# Routing of Structured Queries in Large-Scale Distributed Systems

Judith Winter
Department of Computer Science
University of Frankfurt, Germany
winter@tm.cs.uni-frankfurt.de

## ABSTRACT

In order to search XML-document collections, structural information – given by a user in the form of a structured query or provided by the self-describing structure of XML-documents – have been used in the past years to improve Information Retrieval (IR) quality in terms of recall and precision. However, all known approaches have only been used in classical client-/server (C/S) architectures. None have ever been applied to improve retrieval in large-scale distributed systems such as Peer-to-Peer (P2P) networks, where efficiency issues have to be dealt with carefully, e.g. in order to reduce communication overhead between distributed nodes. As P2P networks can be considered promising alternatives to C/S-systems for storing large amounts of information including XML-documents, possibilities for improving the retrieval in such networks should be investigated.

In this paper, we concentrate on query routing in such a scenario and raise the question, how structured queries can be routed in a highly distributed environment so as to increase both efficiency and effectiveness. We provide an infrastructure for investigating this question and propose techniques for performing routing based on a mixture of document-, element-, collection- and peer-evidence. We also report on preliminary evaluation results with the INEX collection.

## Categories and Subject Descriptors

H.3.3 [**Information Search&Retrieval**]: Retrieval models, Search process; H.3 [**Information Storage&Retrieval**]: Indexing Methods; C.2 [**Communication/Networking&IT**]: Distributed Systems; E.1 [**Data Structures**]: Distributed data structures

## General Terms:

Algorithms, Performance

## Keywords

Information Retrieval, Peer-to-Peer, XML Information Retrieval, Distributed Search, XML-Retrieval

## 1. INTRODUCTION

*Peer-to-Peer Systems* are an emerging infrastructure for distributed computing, in which large sets of equal and autonomous nodes – the peers – are pooled together to share resources such as storage and power capacity. Unlike traditional client/server systems, there is no central control and peers can decide autonomously which services and resources to contribute to the system. Due to their self-organization, P2P systems bear the potential to realize robustness and fault-tolerance, and may scale to theoretically unlimited numbers of participating nodes.

Algorithms for *Information Retrieval in P2P networks* (P2P-IR) are a recent field of research. Existing commercial or open source search engines with efficient response times only exist for simple pattern matching between queries and documents: documents are located in the P2P network by their name and only if they match the query exactly. Research approaches for content-oriented search in P2P can be classified into unstructured systems with a clever replication of objects, such that only a few peers need to be contacted, into semantic routing based on summaries where the network is navigated using hints such as search history, and into structured systems with distributed indexes, i.e. techniques based on distributed hash tables (DHTs) [20]. In P2P-networks based on DHTs, lookup of resources can be performed efficiently in log(n) hops on average by structuring the network such that a set of peers act on a distributed data structure with well-defined operations and supporting joining, leaving, and routing between the peers [8]. Chord is a P2P protocol that implements such a DHT [24]. The general idea of P2P-IR and criteria for the classification of such approaches are presented in [20]. An overview of how to look up data in P2P systems based on DHTs can be found in [4]. A summary of approaches for IR in unstructured P2P networks is given in [28]. An efficient P2P search engine using IR techniques is presented in [18]. They propose a key-based indexing strategy instead of single-term indexing, with keys as term sets that appear only in a restricted number of documents, thus being highly discriminative keys (HDKs).

Searching document collections based on Information Retrieval (IR) is a useful feature for the user. However, one of the main difficulties in distributed systems is searching efficiently while maintaining scalability [12]. Even in centralized approaches, query processing currently consumes a significant amount of resources; additional challenges come up in highly distributed P2P networks with bandwidth and latency constraints. The additional task of efficiently locating useful index information arises, before relevant results can be ranked and retrieved. In order to reduce bandwidth consumption and to guarantee scalability, P2P-IR solutions use only selected information and hence inevitably lose retrieval quality in terms of precision and recall. Selection techniques usually involve contacting only a carefully chosen subset of participating peers or top-k pruning (cutting off posting lists) [29]. For example, [15] presents a framework for distributed top-k query processing in a P2P setting for text documents using filters. Peer descriptions to efficiently reduce network traffic when routing in file sharing systems are used in [27] but no IR techniques for content-oriented search are applied. In general, various techniques from the area of IR, databases, and distributed systems are tried in order to minimize quality loss. None of them takes into account the structure of XML-documents yet.

In order to search for and in XML-documents, quite a few approaches have been proposed in the past that suggest that structural features are a good source for achieving more precise and focused IR results. Methods for exploiting the self-describing structure of XML include weighting diverse parts of documents differently. Furthermore, queries can contain structural constraints about what to search and what to retrieve. Retrieval units can consist of entire documents or only the most relevant parts of a document [13]. Overviews of existing approaches that apply IR methods to the retrieval of XML-documents can be found in [2] and [16]. Evaluation has shown successful application of XML IR e.g. at INEX, the INitative for the Evaluation of XML-Retrieval [9]. However, all of these approaches have only been applied in traditional c/s-architectures. None have ever been used to improve XML-retrieval in large-scale distributed systems.

So far, no P2P solutions for IR of XML-documents exist. *Schema-based P2P-networks* [23] provide techniques for the lookup of semi-structured documents, e.g. by using XML P2P databases. They allow for exact or partial matches and consider hints about the desired document structure but do not yet provide the means to compute the relevance of documents. For example, [1] proposes a scalable solution to query XML data in P2P networks based on DHTs. However, their routing is based on structure instead of content. A survey about indexing, query routing, and query processing of XML data in a P2P setting is conducted in [10], though none of the surveyed approaches features relevance computing with IR techniques.

We therefore propose to investigate, if and how structural features can be used to perform and improve IR of XML-documents in P2P networks. How can structured queries be routed and executed in a distributed setting such as to increase efficiency and effectiveness? Can the structure that is given both in the query and in the document help to improve the routing and ranking by reducing the routing effort, that is, the amount and size of messages between the peers, in order to route a given query to those peers that can rank possible relevant results, without losing precision and recall?

In this paper, a first P2P-IR approach for XML-documents is proposed based on the architecture described in [26]. We provide a distributed infrastructure to investigate the raised research question. Furthermore, techniques for the routing of queries that consist of both content and structure are proposed such that the given structure can be used in the process of selecting useful information. As the infrastructure is based on a DHT, locating known resources is not a problem. The challenge therefore is to deal with the huge posting lists of large-scale document collections. We provide mechanisms to choose promising postings according to evidence from XML elements, XML documents, the document collection, and storing peers. Structural features are exploited to minimize retrieval quality loss when building term combinations at indexing time, when pruning distributed posting lists; when selecting peer subsets; and when performing the final ranking. The proposed infrastructure is used to evaluate these techniques and some preliminary results are presented.

## 2. PROPOSED RESEARCH

The impact of using structural information for retrieving XML-documents will be examined in a P2P environment, where – due to performance issues, communication costs and lack of a central index – only selected information can be used. An analysis will be conducted on how XML IR techniques can be applied in such a scenario to improve the routing of structured queries.

### 2.1 Assumptions and Requirements

Putting much effort into the indexing process can buy performance at querying time and vice versa. The decision as to which aspect to focus on usually depends on the application and the environment – highly dynamic systems require quick and cheap indexing, whereas stable environments can afford a more extensive indexing stage, particularly if there are many more queries to be executed than new documents to be indexed. In this proposal, a dynamic but relatively stable P2P environment is assumed – peers will join and leave the network dynamically but often stay online for long periods, especially when providing large document collections such as digital libraries. Thus the aim is to allow an efficient retrieval at querying time by accepting a more intensified indexing. However, the indexing must scale, i.e. a linear growth of indexed documents should lead to a logarithmical growth of resource usage only.

The target collection consists of documents structured with XML. Most XML-Retrieval algorithms are aimed at collections with a homogeneous schema, as is the case, for example, with most INEX tracks. In practical environments, such a restriction will hold in rare cases only [11]. Especially in distributed systems such as P2P networks, collections usually are not homogeneous but based on a wide spectrum of schemas. Accordingly, the underlying algorithms should be capable of dealing with heterogeneous collections and not rely on a consistent document schema. In particular, no assumptions can be made about possible elements and tags. Instead, functions have to be applied that can detect and compute structural similarity.

In this proposal, a user model is assumed where the user has only a vague idea about the contents and structure to be retrieved, and is thus interested in everything relevant to the given query. This user is not an expert with the ability to express complex queries (e.g. in XQuery style), or with exact knowledge of underlying document schemas. On the other hand, the user is assumed to be informed well enough to use structural hints in addition to the query contents. Thus, the option of using structural hints should be supported. This exceeds the plain use of metadata fields such as author or title. The applied IR model must be able to handle structural constraints in addition to content hints.

A particularly problematic task in P2P-IR systems is the support of multi term queries, where information about more than one query term is needed. This is the case for more than 80% of all queries [17]. Users who are able to specify queries by structural hints are likely to use more terms than average to specify a query. Accordingly, special effort should be put into handling and optimizing multi term queries. Most existing P2P approaches produce large amount of traffic by locating and sending long posting lists for the different terms over the network and perform expensive joins of these posting lists at querying time. However, quick access to combinations of index terms is desirable, especially to reduce bandwidth consumption and support efficient querying.

Communication overhead is a cost factor always to be carefully considered in distributed systems. High bandwidth consumption is one of the major bottlenecks for P2P-IR in general [12] and not limited to the problem of multi term queries. Accordingly, a minimization of the number of messages sent over the network should be aimed for, and the size of these messages should be as small as possible. This is especially true for the peer selection process, i.e. for the decision on which peers hold valuable information for answering a given query and can participate in a parallel relevance computing.

Methods for reducing bandwidth consumption such as pruning of long posting lists, top k approaches, and peer selection usually cause a reduction of retrieval quality, since good information might be pruned or left unconsidered. The techniques used for posting list management and peer selection should therefore effect minimum quality reduction and consider measures to improve quality such as using the structure of XML-documents.

## 2.2 Improving IR Quality

How can structural information help to efficiently route structured queries and thus improve the retrieval of XML-documents in Peer-to-Peer systems in terms of precision, recall, and specificity? Does structure help routing queries in an environment in which not all information is centrally accessible? How can XML IR techniques be used in a P2P network while aiming at a minimization of network traffic and regarding performance aspects?

To investigate these questions, a search engine infrastructure has been developed that uses structural hints explicitly given by the user and implicitly by using the self-describing structure of XML-documents. More focused and specific results are obtained by retrieving results that can be either whole XML-documents or the most relevant elements in these documents.

The indexing includes both content and structural information. To support efficient execution of multi term queries, we adapted the approach on highly discriminative keys (HDKs) in [18] to XML-documents as described in section 3.3. Index keys thus consist of rare combinations of (content, structure)-tuples which we call XTerms. Performance is increased by using only fixed-sized posting lists: frequent index keys are combined with each other until the new combination is rare, with a posting list size under a pre-set threshold. Posting lists are sorted based on a scoring that uses traditional IR measures such as term frequency for a document posting but also weights for retrieval units of this document. Postings get a higher score if they regard to a documents on a peer with good collections regarding the current index key and with good peer performance metrics such as response times. When extracting the posting list for a specific query, a re-ordering or pre-ranking is performed that is based on the structural similarity between key and query.

The ranking is performed in parallel on those peers that were selected as holding information about potential relevant documents and retrieval units. An extension of the vector space model is used for the computation, where different weights are applied for different structures of the same content. For judging the relevance, content relevance and structural similarity between a given query and a potential relevant retrieval result are taken into account.

## 3. A P2P-SEARCH ENGINE FOR XML IR

In this section, a distributed infrastructure for a P2P search engine is proposed that uses the techniques described above to efficiently route structured queries.

## 3.1 Infrastructure

Figure 1 shows the architecture of each peer which consists of three layers: the application layer with an evaluation component for test runs, an interface to other applications, and a graphical user interface for indexing documents of the local file system or querying and presenting the retrieval results; the IR layer with components for indexing, retrieval, routing, ranking, and index storing; and the P2P layer that is used for communication between distributed components, e.g. for requesting information or receiving ranked results from other peers in a structured network.

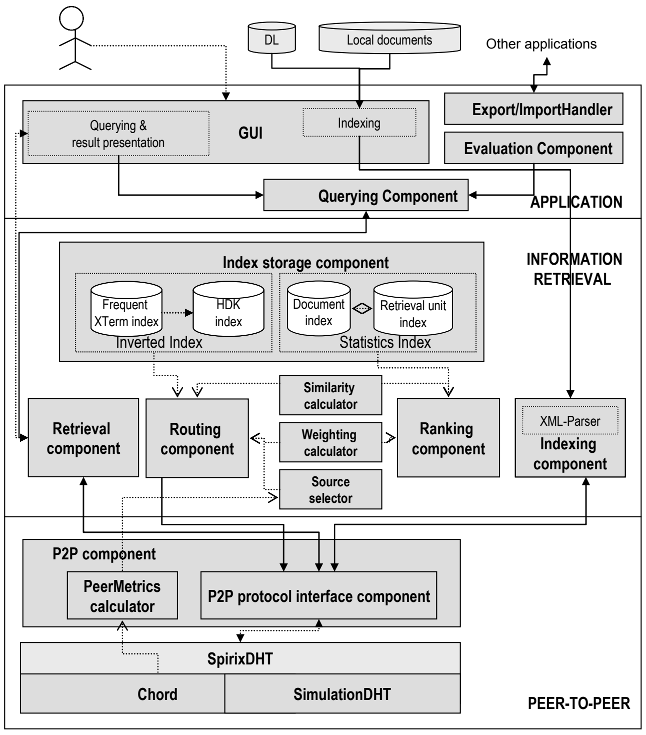

**Figure 1. Architecture of a peer for XML IR**

The indexing workflow is organized as follows: Documents either stored in the local file system or in a digital library are indexed by the local indexing component. All extracted information is stored in the distributed indexes; this includes inverted lists for HDKs and frequent XTerms, as well as document information and various term statistics for the document itself and for its potential retrieval units.

The execution of a structured query *q* is explained in section 3.4 in more detail. Roughly, it will be performed by splitting the query into keys that are either HDKs or frequent XTerms. For each query key $k_i$, the retrieval component of the querying peer will send a request to the peer holding $k_i$'s posting list. This peer will then select potentially relevant documents from the posting list and will redirect the query to peers with term statistics about the selected documents $d_j \in$ posting list($k_i$). This selection process takes into account the structural similarity between query and document, the weight of the document, the weight of potentially relevant retrieval units in this document, the weight of the collection of the peer assigned to the document as well as query-independent evidence about how well this peer performs in a technical sense. Finally, the query will arrive at peers which, in their locally stored part of the distributed index, hold relevant information for documents that contain at least one key $k_i \in q$. The relevance of retrieval units of those documents will be computed by using information such as term weights and by taking into account structural similarity. Ranked results will be sent back to the retrieval component of the querying peer, containing the locations of the most relevant retrieval units. The best results will be merged and prepared for visualisation in the graphical user-interface so that the user can access the retrieval units directly form the peers on which they reside.

## 3.2 Querying

One possibility of using structure is to enable users to enter structural constraints together with the query content. This is common in metadata search engines, though users are restricted to popular fields such as title or author name. In XML IR, content-and-structure (CAS)-queries allow the input of any structural hint whereas content-only (CO)-queries only enable the input of content. The proposed search engine can process CAS-queries as well as CO-queries. Evaluation has to show, if CAS-queries can improve the retrieval results without significantly increasing communication overhead.

We assume the structure given in a CAS-query to be a vague hint in contrary to a more database-related view of seeing structure as a constraint. We furthermore denote structure as the path from document root to the term, expressed with an adaptation of XPath. We do not distinguish between elements with the same name if they are based on the same level of the document-tree and have the same parent-node. Furthermore, stemming, stopword removal, and mapping of structure are performed in order to decrease the number of different structures and to support an efficient calculation of structural similarity.

We define a query as a set of weighted XTerms where the default weight of each XTerm is 1. If the user does not give a structural hint, the structure of the XTerm is empty. If several hints are given (e.g. the user is not sure whether he wants to retrieve a book or an article), an XTerm for each structure is created with a weight divided by the amount of given structures. A query for images of apples in articles about computers could like this: q = {(apple, \image, 5), (computer, \article, 1)}.

## 3.3 Indexing

For the use of structural information in the retrieval and ranking process, the indexing process has to store a term's structure together with its contents, e.g. the XML element that contains the contents. We thus denote the smallest unit to be indexed as an XTerm which we define as a tuple of content and structure. The posting list of an XTerm is limited to a threshold of $PL_{max}$ entries in order to avoid long posting lists that would slow down the retrieval process and increase the amount of query redirections. Only those XTerms whose posting list does not exceed $PL_{max}$, i.e. whose global collection frequency (which is defined in formula 1) is lower than the pre-set threshold, are considered specific and discriminative with regard to the collection; they are indexed as HDKs. If an XTerm is frequent, its posting list will be pruned, so that it contains only the best $PL_{max}$ documents; XTerm and posting list will be stored in the frequent XTerm index in case they can be used to answer a single term query or they can complement a multi term query that is not totally covered by HDKs. To support multi term queries, frequent XTerms of the same document (and the same text window, indicating they appear in the same context and might be queried together) are combined with each other until the posting list for the new combination is rare, i.e. its size does not exceed $PL_{max}$. The new combination can then be stored as a HDK.

Finding the optimal value for $PL_{max}$ is equivalent to finding a balance between two extremes: Either setting $PL_{max}$ extremely low, so that nearly all possible term combinations that might occur together in the same query (impossible to predict!) will be indexed. This demands much effort at indexing time and will incredibly inflate the index space. The other extreme consists in setting $PL_{max}$ extremely high such that all terms will be indexed individually. Their posting lists would then have to be pruned in order to gain a scaling system not overloaded by too much traffic – a pruning that would lead to loss of information and loss of retrieval quality.

Increasing the value of $PL_{max}$ increases the probability that a query $q$ meets the best case: $q$'s key combination has already been computed at indexing time, i.e. $q$ as a whole is stored as a HDK such that an appropriate posting list with the best $PL_{max}$ documents is directly accessible and only the relevance values of their retrieval units have to be computed. This will significantly reduce effort at querying time and will result in the best retrieval quality, as no information is lost by pruned posting lists. Decreasing $PL_{max}$ will save indexing effort and space usage but increase the probability that $q$ meets the worst case: all query terms are frequent XTerms, which results in a potentially high number of query routings at querying time and negatively affects recall and precision by using pruned posting lists.

$$\begin{aligned} CF(t) = {} & \alpha\big(\lambda * DF(t.c) + (1-\lambda) * DF'(t.c)\big) \\ & + (1-\alpha) DF(t) \end{aligned}$$

$$\text{where} \quad DF'(t.c) = \frac{DF(t.c)}{|t_i|} \qquad \forall \text{ XTerms } t_i \text{ with } t_i.c = t.c$$

**Formula 1. Global collection frequency CF of XTerm t**

We define the global collection frequency (CF) of an XTerm $t$ as in formula 1 and use it to detect if $t$ is frequent or rare. CF is proportional to the global document frequency (DF) of $t$, e.g. the number of documents in the collection that $t$ appears in. We did not take into account its global element frequency, e.g. the number of elements of the collections that $t$ appears in (which reflects the fact that results can be either whole documents or retrieval units, e.g. elements). Our reason is that all elements that are not documents themselves are expected to be strongly linked to the documents they appear in. Thus, no postings for retrieval units are stored. [14] and others propose a strong correlation between a document and its contained retrieval units. We follow this lead and assume that most potential relevant retrieval units can be found by identifying their root-documents. What we do take into account is the document frequency of *t*, the DF of *t*'s content (*t.c)* as well as $|t_i|$ which is the number of different structures found for *t*'s content. This number can indicate how discriminative *t* is. Therefore, an XTerm for which the index holds 10 other XTerms with the same content but different structure will have a smaller CF than if it was the only XTerm with exactly this content. With parameter $0 \le \alpha \le 1$, the impact of DF(*t*) versus DF(*t.c*) can be controlled. Parameter $0 \le \lambda \le 1$ sets the impact of $|t_i|$ on *t.c*'s discriminative power.

The decision about which frequent XTerms should be combined together is not trivial. The original HDK-approach constructs HDKs of terms that appear within a window of *w* consecutive terms. [22] propose an improvement of this by using query log analysis. As there are no adequate logs for structured IR queries yet, we follow an approach similar to [5], where the gap between two terms separated by an XML tag, e.g. appearing in different XML elements, accounts for a certain amount of penalty points (equivalent with the same amount of consecutive terms). Using this, we can decrease the probability that they will be combined with each other. Accounting different XML elements with different penalty points might also be a solution. So far, we simply account a gap with (*w*/*p*) penalty points where $p \in \{1,5,10,20\}$.

So far, the same value for parameter $pl_{max}$ is used for all XTerms. Improvements could be made by choosing different truncation levels for important/unimportant terms.

## 3.4 Routing

To find relevant results, a given query *q* must be routed to peers that hold evidence about the relevance of documents and retrieval units in form of term statistics that have been stored at indexing time. In the proposed infrastructure, this routing will be performed in two steps. First, the inverted lists of all query terms $k_i$ must be located and merged with each other. From the resulting final posting list of *q*, promising entries must be selected for further investigation, e.g. requests must be sent to peers holding term statistics of the selected documents $d_j \in$ posting list(*q*) and their retrieval units.

Locating the inverted lists in step 1 presents no problem in a DHT-based infrastructure where items identified by a key can be located in log(n) hops between peers. We support efficient routing by storing HDKs and frequent XTerms on the same peer if they relate to the same content. This is handled by assigning peers to the hash-value of the content of the first XTerm in a key whose combined XTerms are ordered alphabetically. For load-balancing purposes, it might be useful to distribute some of the HDKs constructed of more than one XTerm on neighbouring peers. All posting lists of XTerms with the same content will be stored on the same peer and can be easily located together, so that postings can be selected not only from those lists where the structure of a query key exactly matches the structure of the XTerm of the inverted list but also from lists with similar structure.

The posting lists of all query terms will be processed in a pipelining manner as in [3], after sorting the query keys according to their posting list length. Thus, a querying peer $p_q$ will first send the sorted query to the peer $p_{i1}$ with the shortest posting list. After selecting promising postings from this list, $p_{i\_1}$ will mark the first query key as processed and send the query together with the selected postings to peer $p_{i2}$ assigned to the next query key and so on. Thus $p_{i\_n}$, the last peer in this pipeline, can select the final posting list and request the relevance computing of the selected postings. For this second step of the routing, the statistics for each posting must be located. Again, this can be done easily as they are distributed according to the hash-value of the assigned unique document ID and thus can be found with DHT-techniques in log(n) hops.

Figure 2 illustrates the routing process of a query *q*. Let q consist of keys $k_2$, $k_1$ and $k_3$ with each key being either a HDK or a frequent XTerm. Let $k_1$ be the key with the shortest posting list. The final posting list for all three keys is then created by processing their posting lists and selecting postings form merged lists in a pre-ranking. The final ranking is performed on peers assigned to selected documents such as peer $p_{d\_n}$ assigned to document $d_n$ of the final posting list.

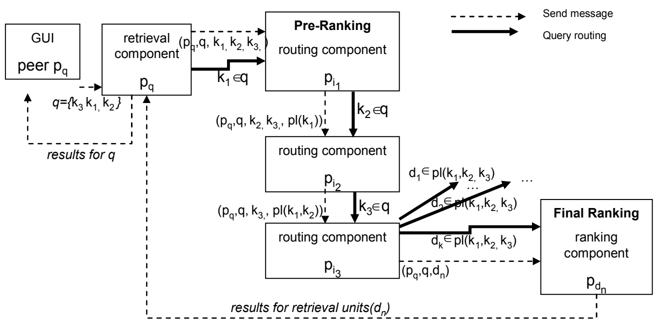

**Figure 2. Pipelined routing of a query with pre-ranking**

The problem that arises from query routing in a DHT-based system is narrowed down to the challenge of determining which postings to select from the inverted lists. Both efficiency and effectiveness issues have to be considered. In a large-scale distributed system with long posting lists due to a huge collection, it would not be feasible to select all postings. This would result in too much traffic by sending large posting lists between the peers and by routing the query to all peers assigned to the selected documents of the final posting list, not to mention the costs for expensive joins at querying time in order to merge the lists of different query keys that have not been pre-computed as HDKs at indexing time. The system thus would not be efficient and scalability could not be guaranteed. On the other hand, the retrieval quality in terms of recall and precision will suffer if too few or the wrong postings are selected.

We propose the following for selecting postings:

- Pre-computation of posting lists for HDKs as described in section 3.3. The posting list of a HDK *h* is constructed from the union of all posting lists for the different XTerms $t_i$ that are part of *h*. For each $t_i$ there will be a set of posting lists that relate to the same content but different structures. Only those postings will be stored in the posting list of an XTerm that occur in at least one other posting list of each set in *h* (as only those XTerms are combined with each other that appear in the same document). Thus for the content of each $t_i$, there exists a posting list for each of its structures containing only postings that also appear in all other sets in *h*. Thus, a posting list for *h* is created as the intersection of the posting lists of all contained XTerms regarding their content while at the same time each XTerm keeps its own posting list to support posting selection according to structural similarity as described later in this section.

- Keeping pruned posting lists for frequent XTerms with the cut-off point being $PL_{max}$ based on the global collection frequency as in formula 1.
- Ordering of posting lists by a query-independent score as in formula 2 that is computed at indexing time and takes into account weights for both document $d_i$ and its highest scoring retrieval unit as well as a score for the peer assigned to $d_i$..
- Selecting the top $n$ results of such ordered posting lists by performing a pre-ranking at querying time that adds up all scores of those postings from lists with a certain structural similarity between query key and posting key. This similarity is computed such as with formula 3.

Formula 2 shows the scoring function used for posting list sorting as described in [25]. It is based on the weights for both document $d_i$ and its best scoring retrieval unit $ru_{best}$. $Score_k(d_i)$ for key $k$ is computed at indexing time and based on an adaptation of the BM25E formula [21] to XML IR such that different elements can be weighted differently, e.g. a title will be given a much higher weight than a footnote. Also, the weight of an XTerm is computed as a combination of element weight and path weight. Parameters include the global collection frequency (CF) of $k$; the term frequency of $k$ in $d_i$; the term frequency of $k$ in $d_i$'s best scoring retrieval unit $ru_{best}$, or alternatively: $tf$ of the best $x$ retrieval units; as well as a *peerScore* that is high for peers with good collections regarding the current key (e.g. computed with CORI [6]) and with good peer metrics such as available bandwidth and online time (for more information on the peer metrics see [25]).

$$\begin{aligned} score_k(d_i) = \ & \delta 1 * BM25E(\, tf_k(d_i),\, CF_k) \\ & + \ \delta 2 * BM25E(\, tf_k(ru_{best}), CF_k) \\ & + \ \delta 3 * peerScore_k(d_i) \end{aligned}$$

**Formula 2. Scoring posting list entries for ordering**

Structural information given in the CAS query can also be useful for posting selection. Better results can be expected by re-ordering the posting list by a structure-based pre-ranking than if the top $n$ documents scoring best were simply selected according to formula 2.Therefore, a pre-ranking is proposed that increases the score for those $d_i$ with high structural similarity between the structural hint given in the query and the structure for a term (as part of the key content) in $d_i$.

The final posting list from which the best top $n$ postings will be selected is thus constructed as the union of all selected postings of each query key $k$. If $k$ is a frequent XTerm, that is there exists only one single posting list for $k$, all scores for its postings will be calculated as $score_k(d_i) * sim(s_k, s_q)$ with $s_k$ being the structure of $k$, $s_q$ being the structure of $k$'s content in the query, and $sim(s_k, s_q)$ being the structural similarity between both structures. The best postings according to this calculation are then selected.

$$score_{h \in q}(d_i) = \sum_{\text{XTerm}\, t \in h} \begin{cases} score_t(d_i) * sim(s_t, s_q), & (*) \\ 0 & \text{else} \end{cases}$$

$$(*) \text{ if } sim(s_t, s_q) > thr_{sim} \text{ or if } d_i \notin postinglist(t)$$

**Formula 3. Scoring postings for pre-ranking**

In case $k$ is a HDK $h$, that is its posting list consists of several sets of posting lists, a new single posting list is build by uniting all postings of lists relating to a structure $s_t$ with $sim(s_t, s_q)$>threshold $thr_{sim}$. For this single posting list, the new scores are accordingly calculated as in formula 3.

## 3.5 Ranking

Both the pre-ranking to select postings and the final ranking to compute the accurate relevance of documents and retrieval units take into account structural similarity as described in formula 4 that was proposed in [26].

$$sim(s_q, s_t) = \begin{cases} \left( \dfrac{1 + |s_q|}{1 + |s_t|} \right)^{\alpha}, & \text{if } s_q \text{ is sub-sequence of } s_t \\ \beta * sim(s_t, s_q), & \text{if } s_t \text{ is sub-sequence of } s_q \\ \varepsilon, & \text{else (with } \varepsilon \text{ being very small)} \end{cases}$$

**Formula 4. Function to compute structural similarity**

Formula 4 is a sub-sequence-based attempt to measure the structure similarity $sim(s_q, s_t)$ between two structures $s_q$ of a query key and $s_t$ of an XTerm $t$. It is determined as a value between 1 (for $s_q = s_t$) and 0 (no similarity between $s_q$ and $s_t$). $|s_f|$ is the amount of tags in $s_f$. Parameter $0 \leq \beta \leq 1$ can be used to reduce the similarity in case not all tags of $s_q$ are matched by $s_t$ ($s_t$ = sub-sequence of $s_q$), as this should result in a lower value than the case that $s_q$ is a sub-sequence of $s_t$.. Parameter $\alpha$ sets the impact and strictness of the calculated structural similarity. For $\alpha$=0, the structural similarity is always 1 and therefore has no impact at all; $\alpha$=1 is the standard impact. With any other $\alpha$, the similarity value can be increased or reduced.

The final ranking is performed using XML IR techniques that provide the opportunity to retrieve more focused results by favouring retrieving a high relevant document part of $d_i$ over retrieving a less relevant document $d_i$. In other words, only the most relevant XML element of a document should be retrieved – which can result in retrieving the whole document, obviously. Relevance is therefore computed for documents and carefully chosen retrieval units. The proposed architecture supports this by indexing appropriate statistics and storing them close to the document statistics for easy access. For the calculation itself, the vector space model has been extended in order to take into account the structure similarity in formula 3. For the weighting, an extension of the BM25E formula is described for formula 2.

## 4. IMPLEMENTATION & EVALUATION

Currently, a search engine for P2P Information Retrieval of XML-documents called SPIRIX is being implemented in order to evaluate the impact of structural features in P2P-IR. The components according to figure 1 have already been integrated. All proposed techniques of this paper can be used except for the peer-scoring. However, some of the algorithms are only implemented in simple versions and need to be extended. We are also experimenting with different solutions for the building of HDKs in order to gain more efficiency while indexing but are still in the process of doing so. A P2P protocol named SpirixDHT has been developed to support efficient transport of messages between peers when retrieving XML-documents. The protocol is based on Chord and adapted to special requirements of XML-retrieval, e.g. the distribution of information is performed as described in this paper. It also collects some of the peer performance metrics such as online time and latency that we proposed to use for the peer-score in section 3.4. We have not finished integrating SpirixDHT into Spirix, so all evaluation so far is based on a SimulationDHT (see figure 1).

Some preliminary evaluation has been performed in order to investigate how useful structure can be exploited with SPIRIX. We recently participated in INEX 2008 to see how well our various ranking strategies work in comparison to centralized XML IR solutions. The results have not been released yet. However, some preliminary results can be reported that were observed when training our system with the evaluation mechanisms of INEX2007.

22 topics (CAS-queries) were chosen randomly out of the assessed 100 topics of last year's competition and ran on the main Wikipedia collection provided by INEX. The collection consists of 659.388 XML-documents (4.6GB); a more detailed description of the collection can be found in [7]. We concentrated on the XML IR aspect of our work and used SpirixDHT in a simulation mode, where messages have not been sent over the network but re-routed directly back to the same peer that contained the whole collection. That is, only a single machine was used. Therefore, only the amount of messages instead of the overall network traffic was measured.

Figure 3 shows an early run used to investigate structural similarity functions. Only entire documents were retrieved, that is no retrieval units. The amount of messages sent for ranking requests was set to 500, that is, only 500 postings were chosen from the final posting list. Hops between peers for message transport (log(n) on average) or messages to build the final posting list were not taken into account. For selecting of postings, a simple version of formula 2 and 3 was performed, where document weight and structural similarity were applied but no peer or element evidence ($\delta_2$ and $\delta_3$ were set to 0). $PL_{max}$ was set to indefinite, so all keys where indexed as HDKs consisting of one XTerm. Runs were conducted with different values for α and β to measure structural similarity and compared with a baseline, where structure was not taken into consideration (α=0). As an indicator for good retrieval quality, the mean average interpolated Precision (MAiP) of all 22 topics over all 101 recall level was calculated as well as early precision at recall-level 1%, 5% and 10% (interpolated Precision iP over all topics).

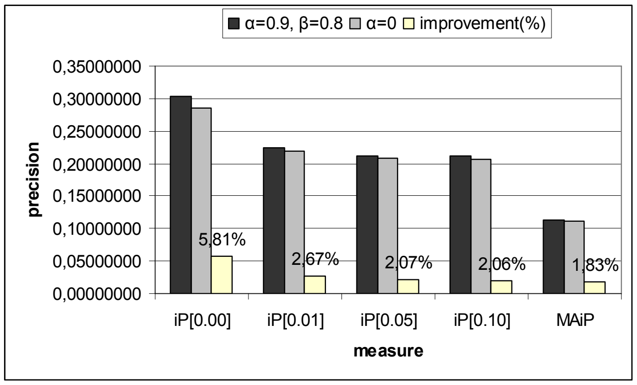

**Figure 3. MAiP and iP for 22 INEX07-topics with #ranking messages=500, num_q=22.**

When comparing precision at different recall-levels, it seems that the proposed use of structure can help improve early precision (5,81% at level 0%). At higher recall-levels, the impact decreases such that the difference for MAiP is not as great (0,11310817 versus 0,11107674, i.e. improvement of 1,83%).

For the randomly chosen 22 topics, we have thus shown that structural features can indeed improve precision. If applied in a pre-ranking as proposed in this paper, a better posting selection can be performed. This can be used to select fewer postings, that is, to reduce the amount of ranking messages, hence reducing bandwidth consumption without losing precision.

The results for α=0.9 and β=0.8 are displayed in figure 4.

```
<eval run-id="p_1"
file="/spirix/eval/withSIM/Results22.xml">
num_q         all    22
iP[0.00]      all    0.302794577560296
iP[0.01]      all    0.223953490176856
iP[0.05]      all    0.212575139228394
iP[0.10]      all    0.211245542885678
MAiP          all    0.1131081667767
...
ircl_prn.0.20   all    0.194176768102583
ircl_prn.0.30   all    0.171990751431073
ircl_prn.0.40   all    0.147907950621928
ircl_prn.0.50   all    0.109207482772513
ircl_prn.0.60   all    0.0713876658048743
ircl_prn.0.70   all    0.0576566315713854
ircl_prn.0.80   all    0.0354435964472534
ircl_prn.0.90   all    0.0188334335057359
ircl_prn.0.99   all    0.000250266974607499
ircl_prn.1.00   all    0.000108138199168743
</eval>
```

**Figure 4. MAiP and iP for 22 INEX07-topics with sim-α=0.9, sim-β=0.8, #ranking messages=5000.**

The evaluation is still in a very early stage. Future evaluation will be performed on all available INEX topics (years 2006-2007 for the Wikipedia collection). Also, we will conduct tests on different numbers of messages and consider message sizes. Finally, real bandwidth consumption will have to be considered. Only by conducting further experiments is to be seen if the observed improvement can be achieved in general, which would allow to conclude that structure can indeed help to improve query routing.

## 5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed to investigate how structured queries can be efficiently routed in a highly distributed environment such as a P2P system by taking advantage of the structural feature in the routing process. An infrastructure was provided to investigate this question and techniques have been proposed to perform routing based on a combination of scoring performed both at indexing and querying time. This scoring takes into account a mixture of document-, element-, collection- and peer-evidence. Furthermore, it was reported on very preliminary evaluation results performed with the INEX collection. These results indicate that structural hints might indeed help to route queries in large-scale P2P systems.

We have only just begun to evaluate the proposed techniques and will conduct a full evaluation in the coming weeks. Future work will include fully implementing the routing strategy proposed in this paper for analyzing which methods are the most promising and can significantly improve efficiency. This will include evaluation of the impact for different values of the used parameters, especially the impact of changing CF and $PL_{max,}$; investigation of how to take into account the peer-score (both in terms of the peer's collection and in terms of peer performance metrics); and evaluation of different posting list merging and scoring strategies for the HDKs. Furthermore, we will participate in the new Efficiency Track of INEX in August 2008 to compare different variants of the proposed similarity function.

## 6. REFERENCES

[1] Abiteboul, S.; Manolescu, I.; Polyzotis, N.; Preda, N.; Sun, C.: ***XML processing in DHT networks***. IEEE 24th Internat. Conference on Data Engineering (ICDE2008), Cancun, Mexico, 2008.

[2] Amer-Yahia, S.; Lalmas, M.: ***XML Search: Languages, INEX and Scoring***. SIGMOD Rec. Vol. 35, No. 4, 2006.

[3] Baeza-Yates, R.; Castillo, C.; Junqueira, F.; Plachouras, V.; Silvestri, F.: ***Challenges on Distributed Web Retrieval.*** IEEE Int. Conf. on Data Engineering (ICDE07), Turkey, 2007.

[4] Balakrishnan, H; Kaashoek, F.; Karger, D.; Morris, R.; Stoica, I.: ***Looking Up Data in P2P Systems***. Communications of the ACM, Vol. 46, No. 2, 2003.

[5] Broschart, A.; Schenkel, R***.: Proximity-Aware Scoring for XML Retrieval.*** In: Proc. of 31$^{st}$ ACM SIGIR, Singapore, 2008.

[6] Callan, J.; Lu, Z.; Croft, W.:***Searching distributed collections with inference networks.*** In: Proc. of the 18th ACM SIGIR, New York, 1995.

[7] Denoyer, L.; Gallinari, P.: ***The Wikipedia XML Corpus.*** In: LNCS, vol. 4528, Springer-Verlag, 2006.

[8] El-Ansary, S.; Haridi, S.: ***An Overview of Structured Overlay Networks.*** In: Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks, CRC Press, 2005.

[9] Fuhr, N.; Gövert, N.; Kazai, G.; Lalmas, M. (eds.): ***INitiative for the Evaluation of XML Retrieval (INEX)***. In: Proc. of the First INEX Workshop, Dagstuhl, Germany, 2002.

[10] Koloniari, G.; Pitoura, E.: ***Peer-to-Peer Management of XML Data: Issues and Research Challenges***. SIGMOD Rec. Vol. 34, No. 2, 2005.

[11] Larson, R.: ***XML Element Retrieval and Heterogeneous Retrieval - In Pursuit of the Impossible***. In: Proc. of INEX 2005, 2nd Edition, Dagstuhl, Germany, 2005.

[12] Li, J.; Loo, B.; Hellerstein, J.; Kaashoek, F.; Karger, D.; Morris, R.: ***On the Feasibility of Peer-to-Peer Web Indexing and Search***. In: Proc. of the Second International Workshop on Peer-to-Peer Systems, 2003.

[13] Malik, S.; Trotman, A.; Lalmas, M.; Fuhr, N.: ***Overview of INEX 2006.*** In: Proc. of INEX, Dagstuhl, Germany, 2006.

[14] Mass, Y.; Mandelbrod, M.: ***Component Ranking and Automatic Query Refinement for XML Retrieval.*** LNCS, Vol. 3493/2005, Springer-Verlag, 2005.

[15] Michel, S.; Triantafillou, P.; Weikum, G.: ***KLEE - A Framework for Distributed Top-k Query Algorithm***s. In: Proc. of 31st VLDB Conference, Trondheim, Norway, 2005.

[16] Pal, S.: ***XML Retrieval – A Survey***. Technical Report, CVPR, http://www.isical.ac.in/~sukomal_r/survey.pdf, 2006.

[17] Pass, G.; Chowdhury, A.; Torgeson, C.: ***A Picture of Search***. In: Proc. of Infoscale'06, Hong Kong, 2006.

[18] Podnar, I.; Rajman, M.; Luu, T.; Klemm, F.; Aberer, K.: ***Scalable Peer-to-Peer Web Retrieval with Highly Discriminative Keys***. In: Proc. of IEEE 23rd International Conference on Data Engineering (ICDE 2007), 2007.

[19] Ramirez, G.; Westerveld, T.; de Vries, A.P.: ***Structural Features in Content Oriented XML Retrieval***. In: Proc. of CIKM '05, ACM Press, New York, USA, 2005.

[20] Risson, J.; Moors, T.: ***Survey of research towards robust peer-to-peer networks – search methods***. In: Technical Report UNSW-EE-P2P-1-1, Uni. of NSW, Australia, 2004.

[21] Robertson, S.; Zaragoza, H.; Taylor, M.: ***Simple BM25 extension to multiple weighted fields***. In: Proc. of CIKM'04, ACM Press, New York, USA, 2004.

[22] Skobeltsyn, G.; Luu, T.; Podnar, I.; Rajman, M.; Aberer, K.: ***Web text retrieval with a P2P query-driven index.*** In: Proc. of 30th ACM SIGIR, Amsterdam, NL, 2007.

[23] Steinmetz, R.; Wehrle, K. (eds.): ***Peer-to-Peer Systems and Applications.*** LNCS 3485, Springer-Verlag, 2005.

[24] Stoica, I.; Morris, R.; Liben-Nowell, D.; Karger, D.; Kaashoek, M. F.; Dabek, F.; Balakrishnan, H.: ***Chord - A Scalable Peer-to-peer Lookup Protocol for Internet Applications***. IEEE/ACM Transactions on Networking, Vol. 11, No. 1, 2003.

[25] Winter, J.; Drobnik, O***.: A Distributed Indexing Strategy for Efficient XML Retrieval.*** Efficiency Issues in Information Retrieval Workshop (EIIR2008) at ECIR2008, Glasgow, Scotland, 2008.

[26] Winter, J.; Drobnik, O.: ***An Architecture for XML Information Retrieval in a Peer-to-Peer Environment.*** First PhD Workshop at CIKM 2007, Lisbon, Portugal, 2007.

[27] Yee, W.G.; Nguyen, L. T.; Jia, D.; Frieder, O.: ***Efficient Query Routing by Improved Peer Description in P2P Networks***. In: Proc. of ACM/ICST Infoscale, 2008.

[28] Zeinalipour-Yazti, D.; Kalogeraki, V.; Gunopulos, D.: ***Information Retrieval in Peer-to-Peer Networks***. IEEE CiSE Magazine, Special Issue on Web Engineering, 2004.

[29] Zhang, J.; Suel, T.: ***Optimized Inverted List Assignment in Distributed Search Engine Architectures.*** In: Proc. of 21th IPDPS 2007, California, USA, 20