# Towards Virtual Knowledge Communities in Peer-to-Peer Networks

Melanie Gnasa, Sascha Alda, Jasmin Grigull, and Armin B. Cremers

Institute of Computer Science III, University of Bonn, Römerstrasse 164,
53117 Bonn, Germany
{gnasa, alda, grigull}@cs.uni-bonn.de

**Abstract.** As a result of the anonymity in todays Web search, it is not possible to receive a personalized search result. Neither prior search results nor search results from other users are taken into consideration. In order to resolve this anonymity towards the search engine, a system is created which locally stores the search results in the scope of a peer-to-peer network. Using the Peer Search Memory (PeerSy) all relevant results are stored and associated with the corresponding queries. By this means, repeated access is facilitated. Furthermore, sharing of associations between queries and relevant results in the peer-to-peer network allows grouping of Virtual Knowledge Communities (VKC) in order to obtain a surplus value in knowledge sharing on the Web.

## 1 Introduction

The rapid development of technologies associated with the Web offers the possibility of a new personalized search in a distributed environment. Today, Web search engines have to deal with two main problems: (1) no personalized search results based on preceding queries of the user exist and (2) regardless of the information need of other users these results are not taken into account as feedback information. In this light, we present a personalized approach for building Virtual Knowledge Communities (VKCs) based on personal search memories.

These VKCs represent a variety of topics, which are interesting for a multiplicity of searches. The interest is measured by the frequency of queries to a special topic. The Google search statistics Zeitgeist[1] reveals that different topics are requested more frequently in certain weeks. In addition the major search interests diverge throughout different countries. For example, the Google Zeitgeist statistics 2002 summarizes that the Google search traffic follows the Las Ketchup craze as it circles the globe. Due to the fact that today usually no personalized search is performed it is not possible to profit from the results of previous queries to the same topic. The search engine Google processes 250 million queries daily. However, these queries are not made available to the public. The benefit of these queries would only be perceivable if additional feedback information for relevant hits had been logged. So far, each user stores high-quality links locally in his

[1] http://www.google.com/press/zeitgeist.html

bookmark or favorite list [Jones et al., 2001]. Thereby an association between queries and relevant results is not provided. For the surplus value of these associations throughout the multiplicity of searches two possibilities do exist: (1) the search engine provider could save the relevant hits next to all queries in a central storage or (2) each user could save his or her relevant results locally according to the requested queries, which in turn could be shared in a distributed environment. The advantage of a pure central system is the high availability and the multiplicity of data that can be collected and retrieved. However, these systems exhibit some drawbacks, such as the existence of a single-point-of-failure: if the server fails, the whole systems will become unavailable for all depending clients. Furthermore, no support is usually provided to create VKCs to subdivide clients according to their interests or specific knowledge.

Recent peer-to-peer architectures [Barkai, 2002] are conceived to support decentralize systems as described in the second possibility. As far as reasonable, these architectures abandon from having any kind of central servers to store data. The data remains at the edges of the Internet (personal computers), thus, eliminating the single-point-of-failure problem. Each peer is thereby autonomous in the decision, to what time and to what extend data is to be shared with the environment. Another promise of peer-to-peer is the facility to build so-called peer groups, which allows groups of peers to restrict the access to distinct data to authorized peers only. In the course of our research, we propose the adoption of the peer-to-peer ideology to realize distributed knowledge communities that accomplish their members to share knowledge of Web content in terms of enriched links to specific topics.

The rest of this paper is organized as follows. Some basic assumptions about distributed knowledge sharing are illustrated in chapter 2. Chapter 3 presents PeerSy, our tool to save query-result associations in a Peer Search Memory. Our approach for Virtual Knowledge Communities is elucidated in Chapter 4. Chapter 5 gives a short survey on related work to our research. A conclusion finally sums up the main aspects of this paper and presents the future work.

## 2 Distributed Knowledge Sharing

In order to assist a user with his daily usage of the Web we need more than simple file sharing via search engines or peer-to-peer networks like Gnutella and Freenet. For the sharing of knowledge common Web technologies have to be adapted. This is the reason why we propose two techniques which are integrated into the knowledge creation cycle. These techniques are motivated by the transformations between implicit and explicit knowledge and the current state of the art of searching the Web.

The human knowledge can be divided into two categories: implicit and explicit knowledge (cf. [Nonaka, 1991], [Stanoevska-Slabeva et al., 1998]). The implicit knowledge depends on a person and is embedded into a certain context. In contrast, explicit knowledge is the externalisation of implicit knowledge, i.e. information. Explicit knowledge is perceived when information, context and ex-

periences are combined to form new implicit knowledge [Nonaka, 1991]. The creation and classification of documents is part of the externalisation process. The retrieval is part of the internalisation process. The general advantage of *full text search* in the Web in the context of knowledge sharing is achieved through the automatic indexing accomplished by robots. The automatic classification makes this part of the externalisation very efficient. The general disadvantage of the common full text search engines is the lack of effectiveness of the search results (e.g. caused by synonymy and polysemy of query terms). For this reason this approach is very weak for the internalisation process in the knowledge cycle (cf. [Stanoevska-Slabeva et al., 1998]).

Ideally a system should fulfill both requirements. The full text search is on the one side highly effective in the externalisation of knowledge, on the other side, however, highly ineffective in the internalisation. In order to close this gap, we have developed two approaches for a distributed knowledge sharing:

1. **Peer Search Memory (PeerSy)**
   Each user produces individual needs for his own image of the Web through evaluated results. Thus, the problem of polysemy that occurred in the context of search engines reduces itself to a local result set in a specified context. Once an ambiguous query is requested, the anonymity against the search machine can be averted and the user receives an answer, which is filtered for his or her own context. Through the additional storage of associations between queries and results lexicographic indexing can be replaced and a grouping of synonymous terms is possible. This way search histories form the basis for effective internalisation and externalisation.
2. **Virtual Knowledge Communities (VKC)**
   The structure of a distributed retrieval environment in the Web reveals the possibility to form Virtual Knowledge Communities on the basis of search memories. Each participant can thus profit from the knowledge of the group. In contrast to the classical full text search the process of the internalisation becomes more effective.

Both techniques can be integrated into the cycle of knowledge generation. Figure 1 depicts this process. Through PeerSy implicit knowledge can be transferred into explicit knowledge. A distributed Information Retrieval System based on Virtual Knowledge Communities makes the internalisation more effective.

## 3 PeerSy - Peer Search Memory

PeerSy is developed to overcome the weaknesses of the conventional search machines and the current Web browsing technology with the access to well-known Web sites. PeerSy is integrated into the system ISKODOR which is being presently developed at the University of Bonn. ISKODOR is a knowledge tool for the World Wide Web on the basis of peer-to-peer technologies. With the help of the search interface MySearch, which is part of the ISKODOR system, the user can look for new or well-known Web sites. With MySearch the

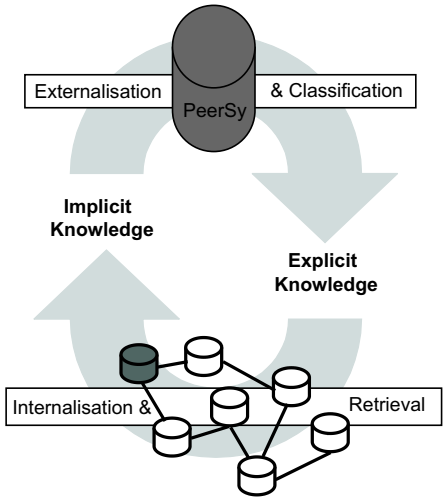


**Fig. 1.** Knowledge generation cycle

searching process is extended to not only take place in the Web, but also in the local PeerSy database and the PeerSy contents of other peers. All sites, which have been evaluated and rated interesting by the user are stored in the personal database PeerSy. The user this way assembles his "own personal Web", which on the one hand consists only of a tiny fraction of the entire Web and on the other hand merely contains sites, which are interesting to the user. The search for well-known sites takes place in the small cutout of the Web, in his personal database only. All associations between queries and relevant results contained in PeerSy can formally be defined as follows:

**Definition 1 (Single Associations)** *A set of links $L$ which represent relevant results and a number of $n$ query terms $T$ for a peer $p$ are given. A query consists of at least one query term. The set $Q$ of all possible queries over all terms $t \in T$ is*

$$Q_p = P(T) \setminus \{\}$$

*The set $B$ of all possible relevant links to a query is*

$$B_p = P(L) \setminus \{\}$$

*The relation $SingleA$ between the sets $Q$ and $B$ describes all possible* ***single associations*** *between a query and a set of relevant results, which are stored in PeerSy for peer $p$:*

$$SingleA_p(Q_p, B_p) = \{(q, b) \in Q_p \times B_p | (q, b) \in PeerSy_p\}$$

Thus, for each peer the individual information need is identified. A further summary of these search interests on the local peer level is waived. It cannot be

assumed that certain topics are investigated representatively. Only the summary of all query-link associations over all peers makes it possible to form meaningful emphases. How these knowledge communities are formed and how the retrieval is realized to enhance the internalisation process, is described in the next section.

## 4 Virtual Knowledge Communities

As already brought up in the introducing chapter, peer-to-peer architectures serve as the best candidate to build Virtual Knowledge Communities. Peers span a decentral virtual network over some existing physical networks, thus, hiding the complexity of connecting peers despite firewalls or subnets (see figure 2). The ability of peers to self-organize into peer groups can be utilized to build private Virtual Knowledge Communities on top of these peers. Communities represent a place for peers sharing interests or competencies within a common knowledge domain. Appropriate authorization routines thereby have to prevent the non-restrictive access to knowledge or to resources. Note that each peer can necessarily belong to multiple groups. All these concepts have been conceptualized for instance in JXTA ([Sun, 2003]), the de-facto standard protocol suite to build peer-to-peer architectures.

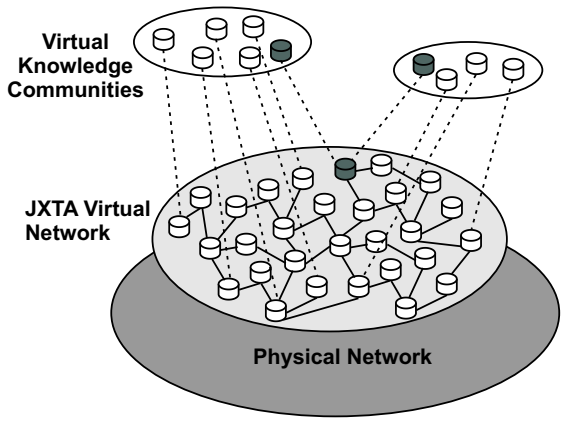


**Fig. 2.** Peer-to-Peer Network with semantic layer for Virtual Knowledge Communities

### 4.1 Building of Virtual Knowledge Communities

Similar to the clustering of data and documents [Jain and Dubes, 1988] a Virtual Knowledge Community (VKC) is described through a selected representative. The clustering of documents results in a partition of documents, which can be described by a representative. For the building of a VKC this procedure

is reversed and first the possible representative of a group is computed. All peers, which have single associations that overlap with the representative can be member of this group. The growth of such Virtual Knowledge Communities is described in detail in the next subsection. The creation of representatives of VKCs from a set of $n$ peers takes place in four steps.

In **step 1** all single associations from all peers $p$ are unified. Thus the multi-set $A$ contains all single associations from all peers:

$$A(Q,B) = \bigcup_{p \in P} SingleA_p(Q_p, B_p)$$

with $Q = \bigcup_{p \in P} Q_p$ and $B = \bigcup_{p \in P} B_p$. The multi-set characteristic is very important in this step in order to measure the frequency of queries and the relevant results. For example it has arisen the following multi-set $A_{exp}$:

$$\begin{aligned} A_{exp} = \{\ & (\{java\}, \{java.sun.com\}), (\{jvm\}, \{java.sun.com\}), \\ & (\{java\}, \{java.sun.com, www.javaworld.com, java.apache.org\}), \\ & (\{haskell, tutorial\}, \{www.haskell.org/tutorial\}), \\ & (\{java\}, \{www.sun.com, java.sun.com\}), \\ & (\{php, tutorial\}, \{www.php.net/tut.php\}), \\ & (\{java, tutorial\}, \{java.sun.com/doks/tutorials\})\} \end{aligned}$$

In **step 2** all overlapping queries and links are unified. Through the combination of all local peer interests two kinds of overlaps can be identified between elements of A. Through the combination of all local PeerSy contents it is now possible that individual query terms emerge in different queries. Therefore it is important to form the set of overlapping queries as large as possible. The combination of all these query sets is denoted *QCloud*. For example an element from *QCloud* would be the set of queries $\{\{java\}, \{java\}, \{java\}, \{java, tutorial\}\}$ for the example set $A_{exp}$. Still the multi-set property of $A$ is important and necessary to reflect repeated queries from the peers. The function *QCloud* over $A$ describes the set of all overlapping queries $u \in U = P(Q) \setminus \{\}$:

$$\begin{aligned} QCloud(A) = \{u \in U | & \bigcap u \neq \emptyset, u \subseteq domA, \\ & \forall v, v \subseteq domA, \bigcap v \neq \emptyset, u \subseteq v \Rightarrow u = v\} \end{aligned}$$

In contrast to the union over the query terms, identical links can occur in the tuples $(q, b) \in A$ with $q \in Q$ and $b \in B$. With the function *LCloud* the set of all grouped links can be described:

$$LCloud[u] = \bigcup Links[u]$$

The function Links[u] is an auxiliary function and supplies the union of the pertinent link associations from the relation $A$ to each element $u \in U$ with:

$$\begin{aligned} & Links[u] = \bigcup A[u] \\ & A[u] \text{ refers to the second position in the tuple } (q, b) \in A \\ & A[u] = \{b | \exists q \in u : qAb\} \end{aligned}$$

For example an element from $LCloud$ would be the set of results

$$\{java.sun.com, java.sun.com, java.sun.com, java.sun.com, \\ www.sun.com, www.javaworld.com, java.apache.org\}$$

for the example set $A_{exp}$.

In **step 3** all overlapping query and link sets are summarized. By means of the relation $SAQ'$ (Seldom Asked Queries) all queries with the corresponding links are summed up.

$$SAQ'(u,b) = LCloud|_{QCloud(A)} = \{(u,b)|u \in QCloud(A) \wedge b = LCloud[u]\}$$

It can occur that the same links are stored to different queries. For this reason all queries for these links must be summarized in $SAQ$. This case appears, if the same information need with different queries is described.

$$SAQ(u,b) = \{(u,b)|b \in rangeSAQ' \wedge u = \bigcup SAQ'^{-1}[b]\}$$

For the example set $A_{exp}$ the following SAQ set can be computed:

$$SAQ_{exp} = \Big\{ \big(\{\{java\},\{java\},\{java\},\{java,tutorial\},\{jvm\}\}, \\ \{java.sun.com, java.sun.com, java.sun.com, java.sun.com, \\ www.sun.com, www.javaworld.com, java.apache.org\}\big), \\ \big(\{\{java,tutorial\},\{haskell,tutorial\},\{php,tutorial\}\}, \\ \{www.haskell.org/tutorial, java.sun.com/doks/tutorial, \\ www.php.net/tut.php\}\big)\Big\}$$

**Step 4** consists of the assignment of all VKC representatives. In the last step all elements from the relation $SAQ$ are selected that possess a certain size of associated queries and links. This size depends on $x$, the minimum number of queries, and $y$, the minimum number of links. Both values again depend on the total number of peers. At present the experimental investigation of these values takes place.

$$VKC(U,B) = \{(u,b)|\ u \in domSAQ, b \in rangeSAQ, \\ \|u\| > x \wedge \|b\| > y\}$$

After the four steps, a set of VKC representatives is found. For instance the following representative can be selected from $SAQ_{exp}$:

$$VKC_{exp} = \Big\{ \big(\{\{java\},\{java\},\{java\},\{java,tutorial\},\{jvm\}\}, \\ \{java.sun.com, java.sun.com, java.sun.com, java.sun.com, \\ www.sun.com, www.javaworld.com, java.apache.org\}\big)\Big\}$$

The problem of different verbalizations of the same information need on individual peers can be intercepted by viewing the entire peer-to-peer net. Over

*QClouds* alternative queries can be reflected and over *LClouds* frequently and good-evaluated links are collected. Both functions detect synonymous descriptions of a special issue. The problem of polysemy cannot be solved by this procedure so far. However, in the future a detailed analysis of each VKC on a semantic level is planed in order to achieve a further partition.

### 4.2 Growing of Virtual Knowledge Communities in a Peer-to-Peer Network

The growing of a VKC over some phases is visualized in figure 3.

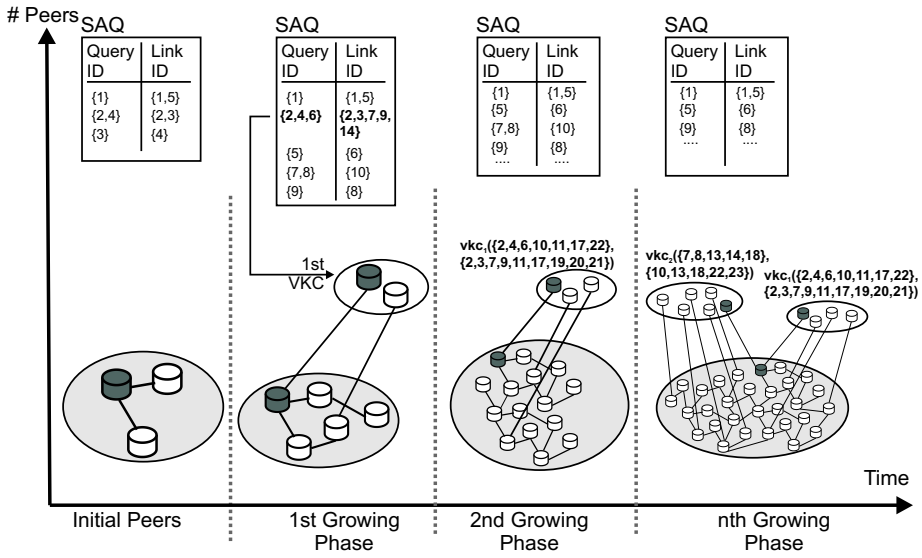


**Fig. 3.** Growing of Virtual Knowledge Communities

In the upper half one can see the SAQ relation – depicted as a table – for each phase containing query and link IDs, which have been collected by the individual peers. One can consider the SAQ table as a database that has been realized on a well-known peer existing in the virtual network. This well-known peer is capable to receive so-called seldom asked queries from peers, which could not be assigned directly to a certain existing group. As time continues, the SAQ table grows with more links and queries, but also the number of peers having joined the virtual network. If a certain number of queries and links is transcended, representatives for Virtual Knowledge Communities can be identified (step 4 in section 4.1). In the given scenario in figure 3, a representative $vkc_1 \in VKC$ is computed in the first growing phase, after the virtual network has been initialized. The pertaining peer group is built afterwards, containing the respective

peers that have previously pushed the query-link associations to the SAQ table. The generation of this new peer group will then be announced to all interested peers. Peers can pre-select the announcements for new Virtual Knowledge Communities through personalized filters that are placed on the same well-known peer. Once a VKC is created the respective representative for it is removed from the SAQ ($SAQ \setminus vkc_x$).

### 4.3 Distributed Retrieval System

Figure 4 demonstrates how a single query is being processed by ISKODOR. The user initiates the process by sending out a query-request. This request is then being sent to three different working units.

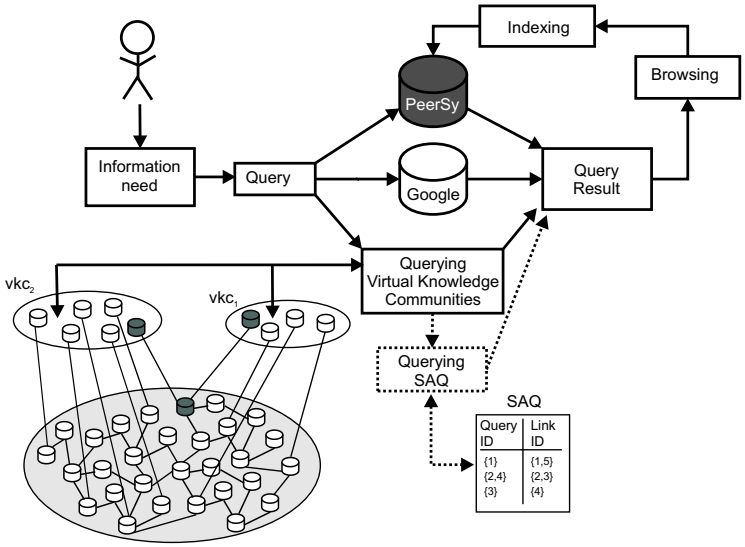


**Fig. 4.** Architecture of ISKODOR

A common internet search engine, like Google, executes its general query-process. At the same time, the PeerSy database is activated, looking for memorized links that the user has used formerly. In parallel to those two processes, the query is being sent out to the peer network. The query is being matched to the representative $vkc_x \in VKC$ of the different VKCs, starting with those groups, the user is already member of and then extending the matching onto the rest of the groups. There are two possibilities to consider now: (1) In case there is a match and the user/peer is already part of this group, he simply gets the appropriate links to his query. If the user/peer is not part of the group, the group itself can decide on how the peer could now join the group and on how many and what kind of links the user gets an inside on. This way the privacy of

the group is assured and the group itself can decide on what kind of protection is needed. (2) The other case occurs if no VKC exists to this specific query and all matches with the representative were unsuccessful. In this case the query is being matched to the SAQ-list to find out if queries and links exist, that were just too few in number to create a Virtual Knowledge Community. If this matching has been successful, the query is being inserted to the corresponding queries and links. At this point the growing of the queries for this field of interest can even lead to the creation of a new peer group based on the queries and links concerned. If no match was found, the query is not being inserted into the SAQ-list. The network cannot satisfy the query-request and no corresponding links are sent back to the initiator.

After these processes are being run in parallel, the links from the three sources are being represented to the user. The user can now navigate through the hits and put relevant links into the PeerSy database. Once a new query-link entity has been inserted that had been found by an internet searching machine, a matching with the representative of the peer groups is initiated as described above. If a matching is found, the corresponding peer has the possibility to join this group. If no match is found, the query-link association is being inserted into the SAQ-list.

### 4.4 Design Improvements

This section discusses some design improvements of the overall ISKODOR system, which we will take into further consideration during the realization of the system.

The realization of the SAQ table as a single database is intuitive, but breaks the original notion of the peer-to-peer paradigm to abandon from having any kind of central server. If the SAQ server failed, the entire architecture as elucidated in figure 4 would fail. Moreover, the SAQ table may grow to a very large and unmanageable table after the nth growing phase. A possible solution we are currently analyzing is the implementation of the SAQ table as a distributed hash table (DHT). Likewise to conventional hash tables, a DHT is a data structure that maps "keys" to "values". The difference is that DHTs can be distributed over several nodes within a network. A concrete implementation of such a DHT is realized in the Content-Addressable Network (CAN) model by Ratnasamy et al. [Ratnasamy et al., 2001]. In this model, all peers are arranged in a (logical) $d$-dimensional Cartesian coordinate space. The entire coordinate space is dynamically partitioned into so-called zones among all the nodes in the system. Each node is dedicated as the owner of exactly one zone. The partition into zones is utilized to conduct requests (insert, lookup, or delete) to key/value pairs. Each key is mapped onto a point in the coordinate space through a common hash function. This point does also correspond to a distinct zone that is maintained by a peer. Following the CAN model the value of this key can be inserted or retrieved in the hash table of this peer. If this zone is not maintained by the requested node, the request is routed through a range of intermediate nodes towards the

nodes that contains the key in his local hash table. To do so, each node additionally maintains a routing table that contains of a number of adjacent nodes in the table. The topology of a CAN-based system is not fixed, new nodes can be inserted, existing nodes can be deleted and so on. For more information about the algorithms for altering the topology see [Ratnasamy et al., 2001].

In order to improve our approach we are about to adopt the CAN model to store seldom asked queries in a decentral manner. Query-link associations are thereby represented by key/value pairs. During the initial phase, the SAQ table still remains on an initial peer, which is accessible through a well-known address. If a VKC is created, the initial table is splitted into two equally sized tables: one table remains on the initial peer, while the other table is migrated to the respective rendezvous peer that does represent and manage the VKC. In the following phases, the procedure of splitting and migrating a given SAQ table is conducted each time a VKC is identified within a SAQ table on a distinct rendezvous peer. Each rendezvous will also maintain a routing table consisting of adjacent rendezvous peer. This table will be updated each time a new SAQ table has been created. A query to a SAQ table can start from an arbitrary rendezvous peer. If the query cannot be matched in the local SAQ table, the query will be routed through all adjacent rendezvous peers.

In this regard, one essential question that has to be faced concerns the scalability of the presented model. According to the CAN model each peer has to maintain only $2d$ adjacent peers. Note that this number, in contrast to other routing strategies, is independent from the overall number of peers in the system. Thus, the number of nodes may grow without increasing the size of a routing table. One can conclude that the CAN model but also our adoption of CAN to our approach does scale even for huge numbers of new peers.

## 5 Related Work

In the following section some related systems are presented. Basically we refer to other studies which outline different methods to memorize and share bookmarks and to form groups based on interest. Particularly, we portray three systems that cover these areas of interest.

**WebView** (cf. [Cockburn et al., 1999]) is a prototype designed to improve the efficiency and usability of page revisitation. It does this by integrating many revisitation capabilities into a single display space, an add-on window that interacts with unaltered versions of Netscape Navigator. Whenever the user visits a page in Netscape, WebView keeps track of the page and applies a tree-like structure to the stored page-links. In contrast to PeerSy, this tool keeps track of every page that has been visited and it does not link the query that led to the page to the appropriate bookmark. Furthermore, this system does not support the collaborative aspect of sharing bookmarks between group members or the creation of groups of interest.

The system **CIRE (Collaborative Information Retrieval Environment)** (cf. [Romano et al., 1999]) is dedicated to support collaborative informa-

tion seeking and retrieving. It constitutes the implementation of an integrated knowledge creation environment in which IR (Information Retrieval) and GSS (Group Support Systems) are combined to provide integrated group support for all tasks required for teams to work together, including information retrieval. The difference to ISKODOR appears to be the client-server architecture the system is build on. The ISKODOR architecture exploits both the extensive distributed resources available at the peers in addition to a centralized repository of the SAQ-list. This minimizes the role of centralized resources for low cost and scaling.

**YouSearch** (cf. [Bawa et al., 2003]) is a distributed (peer-to-peer) search application for personal Web servers operating within a shared context. It supports the aggregation of peers into overlapping (user defined) groups and the search over specific groups. The hybrid peer-to-peer architecture is augmented with a light-weight centralized component. In comparison to ISKODOR, YouSearch does not provide bookmark-sharing, but more or less file-sharing which is supported by search mechanisms. Another difference is that groups are formed via manual user action only and the system does not conduct any proposals (e.g. directly approaches the peers concerned to recommend a group creation) to support and enhance this process.

## 6 Conclusion

In this paper we presented our notion of Virtual Knowledge Communities based on peer-to-peer networks enabling users to share knowledge about web content on the basis of personal search memories. As future work we see the transformation of the presented concepts towards a prototype based on the Java platform and on the JXTA framework for peer-to-peer architectures. We intend to evaluate both the concepts and the prototype in a project made up of lawyers, who aim to share query-link associations for wordings of a law or annotations of convictions.

## References

[Barkai, 2002] Barkai, D., editor (2002). *Peer-to-Peer Computing. Technologies for sharing and collaboration on the Net.* Intel Press.

[Bawa et al., 2003] Bawa, M., Bayardo, R. J., Rajagopalan, S., and Shekita, E. (2003). Make it fresh, make it quick - searching a network of personal webservers. In *Proc. ACM in Budapest, Hungary.*

[Cockburn et al., 1999] Cockburn, A., Greenberg, S., McKenzie, B., Smith, M., and Kaasten, S. (1999). Webview: A graphical aid for revisiting web pages. In *Proc. of the OZCHI'99 Austalian Conference on Human Computer Interaction.*

[Jain and Dubes, 1988] Jain, A. and Dubes, R. C., editors (1988). *Algorithms for Clustering Data.* Prentice Hall, pub-prentice:adr.

[Jones et al., 2001] Jones, W., Bruce, H., and Dumais, S. (2001). Keeping found things found on the web. In *Proc. of the 10th Int. Conf. on Information and Knowledge Management.*

[Nonaka, 1991] Nonaka, I. (1991). The knowledge creating company. *Harvard Business Review*, pages 96–104.

[Ratnasamy et al., 2001] Ratnasamy, S., Francis P., Handley, M., Karp, R., and Shenker, S. (2001). A Scalable Content Addressable Network. In *Proceedings of ACM SIGCOMM 2001.*

[Romano et al., 1999] Romano, N. C. J., Roussinov, D., Nunamaker, J. F. J., and Chen, H. (1999). Collaborative information retrieval environment: Integration of information retrieval with group support systems. In *Proc. of the 32nd Hawaii Int. Conf. on System Science 1999.*

[Stanoevska-Slabeva et al., 1998] Stanoevska-Slabeva, K., Hombrecher, A., Handschuh, S., and Schmid, B. (1998). Efficient information retrieval: Tools for knowledge management. In *Proc. the 2nd Int. Conf. on Practical Aspects of Knowledge Management (PAKM98)*, pages 23.1–23.6, Basel (Switzerland).

[Sun, 2003] Sun Microsystems Corp. (2003). Jxta v2.0 protocols specification. URL: http://spec.jxta.org/v2.0/.