An Architecture for Peer-to-Peer Information Retrieval

Karl Aberer, Fabius Klemm, Martin Rajman, Jie Wu

School of Computer and Communication Sciences
EPFL, Lausanne, Switzerland

July 2, 2004

Abstract

Peer-to-Peer (P2P) systems are very large computer networks, where peers collaborate to provide a common service. Providing large-scale Information Retrieval (IR), e.g. for searching the Word Wide Web, is an appealing application for P2P systems. The research community has presented several proposal for P2P-IR. However, so far the concepts of P2P and of IR have been intermingled. In this paper, we propose an architecture to structure P2P-IR systems. We differentiate between concepts belonging to the construction and maintenance of a P2P overlay network, and those belonging to IR. Furthermore, we distinguish basic P2P-IR concepts, which are likely to be needed in all P2P-IR systems, and advanced P2P-IR concepts, that rather depend on the flavor of the system. This decomposition of the P2P retrieval process is an important step towards a structured implementation of such systems. Furthermore, it allows a systematic sharing of methods and resources needed to perform retrieval. The next generation of global information retrieval systems will combine these distributed resources in new ways to provide more efficient web search.

Keywords: Peer-to-Peer Information Retrieval (P2P-IR), architecture, key-based routing (KBR), P2P web search

1 Introduction

Peer-to-Peer (P2P) systems are decentralized, large-scale computer networks, where peers operate as clients and servers at the same time. Peers can join and leave the system at any time. The power of P2P systems lies in their capability to provide services with practically unlimited scalability based on the principle of resource sharing. In the context of *information retrieval (IR)* the principle of sharing also applies to the knowledge on document collections and retrieval models. Existing P2P systems form already very large networks with hundreds of thousands or even million of computers participating. Combining the resources of a very large number of peers, we can expect a qualitative shift in information retrieval systems of the future.

The World Wide Web is growing at such a pace that even the biggest centralized search engines are able to index only a small part of the available documents. Federated, decentralized search engines, where the effort is shared among a very large number of computers, seem to have the potential of building IR systems with almost unlimited capacity.

The research community has presented several approaches to perform P2P-IR. The proposed systems can be categorized into IR over *unstructured* P2P systems [4, 6] and IR over *structured* P2P systems [10, 11]. These approaches

focus, however, mostly on the use of P2P overlay networks for distributed indexing of document collections.

While planning the implementation of a P2P-IR system, we noticed that in the proposed solutions, different concepts are not clearly separated into levels of abstraction. We therefore started to develop a generic architecture for a P2P-IR system. In this architecture we propose to decompose the IR process in a P2P environment into four different layers: 1) Transport Layer Communications, 2) Structured Overlay Networks, 3) Document and Content Management, and 4) Retrieval Models. In this way, we separate different concerns and allow different solutions at the higher layers to take advantage of the same infrastructure provided at the lower layers. This separation of concepts increases modularity of design and thus reusability of components offering basic services. Furthermore, it provides a step towards making different P2P-IR solutions interoperable, as it is unlikely that a single approach will prevail.

In this paper we give an overview of the key concepts of our architecture. From this concepts we derive prototypical interfaces between the different architectural layers. We concentrate particularly on layer 3 (document and content management), which plays an important role with respect to performance and flexibility when implementing a specific retrieval model in a P2P environment. We also identify *key-based routing* (KBR) of (structured) P2P overlay networks as the key contribution of P2P systems to support P2P-IR efficiently, and clarify how P2P-IR solutions can take advantage of this functionality. Last but not least, as a proof-of-concept, we discuss a case study based on [10], a P2P-IR proposal from outside our group, to demonstrate that our architecture has the potential to accommodate a wide variety of systems. Our architecture will be the basis to explore a wide range of potential solutions for efficient and scalable P2P-IR. We explicitly turn our attention to *text-based retrieval* in this paper. We do not explicitly address, for example, link-based ranking. However, after only slight extensions, it will be also possible to fit such advanced retrieval concepts in our model.

This paper is structured as follows: Section 2 gives an overview of our P2P-IR architecture. In Section 3, we provide the detailed concepts and interfaces between the layers. Section 4 describes how document indexing and query processing is carried out within the architecture. Section 5 introduces the case study. We finish with our conclusions and future work in section 6.

2 Overview of the P2P-IR Architecture

In this section we introduce the proposed P2P-IR architecture. Figure 1 shows the architecture, which consists of four layers. Each layer can be conceived as a space of objects with a specific topology (e.g. a similarity measure or distance function), and with certain operations to access and manipulate these objects. Each layer primarily uses operations of the layer underneath it to implement its operations. Objects at higher layers are related to (one or more) objects at lower layers. The problem of P2P-IR consists essentially in locating document objects (on layer 4) that are semantically close to a given document (respectively query) at peers (on layer 1) that store these documents.

On layer 4, the objects are *documents*. The ranking functions of specific IR models define the semantic distance between two documents. The algorithms of these ranking functions are typically based on computations performed on *keys* (short for keyword sets), which are extracted from the documents. *Keys* are the objects on layer 3 and serve as the building blocks for semantic distance on layer 4.

In a P2P system a large number of peers shares resources (and the execution of tasks on them). The key idea of structured P2P overlay networks is to partition resources. We achieve such a partitioning by associating peers and resources with identifiers from the same (application-specific) space. The basic function of a structured overlay network is to efficiently route resource

requests, which any peer can submit, to responsible peers in the network. In the literature this service is referred to as *key-based routing* (KBR)[1] [2]. By mapping *keys* of layer 3 to *identifiers* of layer 2, we associate certain document management tasks to specific peers. In this paper, we concentrate on structured overlay networks for layer 2. An adaptation to unstructured overlays is part of future work.

The purpose of layer 2 is to provide a logical network, which is independent of the inherent dynamics (e.g. because of network failures, dynamic IP, or mobility of peers) of the physical network (layer 1). The reason for choosing different spaces for *identifiers* on layer 2 and *keys* on layer 3 is that methods for managing a structured overlay network impose certain properties on the identifier space that are not necessarily satisfied by a key space, which has to fulfill IR requirements. Nevertheless, mappings with distance-preserving properties are an important tool for optimizations in P2P-IR. Similarly, distance-preserving mappings from *identifiers* on layer 2 to *physical addresses* on layer 1 support optimizations of the physical access to peers.

In the following section, we will introduce the four different layers in more detail, with a particular emphasis on layers 2 and 3 and their relationship.

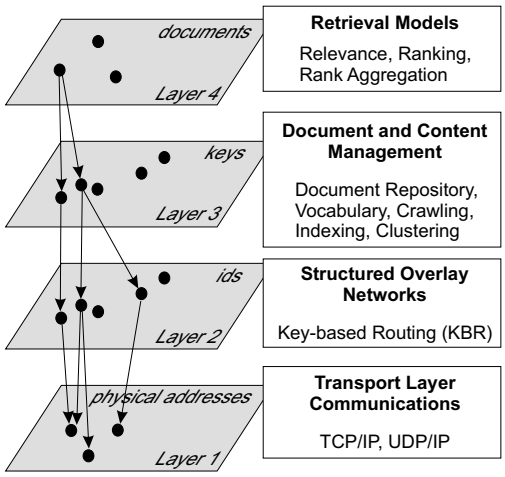

Figure 1: Overview of the peer-to-peer search engine architecture.

3 Detailed P2P-IR Architecture

3.1 Layer 1: Transport Layer Communications

Layer 1 handles the communication between two peers. We assume that two applications communicate via the UDP/IP and TCP/IP protocols, which are message-based and asynchronous. A peer runs an application, which is a program that can be accessed via the network by a, possibly temporary, IP address and port number. Thus, IP address and port number serve as layer-1 identifier for an application running on a peer. As this identifier can change (many computers have dynamic IP addresses), layer 2 has to keep track of the mapping between layer-2 identifiers and layer-1 *physical addresses*.

An application can exchange messages with any other application in the network. Current implementations of P2P systems typically use proprietary protocols based directly on UDP and TCP. UDP packets are neither acknowledged nor automatically retransmitted in case of loss. To send data via TCP,

[1]Note that we use the term *identifier* instead of *key* on layer 2, as we already use the term *key* on layer 3.

an application first opens a connection with the destination using a three-way handshake protocol. TCP packets are acknowledged and lost packets are automatically retransmitted. A two-message protocol is used to close a TCP connection.

Layer 1 provides TCP and UDP sockets as interface to the upper layers.

3.2 Layer 2: Structured Overlay Networks

Layer 2 deals with the P2P programming concepts we rely on. Structured P2P systems, such as CAN [7], Chord [9], Pastry [8], and P-Grid [1], provide an abstraction for overlay networks for resource location called *key-based routing* (KBR), in our context more suitably called *identifier-based routing*. Each peer is responsible for a certain range of identifiers. The cost of routing a message to a peer responsible for a given identifier scales gracefully with the size of the network and is usually $O(Log(N_P))$, where N_P is the number of peers in the system. The implementation of structured overlay networks relies on the transport layer communication mechanisms.

It is important to observe that a service is provided by a *group of undistinguishable peers*, i.e. the service can be requested from each peer of the group in a uniform way.

In the following, we first introduce the *abstract concepts* that underlie identifier-based routing. From these concepts we then derive a prototypical interface, which layer 2 provides to upper layers for the implementation of higher-level services. We present these interfaces in an object-oriented style (e.g. in the spirit of IDL, JAVA, or UML notations) in figure 3.2. Note that the interfaces are prototypical: the specification of such interfaces would be typically part of a standardization process. Furthermore, we omit details of data and function specifications as well as possible refinements and extensions. Our main intention is to outline the principles of our architecture proposal, i.e., we concentrate on the essential core concepts. Also note that the structures we introduce, such as the set of peers, are in general dynamic.

Virtual Identifiers: We use a large space of virtual identifiers Id to address peers and data in the system. All peers participating in a structured overlay network agree on a *common identifier space*. The identifier space has a *distance function* to exploit proximity for efficient routing.

Peers: A structured overlay network results from the cooperation of a set of peers. Each peer has a unique associated identifier $id_x \in Id$ and is, depending on id_x, responsible for a defined subset of identifiers in Id. Peers connect to each other using layer-1 primitives and construct a layer-2 network to implement (efficient) routing protocols.

Network Maintenance: A peer can join and leave the group of peers forming the overlay network. Maintenance ensures the integrity of the structured overlay network under such changes. It also performs load balancing. To join a group of peers, a new peer has to be able to communicate with a least one peer in the group. The joining peer therefore uses an outside bootstrap mechanism to obtain the physical address of a peer in the group. Cooperative peers may announce when they leave the network to support maintenance.

Routing: Given an identifier $id_i \in Id$, a peer can route a message with arbitrary *payload* to a peer responsible for the identifier (method *route*()). The interpretation of the payload is left to applications on higher layers. More refined versions of the routing method might return the physical address of the target peer to allow the implementation of complex protocols directly with this peer. To route a message with identifier id_i, each peer maintains a *routing table*, which contains links to peers that are closer to a peer responsible for

```
Class Identifier
    equal(id: Identifier): Boolean;
    distance(id: Identifier): Real;

Class Peer
    identifier(): Identifier;
    address(): IPAddress;
    neighbors(): {Peer};
    join(addr: IPAddress);
    leave();
    route(id: Identifier, payload: ByteString);
    broadcast(payload: ByteString);
    history(id: Identifier, query: HistoryQuery): HistoryData;
```

Figure 2: Interfaces supported by peers on layer 2.

id_i. The construction and maintenance of these routing tables, as well as the routing strategy itself, depend on the implementation of the structured overlay network and are hidden from upper layers. In a network of N_P peers, the average cost of routing a message to a responsible peer is usually $O(Log(N_P))$ overlay hops.

Broadcasting: This method efficiently broadcasts a message to all peers in the structured overlay network. Such a broadcast is typically possible with $O(N_P)$ messages and with a delay of $O(Log(N_P))$. [3] present, for example, such an algorithm for standard structured overlay networks.

History: When routing messages, each peer maintains a history of the identifiers and payloads of messages it has processed. Peers can query the histories of other peers. A peer p_i can, for example, query another peer p_j about how many identifiers id_i peer p_j has forwarded during a certain period of time. Among other things, such a history is important to analyze the load distribution in the network or to allow higher layers to generate usage patterns on document collections. The specific format of history data and the history query language is subject of a refined design.

The services we described in this subsection are implemented by peers, which communicate through layer-1 primitives. The protocol to build and maintain the overlay and the message types that are exchanged between peers depend on the implementation of the specific overlay, e.g. CAN, Chord, Pastry, or P-Grid [7, 9, 8, 1].

3.3 Layer 3: Document and Content Management

Independent of which semantic models are used for retrieval, certain tasks of document management are *common to all models* and will serve as building blocks for the definition of retrieval models. Such common tasks are, for example, maintaining a distributed document repository, associating term sets with documents, maintaining vocabularies, and cluster hierarchies. We therefore separately identify these tasks through a set of generic primitives that view documents as structured data objects. The specific choice of these primitives has been driven by an analysis of the needs of common retrieval models and by the principle of providing a simple, yet sufficiently powerful model to satisfy the needs of a wide range of models. In particular, we identify *shareable resources* and address issues of *distribution*. The distributed, and thus scalable, implementation of such functions relies on the primitives provided through structured overlay networks on layer 2. From these primitives we derive interfaces, which layer 3 provides to upper layers (figure 3.3.1).

3.3.1 Concepts Supported on Layer 3

Most of the concepts we introduce on layer 3 correspond to standard concepts from text-based information retrieval.

One not so obvious decision was to identify *clusters* as the main abstraction for document management. The reason is that documents in very large collections are often clustered to allow access on higher levels of granularity, e.g. for indexing or query answering. Furthermore, we separate cluster hierarchies from vocabularies because retrieval models frequently apply different document indexing methods to the same hierarchically organized document collection.

From a distributed data management perspective we observed an important requirement for distributed data access: To differentiate between query and data shipping. We also point out some limitations we deliberately introduced, such as the restriction of keys to term sets, which is a frequent approach, however, not a universal principle in text retrieval.

Document Collection: The P2P network provides the capability to store and retrieve documents based on unique identifiers associated with documents. A document is a string, which is not further interpreted, and represents the smallest addressing granularity.

Clusters: A cluster is a set of documents with a unique identifier. A document can be member of multiple clusters. They are the basic unit of document processing and can be organized into hierarchical structures. Clusters are a key mechanism for document providers and document analysis tools, because they make the semantic interpretation of the document collection explicit.

Vocabularies: Vocabularies establish a context for the interpretation of the document collection (or parts of it) by means of derived statistics and ranking functions. Each vocabulary is related to a cluster respectively a cluster hierarchy. In our setting we will use *keys* as the basic feature to represent documents for retrieval. A key is a set of *terms*. Terms are strings (e.g. words, phrases), typically extracted from the document content. In addition to features (i.e. keys), a vocabulary also contains (global) statistical information, such as inverse document frequencies of keys, which is necessary for document interpretation.

Advanced retrieval techniques can put different vocabularies into relation with each other, e.g. with linear transformations. An example is *latent semantic indexing* (LSI). We therefore introduce generic transformation functions to produce *derived vocabularies*. A retrieval model can then first perform searches for suitable vocabularies and document sets before searching for documents directly.

Posting Lists: A posting list manages the association of a key with a set of clusters and statistics of the key (e.g. the frequency of the key in a cluster). We call this information the *cluster digest*. As keys, posting lists are always unambiguously associated with a defined vocabulary.

Posting lists provide key-based retrieval of documents, which is an elementary function in IR. An important aspect in a *distributed environment* is the possibility to send a key-based query together with a filter, which we call *digest query*. Such a digest query can already filter the cluster digests of a posting list in place. Possible criteria passed with digest queries are limitations on the result size or containment in a specific (sub-) cluster. Digest queries thus allow to choose between different *query and data shipping* strategies.

Queries: A query consists of a set of keys from a specific vocabulary. A query retrieves all posting lists that are associated with its query keys. In

addition, vocabularies maintain histories of queries, which can be exploited by retrieval models, e.g. for usage-based ranking or keyword selection (as suggested in [5]).

Document Management: Document management allows to add and delete documents from the document collection. A document is always accessible by its unique identifier.

Document Indexing: An application can join documents from the collection into any cluster using a method *associate*(). This association method indexes the document in the cluster, i.e. it generates a document representation, which consists of a document digest and keys for indexing. The document is indexed for all vocabularies that are associated with the cluster hierarchy of the cluster. In addition, the corresponding vocabularies have to update the statistical information they maintain. Note that a vocabulary is potentially replicated over many peers. Such an indexing operation may therefore affect a number of different peers. *Efficient maintenance strategies* for vocabularies will thus be a key element of successful implementations of P2P-IR systems.

3.3.2 Support Functions of Layer 3

Using a structured overlay network, peers provide support functions to carry out the services of layer 3. These support functions provide mappings of keys and document identifiers to layer-2 identifiers. Furthermore, they support local data management to execute requests on posting lists, vocabularies, cluster hierarchies, and document set. Layer 2 also needs information about processing and storage cost to efficiently manage data and to perform load balancing between different peers.

Mapping of Keys: This function maps a layer-3 key $k_i \in K$ onto a layer-2 identifier $id_i \in Id$. It typically has distance preserving properties, such that close keys are mapped to close (or the same) peers. Higher layer functions use this mapping to access posting list based on keys. A peer maps a key onto an identifier and then uses layer-2 function *route*() to send a request to a responsible peer.

Mapping of Document Identifiers: This function maps a layer-3 document identifier $id_{d_i} \in Id_D$ onto a layer-2 identifier $id_i \in Id$. Here, distance preservation typically does not play a role as document identifiers do not relate to the semantics of documents.

Local data management: Each peers maintains a local database to manage documents, clusters, and vocabularies. When a peer on layer 2 receives (from other peers) messages with insertion and retrieval requests as payload and it is responsible for the identifier included in the messages, it dispatches them to the local data management functions supporting layer 3. These functions correspond to the interfaces provided in the specification of layer 3.

Maintenance: The dynamics of the system on layer 3 is influenced by changes in the peer population and document collection. The responsibility of document management tasks may migrate in case of changing peer population to further ensure load balancing and failure resilience. To support the migration of responsibilities, peers provide special functions: Layer 3 provides, for example, information about the cost of management tasks, such as the cost of maintaining vocabularies, clusters, or posting lists.

Another aspect of maintenance is providing consistency of vocabularies, which is handled with document insertion and deletion.

```
Class Collection
    insert(doc: Document, id: DocId);
    retrieve(id: DocId): Document;

Class Document
    identifier(): DocId;
    equal(id: DocId): Boolean;
    content(): String;
    delete();

Class Cluster
    identifier(): ClustId;
    root(): Boolean;
    createChildCluster(id: ClustId);
    children(): {Cluster};
    containedIn(id: ClustId): Boolean;
    associate(doc: DocId);
    documents(): {Document};

Class Vocabulary subclass-of Document
    keys(): {Key};
    frequency(k: Key): Real;
        (* may be extended to provide
        more statistical information *)
    cluster(): ClustId;
        (* root of a cluster hierarchy the
        vocabulary is associated with *)
    updateStatistics(key: Key, id: ClustID, d: ClustDigest);
    digest(id: ClustId): ClustDigest;
        (* this method provides the document representation
        as generated for the needs of a specific retrieval model *)
    map(t: Transformation): Vocabulary;
        (* generation of derived vocabularies *)

Class PostingList
    retrieve(k: Key, q: DigestQuery): {ClustDigest};
    insertDigest(key: Key, id: ClustID, d: ClustDigest);

Class ClustDigest
    (* data structure together with query language
    to represent indexing information *)

Class Key
    terms(): Terms;
    distance(key: Key): Real;
    vocabulary(): Vocabulary;
```

Figure 3: Interfaces supported by peers on layer 3.

3.4 Layer 4: Retrieval Models

Different retrieval models use the same document and content management functions of layer 3. Thus, it is important to have a common framework, in which we can capture the model we would like to use. Such a framework characterizes notions like ranking, relevance, rank aggregation, etc. and should be probabilistic (in some sense).

Basic concepts on layer 4 are the representation of user queries, the provision of ranking and clustering functions, and interfaces for basic information retrieval tasks. Layer 4 provides functions for constructing vocabularies and document indexing, such as extracting keys from documents. The implementation of these functions relies on the services provided by layer 3. Due to space limitations we omit a detailed specification of this layer.

4 Implementation

In this section we sketch how typical retrieval tasks, namely document indexing and document retrieval, are implemented using our layered architecture.

4.1 Indexing a new Document

The following steps are taken upon insertion of a new document. Even if a peer holds documents that are only accessible to authenticated users, it can still index these documents to make them searchable.

1. A peer p on layer 4 decides to include a document into a search engine using a specific retrieval model. It uses a function to generate the document digest, i.e. the keys and associated statistics representing the document according to the retrieval model. p also provides the cluster(s) in which the document should be included, respectively decides to create a new cluster for the document. It generates a unique key for the document.
2. For implementing the layer-4 operations, p retrieves through layer 3 the cluster hierarchy and vocabulary that the retrieval model is based on. These queries are routed on layer 2 as payload to the corresponding peers and require the layer-3 support functions for converting document and cluster identifiers into layer-2 identifiers. The receiving peers interpret the payload using their local document management functions and return the results (either directly or by routing).
3. The document is now included on layer 3 into the document collection and associated with the corresponding cluster(s). Next the new or updated cluster digest is inserted into the posting list.
4. For inserting a new cluster digest into a posting list, layer-2 routing is used and the keys are converted into layer-2 identifiers.
5. The insertion of new documents also triggers updates on the statistics of the vocabulary. Depending on the usage patterns, vocabularies are possibly widely replicated. Therefore broadcast functions may be used to perform these updates efficiently.

4.2 Retrieval Process with Ranking

The following steps are performed when retrieving a document:

1. On layer 4 a peer p receives a query for the retrieval of documents according to a specific retrieval model.
2. The peer p needs access to the relevant vocabulary to obtain global information on distributional properties of keys. Therefore p performs a search for a peer that stores such a vocabulary.

3. p generates a set of query keys from the application query.
4. The set of query keys is used to retrieve the posting lists using the layer-3 function *retrieve*(), which in turn is executed by using layer-2 routing functionality.
5. After having retrieved the posting lists, which are possibly pre-filtered, p (on layer 4) computes the ranked result according to the specific retrieval model.
6. For producing a more compact or better structured representation of the result, p might exploit the structure of the cluster hierarchy that is associated with the retrieval model.

5 Case Study

We are currently exploring a number of retrieval models that are particularly suited for use in a P2P environment. However, for the verification of the general applicability of the proposed architecture we prefer to check it by matching other approaches from the literature into our architecture as well. In the following we provide, for illustration purposes, one example of how a concrete system proposed independently by C. Tang and S. Dwarkadas [10] can be mapped on the framework we have introduced.

Layer 4 The vector space model (VSM) is used for mapping documents onto keys. Therefore, all terms in a document are weighed using a special algorithm. This algorithm relies on global statistics that are aggregated and maintained by layer 3. Furthermore, the authors use a stemmer and exclude stop words. This is represented in our model by the use of a retrieval-model specific document indexing method.

Layer 3 The authors use keys, which are terms or phrases of terms. Document identifiers and posting lists, as well as their insertion and retrieval, are implicitly introduced. Posting lists also contain document digests instead of only document references. A document digest consists of a complete list of terms in the document. Tang and Dwarkadas also use digest queries to select which document references of the posting lists are returned.

Layer 1 and 2 The authors use Chord [9] as underlying Distributed Hash Table (DHT). Chord builds and maintains a structured overlay network and provides key-based routing (KBR). It can be extended to also provide broadcast and history functionality on layer 2.

Thus, in summary, we have no difficulties in mapping all concepts introduced in this specific approach into our architectural framework.

6 Conclusions and Future Work

In this paper, we have introduced a framework for information retrieval in peer-to-peer systems. With this first proposal of a generic architecture for P2P-IR, we envision to encourage modular design of P2P-IR systems, enabling resource sharing at different levels of abstractions and thus increasing the level of sharing of knowledge and resources in global information retrieval.

An important study we currently perform is the design of retrieval models that allow an efficient implementation on a P2P architecture. In principal, the design of the retrieval model should be logically independent of the infrastructure it is implemented in. However, we believe that in practice infrastructure constraints cannot be ignored.

7 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**).

References

[1] K. Aberer. *P-Grid: A Self-Organizing Access Structure for P2P Information Systems.* Sixth Inter-national Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy 2001

[2] F. Dabek, B. Zhao, P. Druschel, and I. Stoica: *Towards a common API for structured Peer-to-Peer overlays.* In 2nd International Workshop on Peer-to-Peer Systems (IPTPS'02), Feb. 2003

[3] S. El-Ansary, L. O. Alima, P. Brand, S. Haridi: *Efficient Broadcast in Structured P2P Networks.* 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA, 2003

[4] A. Löser, W. Nejdl, M. Wolpers, and W. Siberski: *Information integration in schema-based peer-to-peer networks.* In Proceedings of the 15th Conference On Advanced Information Systems Engineering (CAISE 03) (Klagenfurt/Velden, Austria, June 2003).

[5] F. Klemm, A. Datta, K. Aberer: *A Query-Adaptive Partial Distributed Hash Table for Peer-to-Peer Systems.* International Workshop on Peer-to-Peer Computing & DataBases (P2P&DB 2004), Crete, Greece, March 2004

[6] 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 (CIKM'03). New Orleans, 2003

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *A scalable content-addressable network.* In SIGCOMM, Aug. 2001.

[8] 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), November 2001.

[9] 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.

[10] C. Tang, S. Dwarkadas: *Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval.* First Symposium on Networked Systems Design and Implementation (NSDI'04), San Francisco, March 2004.

[11] C. Tang, Z. Xu, S. Dwarkadas: *Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks.* SIGCOMM 2003