagate the popularity of such keyword pairs.

3.4 Gathering Global Keyword Popularity

Within the structure of a hybrid P2P network we use a variant of the push synopsis gossip algorithm first proposed in [21] to gather global keyword popularity in the Web. The robust algorithm enables every peer to quickly collect the global statistical term frequency in the P2P Web [10, 35].

Considering the example of $|X|$, the global statistical frequency of keyword x, the method first lets all peers in the network check their local index. When the keyword x is found the first time in a document on a peer, this peer does the following experiment: it flips a coin up to t times and counts the number of times the head appears before the first time it sees the tail. It saves this count in a value called *FC(x)*. Then the *FC(x)* is gossiped among the peers in the network. During each round of gossip, each node chooses a random neighbor and sends the neighbor the *FC(x)* value it locally holds. After receiving the *FC(x)* values from a neighbor, a peer computes the maximum value of *FC(x)*, i.e., *maxFC(x)*. The results in [14] show that the robust gossip scheme leads the computation of aggregated information to converge exponentially: after $O(log(n))$ rounds of gossip, where n is the number of nodes in the network, all peers will get $|X|$ with high probability. The frequency of keyword x is roughly $2^{maxFC(x)-1}/0.77351$ with high probability [10, 35].

The pushing synopsis based gossiping algorithm for estimating global statistical keyword frequency has three main operations. *1) Synopsis generation*: When a peer joins the network the first time, it browses its local index and generates a local synopsis by doing the coin flipping experiment. The synopsis structure is designed as {(x, *bitvec_x*)}, where *bitvec_x* is a bit vector for counting the statistical frequency of keyword x. *2) Synopsis disseminating:* The synopses are disseminated among peers using the randomized gossip algorithm proposed in [14]. During gossip round, each node randomly choose a neighbor and sends the selected neighbor its local synopsis. *3) Synopsis merging:* When a peer receives the synopsis from its neighbor, it checks the synopsis it receives against his own synopsis and performs the following synopsis merging operation. For the keyword t in both synopses, it performs the bitwise-or operation on the pair of bit vectors for *bitvec_t*; and for those keywords in the synopsis of the neighbor but not in the local synopsis, it inserts the bit vector into its own synopsis.

4. SIMULATION METHODOLOGY

In this section we first introduce the data set and query logs we use for the evaluation of our design, and how we collect the traces of Gnutella for simulating the P2P topology. We then discuss the design of our simulator for P2P Web multi-keyword search.

4.1 Web Data Collection

There has been no standard data set established for evaluating the performance of content-based P2P web search [18]. We built one based on Text Retrieval Conference (TREC) [12] WT10G web corpus, a large test set widely used for performance evaluation in web retrieval research area. The dataset includes 10 gigabyte, 1.69 million web page documents and a set of queries (we use the "title" field of a TREC topic as a query [12]). The WT10g data was divided into 11,680 collections based on document URLs. Each collection on average has 144 documents with the smallest one having only 5 documents. The average size of each document is 5.91KB.

All data set was stemmed with the Porter algorithm to reduce words to their root (e.g., "putting" becomes "put") and common stop words such as "the", "and", etc. were removed from the data set [12]. Table 1 summarizes the statistics for the test data set.

4.2 Queries

The number of queries provided by U.S. National Institute of Standards and Technology (NIST) for the TREC WT10g web test collection is far from enough to be used in studies on P2P web search. We evaluate our design using the query logs of a commercial search engine which we have analyzed in Section 3.2. The query logs are quite representative for real world systems.

Table 1. Statistics of the WT10G data set

Parameters	Value
Number of documents	1, 692, 096
Number of collections	11, 680
TREC topics	501~550
Average number of documents of a collection	144
Average size of documents	5.91KB

4.3 Gnutella Trace

We have developed a crawler in Java based on the limewire [3] open source client to collect topology information of Gnutella network. According to Gnutella protocol [2], a ping message with settings TTL= 2 and HOP = 0 is regarded as a crawler ping, and peers which receive a crawler ping should respond with appropriate pong messages. Our crawler ran in parallel with 40 threads, and can discover more than 50,000 peers within half an hour. We use the Gnutella topology trace we collected to simulate a real P2P network.

4.4 Hybrid P2P Networks

In order to well 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 a large scale Internet physical topology follows the small world and power law properties.

The topology of a small-world network has the properties of sparseness, short global separation, and high local clustering of nodes while power law denotes the property of the node degree distribution. The study of Tangmunarunkit et. al [33] 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. All P2P nodes in the trace 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. In the simulations, we randomly distribute the WT10G collections into the Gnutella peers. Thus, each peer acts as a web server in the P2P web. We simulate Chord protocol to support single keyword based global inverted index. The gossip algorithm described in Section 3.4 is implemented in an unstructured overlay network.

5. PERFORMANCE EVALUATION

In this section we first introduce the metrics that we use in the evaluation. Then we compare our design with the previous work proposed in [25].

5.1 Metric

In the evaluation, we mainly consider the metric of traffic cost. P2P traffic has a significant impact on the underlying network. Heavy network traffic limits the scalability of P2P networks. In a P2P Web search application, the bandwidth consumption for a search is the main cost [11]. While there are some overheads associated with maintaining an inverted index, we believe this cost is typically offset by avoiding the overhead associated with repeated spidering of web content.

We define the traffic cost as network resource used in a web search process of P2P systems, which is mainly a function of consumed network bandwidth and other related expenses. 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: $Tc = M \sum_i L_i / B_i$ [16], 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.

5.2 Results

We first show how we achieve the optimal settings of BF by analyzing the targeted function defined in Section 3.2 for minimizing communication cost using Matlab. Based on the analysis results, we then compare our optimal BF design with the straightforward BF algorithm [25], which reduces the communication cost by minimizing the false positive of a BF using comprehensive simulations.

5.2.1 Optimal setting of bloom filter

In this section, we show how to achieve the minimized communication cost defined in Section 3.2 by using optimal settings of BFs. We analyze the communication cost quantified by Eq. (4) with Matlab. We consider three typical situations $|X|<|Y|$, $|X|=|Y|$, and $|X|>|Y|$. We set r to 250 bits based on the research results conducted on Google search engine, which show that the average URL length measured in character is 31.2 characters [13]. We adjust the parameters m and k and examine how the value of $f(m, k)$ changes.

We find that the intersection order is critical for minimizing the communication cost. When $|X|$ is not greater than $|Y|$, the communication cost can be minimized. The value of $f(m, k)$ is significantly influenced by the variable m. The minimal value of $f(k, m)$ can be achieved when m is set as an optimal value. The minimal communication cost changes very slightly when we adjust the value of parameter k while fixing the value of parameter m.

The results demonstrate that the optimal BF is determined by the popularities of keywords and the intersection order. Much benefit can be achieved if we transmit the BF for the set of a less popular keyword to the DHT node responsible for a popular keyword during the process of distributed intersection. Based on these observations, given $|X|$, $|Y|$ and k, the objective of our optimal BF based intersection algorithm is to enable each node intelligently choose the optimal m and the intersection order to achieve the minimal communication cost.

In this design we first sort the keywords for an intersection operation in increasing order according to their popularities, $|X|<|Y|$. By varying the values of $|X|$ and $|Y|$ we obtain a set of sample values for optimal m.

Figure 4 plots the optimal values of m for any given $|X|$ and $|Y|$. The results show that with same values of $|X| / |Y|$, the value $m / |X|$ is a constant, where m is the optimal setting. For simplicity, we use u to denote $|X| / |Y|$ and v to denote $m / |X|$. Thus, we can derive a function $v = f(u)$.

We use Matlab least-squares polynomial curve-fitting tools to find best fits. Figure 5 shows the curves for fits. The three cubed curve $v = 0.0001u^3 - 0.01u^2 + 0.42u + 11.06$ better fits the distribution of the optimal m. Thus each node can determine the optimal settings of BF according to the popularities of query keywords with no extra configuration cost. In the rest of simulations, every DHT node calculates the optimal m by: $m = f(u)\,|X|$.

5.2.2 Comparisons

To better examine the performance of our popularity-aware distributed intersection using optimized BF settings, we compare the performance achieved by our method with that of the previous work [25] as the baseline approach. In the baseline approach, the parameter is adjusted to minimize the false positive of a BF and the intersection order is random without popularity statistics of the keywords and the intersection size estimation.

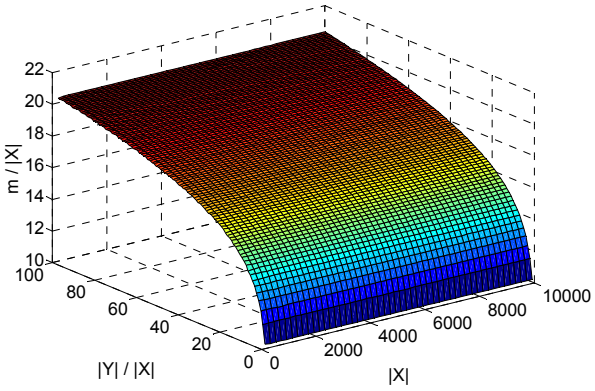

Figure 4. Distribution of optimal *m* settings.

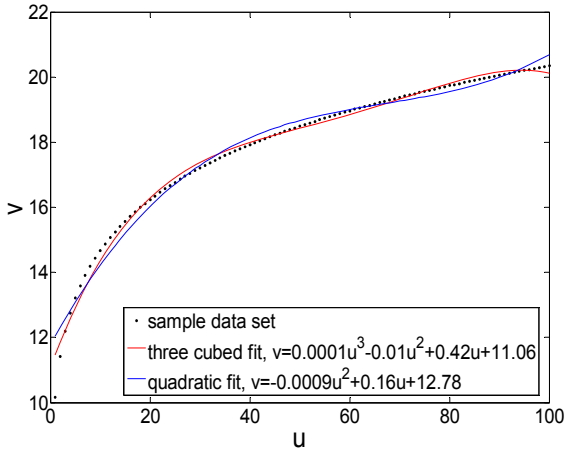

Figure 5. Polynomial curve-fitting for distribution of optimal *m*

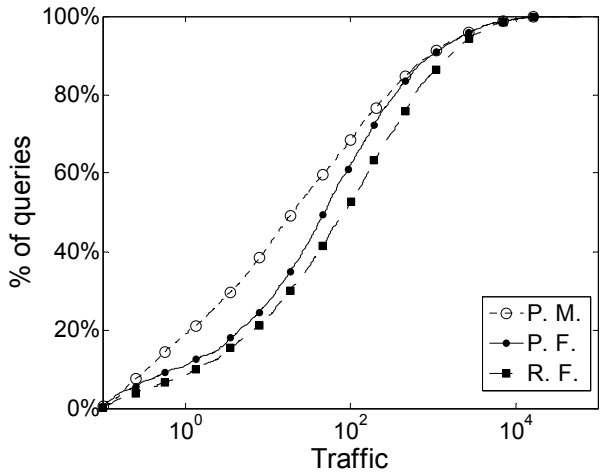

Figure 6. Traffic for distributed intersection using optimal BF.

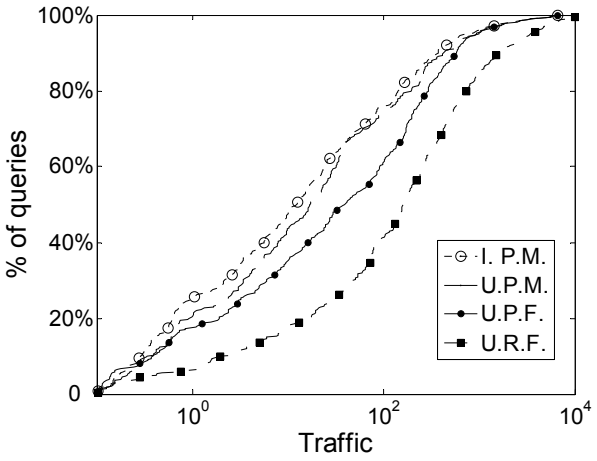

Figure 7. Traffic for distributed intersection using intersection size estimation.

To make this comparison clear, we first examine all the two-keyword queries in the query logs.

Figure 6 plots traffic cost of all the tested queries. For legends "*A.B.*", *A* and *B* are sets, where *A* = {*P*: sorting query terms according to popularity for distributed intersection, *R*: randomly sorting query terms for distributed intersection}, *B* = {*M*: optimizing *m* for minimizing communication cost, *F*: optimizing *m* for minimizing false positive of BF}. For example, *P.M.* stands for the method that configures the BF parameter *m* for a minimal communication cost and uses a popularity-aware term order for distributed intersection.

The results show that the insight proposed in our work is quite valid. The policies with *P.* and *M.* both improve the search performance of BF based distributed intersection greatly. We can use *R. F.* to denote the baseline, which adjust the parameter of a BF by optimizing false positive and search query terms in a random order. The result shows that with our strategy of *P.*, the statistical average query traffic is decreased by 37.78%. With both strategies of *P.* and *M.* the statistical average query traffic is significantly reduced by 63.75%. About 65.4% queries using the optimal strategy have a traffic cost less than 100, while 45.7% queries of the baseline achieve such low traffic cost.

To examine the benefit of the optimal intersection order for multiple keyword queries based on the intersection size estimation method described in Section 3.3, we conduct experiments with all the queries that have more than two keywords in the query logs. In Figure 7, we plot the traffic cost of the proposed strategies and compare them with the baseline approach, where the legends *A.B.* have the same meaning with Figure 6, and *C.* is a set, where *C* = {*I*: keyword pairs with smaller intersection size are searched together, *U*: unaware of the intersection set size}.

We compare the performance achieved by our strategies with the baseline approach. The baseline for multi-keyword search proposed in [25] can be denoted as *U. R. F.* The result in Fig. 7 shows that with our strategy of *P.*, the average query traffic is decreased by 71.51%. With strategies of *P.* and *M.* the average query traffic is reduced by 81.75%. With all the strategies of *I. P. M.* the average query traffic is significantly reduced by 84.23%. About 76.51% queries using the optimal strategy have a traffic cost less than 100, while 41.89% queries of the baseline achieve such low traffic cost.

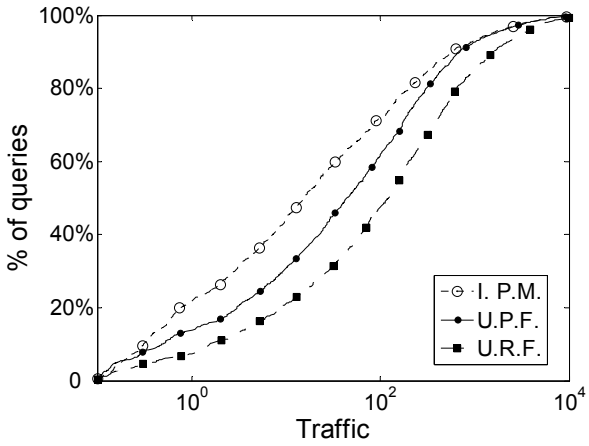

Figure 8. Traffic of distributed intersection for all the logs.

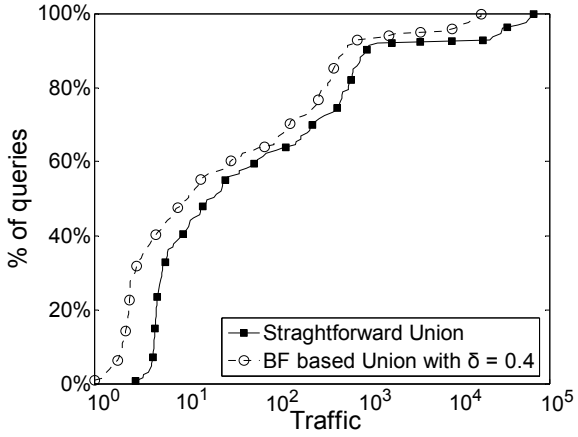

Figure 9. Traffic for distributed union.

Figure 8 plots the traffic cost for all the queries we collected, where our algorithm greatly outperforms the baseline. Figure 8 shows that with our strategy of *P*., the average query traffic for all the query logs is decreased by 53.01%. With all the strategies of *I. P. M.* the statistical average query traffic is significantly reduced by 73.03%. About 72.86% queries using the optimal strategy have a traffic cost less than 100, while 47.45% queries of the baseline achieve such low traffic cost.

We further examine the performance of our distributed union algorithm based on BF described in Section 3.3.2 by using the straightforward union operation as the baseline. Figure 9 shows the performance of distributed union algorithm, where the threshold is fixed at $\delta = 0.4$. Figure 9 shows that the traffic cost of the involved queries are effectively reduced. Statistically the traffic cost is reduced by 49.84%. About 84.9% queries using our distributed union algorithm have a traffic cost less than 500, while only 75.1% involved queries of the baseline achieve such low traffic cost.

We adjust the value of δ . Figure 10 plots the reduction of traffic for different values of δ . Results show that when the threshold increases, the reduction of traffic cost increases apparently.

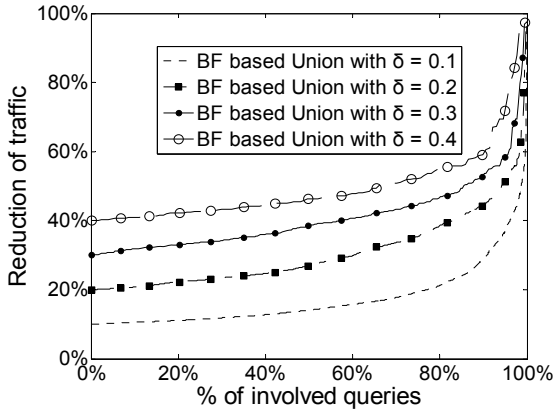

Figure 10. Traffic cost when varying the threshold.

5.2.3 Analysis of the communication cost of gossip

Another cost of this design is the synopsis for the gossip algorithm. In this design the bit vector for each unique keyword takes 4 bytes. The average word length is about five. Each term will take 10 bytes in the synopsis. There are about 600,000 distinct terms in the Oxford dictionary. Hence, each peer needs a storage size below the bound of 6MB for gathering the global information.

Text compression using Burrows-Wheeler Transform (with measured compression ratios of 2.95 [22]) can reduce the storage size to about 2MB. On the other hand, the communication cost for the gossiping algorithm is quite acceptable for a real world system due to the fact that the global statistics in the World Wide Web are slowly changing (on the time scale of weeks). Therefore, infrequent and approximate computation of these statistics is sufficient for good performance.

6. CONCLUSIONS

In this paper we proposed an efficient multi-keyword search mechanism over P2P Web environments. We showed mathematically that the communication cost for multi-keyword search in P2P Web using BF is determined by the popularities of the query keywords, and designed a method to achieve optimal settings for a BF to obtain a minimal communication cost. We also proposed the optimal order strategies for both "and" and "or" queries. We conduct comprehensive simulations based on TREC WT10G test collection and the query log of a commercial search engine. Simulation results show that our design outperforms existing work, and wide deployment of this design will significantly improve multi-keyword search for P2P Web.

7. ACKNOWLEDGEMETS

This work was supported in part by National 973 Key Basic Research Program under grant No.2003CB317003, the Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China under grant No.705034, Hong Kong RGC Grant N_HKUST614/07 and HKUST Nansha Research Fund NRC06/07.EG01

8. REFERENCES

[1] "BRITE, http://www.cs.bu.edu/brite/." 2007.

[2] "The Gnutella protocol specification 0.6," 2002.

[3] "Limewire, http://www.limewire.com." 2007.

[4] Bender, M., Michel, S., Triantafillou, P., Weikum, G., and Zimmer, C., P2P Content Search: Give the Web Back to the People, in Proceedings of IPTPS, Santa Barbara, CA, USA, 2006.

[5] Bloom, B.H., Space/Time Trade-offs in Hash Coding with Allowable Errors, *Communication of the ACM*, 13 (7):422-426, 1971.

[6] Broder, A. and Mitzenmacher, M., Network Applications of Bloom Filters: A Survey, *Internet Mathematics*, 1 (4):484-509, 2005.

[7] Cuenca-Acuna, F.M., Peery, C., Martin, R.P., and Nguyen, T.D., Planetp: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities, in Proceedings of HPDC, 2003.

[8] Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., and Stoica, I., Wide-area Cooperative Storage with CFS, in Proceedings of SOSP, 2001.

[9] Fan, L., Cao, P., Almeida, J., and Broder, A.Z., Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol, *IEEE/ACM Transactions on Networking*, 8 (3):282-293, 2000.

[10] Flajolet, P. and Martin, G.N., Probabilistic Counting Algorithms for Data Base Applications, *Journal of Computer and System Sciences*, 31 182-209, 1985.

[11] Gnawali, O.D., "A Keyword-Set search system for peer-to-peer networks," in *Master's thesis, Massachusetts Institute of Technology, June*, 2002.

[12] Hawking, D., Overview of the TREC-9 Web Track, in Proceedings of Text REtrieval Conference (TREC-9), 2000.

[13] Huang, N.-F., Liu, R.-T., Chen, C.-H., Chen, Y.-T., and Huang, L.-W., A Fast URL Lookup Engine for Content-Aware Multi-gigabit Switches, in Proceedings of AINA, 2005.

[14] Kempe, D., Dobra, A., and Gehrke, J., Gossip-Based Computation of Aggregation Information, in Proceedings of IEEE FOCS, 2003.

[15] Li, J., Loo, B.T., Hellerstein, J.M., Kaashoek, M.F., Karger, D., and Morris, R., On the Feasibility of Peer-to-Peer Web Indexing and Search, in Proceedings of IPTPS, 2003.

[16] Liu, Y., Liu, X., Xiao, L., Ni, L.M., and Zhang, X., Location-Aware Topology Matching in P2P Systems, in Proceedings of IEEE INFOCOM, 2004.

[17] Loo, B.T., Hellerstein, J.M., Huebsch, R., Shenker, S., and Stoica, I., Enhancing P2P File-Sharing with an Internet-Scale Query Processor, in Proceedings of VLDB, 2004.

[18] Lu, J. and Callan, J.P., Content-Based Retrieval in Hybrid Peer-to-Peer Networks, in Proceedings of CIKM, 2003.

[19] Lu, J. and Callan, J.P., User Modeling for Full-text Federated Search in Peer-to-Peer Networks., in Proceedings of SIGIR, 2006.

[20] Mitzenmacher, M., Compressed Bloom Filters, *IEEE/ACM Transactions on Networking*, 10 (5):604-612, 2002.

[21] Nath, S., Gibbons, P.B., Seshan, S., and Anderson, Z.R., Synopsis Diffusion for Robust Aggregation in Sensor Networks, in Proceedings of ACM SenSys, 2004.

[22] Nelson, M., Data Compression with the Burrows-Wheeler Transform, *Dr. Dobb's Journal*, 1996.

[23] Podnar, I., Rajman, M., Luu, T., Klemm, F., and Aberer, K., Scalable Peer-to-Peer Web Retrieval with Highly Discriminative Keys, in Proceedings of IEEE ICDE, 2007.

[24] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S., A Scalable Content-addressable Network, in Proceedings of ACM SIGCOMM, 2001.

[25] Reynolds, P. and Vahdat, A., Efficient Peer-to-Peer Keyword Searching, in Proceedings of Middleware, 2003.

[26] Robertson, S., Understanding Inverse Document Frequency: on Theoretical Arguments for IDF, *Journal of Documentation*, 60 503-520, 2004.

[27] Rowstron, A. and Druschel, P., Pastry: Scalable, Decentralized Object Location, and Routing for Large scale Peer-to-Peer Systems, in Proceedings of Middleware, 2001.

[28] Shen, H.T., Shu, Y.F., and Yu, B., .Efficient Semantic-based Content Search in P2P Network, *IEEE Transaction on Knowledge and Data Engineering*, 16 (7):813-826, 2004.

[29] Skobeltsyn, G., Luu, T., Zarko, I.P., Rajman, M., and Aberer:, K., Web text retrieval with a P2P query-driven index, in Proceedings of SIGIR, 2007.

[30] Sripanidkulchai, K., Maggs, B., and Zhang, H., Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems, in Proceedings of IEEE INFOCOM, 2003.

[31] Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H., Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, in Proceedings of ACM SIGCOMM, 2001.

[32] Tang, C. and Dwarkadas, S., Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval, in Proceedings of NSDI, 2004.

[33] Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S., and Willinger, W., Network Topology Generators: Degree-Based vs. Structural, in Proceedings of ACM SIGCOMM, 2002.

[34] Yang, B. and Garcia-Molina, H., Designing a Super-Peer Network, in Proceedings of ICDE, 2003, 49-60.

[35] Zaharia, M. and Keshav, S., Gossip-based Search Selection in Hybrid Peer-to-Peer Networks, in Proceedings of IPTPS, Santa Barbara, CA, USA, 2006.

[36] Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., and Kubiatowicz, J.D., Tapestry: A Resilient Global-Scale Overlay for Service Deployment, *IEEE Journal on Selected Areas in Communications (JSAC)*, 2004.

[37] Zhu, Y. and Jiang, H., False Rate Analysis of Bloom Filter Repoicas in Distributed Systems, in Proceedings of ICPP, 2006.

[38] Zhuge, H., Sun, X., Liu, J., Yao, E., and Chen, X., A Scalable P2P Platform for the Knowledge Grid, *IEEE Transactions. Knowledge and Data Engineering*, 17 (12):1721-1736, 2005.