Bookmark-driven Query Routing in Peer-to-Peer Web Search *

Matthias Bender, Sebastian Michel, Christian Zimmer, Gerhard Weikum
{mbender, smichel, czimmer, weikum}@mpi-sb.mpg.de
Max-Planck-Institut für Informatik, 66123 Saarbrücken

Abstract: We consider the problem of collaborative Web search and query routing strategies in a peer-to-peer (P2P) environment. In our architecture every peer has a full-fledged search engine with a (thematically focused) crawler and a local index whose contents may be tailored to the user's specific interest profile. Peers are autonomous and post meta-information about their bookmarks and index lists to a global directory, which is efficiently implemented in a decentralized manner using Chord-style distributed hash tables. A query posed by one peer is first evaluated locally; if the result is unsatisfactory the query is forwarded to selected peers. These peers are chosen based on a benefit/cost measure where benefit reflects the thematic similarity of peers' interest profiles, derived from bookmarks, and cost captures estimated peer load and response time. The meta-information that is needed for making these query routing decisions is efficiently looked up in the global directory; it can also be cached and proactively disseminated for higher availability and reduced network load.

1 Introduction

The peer-to-peer (P2P) approach has become popular in the context of file-sharing systems such as Gnutella or KaZaA. In such a system, all peers are equal and all of the functionality is shared among all peers so that there is no single point of failure and the load is balanced across a large number of peers. These characteristics offer potential benefits for building a powerful search engine in terms of scalability, resilience to failures, and high dynamics. In addition, a P2P search engine can potentially leverage the intellectual input from a large user community, for example, prior usage statistics, personal bookmarks, or implicit feedback derived from user logs and click streams.

This paper presents the architecture of a P2P Web search federation that we are currently building, and it proposes a strategy for routing queries to peers based on bookmarks maintained at peers. Each peer is autonomous and has its own local search engine with a crawler and a corresponding local index. Peers share their local indexes (or specific fragments of local indexes) by posting meta-information into the P2P network. This meta-information contains compact statistics and quality-of-service information, and effectively

*This work is partially supported by the EU Integrated Project DELIS on Dynamically Evolving, Large-scale Information Systems.

forms a global directory. However, this directory is implemented in a completely decentralized and largely self-organizing manner. More specifically, we maintain it as a distributed hash table (DHT) using the (re-implemented and adapted) algorithms of the Chord system [SMK+01]. Our per-peer engine uses the global directory to identify candidate peers that are most likely able to provide good query results. A query posed by a user is first executed on the user's own peer, but can be additionally forwarded to these peers for better result quality. The local results obtained from there are merged by the query initiator.

The rationale for our query routing strategy, i.e., the selection of the most promising peers among the, possibly large, set of candidates, is based on the following three observations:

1. The query initiator should prefer peers that have *similar interest profiles* and are thus likely to hold thematically relevant information in their indexes.
2. On the other hand, the query should be forwarded to peers that offer *complementary results*. If the remote peer returns more or less the same high-quality results that the query initiator already obtained from its own local index, then the whole approach of collaborative P2P search would be pointless.
3. Finally, all parties have to be cautious that the *execution cost* of communicating with other peers and involving them in query processing is tightly controlled and incurs acceptable overhead.

We address the first two points by defining the *benefit* that a remote peer offers for the given query to be proportional to the thematic similarity of that peer and the query initiator and inversely proportional to the overlap between the two peers in terms of their local index contents. To limit the overhead of estimating these measures, we use the Kullback-Leibler divergence [Ku59] between the bookmark documents of the two peers and the overlap in their bookmarks as the basis for estimating benefit. Here we view the index contents of a peer as being generated by the peer's bookmarks, which served as seeds for the peer's Web crawls and possibly also as training data for a thematically focused crawler [Ch02, SBG+]. We reconcile this notion of benefit with the third of the above observations by considering the benefit/cost ratio of peers, where cost is estimated based on tracking the utilization and resulting response time of different peers.

These guidelines for query routing lead to a specific strategy that is developed in this paper. The outlined system architecture is fully implemented and will serve as an experimental testbed for studying the viability of the proposed strategy. The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the system architecture in more detail. Section 4 develops the query routing strategy. Section 5 discusses our prototype implementation.

2 Related Work

Recent research on P2P systems, such as Chord [SMK+01], CAN [RFH+01], Pastry [RD01], or P-Grid [APHS02], is based on various forms of distributed hash tables (DHTs)

and supports mappings from keys, e.g., titles or authors, to locations in a decentralized manner such that routing scales well with the number of peers in the system. Typically, an exact-match key lookup can be routed to the proper peer(s) in at most $O(log\ n)$ hops, and no peer needs to maintain more than $O(log\ n)$ routing information. These architectures can also cope well with failures and the high dynamics of a P2P system as peers join or leave the system at a high rate and in an unpredictable manner. Earlier work on scalable distributed storage structures, e.g., [LNS96, VBW98], addressed similar issues. However, in all these approaches searching is limited to exact-match queries on keys. This is insufficient for text queries that consist of a variable number of keywords, and it is absolutely inappropriate when queries should return a ranked result list of the most relevant approximate matches [Ch02]. Our work makes use of one of these systems, namely Chord, for efficiently organizing a distributed global directory; our search engine is layered on top of this basic functionality.

In the following we briefly discuss some existing approaches towards P2P Web search.

PlanetP [CAPMN02] is a publish-subscribe service for P2P communities and the first system supporting content ranking search. PlanetP distinguishes local indexes and a global index to describe all peers and their shared information. The global index is replicated using a gossiping algorithm. The system, however, is limited to a few thousand peers.

Odissea [SMW$^+$03] assumes a two-layered search engine architecture with a global index structure distributed over the nodes in the system. A single node holds the entire index for a particular text term (i.e., keyword or word stem). Query execution uses a distributed version of Fagin's threshold algorithm [Fa99]. The system appears to cause high network traffic when posting document metadata into the network, and the query execution method presented currently seems limited to queries with one or two keywords only.

The system outlined in [RV03] uses a fully distributed inverted text index, in which every participant is responsible for a specific subset of terms and manages the respective index structures. Particular emphasis is put on three techniques to minimize the bandwidth used during multi-leyword searches: Bloom filters [Bl70], caching, and incremental result gathering. Bloom filters are a compact representation of membership in a set, eliminating the need to send entire index lists across servers. Caching reduces the frequency of exchanging Bloom filters between servers. Incremental result gathering allows search operations to halt after finding a certain number of results.

[LC03] considers content-based retrieval in hybrid P2P networks where a peer can either be a simple node or a directory node. Directory nodes serve as super-peers, which may possibly limit the scalability and self-organization of the overall system. The peer selection for forwarding queries is based on the Kullback-Leibler divergence between peer-specific statistical models of term distributions. The approach that we propose in this paper also uses such statistical measures but applies them in a much more light-weight manner for better scalability, primarily using bookmarks rather than full index information and building on a completely decentralized directory for meta-information.

Strategies for P2P request routing beyond simple key lookups but without considerations on ranked retrieval have been discussed in [YGM02, CGM02a, CFK03], but are not directly applicable to our setting. The construction of semantic overlay networks is addressed in

[LNS+, CGM02b] using clustering and classification techniques; these techniques would be orthogonal to our approach. [TXD03] distributes a global index onto peers using LSI dimensions and the CAN distributed hash table. In this approach peers give up their autonomy and must collaborate for queries whose dimensions are spread across different peers.

In addition to this recent work on P2P Web search, prior research on distributed IR and metasearch engines is potentially relevant, too. [Ca00] gives an overview of algorithms for distributed IR like result merging and database content discovery. [Fu99] presents a formal decision model for database selection in networked IR. [NF03] investigates different quality measures for database selection. [GBS01, MRYGM01] study scalability issues for a distributed term index. GlOSS [GGMT99] and CORI [CLC95] are the most prominent distributed IR systems, but neither of them aimed at very-large-scale, highly dynamic, self-organizing P2P environments (which were not an issue at the time these systems were developed).

A good overview of metasearch techniques is given by [MYL02]. [WMYL01] discusses specific strategies to determine potentially useful local search engines for a given user query. Notwithstanding the relevance of this prior work, collaborative P2P search is substantially more challenging than metasearch or distributed IR over a small federation of sources such as digital libraries, as these approaches mediate only a small and rather static set of underlying engines, as opposed to the high dynamics of a P2P system.

3 System Architecture

3.1 Rationale and Overview

Figure 1 illustrates our approach which closely follows a publish-subscribe paradigm. We view every peer as autonomous. Peers can post meta-information at their discretion without a globally enforced assignment of directory data onto peers. Our conceptually global but physically distributed directory does not hold information about individual documents previously crawled by the peers, but only very compact aggregated information about the peers' local indexes and only to the extent that the individual peers are willing to disclose to other peers. We use distributed hash tables to partition the term space, such that every peer is responsible for a randomized subset of terms within the global directory. For failure resilience and availability, the entry for a term may be replicated across multiple peers.

Every peer publishes a summary (*Post*) for every term in its local index to the underlying overlay network, which is routed to the peer currently responsible for this term. This peer maintains a *PeerList* of all postings for this term from across the network. Posts contain contact information about the peer who posted this summary together with IR-style statistics (e.g., TF and IDF values [Ch02]) for a term and other quality-of-service measures (e.g., length of the index list for a given term, or average response time for remote queries). Analogously, users can also post their bookmark URLs to the overlay network; this may be only a subset at the user's discretion. Every URL is routed to a peer responsible

for this URL, and that peer maintains a list of peers that have this URL bookmarked. This second form of PeerLists can be either handled in the same hash-key space as the term-based lists or in a second overlay network using the same DHT software with a different hash function.

The querying process for a multi-term query proceeds as follows: First, the querying peer retrieves a list of potentially useful peers by issuing a *PeerList request* for each query term and/or bookmark of the query initiator to the underlying overlay network. Next, a number of promising peers for the complete query is selected from these PeerLists (e.g., based on the quality-of-service measures associated with the Posts). Subsequently, the query is forwarded to these carefully selected peers and executed based on the their local indexes. Note that this communication is done in a pairwise point-to-point manner between the peers, allowing for efficient communication and limiting the load on the global directory. Finally, the results from the various peers are combined at the querying peer into a single result list. Section 4 gives details about the peer selection strategy.

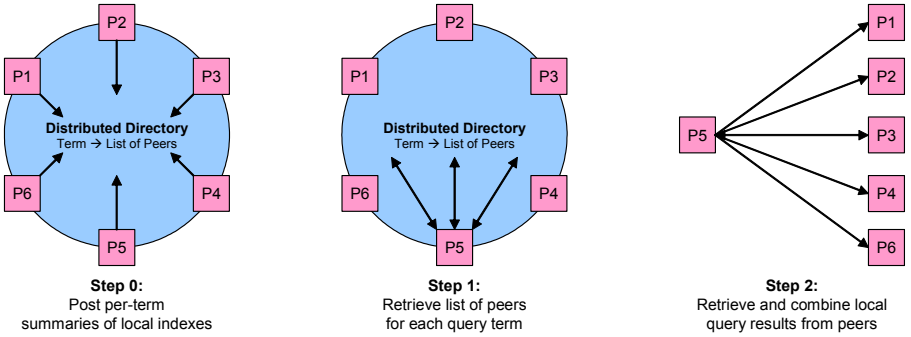

Figure 1: P2P Query Routing

We have chosen this approach because the goal of finding high-quality search results with respect to precision and recall cannot be easily reconciled with the design goal of unlimited scalability, as the best information retrieval techniques for query execution rely on large amounts of document metadata. In contrast, posting only aggregated information about local indexes and bookmarks and executing queries at carefully selected peers exploits global statistical knowledge for good query results while, at the same time, limiting the size of the global directory and, thus, consuming only little network bandwidth. We expect this approach to scale very well as more and more peers jointly maintain this moderately growing global directory.

The approach can easily be extended in a way that multiple distributed directories are created to store information beyond local index summaries or bookmarks, such as information about relevance assessments derived from peer-specific query logs, click streams, or explicit user feedback. This information could be leveraged when executing a query to further enhance result quality.

3.2 Global Directory

In this section we formalize the system architecture that we have introduced in the previous subsection. Let $P := \{p_i | 1 \leq i \leq r\}$ be the set of peers currently attached to the system. Let $D := \{d_i | 1 \leq i \leq n\}$ be the global set of all documents; let $T := \{t_i | 1 \leq i \leq m\}$ analogously be the set of all terms.

Each peer $p_i \in P$ has one or more of the following local data available:

- Local index lists for terms in $T_i \subseteq T$ (usually $|T_i| \ll |T|$).
  The local index lists cover all terms in the set of locally seen documents $D_i \subseteq D$ (usually $|D_i| \ll |D|$).
- Bookmarks $B_i \subseteq D_i$ ($|B_i| \ll |D|$)
  Bookmarks are intellectually selected links to selected documents or other peer profile information and, thus, are a valuable source for high-quality search results as well as for the thematic classification of peers.
- Cached documents $C_i \subseteq D$
  Cached documents are readily available from a peer

Separate hash functions

$$hash_{terms} : T \rightarrow ID$$

$$hash_{bookmarks} : D \rightarrow ID$$

$$hash_{cached} : D \rightarrow ID$$

can be used in order to build conceptually global, but physically distributed directories that are well-balanced across the peers in the ID space.

Given these hash functions that assign identifiers to keys using

$$id := hash_j(k)$$

the underlying distributed hash table offers a function

$$lookup : ID \rightarrow P$$

that returns the peer p currently responsible for an id.

Building on top of this basic functionality, different PeerList requests plr_j with $j \in \{terms, bookmarks, ...\}$ can be defined as functions

$$plr_{terms} : T \times P \rightarrow 2^P$$

$$plr_{bookmarks} : D \times P \rightarrow 2^P$$

$$plr_{cache} : D \times P \rightarrow 2^P$$

that, from a peer p previously determinded using $lookup$, return lists of peers that have posted information about a key with ID id in dimension j. Note that id for a specific key k is unambiguously defined across the directory using $hash_j(\mathrm{k})$.

In order to form a distributed directory, each peer p_i at its own discretion globally posts subsets

$$T_i' \subseteq T_i$$
$$B_i' \subseteq B_i$$
$$C_i' \subseteq C_i \subseteq D$$

(potentially along with further information or QoS statistics) forming the corresponding global directories:

- $systerms : T \rightarrow 2^P$ with $systerms(t) = plr_{terms}(t, lookup(hash_{terms}(t)))$
  This directory provides a mapping from terms to PeerLists and can be used to identify candidate peers that hold index information about a specific term.
- $sysbm : D \rightarrow 2^P$ with $sysbm(d) = plr_{bookmarks}(d, lookup(hash_{bookmarks}(d)))$
  This function provides information about which peers have bookmarked specific documents and is a combination of the above methods analogously to $systerms(t)$
- $syscd : D \rightarrow 2^P$ with $syscd(d) = plr_{cached}(d, lookup(hash_{cached}(d)))$
  This function provides information about the availability of documents in the caches of local peers, which is a valuable information for the efficient gathering of results.

We consider a query q as a set of $(term, weight)$-pairs and the set of available queries as $Q := 2^{T \times \mathbb{R}}$. In order to process a query q, first a candidate set of peers that are confronted with the query has to be determined. This can be done using the functions

$$selection_{terms} : Q \rightarrow 2^P$$
$$selection_{bookmarks} : 2^D \rightarrow 2^P$$
$$selection_{cached} : 2^D \rightarrow 2^P$$

that select candidate subsets for each dimension by appropriately combining the results returned by $systerms$, $sysbm$, and $syscd$, respectively. These candidate subsets are combined (e.g., by intersection or union) using a function

$$comb : 2^P \times 2^P \times 2^P \rightarrow 2^P$$

Combining the above, the final candidate set is computed using a function

$$selection : Q \times 2^D \times 2^D \rightarrow 2^P$$

$$selection(q, B_0'', C_0'') = comb(select_{terms}(q), select_{bookmarks}(B_0''), select_{cached}(C_0''))$$

where $B_0'' \subseteq B_0$ and $C_0'' \subseteq C_0$ are the bookmarks and cached documents, respectively, that the querying peer has chosend to support query execution. For example, a peer may choose its own bookmarks and a sample of its cached documents as B_0 and C_0, respectively.

The execution of a query is a function

$$exec : 2^P \times Q \rightarrow 2^D$$

that combines the local results returned by the peers that are involved in the query execution into one single final result set. Finally, we can define the global query execution function

$$result : Q \times 2^D \times 2^D \rightarrow 2^D$$

that is evaluated as

$$result(q, B, C) := exec(selection(q, B, C), q)$$

$$= exec(comb(select_{terms}(q), select_{bookmarks}(B), select_{cached}(C)))$$

4 Query Routing

Query routing is the problem of finding appropriate peers that can answer the query of a given peer p_0 with high result quality at low execution costs. Our approach is decomposed into three steps:

1. *looking up* candidates that may be selected,
2. *pruning* the set of candidates to a manageable number, and
3. *assessing* the remaining candidates in terms of their benefit/cost ratio, and then choosing the best k peers where k is a system-configuration parameter.

For the first step we can simply look up the global directory to retrieve all peers that contain one of the query keywords, and we can merge the resulting PeerLists to obtain the intersection, i.e., those peers that contain all of the query keywords. For efficiency, we could limit the directory lookup to return only the best peers per keyword, in terms of index list length and other quality measures of peers. However, this may still produce a fairly large candidate set, and the assessment of a peer's information quality for the given query may involve statistical comparisons and computations with non-negligible overhead. This calls for the second step, the pruning of candidates and selection of a much smaller peer set. To this end we consider the ratio $\frac{result\ quality}{execution\ cost}$ as a benefit/cost measure.

We use dynamic estimates of a peer's utilization and resulting average response time as a measure of cost. The necessary load information is disseminated through the global directory; as it does not have to be perfectly up-to-date and accurate we can piggyback much of it upon messages that are needed to maintain the underlying overlay network anyway (e.g., the Chord-style stabilization protocol).

As for the benefit measure, peers are likely to yield high-quality results to the query if they have thematically related but complementary information. Subsection 4.1 elaborates on this approach, based on the bookmarks of the peers. Bookmarks reflect a user's interest profile and are query-independent, so we can precompute various statistical measures and efficiently use them for peer selection. Once we have largely pruned the candidate set, we then switch to a query-oriented model for assessing the remaining peers' benefit and making the final selection. This last stage of our query routing method is presented in Subsection 4.2.

4.1 Bookmark-driven Peer Selection

We measure the thematic similarity between two peers by comparing their bookmarks. More specifically, we compare the peers in two aspects: 1) regarding their URL sets, and 2) regarding the term frequency distributions in the documents referenced by the bookmark lists.

As for term distributions, we use the relative entropy of distribution f with respect to distribution g, also called the Kullback-Leibler distance [Ku59], as a measurement for information inequality. It is defined by

$$KL(f,g) := \sum_x f(x) log \frac{f(x)}{g(x)}$$

where f and g are discrete probability distributions. The relative entropy has important mathematical properties; for example, it is non negative and equals zero if and only if $f = g$. The peer p_0 that issues the query aims to find peers that are thematically related; this suggests that the benefit of a candidate peer p_i is inversely proportional to $KL(B_0, B_i)$. B_0 and B_i denote the term frequency distributions constructed using all documents referenced by the corresponding bookmark lists (and optionally also all hyperlink successors of these pages). Note that this measure does not only reflect the similarity between the bookmark lists themselves, but actually compares the local index contents of peer p_0 with the index contents of peer p_i, simply because the index of a peer p_j has been constructed by crawling the Web with the local bookmarks B_j as crawl seeds. Clearly, comparing bookmarks is much more efficient than comparing entire indexes.

Obviously, if a peer p_i had exactly the same bookmarks as p_0 it would seem to be a perfect match for routing the query to. However, with the same bookmarks, p_i is likely to have crawled more or less the same Web pages as p_0, so it would almost be redundant to query p_i, too. Therefore, our notion of benefit also considers the overlap between the bookmark lists, again as representatives of the peers' actual index contents. We define $overlap$ as

$$overlap(B_0, B_i) := card(B_0 \cap B_i)$$

and we claim that the benefit is inversely proportional to the overlap.

The way we defined overlap prioritizes peers with large bookmark lists; if this is undesirable the overlap can be normalized by dividing it by the cardinality of B_i. Moreover,

instead of using the bookmarks themselves, we could compare the cardinalities of the successors of the Web pages referenced by the bookmarks. This would take into account the fact that sometimes several bookmarks point to the same site or quickly lead to common pages when following their hyperlinks.

These considerations lead to the following definition of the benefit that peer p_0 receives when querying peer p_i:

$$benefit(p_i) := \frac{1}{KL(B_0, B_i)} * \frac{1}{overlap(B_0, B_i)}$$

4.2 Query-oriented Peer Assessment

The bookmark-driven peer selection of Subsection 4.1 is query-independent and can be viewed as a means of establishing a "semantic overlay network" among peers. Now we consider a query q initiated by p_0 and a set of candidate peers p_i selected by the bookmark-driven pruning. Which are the best peers in terms of providing highly relevant results to q and are complementary to the results that p_0 already obtained locally? It seems natural to use a model similar to that of Subsection 4.1, centered on the notions of thematic similarity and overlap.

The similarity could be measured by $KL(q, D_i)$ where D_i is the index contents of peer p_i. However, q is way too small (only a few keywords) for a meaningful statistical comparison and D_i is way too big for an efficient comparison. Therefore, we enhance the model for q by enhancing the term distribution statistics with the best k (e.g., k=10) query results that p_0 found in its local index. This resembles IR methods for pseudo-relevance feedback, but it serves a different purpose here. As for the remote peer's D_i term distribution, we need a sample of D_i as a compact representation. Since we view D_i as being generated by p_i's bookmarks (i.e., the crawl seeds for building the index of p_i), we simply use B_i for this purpose. As the bookmark lists B_i are rather static they can be proactively disseminated among peers offering a potential for inexpensive access by p_0.

Finally, the overlap between p_0 and a candidate p_i, now in the query-specific context, is difficult to assess. It seems, however, that the definition that we used in Subsection 4.1 is still meaningful here; so we simply use the overlap between the bookmark lists B_0 and B_i as a decision criterion.

Putting everything together leads us to finally choosing the peers with the highest benefit/cost ratio with

$$benefit(p_i) := \frac{1}{KL(Q, B_i)} * \frac{1}{overlap(B_0, B_i)}$$

where Q denotes the term distribution in the best local matches to q.

5 Prototype Implementation

Figure 2 illustrates the architecture of a single peer as part of our distributed system. Each peer works on top of the global directory which is organized as a distributed hash table (DHT) that provides mappings from terms and bookmark URLs to peers by returning a *PeerDescriptor* object representing the peer currently responsible for a term or a directory entry for a URL. A *Communicator* can be established to send messages to other peers. Every peer has an *Event Handler* that receives incoming messages and forwards them to the appropriate local components.

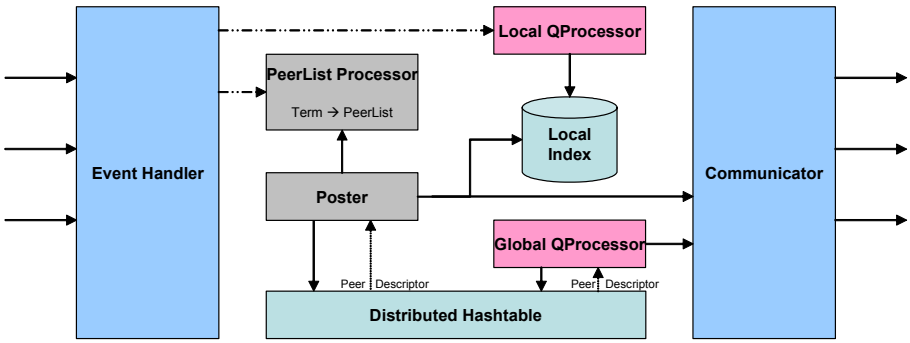

Figure 2: System Architecture

Every peer has its own local index that can be imported from external crawlers and indexers. The index is used by the *Local QueryProcessor* component to answer queries locally and by the *Poster* component to publish per-term summaries or bookmark URLs (*Posts)* to the global directory. To do so, the Poster uses the underlying DHT to find the responsible peer; the *PeerList Processor* at this peer maintains a PeerList of all Posts for this term or bookmark URL from across the network. When the user poses a query, the *Global QueryProcessor* component analogously uses the DHT to find the responsible peer and retrieves the respective PeerLists from the PeerList Processors using Communicator components. After running the peer selection and assessment strategies on these lists, the Global QueryProcessor forwards the complete query to selected peers, which in turn process the query using their Local QueryProcessors and return their results. Finally, the Global QueryProcessor merges these results and presents them to the user.

We have built a prototype system that handles the above procedures. Our system uses a Java-based reimplementation of Chord [SMK+01] as its underlying DHT, but can easily be used with other DHT's providing a $lookup(key)$ method. Communication is conducted socket-based, but Web-Service-based [ACK04] peers can easily be included to support an arbitrarily heterogeneous environment. The local index is stored in a database, using standard IR measures, such as TF*IDF and PageRank-style authority. Result ranking is currently based on a smoothed TF*IDF quality measure. Figure 3 shows a screenshot of the user interface of our prototype. The user creates a peer by either creating a new Chord ring or by joining an existing system. Both actions require the specification of a local Chord port for communication concerning the global directory and a local application port

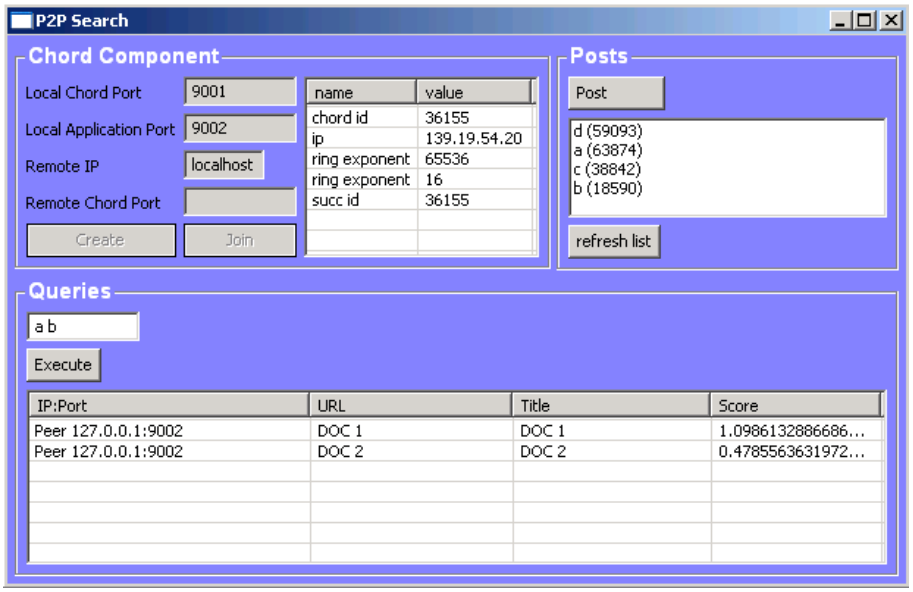

Figure 3: Prototype GUI

for direct peer-to-peer communication. The join operation requires additional information on how to find an already existing peer. Status information regarding the Chord ring is displayed and continuously updated. The Posts section of the GUI provides information about the terms or bookmark URLs that a peer is currently responsible for, i.e., for which it has received Posts from other peers. The button *Post* posts the information contained in the local index to the DHT. The Queries section can be used to execute queries with multiple keywords entered into a form field. After query execution, the results obtained from the system are displayed order by their scores.

6 Concluding Remarks

We have implemented the presented system architecture for P2P Web search as a framework for query routing and query execution strategies and a platform for experimentation. We plan to experiment with the bookmark-driven routing strategy developed in this paper, and we expect that the experiments will lead to new insights that should affect our strategies. This work in progress raises the following key issues for future research:

- How can we guarantee that the overhead of our routing strategy is bounded and can be adapted to the high dynamics in a P2P network (e.g., automatically throttled under high load conditions or in the presence of many failures)?
- How should we precompute, cache, and proactively disseminate the information that is needed by the proposed query routing strategy?

- How can we factor quality of service measures on the peers' information, such as score distributions in index lists or PageRank-style authority distributions, into the proposed bookmark- and query-driven routing strategy?

References

[ACK04] Alonso, G., Casati, F., and Kuno, H.: *Web Services - Concepts, Architectures and Applications*. Springer. Berlin;Heidelberg;New York. 2004.

[APHS02] Aberer, K., Punceva, M., Hauswirth, M., and Schmidt, R.: Improving data access in p2p systems. *IEEE Internet Computing*. 6(1):58–67. 2002.

[Bl70] Bloom, B. H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*. 13(7):422–426. 1970.

[Ca00] Callan, J.: Distributed information retrieval. *Advances in information retrieval, Kluwer Academic Publishers*. S. 127–150. 2000.

[CAPMN02] 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. Technical Report DCS-TR-487. Rutgers University. September 2002.

[CFK03] Cohen, E., Fiat, A., and Kaplan, H.: Associative search in peer to peer networks: Harnessing latent semantics. In: *Proceedings of the IEEE INFOCOM'03 Conference, April 2003*. April 2003.

[CGM02a] Crespo, A. and Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: *Proc. of the 28th Conference on Distributed Computing Systems*. July 2002.

[CGM02b] Crespo, A. and Garcia-Molina, H.: Semantic Overlay Networks for P2P Systems. Technical report. Stanford University. October 2002.

[Ch02] Chakrabarti, S.: *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann. San Francisco. 2002.

[CLC95] Callan, J. P., Lu, Z., and Croft, W. B.: Searching distributed collections with inference networks. In: *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. S. 21–28. ACM Press. 1995.

[Fa99] Fagin, R.: Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.* 58(1):83–99. 1999.

[Fu99] Fuhr, N.: A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*. 17(3):229–249. 1999.

[GBS01] Grabs, T., Böhm, K., and Schek, H.-J.: Powerdb-ir: information retrieval on top of a database cluster. In: *Proceedings of the tenth international conference on Information and knowledge management*. S. 411–418. ACM Press. 2001.

[GGMT99] Gravano, L., Garcia-Molina, H., and Tomasic, A.: Gloss: text-source discovery over the internet. *ACM Trans. Database Syst.* 24(2):229–264. 1999.

[Ku59] Kullback, S.: *Information Theory and Statistics*. Wiley. New York. 1959.

[LC03] Lu, J. and Callan, J.: Content-based retrieval in hybrid peer-to-peer networks. In: *Proceedings of the twelfth international conference on Information and knowledge management*. S. 199–206. ACM Press. 2003.

[LNS+] Löser, A., Naumann, F., Siberski, W., Nejdl, W., and Thaden, U.: Semantic overlay clusters within super-peer networks. In: *Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing, 2003 (DBISP2P 03)*. S. 33–47.

[LNS96] Litwin, W., Neimat, M.-A., and Schneider, D. A.: Lh* – a scalable, distributed data structure. *ACM Trans. Database Syst.* 21(4):480–525. 1996.

[MRYGM01] Melnik, S., Raghavan, S., Yang, B., and Garcia-Molina, H.: Building a distributed full-text index for the web. *ACM Trans. Inf. Syst.* 19(3):217–241. 2001.

[MYL02] Meng, W., Yu, C. T., and Liu, K.-L.: Building efficient and effective metasearch engines. *ACM Computing Surveys*. 34(1):48–89. 2002.

[NF03] Nottelmann, H. and Fuhr, N.: Evaluating different methods of estimating retrieval quality for resource selection. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. S. 290–297. ACM Press. 2003.

[RD01] Rowstron, A. and Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*. S. 329–350. 2001.

[RFH+01] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S.: A scalable content-addressable network. In: *Proceedings of ACM SIGCOMM 2001*. S. 161–172. ACM Press. 2001.

[RV03] Reynolds, P. and Vahdat, A.: Efficient peer-to-peer keyword searching. In: *Proceedings of International Middleware Conference*. S. 21–40. June 2003.

[SBG+] Sizov, S., Biwer, M., Graupmann, J., Siersdorfer, S., Theobald, M., Weikum, G., and Zimmer, P.: The bingo! system for information portal generation and expert web search. *First Semiannual Conference on Innovative Data Systems Research (CIDR), Asilomar(CA), 2003*.

[SMK+01] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*. S. 149–160. ACM Press. 2001.

[SMW+03] Suel, T., Mathur, C., Wu, J., Zhang, J., Delis, A., Kharrazi, M., Long, X., and Shanmugasunderam, K.: Odissea: A peer-to-peer architecture for scalable web search and information retrieval. Technical report. Polytechnic Univ. 2003.

[TXD03] Tang, C., Xu, Z., and Dwarkadas, S.: Peer-to-peer information retrieval using self-organizing semantic overlay networks. In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. S. 175–186. ACM Press. 2003.

[VBW98] Vingralek, R., Breitbart, Y., and Weikum, G.: Snowball: Scalable storage on networks of workstations with balanced load. *Distributed and Parallel Databases*. 6(2):117–156. 1998.

[WMYL01] Wu, Z., Meng, W., Yu, C. T., and Li, Z.: Towards a highly-scalable and effective metasearch engine. In: *World Wide Web*. S. 386–395. 2001.

[YGM02] Yang, B. and Garcia-Molina, H.: Improving search in peer-to-peer networks. In: *Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*. S. 5–14. IEEE Computer Society. 2002.