ALVIS Peers: A Scalable Full-text Peer-to-Peer Retrieval Engine

Toan Luu, Fabius Klemm, Ivana Podnar, Martin Rajman, Karl Aberer
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
{vinhtoan.luu, fabius.klemm, ivana.podnar, martin.rajman, karl.aberer}@epfl.ch

ABSTRACT

We present ALVIS PEERS, a full-text P2P retrieval engine designed to offer retrieval performance comparable to centralized solutions while scaling to a very large number of peers. It is the result of our research efforts within the project ALVIS[1] that aims at building a truly-distributed semantic search engine. To cope with problem of unscalable bandwidth consumption in the P2P network, the engine implements a novel retrieval model that indexes highly-discriminative keys (HDKs)—terms and term sets appearing in a limited number of collection documents. Our prototype is a fully-functional retrieval engine built over a structured P2P network. It includes a component for HDK-based indexing and retrieval, and a distributed content-based ranking module. Such an integrated system represents a substantial contribution to the design and development of realistic P2P retrieval systems.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; D.3.4. [**Information Storage and Retrieval**]: Systems and Software—*Distributed Systems*

General Terms

Architecture, Design, Performance

Keywords

peer-to-peer information systems, distributed information retrieval, scalability

1. INTRODUCTION

The state-of-the-art information retrieval (IR) engines are built upon centralized or clustered architectures to achieve extremely efficient and high-performance retrieval functionality. However, this comes at the price of maintaining a huge index that has to deal with frequent document collection changes, which is contradictory to the inherently distributed nature of information sources and the randomized process of information generation. Peer-to-peer (P2P) information systems offer an alternative decentralized architecture for building truly-distributed IR engines in accordance with an ever-increasing number of heterogeneous information sources.

In our view, the P2P-based retrieval engines do not directly compete with traditional clustered engines, but rather offer an alternative solution in support of novel usage scenarios. We assume each peer in the P2P network contributes documents to the global document collection and, in exchange for making its local documents globally searchable (it is in a way advertising the local content through the global search facility), the peer invests computing resources to build the P2P network and attain part of the querying load. The P2P retrieval engine can offer free access to the querying facility to the broader public, while the retrieval of documents remains under the responsibility of individual peers. Moreover, information sources themselves decide on the content they want to make globally searchable. Consider, for example, the following motivating scenarios:

An integrated digital library. Today digital libraries offer sophisticated means to query documents from their local collections, and may use a P2P-based engine as the common media for an integrated search facility. In this perspective, the P2P engine is used to identify the set of digital libraries with the relevant content, while the semantic-based querying and retrieval is performed using the sophisticated facilities of the individual digital libraries. We have presented a solution for collaborative search over digital library collections based on structured P2P networks in [15].

P2P-based news search engine. A P2P news search engine offers an integrated search facility over a large number of news sources. As news articles are a typical example of highly-dynamic information, a P2P-based news search engine would dynamically update the index at the time when a new article is published.

Blog search. The blogging software may be extended to offer a P2P-based integrated search facility over the blogging sites, while blog creators would be given a chance to decide whether they want to make their content globally searchable.

[1]European FP 6 STREP project ALVIS, http://www.alvis.info/

In this paper we present the design of our full-text P2P retrieval engine, Alvis peers. It is the result of our research efforts within the project Alvis that aims at building an open-source semantic search engine with P2P and topic specific technology at its core [5]. In contrast to the majority of proposed solutions that rely on simulations, we demonstrate one of the few complete solutions for IR over P2P networks. It is implemented using our P-Grid P2P platform[2] and integrates a solution for distributed indexing, retrieval, and content-based ranking.

The prototype is based on a novel indexing mechanism that reduces the size of posting lists associated with indexing features to limit the generated traffic when processing queries. To obtain short posting lists, we propose to index not only single terms from the collection vocabulary T, but also *keys*, i.e. sets of terms that occur simultaneously in documents. The interesting keys appear in a limited number of documents from the collection D because they are *discriminative* wrt D. For example, let us assume that two terms from T, t_1 and t_2, appear in D. Instead of using two long posting lists $t_1 \rightarrow \{d_1, d_2, d_4, d_6, d_8, d_{10}\}$ and $t_2 \rightarrow \{d_2, d_3, d_6, d_7, d_9\}$, we might index the term set $\{t_1,\ t_2\}$, which is associated with a shorter posting list, $\{t_1, t_2\} \rightarrow \{d_2, d_6\}$, and thus an answer to the query $q = \{t_1, t_2\}$ is already pre-computed and available for retrieval. To answer queries containing only t_1 or t_2, the index can store truncated posting lists as most users are only interested in the top-k matching documents. In other words, we propose to identify discriminative term combinations at indexing time instead of performing long posting list intersections at query processing time because it generates unscalable network traffic [11, 23]. Discriminative keys may be observed as queries associated with pre-computed answers facilitating highly-efficient retrieval as the (*query*, *answer*) pairs are stored in a Distributed Hash Table (DHT).

The major concern is that indexing with term combinations might lead to a key vocabulary of unmanageable size because it theoretically grows with $2^{|T|}$. To limit this effect, we introduce a concept of *highly discriminative keys* (HDKs) and design a scheme for careful key selection that removes redundant keys from the key vocabulary while preserving good retrieval performance properties. We have reported a scalability analysis in [17] that proves the HDK vocabulary grows linearly with the document collection size, i.e. a peer indexing locally a document collection of a limited size will produce a bounded number of HDKs with a reasonable upper bound. Furthermore, the measured retrieval performance is comparable to the one achieved with a centralized single-term engine using the BM25 ranking scheme, while the crucial property confirmed by our experiments is scalable retrieval in contrast to the distributed single-term indexing approach. Finally, the presented indexing model integrates gracefully with our distributed ranking schema because it provides global statistics and enables aggregation of content-based ranking scores in line with those obtained by the centralized system.

The paper is structured as follows: Section 2 provides an overview of our HDK-based indexing model. We describe the layered architecture of the prototype in Section 3. Section 4 gives a summary of our experimental results, while Section 5 provides an overview of existing systems for P2P IR. Section 6 concludes the paper with a summary and short discussion of future work.

[2]http://www.p-grid.org/

2. INDEXING WITH HIGHLY DISCRIMINATIVE KEYS

The standard solution for text retrieval in structured P2P networks is to distribute the inverted index among the peers in the network [18, 22], such that each peer is responsible for storing a part of the vocabulary and their associated posting lists as depicted in the upper part of Figure 1. Given a term t, such a P2P system must guarantee that at least one of the peers responsible for t is available at any time to retrieve the associated posting list.To achieve fault tolerance, terms and posting lists are replicated at several peers. Furthermore, as such a peer can be found very efficiently, i.e. typically in $O(Log(N))$ routing hops (where N is the number of peers), a simple mechanism to process a query is to retrieve and intersect the posting lists associated with the query terms.

The major problem with this approach also called the naïve approach is that some terms have very long posting lists. As answering queries requires posting lists to be shipped over the network, this approach generates unscalable traffic during retrieval [11]. A recent study has shown that, even when a carefully designed and optimized protocol is used in this setting (peers are contacted in a 'round-robin' fashion; terms from a query are ordered by the size of associated posting lists in the index, and they determine the sequence of contacted peers; each peer performs partial intersection and forward the query to the subsequent peer on the list) the generated query traffic is unacceptably high for web-scale document collections [23].

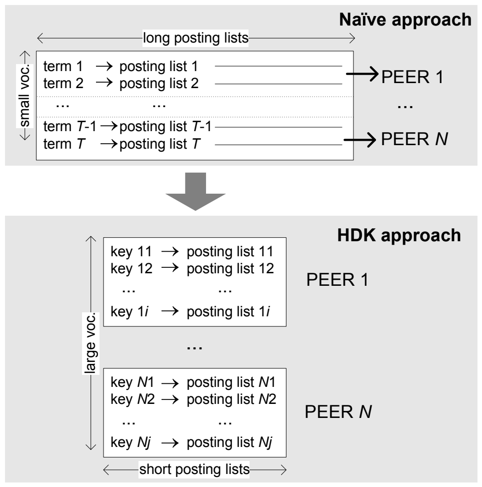


Figure 1: The basic idea of indexing with HDKs

The central idea of our indexing strategy is to limit the posting list size associated with keys to a constant predefined value and extend the index vocabulary to improve retrieval effectiveness, as depicted in the lower part of Figure 1. The key index only contains single terms and term sets that are *discriminative* with respect to a document collection. In this perspective, we categorize a key on the basis of its *global document frequency* (DF), and define a threshold DF_{max} to divide the set of keys into two disjoint classes, discriminative and non-discriminative keys. If a key k appears in more than DF_{max} documents, the key is *non-discriminative*. Otherwise, k is *discriminative* and specific with respect to the documents it is associated with in the document collection. This property is in line with the capacity constraints of P2P networks that are capable of storing and transmitting posting lists of limited size.

Notice that the extension of the index entries to discriminative terms and term sets, while limiting the size of the posting lists, entails the expansion of the key vocabulary. It is therefore important to select among the potential key candidates those that have favorable properties for the retrieval performance of the P2P search engine. We currently use three filtering techniques to produce *highly-discriminative keys* (HDKs) indexed by our search engine. (1) *Size filtering* limits the size of the keys (number of terms forming a key) to be considered to a maximal size s_{max}. (2) *Proximity filter* uses textual context to reduce the size of the key vocabulary and retains keys built of terms appearing in the same textual context—a document window of predefined size w. (3) *Redundancy filter* removes supersets of discriminative keys from the vocabulary as such keys are redundant and only increase the vocabulary size without improving retrieval performance.

HDKs are good and discriminative indexing features, however, user may still submit non-discriminative queries. Therefore, to improve the retrieval performance, we also index DF_{max}-best documents for non-discriminative keys. The experimental results and scalability analysis demonstrate the growth of such a key vocabulary and associated index is scalable with reasonable and manageable real bounds for very large document collections [17].

3. DESIGN OF THE ALVIS P2P RETRIEVAL ENGINE

The Alvis P2P retrieval engine is built of peers extended by the IR functionality. The peers are part of a structured P2P network, as depicted in Figure 2, and are responsible to build and maintain a global index associating keys to posting lists. Each peer is a stand-alone component that can index and search its local document collection. The peer is responsible to compute keys and associated postings from its local collection and to insert them as a contribution to the global index. On the global level, the peer maintains part of the global index, and as far as retrieval is concerned, it interacts with the global network to retrieve the list of documents that contain indexing keys with maximal overlap with the submitted query.

The architecture is layered to enable clean separation of different concepts related to network communication, P2P networks, IR-related tasks such as indexing and ranking, as advocated in [2]. The system is decomposed into the following layers: 1) DHT layer building a Distributed Hash Table for storing global index entries; 2) HDK layer for building the key vocabulary and corresponding posting lists, and mapping queries to keys during retrieval; and 3) Ranking layer that implements distributed document ranking. Each peer also runs a Web services to accept queries and documents from remote hosts.

Each peer incorporates a local and global system view. The HDK layer focuses on the local view and builds the key index from a received single-term index for a local collection.

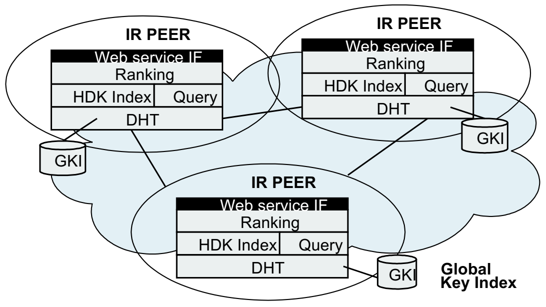


Figure 2: System architecture

The received single-term index must contain a positional index needed for the key computation, and provide relevance scores for (key, document) pairs. The DHT layer provides a global system view by maintaining the global key index with information about key document frequencies and other statistics used by the ranking layer. We now describe in detail the functionalities of each layer of our P2P-IR prototype.

3.1 DHT Layer

The DHT layer provides the following essential service: $route(id, data)$, which allows a peer to send data to a peer responsible for *id*, while the content of *data* is application-specific, i.e. it can be a lookup, insert, update, delete, etc. The route operation requires $O(log(N))$ overlay hops to find a responsible peer (N is the number of peers in the network). Each peer maintains a routing table with $O(log(N))$ routing entries. The routing entries are chosen in such a way that the resulting network topology has small-world characteristics [8]. Routing is done in a greedy fashion, i.e. given a message with destination *id*, a peer that is not responsible for *id* forwards the message to the neighbor in its routing table that is closest to *id*. Popular DHTs with such characteristics are among others Chord, Pastry, and P-Grid [21, 20, 1].

P2P IR applications have high performance demands for DHTs. The DHT has to handle a very large number of small messages: when inserting the local key index into the DHT, a peer generates hundreds of thousands of relatively small messages that consist of a key and a document reference. Another example is aggregation of global statistics, e.g. the document frequencies associated with keys, which also requires a peer to insert and request many values in the DHT.

In an IR environment, DHTs have to handle a much higher rate of service requests than in other applications, such as file-sharing systems. To handle such large amounts of data, we have extended our DHT with special congestion control mechanisms to better manage the flow of messages. In P2P environments, the network capacity is usually the bottleneck. When peers generate requests faster than the network can handle, without congestion control, the DHT can suffer a congestion collapse. Each peer therefore receives positive or negative feedback from the network: positive feedback allows the peer to increase its request rate, whereas negative feedback indicates congestion in the network and the peer will reduce its request rate to alleviate the congestion.

Without such congestion control mechanisms a DHT cannot handle the large amount of traffic generated by an IR application. Further details on congestion control for DHTs can be found in [10].

We have implemented further mechanisms to improve the throughput of our DHT [9]: each peer has a queue to buffer messages for a short time before they are forwarded to the next hop in the DHT. All messages that will be sent to the same next hop are buffered in the same queue. Thus, we have a queue for each entry in the routing table, i.e. $O(log(N))$ queues. If the queue reaches a given threshold (e.g. 100 messages), all queued messages are packed together and compressed to be send in a single message to the neighbor. We can thus better use the available bandwidth in the network. To avoid that messages are delayed too long in a queue, the process of packing and compressing can also be triggered by a timeout even if the threshold is not reached.

3.2 HDK Layer

The HDK layer deals with the task of key-based indexing, i.e. finding the set of keys and associated postings lists for given a document collection, and the querying task, i.e. given a query, finding relevant keys in the global P2P index, retrieving the postings associated with those keys, and merging the result set.

3.2.1 Distributed indexing

During the indexing process, peers share the computational load and build the HDK vocabulary in parallel. Initially, each peer builds a local index from its local collection by using the standard single term indexing mechanism. The local index is composed of a lexicon, an inverted index with term positions, a document index etc. The distributed indexing process is triggered when a peer joins the P2P-IR network. Each peer starts creating HDKs from the its local index in several iterations by computing single-term keys, then 2-term keys, ..., and finally s_{max}-term keys. For any current key-size s, a peer computes its local discriminative and non-discriminative keys, and inserts their document frequencies into the global P2P index. Subsequently, it inserts posting lists for its local HDKs and top-DF_{max} postings for its local non-discriminative keys into the P2P overlay. Notice that a locally discriminative key may be globally discriminative, thus the P2P layer stores all received posting lists for global HDKs. However, it is possible that some locally discriminative keys are globally non-discriminative, and for such keys and for those that are locally non-discriminative (therefore also globally non-discriminative) the responsible peer selects top postings from all received postings. To support both indexing and ranking, the P2P layer maintains the global document frequencies for all received keys, and notifies the indexing layer when discriminative keys become non-discriminative due to the addition of new documents.

Figure 3 shows an example for computing HDKs in a distributed manner. Each peer starts by inserting local document frequencies for its single term vocabulary (a term is a key of size 1). The messages of the form (key, $frequency$, $peerID$) are routed to the peer responsible for the key and this peer aggregates the received frequency values. In this example, $peer_x$ is responsible for the global index of term t_1, so it aggregates the received values (3+4+4=11) and stores the list of peers ($peer_i, peer_j, peer_k$) term t_1 was reported from. We assume that DF_{max} is set to 10, thus t_1 is a non-discriminative key. Therefore, $peer_x$ will send notification messages to the peers in the peer list associated with t_1. When $peer_i, peer_j$ and $peer_k$ receive the notification, they start generating new keys by combining t_1 with other globally non-discriminative terms from the single term vocabulary under the condition that they satisfy the conditions imposed by the size and proximity filter. In this example, t_2, t_3, t_4, t_5 can be combined with t_1 to generate new keys. The following properties must be satisfied to obtain the candidate HDKs:

- All terms must in the same document and within a window of size w (proximity filter). The positional term index is used to check this property.

- For a new key k of size s, all properly contained keys of size $(s-1)$ must be non-discriminative (redundancy filter).

Then newly generated keys are inserted in the P2P network with their local frequency used for the vocabulary aggregation. The process continues until the key size reaches s_{max} and can be triggered again by notification messages reporting new non-discriminative keys in the P2P network. Such keys have previously been discriminative, but due to the addition of new documents or new peer joins become non-discriminative.

Finally, the posting lists (list of documents containing the keys) are inserted into the P2P network. For the non-discriminative keys, a posting contains $docID$ and a relevance score of the key wrt document. The peer responsible for the non-discriminative keys will choose DF_{max}-best documents based on the received scores.

3.2.2 Querying

The querying part of our prototype deals with the problem of mapping of a query to keys and subsequent retrieval of postings from the P2P overlay.

Figure 4 shows the querying process when peer P_i receives the query Q. We assume Q is a simple set of terms, and it must be mapped to a set of keys present in the P2P global index. All term subsets of a query are possible key candidates that may be stored in the P2P index, and P_i needs to explore the lattice of term combinations and check which key candidates are indeed keys from the global index.

The optimistic strategy is to start with the largest possible term set (marked as level-1), limited either by the query size q (as it is assumed in Figure 4) or the maximal number of terms forming a key (s_{max}). P_i will try to retrieve the keys sequentially by exploring initially level-1 keys, and then depending on the result of the previous step continue with level-2 keys, ..., level-q keys.

The perfect situation occurs when $k_{11} = \{t_1, t_2, \ldots, t_q\}$ is a key in the global index, in other words, a user has posed a good query for the indexed document collection: The posting list is readily available and is simply retrieved from the global index by issuing a single request to the P2P network. Indeed, this will not happen with all user queries. Therefore, level-2 set of potential key candidates of size $q-1$ is explored. If the P2P network stores postings associated with level-2 keys that cover all terms in Q, the hit list is the union of the retrieved postings. However, it is possible that either (a) no level-2 keys exist in the index, or (b) that some $t_i \in Q$ are not covered by the retrieved keys. In case

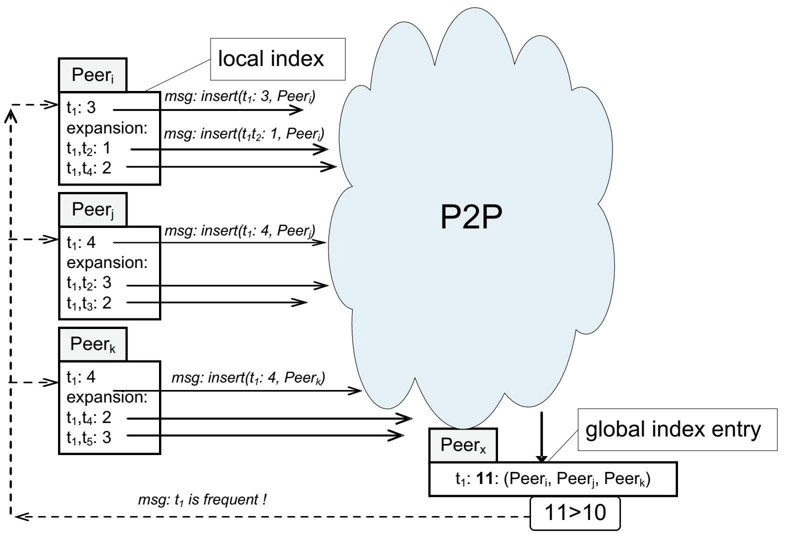


Figure 3: Distributed key generation

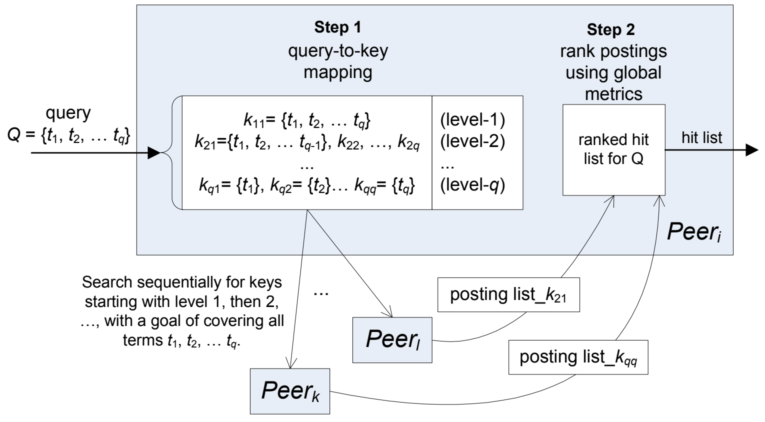

Step 1
query-to-key mapping
Step 2
rank postings using global metrics
query
$Q = \{t_1, t_2, \ldots t_q\}$
$k_{11} = \{t_1, t_2, \ldots t_q\}$
$k_{21} = \{t_1, t_2, \ldots t_{q-1}\}, k_{22}, \ldots, k_{2q}$
...
$k_{q1} = \{t_1\}, k_{q2} = \{t_2\} \ldots k_{qq} = \{t_q\}$
(level-1)
(level-2)
...
(level-q)
ranked hit list for Q
hit list
Peer$_i$
Search sequentially for keys starting with level 1, then 2, ..., with a goal of covering all terms $t_1, t_2, \ldots t_q$.
...
posting list_k_{21}
Peer$_l$
posting list_k_{qq}
Peer$_k$

Figure 4: P2P-IR retrieval

of (a), P_i explores all level-3 keys and subsequently higher level keys until finally reaching level-q. In case of (b), P_i explores level-3 keys but only those that contain non-covered terms from the previous levels. Depending on the retrieved set of keys, the procedure is extended to higher-level keys until all $t_i \in Q$ are covered by the retrieved key set. The resulting answer set is the union of postings associated with the retrieved keys.

Let us again consider the example in Figure 4. The P2P network does not contain k_{11}, and P_i searches for level-2 keys. Out of q possible level-2 keys, only k_{21} is an HDK and its posting list is retrieved from P_l. However, t_q is still not covered, and P_i unsuccessfully explores keys on higher levels containing t_q, until finally reaching level-q where a posting list of size DF_{max} is associated with k_{qq} because k_{qq} is a non-discriminative key. Finally, both posting lists are merged by a simple union procedure and this posting list is used as the input information to the ranking layer.

3.3 Ranking Layer

The ranking layer produces the final ranked result set according to the relevance of a document wrt Q. The ranking procedure must be implemented in a distributed fashion using global document collection statistics available from the P2P index. The following example illustrates how the distributed ranking function is implemented in our prototype:

Given a query Q that is submitted to peer P_i, assume the HDK layer of P_i has obtained two posting lists from the global P2P network and these postings are associated with keys k_{21}, k_{qq} (as in Figure 4). Each posting in the posting list is a pair ($peerID$, $docID$):

$k_{21} : (P_i, d_1), (P_j, d_2)$,

$k_{qq} : (P_j, d_2), (P_j, d_4)(P_k, d_5)$.

The ranking layer receives the listed pairs, and first, creates the union of the posting lists. Then it groups the $docIDs$ according to their $peerIDs$. In the example, there are 3 groups: $(P_i : d_1)$, $(P_j : d_2, d_4)$ and $(P_k : d_5)$.

Next, the messages with the original query Q and the list of document identifies (e.g (Q, d_2, d_4)) will be sent to the corresponding peers (e.g P_j)to compute the relevant scores of the query wrt documents from the list by using global and local statistics. This step is performed by peers that have originally indexed the identified documents because only these peers have the complete information about the documents (it is not available in the global P2P index). Finally, the ranking peer (P_j) sends back the list of document digests containing (URL, title, snippet, relevant scores) to the requesting peer P_i. Finally, the results are merged, sorted and displayed to the user. Document digests are used to present the final answer to end users.

This scenario may lead to a large number of ranking requests sent to indexing peers. Here are the reasons to perform ranking in such a way:

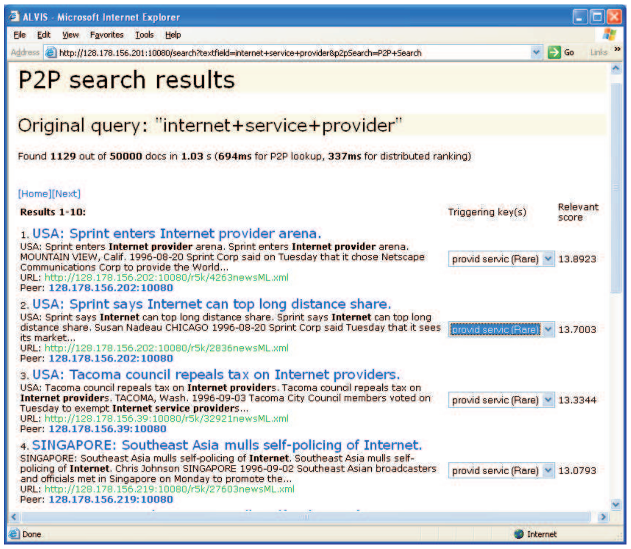
- *Displaying complete information about the documents to the user.* Following the practice used for standard web search engines, our prototype display results in such a way to include document digest consisting of a document URL, peer URL, document title, and snippet. Additionally, term positions in documents can also be stored in document digests to support ranking functions that use the distance between query term or sentence search. Therefore, for each document, a document digest contains large amounts of information that can be an overhead if is stored as part of a posting list in the global P2P index. We have decided to store document digests by peers indexing the documents, and it is retrieved together with the computed relevance score during the ranking step.

- *Using global and local statistics for ranking.* For the content-based ranking, the relevance scores between a given query and a document from the collection is a function with global and local values. Global values are related to the whole document collection (examples are document frequency, number of documents in the global collection), while local values are document-based (e.g. term frequency, document length). *Local statistics* can only be extracted from the original document, and, as we do not globally store or use the knowledge about the document, it is only available to the indexing peers. Notice that this property is in line with our assumption that the global index has no complete knowledge about the indexed documents. *Global statistics* can only be obtained from the global P2P index, and we already maintain some collection related statistics for the purpose of indexing (document frequencies). The standard TF-IDF scores can already be computed to obtain global rankings using only this global knowledge (the total number of documents is used for normalization purpose). However, more sophisticated ranking functions such as BM25 [19] require other global statistics that are difficult to maintain in a P2P network. Aggregation procedures in P2P networks are needed to obtain such global statistics as for example the solution proposed in [14].

- *The bandwidth consumption during the ranking process is scalable.* The number documents needed to be ranked are of the same size as the union set of posting lists after the querying phase. As the size of posting lists are bounded by DF_{max} and each query is expended to a limited number of keys, the total bandwidth consumption during the ranking process remains scalable for each query (the list of contacted peers will be bounded). Our experiment in section 4 will show this aspect.

4. PROTOTYPE EVALUATION

Figure 5 shows the results page of our prototype Alvis peers. The query was performed globally on the P2P network with 10 peers hosting 50,000 document. We display the URL of a hit document with the keys found by mapping the original query to keys available in the global index. We also display the global relevance score (in this case TF-IDF) and the URL of the peer that has indexed the document. The user can thus perform a local search on the peer if she thinks the documents of a particular peer are interesting.


ALVIS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go
Links
Done
Internet

http://128.178.156.201:10080/search?textfield=internet+service+provider&p2pSearch=P2P+Search

P2P search results

Original query: "internet+service+provider"

Found **1129** out of **50000** docs in **1.03** s (**694ms** for P2P lookup, **337ms** for distributed ranking)

[Home][Next]

Results 1-10: Triggering key(s) Relevant score

1. USA: Sprint enters Internet provider arena.
USA: Sprint enters **Internet provider** arena. Sprint enters **Internet provider** arena. MOUNTAIN VIEW, Calif. 1996-08-20 Sprint Corp said on Tuesday that it chose Netscape Communications Corp to provide the World...
URL: http://128.178.156.202:10080/r5k/4263newsML.xml
Peer: **128.178.156.202:10080**
provid servic (Rare) 13.8923

2. USA: Sprint says Internet can top long distance share.
USA: Sprint says **Internet** can top long distance share. Sprint says **Internet** can top long distance share. Susan Nadeau CHICAGO 1996-08-20 Sprint Corp said Tuesday that it sees its market...
URL: http://128.178.156.202:10080/r5k/2836newsML.xml
Peer: **128.178.156.202:10080**
provid servic (Rare) 13.7003

3. USA: Tacoma council repeals tax on Internet providers.
USA: Tacoma council repeals tax on **Internet providers**. Tacoma council repeals tax on **Internet providers**. TACOMA, Wash. 1996-09-03 Tacoma City Council members voted on Tuesday to exempt **Internet service providers**...
URL: http://128.178.156.39:10080/r5k/32921newsML.xml
Peer: **128.178.156.39:10080**
provid servic (Rare) 13.3344

4. SINGAPORE: Southeast Asia mulls self-policing of Internet.
SINGAPORE: Southeast Asia mulls self-policing of **Internet**. Southeast Asia mulls self-policing of **Internet**. Chris Johnson SINGAPORE 1996-09-02 Southeast Asian broadcasters and officials met in Singapore on Monday to promote the...
URL: http://128.178.156.219:10080/r5k/27603newsML.xml
Peer: **128.178.156.219:10080**
provid servic (Rare) 13.0793

Figure 5: Screen shot of the results page

We performed experiments to investigate the performance of our P2P retrieval engine using a subset of documents from the Wikipedia corpus[3] and news from the Reuters corpus [4]. The experiments were performed on our campus intranet running on up to 28 peers (Linux RedHat PCs with 1GB of main memory). To simulate the evolution of a P2P system, i.e. peers joining the network and increasing the document collection, we started each experiment with 4 peers, and added additional 4 peers at each new experimental run. Each peer contributes 5,000 documents to the global collection.

Our experiments investigate indexing and retrieval costs in terms of bandwidth and storage consumption, and compare retrieval performance of our P2P search engine to a centralized engine. Here we report our major findings. The detailed results for the Reuters collection can be found in [16], and for Wikipedia in [17].

- We observe an enormous reduction of bandwidth consumption per query of the HDK-based approach compared to the distributed single term indexing. The

[3]http://www.wikipedia.org/
[4]http://about.reuters.com/researchandstandards/corpus/

single term index is unscalable during querying because the traffic for a query grows linearly with respect to the collection size. For our HDK-based approach, however, the traffic remains constant. Figure 6 shows the average number of transmitted postings per query during both retrieval and ranking for different collection sizes. We use a set of 3000 queries with more than 2 terms per query, randomly chosen from a real Wikipedia query log.

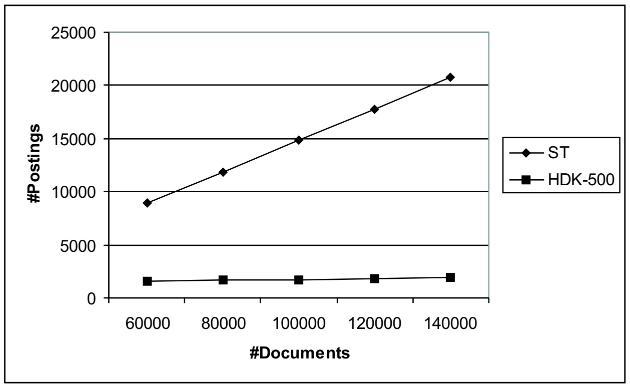


Figure 6: Number of transmitted postings per query during retrieval and ranking

- The overlap of the top-20 documents retrieved by our system and the centralized Terrier search engine[5] is quite satisfactory when using BM25 for ranking (more than 90%). The retrieval performance is better for larger DF_{max} as we are approaching single-term indexing.
- The size of the total key vocabulary and associated index maintained in the global P2P network increases linearly with the number of documents for large collections. Therefore, each peer is responsible for a constant index size when the global collection grows by inserting new peers each adding a limited number of documents to the collection.
- The number of inserted messages per peer during the indexing process grows linearly with respect the collection size. In other words, when a new peer joins the network with its local index, the bandwidth consumption for generating the new keys is independent of the size of the P2P system.
- The indexing costs of the HDK-based approach are significantly larger than for single-term indexing. However, we believe that these costs are still feasible and profitable as the gains in terms of bandwidth consumption during the query phase are huge, and thus compensate for the increased indexing costs.
- With our DHT layer with an integrated solution for congestion control, our DHT can process a total of 25'000 insertions and 6'500 requests of (key, frequency) per second in the experimental setup described above.

[5]Terrier search engine, http://ir.dcs.gla.ac.uk/terrier/

5. RELATED WORK

A number of solutions have been proposed for full-text search in P2P systems. However, most of the presented solutions use simulations to investigate system performance, and few fully-functional systems have been reported. We are aware of the following systems PlanetP [6], ODISSEA [22]and MINERVA [4].

PlanetP is a solution for unstructured networks that gossips compressed information about peers' collections. As large posting lists are the major concern for global single-term indexing, ODISSEA [22] proposes top-k posting list joins, Bloom filters, and caching as promising techniques to reduce search costs for multi-term queries. However, a study reported in [23] shows that this solution cannot scale to web-size document collections. MINERVA [4] maintains a global index with peer collection statistics in a structured P2P overlay to facilitate the peer selection process. It implements a method which penalizes peers holding overlapping document collections. The Keyword-Set Search System (KSS) [7] pre-computes and stores intersection results of inverted lists of popular query keywords in advance. KSS is quite similar to our system, however, it does not provide a ranking module. Furthermore, the authors generate exhaustive term combinations, which leads to unrealistic storage requirements for the index.

Another approach for resource selection is presented in [12, 13] in the context of hierarchical P2P networks: here, special directory nodes route queries to appropriate peers having high chances of answering a query. A recent solution builds an index dynamically following user queries [3]: a super-peer backbone network maintains the information about good candidates for answering a query, while peers answer the queries based on their local document collections.

6. CONCLUSIONS

In this paper, we have presented Alvis peers, a prototype for scalable full-text P2P-IR using Highly Discriminative Keys for indexing. Our solution overcomes the scalability problem of single-term retrieval in structured P2P networks. The HDK-based approach achieves results comparable to standard centralized engines using the BM25 relevance scheme while the indexing costs are scalable with realistic upper bounds.

Alvis peers is a fully-functional retrieval engine built over a structured P2P network. It provides distributed indexing, retrieval, and a content-based ranking module. Our system represents a substantial contribution to the design and development of realistic P2P retrieval systems.

For our implemented prototype, the index size is larger than the single term index. However, storage space is largely available in P2P systems as opposed to network bandwidth. Therefore, the reductions in bandwidth and response time for query answering are more important than the increase of the storage space. Moreover, there exist other mechanisms to further reduce the size of the key index, for example, taking into account user queries or applying more filtering mechanisms when choosing index keys. Some sophisticated techniques to deal with massive data processing need to be added in our prototype to improve its performance. It will allow us to further increase the number of indexed documents per peer.

7. REFERENCES

[1] K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. *Sixth International Conference on Cooperative Information Systems*, 2001.

[2] K. Aberer, F. Klemm, M. Rajman, and J. Wu. An Architecture for Peer-to-Peer Information Retrieval. 2004.

[3] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. DL Meets P2P - Distributed Document Retrieval Based on Classification and Content. In *9th European Conference on Research and Advanced Technology for Digital Libraries, (ECDL)*, pages 379–390, 2005.

[4] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving collection selection with overlap awareness in P2P search engines. In *SIGIR '05*, pages 67–74, New York, NY, USA, 2005. ACM Press.

[5] W. Buntine, K. Aberer, I. Podnar, and M. Rajman. Opportunities from open source search. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 2–8, 2005.

[6] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*. IEEE Press, 2003.

[7] O. D. Gnawali. A keyword set search system for peer-to-peer networks, 2002. Master's thesis, Massachusetts Institute of Technology.

[8] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[9] F. Klemm and K. Aberer. Aggregation of a Term Vocabulary for Peer-to-Peer Information Retrieval: a DHT Stress Test. In *Third International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2005)*, 2005.

[10] F. Klemm, J.-Y. Le Boudec, and K. Aberer. Congestion control for distributed hash tables. In *The 5th IEEE International Symposium on Network Computing and Applications (IEEE NCA06)*, 2006.

[11] J. Li, B. Loo, J. Hellerstein, F. Kaashoek, D. Karger, and R. Morris. The Feasibility of Peer-to-Peer Web Indexing and Search, 2003.

[12] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, 2003.

[13] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *Advances in Information Retrieval, 27th European Conference on IR Research (ECIR)*, pages 52–66, 2005.

[14] N. Ntarmos, P. Triantafillou, and G. Weikum. Counting at large: Efficient cardinality estimation in internet-scale data networks. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006)*, 2006.

[15] I. Podnar, T. Luu, M. Rajman, F. Klemm, and K. Aberer. A Peer-to-Peer Architecture for Information Retrieval Across Digital Library Collections. In *To appear in European conference on research and advanced technology for digital libraries (ECDL 2006)*, September 2006.

[16] I. Podnar, M. Rajman, T. Luu, F. Klemm, and K. Aberer. Beyond term indexing: A P2P framework for web information retrieval. *To appear in Informatica, Special Issue on Specialised Web Search*, 2006.

[17] I. Podnar, M. Rajman, T. Luu, F. Klemm, and K. Aberer. Scalable peer-to-peer web retrieval with highly discriminative keys. Technical Report LSIR-REPORT-2006-009, 2006.

[18] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching, 2003.

[19] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992.

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

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

[22] T. Suel, C. Mathur, J.-W. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram. ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. *WebDB'03*, 2003.

[23] J. Zhang and T. Suel. Efficient query evaluation on large textual collections in a peer-to-peer environment. In *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 225–233, Washington, DC, USA, 2005. IEEE Computer Society.

Acknowledgments

The work presented in this paper was carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European FP 6 STREP project ALVIS (002068).