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

Matthias Bender, Sebastian Michel, Gerhard Weikum, Christian Zimmer

2004
Bookmark-driven Query Routing in Peer-to-Peer Web Search *

Matthias Bender, Sebastian Michel, Gerhard Weikum, Christian Zimmer
{mbender, smichel, weikum, czimmer}@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 thematical 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 contrast to a centralized web search engine, our P2P approach can additionally 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 local 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 decentral-
lized 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 organi-
zed 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 pro-
totype 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 [SMW03] 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-keyword 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
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. GIOSS [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

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 bookmarked this URL. 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 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](image)

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.
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 $\frac{\text{result quality}}{\text{execution cost}}$ 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 term distribution of the local index contents of peer $p_0$ with the term distribution of 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. If bookmark collections are similar regarding their term distribution, we also expect the resulting indexes to be thematically related. 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 $p_i$. 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

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

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 $p_i$:

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

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” [LNS+, CGM02b] 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

\[
\text{benefit}(p_i) := \frac{1}{KL(Q, B_i)} \times \frac{1}{\text{overlap}(B_0, B_i)}
\]

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 \textit{PeerDescriptor} object representing the peer currently responsible for a term or a directory entry for a URL. A \textit{Communicator} can be established to send messages to other peers. Every peer has an \textit{Event Handler} that receives incoming messages and forwards them to the appropriate local components.

Every peer has its own local index, containing standard IR measures like TF and IDF, that can be imported from external crawlers and indexers. The index is used by the \textit{Local QueryProcessor} component to answer queries locally and by the \textit{Poster} component to publish per-term summaries or bookmark URLs (\textit{Posts}) to the global directory. To do so, the Poster uses the underlying DHT to find the responsible peer; the \textit{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 \textit{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. 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 for direct peer-to-peer communication. Joining an existing system 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 ordered by their scores.

6 Concluding Remarks

We have implemented the system architecture presented in this paper. It offers a P2P Web search framework for query routing and query execution strategies and a platform for experimentation. We plan to experiment with the routing strategies 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 strategies?

References


