

# A Distributed Indexing Strategy for Efficient XML Retrieval

Judith Winter<sup>1</sup> and Oswald Drobnik<sup>1</sup>

<sup>1</sup> J.W.Goethe University, Institute for Informatics, Frankfurt, Germany  
{winter, drobnik}@tm.informatik.uni-frankfurt.de

**Abstract.** Using structural information from XML-documents can help to improve retrieval quality of these documents significantly, even in highly dynamic environments such as Peer to Peer (P2P) systems. However, due to the absence of a central index, the index information to be stored in a P2P network must be selected and distributed carefully to reduce communication costs and to guarantee scalability. We propose a novel indexing strategy for XML Information Retrieval in P2P systems. Indexing techniques to realize efficient retrieval include the use of highly discriminative keys (HDKs) which are created from rare combinations of content and structure information and used to support efficient retrieval, in particular for multi term queries. Depending on a peer's status, we index documents either globally into distributed indexes or locally in connection with distributing peer summaries. Finally, structural information from XML-documents is used to select entries for pruned posting and peer lists.

**Keywords:** Information Retrieval, Peer-to-Peer, XML Information Retrieval, XML-Retrieval, Indexing, Peer Selection, Highly Discriminative Keys, Distributed Search

## 1 Introduction and Related Work

Peer-to-Peer networks are a potentially powerful form of distributed computing, where peers – large sets of equal and autonomous nodes – are pooled together to share resources such as storage and computing power. Activities involved in search include issuing requests (“querying”), routing requests (“query routing”), matching results (“ranking”), and responding to requests (“retrieval”) [8].

A precondition for efficient Information Retrieval (IR) is an adequate indexing strategy, which is the process of building an index over the document collection. Typically, an inverted index is used for storing the lexicon (index terms) and the posting lists (occurrences of the index terms in the collection) [2]. In P2P environments, two different methods of distributing an inverted index can be applied: document partitioning which is partitioning the document collection into several smaller sub-collections and indexing each of them, or term partitioning which is partitioning the inverted index by index terms [25].

Searching collections based on IR is a useful feature for the user. However, performing the search efficiently and maintaining scalability is one of the main obstacles in distributed environments [7]. Query processing consumes a considerable amount of resources even with centralized solutions; additional challenges arise in highly distributed P2P systems with bandwidth and latency constraints. The task of locating useful index information comes up, before relevant results can be ranked and retrieved. In order to reduce bandwidth consumption, P2P-IR algorithms aim at improving the efficiency of query evaluation by reducing the volume of index information fetched and processed, e.g. by using only selected information [12]. Selection techniques involve cutting off posting lists (“top-k pruning”) or contacting only a small subset of participating peers that contain many documents relevant to the query (“resource selection”) [8].

Approaches for Information Retrieval in P2P networks (P2P-IR) are a recent field of research. So far, acceptable response times are only achieved by solutions for simple pattern matching: documents are located in the P2P network by their name and only if they match the query exactly. An overview of P2P-IR approaches and criteria for their classification are presented in [17], whereas the issue of looking up data in P2P systems based on distributed hash tables (DHTs) is discussed in [3]. An efficient P2P search engine using IR techniques can be found in [15] where a key-based indexing strategy instead of single-term indexing is proposed, with keys being term sets that appear in a restricted number of documents, thus being highly discriminative keys. Two interesting approaches addressing the problem of efficiently selecting promising peers for answering a query are Minerva [4] and PlanetP [5]. Minerva is a distributed search engine on top of a DHT whose peer selection strategy is based on peer summaries. PlanetP is a content addressable publish/subscribe service for unstructured P2P communities that selects and contacts peers until k documents are found and until retrieving more documents would fail to contribute better results than these k documents.

The self-describing structure of XML can be a valuable source to achieve more precise and focused IR results when retrieving XML-documents. Methods include weighting diverse parts of documents differently; content and structure (CAS) queries enable users to specify structural constraints on what to retrieve; and retrieval units can consist of entire documents or only the most relevant parts of a document. A survey on indexing and searching XML-documents is conducted in [9]. Evaluation has shown successful application of XML Information Retrieval [10]. However, current approaches are all based on traditional client/server architectures.

Schema-based P2P networks or Peer Data Management Systems do apply XML in a P2P environment by providing techniques for the lookup of semi-structured data, e.g. by using XML P2P databases [23]. These systems consider exact or even partial matches and return either documents as a whole or XML fragments. However, they do not provide means to compute content relevance or structure similarity so far. A general idea about indexing, query routing, and query processing of XML data in a P2P network is presented in [6]. None of the analyzed approaches enables the use of IR techniques to compute relevance, though. The central challenge concerning the indexing process is identified as the handling of both value and path indexes. Deriving appropriate mappings of documents to peers is considered as the main problem of structured P2P systems. A fully self-organizing XML P2P database

system for sharing and querying XML data is presented in [19]. Information is requested with XQuery, so that a user can execute very expressive queries, but the results will have to match exactly.

To our knowledge, no P2P solutions for IR of XML-documents exist so far. Schema-based P2P-networks consider hints about the desired document structure but do not yet provide means to compute the relevance of documents. In this paper, we propose the first approach for XML Information Retrieval in a P2P system and concentrate on a novel indexing strategy for efficient retrieval.

## 2 Challenges for XML Indexing in P2P Systems

Approaches for ranking of XML-documents can incorporate structural information to improve retrieval quality. Additionally, retrieval units can be whole documents or document parts, therefore term statistics have to be collected for both. Even for centralized IR approaches, this introduces the challenge of storing the extra information efficiently in terms of storage space and access effort. Optimally, the information available for XML Information Retrieval would consist of all term statistics of all retrieval units in all XML documents for all possible query term combinations and would be accessible by all participating computers. However, this would result in a very exhaustive indexing process, especially if all indexing information is stored in a P2P network, with “churning” peers producing a highly dynamic data flow by joining and leaving the network. Hence, the information to be indexed and its distribution over the network have to be selected very carefully.

In P2P systems, the document collection to be queried is not static but undergoes constant change, e.g. by churning peers. In particular, there is no central index to store term weights and other evidence necessary for relevance computing, but the index is distributed over the network. Each newly indexed document leads to a bunch of messages sent between the peers to distribute the extracted information, and each churn of peers implies a redistribution of the existing information. Thus, communication overhead is a major cost factor [7]. The extra information stored and accessed for XML IR can additionally increase the amount of data transfer. Accordingly, the number and size of messages sent over the network should be minimized.

Due to the absence of a central index, peers cannot access required information directly but must locate it before use. Furthermore, the number of peers to be contacted for a specific query must be limited in order to guarantee scalability of the system – only carefully selected information can be used. Consequently, the index must be designed and distributed such that each peer can easily locate and access all content and structure information required for the peer’s participation in answering a specific query. The peer has to store this information locally or must have knowledge how to get it from other peers. Information relates to the content of documents and document parts as well as their structure.

Around 85% of all queries are multi term queries that consist of more than one query term [14]. Especially in P2P systems, handling of multi term queries turns out to be a further challenge, since posting lists of all query terms must be matched with each other. This is of particular concern for large collections and popular terms, when

very large posting lists are transferred for matching leading to massive network traffic. In most existing P2P-IR approaches, much bandwidth is consumed by locating and sending large posting lists for the different query terms over the network, and expensive joins of these posting lists are performed at querying time. However, quick access to term combinations is desirable, especially to reduce network traffic and to facilitate query execution.

### 3 A Hybrid Indexing Strategy Based on HDKs

Our proposal for an indexing strategy aims at efficient querying by several techniques: pre-computing of posting lists for popular query term combinations; reducing posting list sizes to a fixed limit by indexing only rare terms (or term combinations) - highly discriminative keys; indexing documents either globally to distributed indexes or locally in connection with distributing peer summaries; and using structural information given by the user or by the documents themselves to select the entries that are stored in posting and peer lists.

#### 3.1 Indexing Highly Discriminative Keys

Usually, documents are indexed per single terms. This might result in long posting lists for popular terms and in expensive joins of these posting lists for multi term queries. To avoid these problems, the use of highly discriminative keys was proposed in [15]. The original approach extracts all index terms of a document and distinguishes between rare and frequent keys. Only those keys whose global document frequency ( $df$ ) with respect to the whole collection does not exceed a threshold are considered rare and specific enough to describe the document's content. Frequent keys, i.e. with a global  $df$  above that threshold, are regarded as non-discriminative. They are combined with each other if hints imply that they might be used together in a multi term query, e.g. based on a query log analysis [21] or if they occur in the same window of the same document. Only rare keys and rare key combinations are considered as highly discriminative keys and thus are indexed. As a result, the posting list of a HDK never exceeds the threshold. Multi term queries are supported by those HDKs that consist of several terms. Despite increased indexing costs by indexing several combinations per frequent key, the total traffic generated is notably smaller than when using a distributed single-term indexing strategies [16].

In extension of this approach, our HDK-based indexing strategy is applied to XML-documents, where structure can be taken into account at several steps of the retrieval process and consequently must be indexed. For each term in a document, structural information about the XML elements that contain this term is extracted together with the content. We denote by XTerm a tuple of content (i.e. term) and its structure, which is the path from the document root element to the term element in the XML-document tree expressed with XPath. The extracted XTerms are used to build HDKs by combining XTerms such that each combination is rare. A HDK can either consist of a single rare XTerm, in which case the XTerm's global frequency in the document collection does not exceed the frequency threshold. Or the HDK consists of

a set of frequent XTerms if each XTerm is frequent but the combination of XTerms is rare. We denote an index key as being either a HDK or an XTerm. The global frequency of an XTerm  $t$  is computed as a combination of the  $df$  of  $t$ 's content (as in the original HDK approach) and the  $df$  of  $t$ , i.e. different frequencies for different structures are taken into account. However,  $df$  of  $t$ 's content has more impact on the global frequency, as we focus on content-based search and use structure as vague hint considering that the user might not be able to specify the respective structure precisely.

While extracting and building the HDKs, their term statistics are collected for later ranking. These statistics are not limited to static document statistics but include information about document parts, too: for each document, evidence for several of its potential retrieval units is collected. The decision, which passages of a document are chosen as potentially good retrieval units, is based on several factors, e.g. the depth of a passage's path, the size of the passage, experience from past retrieval, parameters set by the user, or if the passage is very focused regarding a specific topic.

To support efficient retrieval, pruned posting lists are indexed for frequent XTerms in addition to full posting lists for HDKs. There are two reasons for this. First, HDKs composed of more than one XTerm have posting lists that contain only documents in which all XTerms appear: the posting list  $pl(h)$  for a HDK  $h$  is created as intersection of the posting lists of all XTerms  $t_i$  in  $h$ , i.e.  $pl(h) = \bigcap pl(t_i \mid t_i \in h)$ . Entries for documents that contain only one  $t_i$  are therefore not stored; this might be an undesired effect for highly relevant documents. Thus, the most relevant documents are stored for frequent XTerms, too, and can later be included into the ranking. Second, not all queries are composed of HDKs completely. When a given query  $q$  is executed, it has to be split into HDKs. The query terms would ideally form a single existing HDK – the posting list for the query would have been already computed and can be used directly after locating it. In case the query has to be split into several HDKs, it might happen that not all query terms can be covered and non-discriminative XTerms are left. In [24] it was proposed to either ask the user to specify such XTerms by adding further hints on content or structure, or to merge posting lists of HDKs containing those terms. However, this results either in additional effort for the user or in computation costs and network traffic for the posting list merging. Using an additional index for frequent XTerms can increase efficiency. The posting lists for frequent XTerms are pruned to a limit  $PL_{max}$  which is proportional to the frequency threshold used for HDKs so that these posting lists are limited to a fixed number of entries, too. The approach in [22], that extends the original HDK algorithm by applying query-driven indexing, suggests truncating posting lists –if necessary– by ranking the posting list entries according to the BM25 relevance computation scheme. We propose a more elaborated formula for pruning, such as the scoring function shown in formula (1) which is applied to choose the top  $k$  best posting list entries for XTerm  $t$ .

$$\begin{aligned}
score_t(d_i) = & \alpha * w(tf_t(d_i), icf_t) \\
& + \beta * w(tf_t(ru_{best}), icf_t) \\
& + \gamma * score_t(p_i)
\end{aligned} \tag{1}$$

The function is used to sort the posting list of  $t$  by  $score_t(d_i)$  instead of simply using the term frequency as sorting key. This score is computed at indexing time and takes

into account weights for document  $d_i$  and its best retrieval unit  $ru_{best}$ , i.e. the retrieval unit with the highest weight for  $t$ . Alternatively, the average or even the sum of all indexed retrieval units of  $d_i$  could be considered. For the weighting function  $w$ , we adapted the BM25f formula [18] to XML IR. Parameters include the inverted collection frequency ( $icf$ ) of  $t$ ; the term frequency of  $t$  in  $d_i$ ; and the term frequency of  $t$  in  $ru_{best}$ . The  $score_t(p_i)$  depends on the quality of peer  $p_j$  on which  $d_i$  is stored. For instance, the peer score is high for peers with good collections regarding  $t$  and with good performance metrics such as response times. Values for  $\alpha$ ,  $\beta$ , and  $\gamma$  can be chosen such that  $1 \geq \alpha \geq \beta \geq \gamma \geq 0$ , i.e. the document weight will be favoured.

### 3.2 Hybrid Evidence Indexing

A peer can either index information locally or globally, depending on its status at indexing time. The more reliable and valuable a peer is, the more information it is allowed to send over the network (consuming more bandwidth and storage space) at indexing time, and the more information about the peer and its collection is available at querying time. A peer  $p$  is regarded as reliable, if it performed well in the past (e.g. provided highly relevant results for past queries) or if it provides a high quality collection (e.g. can achieve high scores regarding formula (1) for at least one  $t_i$ ). Peer characteristics such as response times, available bandwidth, open IP address (vs. NAT-bound), latency, and CPU/Memory are also considered. To provide adequate peer statistics, we developed a P2P protocol that is based on Kademia [11] and collects the desired information. Those peers will be preferred that have stayed online for at least  $x$  minutes, as the probability that a peer stays online increases with its uptime. For example, a network analysis of three file sharing systems found, that 65% of the peers joined the system online only once, that more than 20% of all connections lasted 1 minute or less, and that around 60% of the peers kept active no longer than 10 minutes each time they joined the system [20]. Of course, the peers' behaviour varies depending on the application. In our proposal, a relatively stable system is assumed where peers usually stay online over longer time periods. However, peers with short online times or which join the system only once and then disappear forever should not be allowed to perform an exhaustive indexing.

Our hybrid strategy distinguishes between peers with different status. Peers with full reliable status can store all extracted evidence into indexes distributed over the network, i.e. all term statistics are stored per document. Peers with low status (or if the user chooses "quick indexing") can store the extracted evidence only locally; a summary of the peer's overall collection will then be stored in the distributed indexes, i.e. all term statistics are stored per local collection. Hence, the indexing process distributes two lists for each key: a posting list with entries for documents (which were indexed per document), and a peer list with entries for peers (which hold documents that were indexed per collection). In the retrieval process, entries from both lists are taken into account, with a bias to posting list entries since these host the more significant information.

## 4 An Architecture Based on Distributed Indexes

Index information will be stored and accessed as shown in the architecture of each peer in figure 1. Black arrows denote local interaction between components of the same peer, whereas dotted arrows denote interaction between components of different peers via the P2P complex. The search engine is accessed by graphical user interface (GUI) which is part of the application complex. All interaction between components of different peers such as lookup and storage of indexed information are handled by the P2P complex.

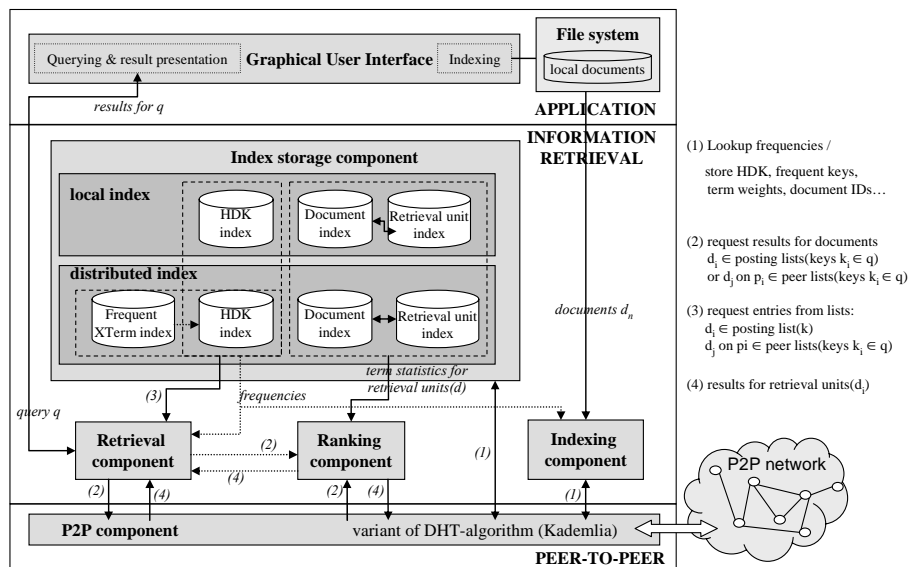


Fig. 1. A peer's architecture with several local and distributed indexes

In the following, we focus on the Information Retrieval complex which performs indexing, querying, and ranking of XML-documents. Both a term-partitioned and a document-partitioned strategy [25] are combined to store (and later retrieve) all indexing information: HDK index and frequent XTerm index act as term-partitioned inverted indexes by storing posting and peer lists of keys; document index and retrieval unit index hold the evidence used for relevance computing and are implemented both in a distributed clustered version and in a local version for sub-collections.

The INDEXING COMPONENT parses XML-documents to extract all XTerms, creates HDKs from the extracted XTerms, and collects term statistics for potential retrieval units. For the HDK creation, global frequencies are requested via the P2P complex and received from the distributed HDK index and the frequent XTerm index. The following information is stored in local and distributed indexes:

- **DISTRIBUTED HDK INDEX:** For each HDK  $h$ , its global frequency, a posting list, and a peer list is stored. The posting list contains documents in which  $h$  occurs and is ordered by term frequency. These documents and their statistics are not stored locally but distributed among other peers. The peer list is ordered by peer score and contains peers that hold documents in which  $h$  occurs.
- **LOCAL HDK INDEX:** The same information as in the distributed version of this index is stored with exception of peer lists. All information relates to locally stored documents of the same peer.
- **FREQUENT XTERM INDEX:** For each frequent XTerm  $t$  with global  $tf$  exceeding threshold  $PL_{max}$ , a pruned posting list for the  $PL_{max}$  best documents and a pruned peer list for the best peers are stored. Links to all HDKs containing  $t$  are stored, too.
- **DOCUMENT INDEX:** For each document, statistics are maintained to compute the relevance of the document in the ranking process. Additionally, links to possible retrieval units of each document are stored. The statistics are represented by vectors to support ranking based on an extended vector space model. The local document index stores statistics of documents stored directly on the same peer (and indexed per collection). The distributed version of the index stores statistics of documents stored on other peers (and indexed per document).
- **RETRIEVAL UNIT INDEX:** Term statistics for the retrieval units of each indexed document are kept in vectors where each vector component represents the weight of an XTerm occurring in this retrieval unit.

In the retrieval process, the indexes are used as follows: Information from HDK index and frequent XTerm index is used by the retrieval component of a querying peer to decide which peers should participate in answering a given query  $q$ . Therefore, the query is sent to all peers with posting lists and peer lists for HDKs covering the query. The retrieval components of these peers will consult their lists to select promising peers and redirect the query to the selected peers. These peers hold statistics of potential relevant documents in their local document index or in their locally stored part of the distributed document index (and also can direct access the retrieval unit statistics). Once they receive the query, their ranking components will perform the ranking and send back retrieval results to the querying peer. In the GUI of the querying peer, a ranked list of links to relevant documents and retrieval units is presented to the user who can access them directly on the source peers.

## 5 Outlook

In this paper, a novel indexing strategy for efficient XML Information Retrieval in P2P systems has been proposed in order to achieve reduction of bandwidth consumption, support of multi term queries, and quick access to distributed information. The strategy is based on highly discriminative keys; posting lists for popular query term combinations are pre-computed at indexing time and their size is limited to a fixed threshold. Indexing is performed according to a hybrid strategy and depends on the indexing peer's status: only reliable peers can distribute exhaustive information; peers with an online time that is expected to be short can only store

evidence summaries. For the pruning of posting and peer lists, structural information is used to select the list entries. Finally, an architecture was designed that supports efficient retrieval by employing the proposed methods.

At present, we are implementing SPIRIX, a P2P search engine for Information Retrieval in XML-documents, on top of the centralized search engine Terrier [13]. A P2P protocol based on Kademia has been developed that collects the peer statistics for the hybrid evidence indexing. For the ranking, we extended the vector space model with BM25F-oriented weighting to the use of XML IR by including weights for structural information. Experiments with our prototype to evaluate optimal parameters for the proposed algorithms will start soon.

## References

- [1] Amer-Yahia, S.; Lalmas, M.: *XML Search: Languages, INEX and Scoring*. SIGMOD Rec. Vol. 35, No. 4, 2006.
- [2] 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.
- [3] 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.
- [4] Bender, M.; Michel, S.; Weikum, G.; Zimmer, C.: *The MINERVA Project - Database Selection in the Context of P2P Search*. BTW Conference 2005, Karlsruhe, Germany, 2005.
- [5] Cuenca-Acuna, F.; Peery, C.; Martin, R.; Nguyen, T.: PlanetP - Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In: Proc. of the 12th IEEE International Symposium on High Performance Distributed Computing, Washington, USA, 2003.
- [6] Koloniari, G.; Pitoura, E.: *Peer-to-Peer Management of XML Data: Issues and Research Challenges*. SIGMOD Rec. Vol. 34, No. 2, 2005.
- [7] 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.
- [8] Lu, J.; Callan, J.: *Federated search of text-based digital libraries in hierarchical peer-to-peer networks*. Information Retrieval Vol. 9, No. 4, Springer-Verlag, 2006.
- [9] Luk, R.; Leong, H.V.; Dillon, T.; Chan, A.: *A Survey in Indexing and Searching XML Documents*. JASIST, Vol. 53, No. 6, 2002.
- [10] Malik, S.; Trotman, A.; Lalmas, M.; Fuhr, N.: *Overview of INEX 2006*. In: Proc. of the Fifth Workshop of the INitiative for the Evaluation of XML Retrieval, Germany, 2007.
- [11] Maymounkov, P.; Mazieres, D.: *Kademia: A peer-to-peer information system based on the xor metric*. In Proceedings of IPTPS02, Cambridge, USA, 2002.
- [12] Moffat, A.; Webber, W.; Zobel, J.; Baeza-Yates, R.: *A pipelined architecture for distributed text query evaluation*. Springer Science + Business Media, LLC 2007.
- [13] Ounis, I.; Amati, G.; Plachouras, V.; He, B.; Macdonald, C.; Lioma, C.: *Terrier: A High Performance and Scalable Information Retrieval Platform*. In: ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006). Seattle, USA, 2006.

- [14] Pass, G.; Chowdhury, Abdur; Torgeson, Cayley: *A Picture of Search*. In: Proc. of First Int. Conference on Scalable Information Systems (Infoscale 2006), Hong Kong, 2006.
- [15] Podnar, I.; Luu, T.; Rajman, M.; Klemm, F.; Aberer, K.: *A Peer-to-Peer Architecture for Information Retrieval Across Digital Library Collections*. In: Proc. of European conference on research and advanced technology for digital libraries (ECDL'06), Alicante, Spain, 2006.
- [16] Podnar, I.; Rajman, M.; Luu, T.; Klemm, F.; Aberer, K.: *Scalable Peer-to-Peer Web Retrieval with Highly Discriminative Keys*. In: Proc. of IEEE 23<sup>rd</sup> International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey, 2007.
- [17] 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.
- [18] Robertson, S.; Zaragoza, H.; Taylor, M.: *Simple BM25 extension to multiple weighted fields*. In: Proc. of CIKM'04, ACM Press, New York, USA, 2004.
- [19] Sartiani, C.; Manghi, P.; Ghelli, G.; Conforti, G.: *XPeer: A Self-organizing XML P2P Database System*. Lecture Notes in Computer Science No. 3268, Springer-Verlag, 2004.
- [20] Sen, S.; Wang, J.: *Analyzing peer-to-peer traffic across large networks*. IEEE/ACM Transactions on Networking, Vol. 12, Issue 2, 2004.
- [21] Skobeltsyn, G.; Aberer, K.: *Distributed Cache Table: Efficient Query-Driven Processing of Multi-Term Queries in P2P Networks*. ACM P2PIR'06, Virginia, USA, 2006.
- [22] Skobeltsyn, G.; Luu, T.; Podnar, I.; Rajman, M.; Luu, T.; Aberer, K.: *Web Text Retrieval with a P2P Quer-Driven Index*. ACM SIGIR'07, Amsterdam, The Netherlands, 2007.
- [23] Steinmetz, R.; Wehrle, K. (eds.): *Peer-to-Peer Systems and Applications*. Lecture Notes in Computer Science No. 3485, Springer-Verlag, 2005.
- [24] Winter, J.; Drobnik, O.: *Peer-to-Peer Cooperation for Content-Oriented XML-Retrieval*. International Workshop on Peer-to-Peer Computing for Information Search (P2PSearch 2007), Jeju-Island, Korea, 2007.
- [25] Zobel, J.; Moffat, A.: *Inverted Files for Text Search Engines*. ACM Computing Surveys, Vol. 38, No.2, Article 6, 2006.