The Robustness of Content-Based Search in Hierarchical Peer to Peer Networks

M. Elena Renda*
I.S.T.I. – C.N.R.
and
Scuola Superiore Sant'Anna
I-56100 Pisa, Italy
elena.renda@isti.cnr.it

Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, US
callan@cs.cmu.edu

ABSTRACT

Hierarchical *Peer to Peer* (*P2P*) networks with multiple directory services have quickly become one of the dominant architectures for large-scale file sharing due to their effectiveness and efficiency. Recent research argues that such networks are also an effective method of providing large-scale content-based federated search of text-based digital libraries. In both cases the directory services are critical resources that are subject to attack or failure, but the latter architecture may be particularly vulnerable because content is less likely to be replicated throughout such networks.
This paper studies the robustness, effectiveness and efficiency of content-based federated search in hierarchical *P2P* networks when directory services fail unexpectedly. Several recovery methods are studied using simulations with varying failure rates. The experimental results demonstrate that quality of service and efficiency degrade gracefully as the number of directory service failures increases.

1. INTRODUCTION

Peer-to-peer (*P2P*) computing is a paradigm for sharing resources, such as data, storage, and bandwidth, through direct exchange, where each participant ("*node*" or "*peer*") acts as both provider and consumer, and performs tasks normally associated with clients, routers and servers. A completely decentralized *P2P* network is called "*flat*" or "*pure*" (see Figure 1).

In pure *P2P* networks, one of the simplest and most common mechanisms for locating content is *flooding*, which is undirected propagation of a search request to neighboring peers until a search horizon is reached. Flooding, adopted for example in Gnutella 0.4 [5], is simple and easy to implement, but does not scale because of the huge number of messages generated for each query. Further, this search mechanism may fail to find content because it extends to only a limited search horizon.

Recent research has focused on developing different type of *P2P* architectures, and on how to route search requests more efficiently and effectively. The two major approaches that have emerged so far are highly-organized networks (*e.g.*, Tapestry [7], OceanStore [11], Pond [18], Pastry [19], Chord [22]), in which content location can be inferred easily due to a well-defined data placement policies, and hierarchical *P2P* networks with directory services. The latter approach has became one of the most popular architecture for widely-deployed *P2P* applications (e.g., Gnutella 0.6 [5], KaZaA [9], Morpheus [13], Napster [14], Project JXTA [15]).

*This work was done when the author was visiting the Language Technologies Institute at the School of Computer Science of Carnegie Mellon University, Pittsburgh, PA, US.

In *Hierarchical P2P* networks there are two types of nodes: *Leaf Peers* and *Ultra Peers*. A *Leaf Peer* can be a content provider ("digital library") and an information seeker (a source of queries). *Ultra Peers* provide directory services, such as resource description, resource selection, and message routing, to other peers. *Leaf Peers* can submit queries only through their *Ultra Peer* neighbors. In decentralized networks there may be multiple *Ultra Peers*, each of which is connected to some subset of the *Leaf Peers* and some subset of the other *Ultra Peers*; the set of all *Ultra Peers* cover the network, forming a "*Pure Ultra Peer*" network (see Figure 2).

P2P networks are a very dynamic world, where peers can leave and join the network at any time (*peer autonomy*). A *P2P* search protocol must deal with this dynamic environment, performing well under any condition. A *robust search protocol* has to guarantee good performance even when the operating conditions are different from the initial ones. For instance, in presence of failures a robust protocol shows a graceful degradation of the performance. A *scalable search protocol* has to guarantee good performance even when the number of peers that participate in the network grows.

For these reasons, the search mechanism must simultaneously:

- be effective in terms of the results returned (quality, quantity and response time);
- maintain a low level of costs (e.g., bandwidth, number of messages through the network, resources);
- allow the system to scale without problem; and
- maintain a high level of robustness in the presence of dynamic conditions and/or failures.

Given a *P2P* network topology and data placement (*i.e.*, how the resources are distributed across the network), we can classify the *P2P* search mechanisms as follows:

1. ***centralized search*:** in this case the *P2P* network is hierarchical, with a unique *Ultra Peer* connected to all the other peers. This *Ultra Peer* acts as a router and manages all the *Leaf Peer* requests. This approach, used, *e.g.*, in Napster [14], is fast and efficient, but not very robust, because of the unique point of failure represented by the central *Ultra Peer* node. For the same reason, it is hardly scalable;
2. ***decentralized search*:**
   - ***without directories*:** in this case the *P2P* network is pure, and all the nodes play the same set of roles, being client when accessing information, router when forwarding information, and server when providing information. Even if this solution is robust in presence of

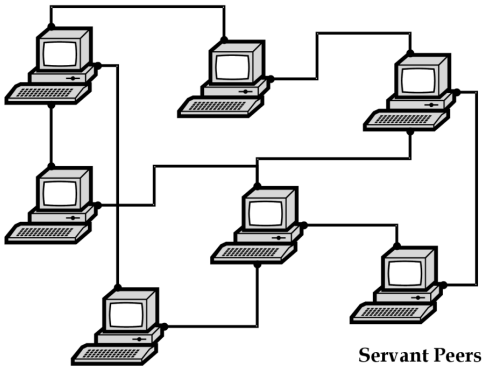

Figure 1: *Pure P2P* Network.

failures, it is not scalable, due to the huge message overhead generated to handle each query. We call this ***Pure P2P Search***;

– ***with multiple directories*:** in this case we have a hierarchical *P2P* network with multiple *Ultra Peer* nodes (also called *Super Peers* or *Hubs*). Only the *Leaf Peers* have content to share and can submit requests through the *Ultra Peers*, while the *Ultra Peers* simply act as request routers. The *Ultra Peers* provide directory services only for defined regions of the network. Hierarchical *P2P* architectures with multiple directories using decentralized search seem to be more robust and scalable than the other approaches. In fact, each *Ultra Peer* provides centralized directory services to a given subset (region) of *Leaf Peers*, and the set of *Ultra Peers* cover the whole network of *Leaf Peers*, thus obtaining a balance between the advantages of the centralized and decentralized approaches: The efficiency of the former combined with the autonomy and the robustness of the latter. In the following, we refer this type of search as ***Hierarchical P2P Search***.

Hierarchical P2P architectures with multiple directory services have become one of the dominant forms of *P2P* network due to their effectiveness and efficiency. Their widespread use in large music file-sharing networks demonstrates that they can also be robust ([9, 13]).

Recent research argues that hierarchical *P2P* networks are also an effective way of providing large-scale content-based federated search of text-based digital libraries [12]. However, in this type of *P2P* search the contents of digital libraries are more likely to be disjoint and content is less likely to be replicated throughout the network, which may make the network more sensitive to disruptions in directory services. Music file-sharing networks are robust due in part to large-scale replication of content throughout the network. When content is not replicated, directory services become critical resources that are subject to attack or failure. The research reported in this paper addresses this issue.

The structure of the paper is the following: Section 2 presents some related work; Section 3 introduces the motivation of the present work; Section 4 describes the search mechanisms; Section 5 presents the behaviour of the search protocol under dynamic conditions; Section 6 describes the data and methodology adopted

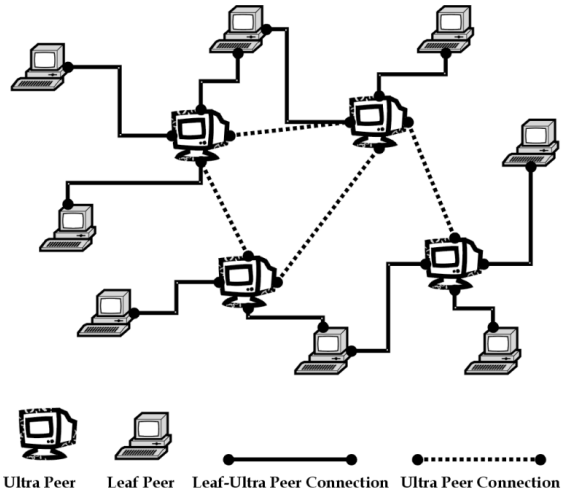

Figure 2: *Hierarchical P2P* Network.

for the experiments; Section 7 reports and discusses the results obtained; and Section 8 concludes.

2. PRIOR RESEARCH

P2P applications for file sharing (such as music, images, news articles, and web pages) have became widespread recently, fostering a lot of work focused on this research area, and in particular on how to improve *P2P* search mechanisms.

Centralized search, adopted first by Napster [14], was shown to be fast, but not very robust because of the central and unique point of failure, and expensive to scale. For this reason, many systems have adopted decentralized search, using different approaches.

Distributed Hash Table (*DHT*) architectures have been proposed to address scalability. Many systems rely on the *DHT* functionality to efficiently find data in *P2P* networks in a small number of hops (e.g., Tapestry [7], OceanStore [11], CAN [16], Chord [22]). In *DHT* Systems, a *key identifier* is associated to each document (produced, *e.g.*, by hashing the document name) and a *node identifier* is associated to each peer (*e.g.*, the IP address). A certain range of keys is assigned to each node in the system (*mapping*). The $lookup(key)$ method returns the identifier of the node responsible for that key. When looking for a document, the $lookup(key)$ message is routed through the overlay network until the node is found. The difference between the various *DHT*-based systems is in the method they use to map key IDs and node IDs.

DHT-based systems are scalable and obtain good search results with limited costs. Although all of the systems proposed in literature are completely decentralized, these *DHT*-based search protocols may be applied both to pure and hierarchical *P2P* networks. However, a drawback of this approach is that they define a very rigid overlay structure on the network, thus limiting peer autonomy. Further, their performance, especially in terms of quality of results returned, in presence of the dynamic *P2P* conditions (such as peers leaving and joining the network) is still unknown [17].

Pure *P2P* Search

Gnutella 0.4 (a protocol developed for *Pure P2P* networks [5]) adopts unstructured search to find content: a peer sends a query

to all its neighbors (*flooding*), and the request is forwarded through the network until it has traveled a certain radius (TTL, *Time to Live*). Even if this solution is really simple and robust, it is not scalable because of the considerable message overhead.

The *Interest-based locality* approach proposed in [21] recalls a collaborative filtering idea: if peer A is looking for the document d and peer B owns it, A considers B a "good" neighbor (creating an *interest shortcut* to it) because it will probably have other documents A is also interested in. The shortcuts provide a loose structure on top of the unstructured overlay of Gnutella 0.4. To avoid flooding, the protocol first looks for content through shortcuts and only when all the shortcuts fail, it uses the Gnutella 0.4 overlay to flood the query. Shortcuts seem to have good load distribution properties, *i.e.* peers that use more the system are more loaded, and they do help to reduce the query scope, *i.e.* the number of peers touched by each query. Furthermore, incurring very little overhead, shortcut-based search mechanism can be considered scalable. However, the results reported in [21] are based on relatively short simulations (one hour long), where the system is always assumed to be running in optimal conditions.

P-Grid [1] is a self-organizing and completely decentralized *P2P* lookup system based on a virtual distributed search tree. It has some similarities with Chord [22], but it exploits local knowledge and does not require central coordination or global knowledge, as Chord does. The P-Grid's search has good characteristics of robustness and scalability, due to the replication distributed over all the peers.

Other approaches [8, 24] try to locate the content selecting the nodes to search based on some specific heuristics (past behaviour, past number of results, past query response time, etc.). These methods obtain good level of efficiency in searching a *P2P* network, but the heuristic-based selection of the nodes to which send the query may fail to find relevant documents.

Hierarchical *P2P* Search

Project JXTA [15] is an ad-hoc, multi-hop adaptive *P2P* network, where a sort of *Ultra Peer* nodes, called *relay peers*, maintain routing tables to relay messages to their destinations. However, peers always attempt direct communication with other peers before using the relay service. In fact, because of the unreliability of peers, in Project JXTA every message carries its own routing information in order to be self-sufficient and protect against relay peer failures.

Gnutella 0.6 (a protocol developed for hierarchical *P2P* networks with multiple directory nodes [5], used, *e.g.*, by KaZaA [9]) adopts the *name-based search* to find content: the *Ultra Peers* flood the query to all the *Ultra Peer* neighbors and select the *Leaf Peers* based on a simple matching between query terms and *Leaf Peer* content, obtained by hashing the name of all the documents each *Leaf Peer* owns.

The solution presented in [12] is based on some Distributed Information Retrieval techniques which are applied in a hierarchical *P2P* network with multiple directories. In particular, techniques of resource description and resource selection are used in order to perform content-based search, both at the *Ultra Peer* and at the *Leaf Peer* level. In the paper are reported several experimental results in which content-based search mechanisms are compared to different resource selection and document retrieval algorithms. The results reported demonstrate that in such networks, content-based search is more accurate and more efficient as compared to the simple name-based search mechanism. Furthermore, the use of resource selection, both at *Leaf Peer* and *Ultra Peer* level, seems to be a good way to improve the efficiency of query routing without degrading the quality of service (*QoS*). However, also in this paper the results reported are based on static simulations, where the system is always assumed to operate under normal conditions.

3. MOTIVATION

Prior work demonstrates that hierarchical *P2P* search is potentially more accurate and scalable than the pure and centralized ones. Furthermore, it has been shown that existing Distributed Information Retrieval techniques can be adapted to implement content-based retrieval in *P2P* systems, maintaining high level of retrieval performance and routing efficiency. However, each technique proposed so far focuses on a different aspect of the *P2P* content location problem (efficiency, or QoS, or ...), while these aspects are not independent [4] and should be evaluated at the same time.

In this work we are interested in developing *P2P* networks of distributed digital libraries containing text-like content, that do not reveal, or reveal only partial information about, their contents. In particular, we are interested in studying the robustness, effectiveness and efficiency of a hierarchical *P2P* network, where the digital libraries represent the information sources (*Leaf Peers*).

Hierarchical *P2P* networks such as KaZaA are effective in finding results even when directories fail. In fact, when the directory services are based on name-based retrieval, they are tiny and easy to replicate or rebuild. Hence, the robustness of these systems could be due to the massive file and/or directory service replication. For systems where the directory services are content-based, much larger and more complex, alternatives to replication and/or rebuilding are desirable.

Our research investigates the robustness issue by studying a network in which there is no replication, so any robustness must be due to the directory services. In such a network, the *Ultra Peers* act as service providers (for resource description, resource selection, etc.) for all the other peers; these directory services are a form of regional centralization that creates potentially critical failure points, but the sensitivity of *Ultra Peer* failure has received little attention. We do not consider failures at the *Leaf Peer* level because this type of failure affects the information available at a certain time on the network, thus having the same effects on all types of search.

4. SEARCH MECHANISMS

In a hierarchical *P2P* network with multiple directories (in the following, simply called hierarchical *P2P* network) there are two different types of search: the *resource selection* at the *Ultra Peer* level and the *document retrieval* at the *Leaf Peer* level. *Ultra Peers* do not own/share data, but provide regional centralized services for a portion of the *Leaf Peer* and the other *Ultra Peer* nodes. In order to do this, they have to maintain information about the content of their *Leaf Peer* and *Ultra Peer* neighbors. The peer content description is acquired by each *Ultra Peer* and used to route the queries only to a selected subset of *Leaf Peer* and *Ultra Peer* neighbors, *i.e.* those that are supposed to be "good peers" (*resource selection*). The *Leaf Peers* represent the information sources in the network (*e.g.*, digital libraries). The regions served by each *Ultra Peer* could be organized along many dimensions, e.g., geographic ("Italy"), temporal ("the 15^{th} century"), content ("medicine"), etc., and all of these could exist simultaneously in the network. Since a *Leaf Peer* could represent a heterogeneous information source, it could belong to different regions, so each *Leaf Peer* is allowed to be connected to more than one *Ultra Peer*. There is no connection/communication between *Leaf Peers*: they can submit requests and provide information about their data only through their *Ultra Peer* neighbors (see Figure 3).

In our model it is assumed that each *Ultra Peer* selects its *Ultra Peer* neighbors based on their acquired language model, while the

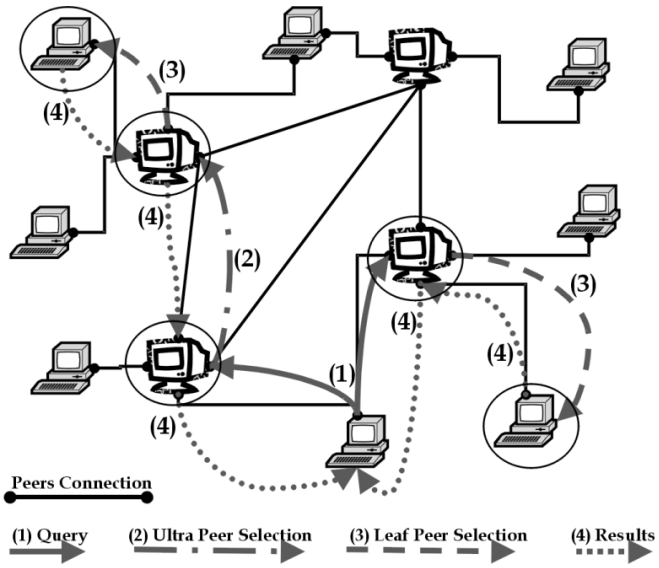

Figure 3: Search Mechanism: (1) Query Submission; (2) *Ultra Peer* Selection; (3) *Leaf Peer* Selection; (4) Results.

Leaf Peers and the documents are selected based on their content. There is no source selection at the *Leaf Peer* level: each *Leaf Peer* simply floods the query to all the *Ultra Peer* neighbors.

In the following, we first describe how an *Ultra Peer* selects the *Leaf Peer* neighbors to which it forwards the queries (Section 4.1), then we describe how an *Ultra Peer* selects the *Ultra Peer* neighbors (Section 4.2), finally we briefly describe the document retrieval at the *Leaf Peer* level (Section 4.3).

4.1 *Leaf Peer* Selection

The simplest and fastest way to search for a document over a *P2P* network is the *name-based* search mechanism, which is based on the simple matching between query terms and the name of the documents owned by a peer. This protocol works well for music filesharing systems such as Napster because a simple "title and/or artist" file-naming protocol (e.g., "Christina + Aguilera + Fighter") is relatively consistent, unambiguous, and easy to remember. However, a similar "title and/or author" file-naming protocol and search mechanism is considered restrictive when applied to textual content, because usually a person does not know the authors or titles of documents that contains desired content. The state-of-the-art for searching text-based digital libraries is content-based search methods (e.g., Web search engines).

Because of the inefficiency of name-based mechanism in searching a *P2P* network of digital libraries, we adopt a different approach for the *Leaf Peer* selection phase: the *Ultra Peers* acquire the *Leaf Peer* content, asking directly each *Leaf Peer* to provide it, *e.g.*, using the STARTS protocol [6]. We adopt two different approaches, in which each *Leaf Peer* can provide:

- its *vocabulary*, defined as the set of all unique terms present in the document collection the *Leaf Peer* owns. In this case the *Ultra Peers* simply forward the query to those *Leaf Peer* neighbors that satisfy the matching between query terms and their vocabulary (*match-based selection*);

- its *description*, defined as the set of all unique terms that appear in the document collection the *Leaf Peer* owns and their correspondent collection frequency. In this case the *Ultra Peers* measure how well each *Leaf Peer* description predicts a query with a Kullback-Leibler (*KL*) divergence-based algorithm, an information theoretic metric used to measure how well one probability distribution predicts another [10]. It has been demonstrate that the K-L method works well for collection selection [20, 23].

  Based on the score computed with the K-L algorithm, each *Ultra Peer* ranks the set of *Leaf Peer* neighbors and sends the query only to those up to a given threshold (*content-based selection*).

We assumed the match-based mechanism as the baseline of our experiments, to be compared with the performance of *content-based* selection. In fact, the match-based mechanism can be seen as an enhancement of the simple name-based one, used in almost all the common *P2P* file sharing systems.

4.2 *Ultra Peer* Selection

Since an *Ultra Peer* does not own data, the *Ultra Peer description* could be represented, for example, by the sum of all the descriptions of *Leaf Peers* it is connected to. However, an *Ultra Peer* can be connected to many *Leaf Peers*, and this means that the information to store about an *Ultra Peer* (term frequency and weight) can become unreliable. For this reason, when selecting the *Ultra Peers* to which forward the requests, each *Ultra Peer* uses a sort of query learning to acquire the *Ultra Peer* collection language model. So, instead of acquiring the whole *Ultra Peer* descriptions, each *Ultra Peer* simply stores the query terms of the queries the *Ultra Peer* neighbors have satisfied in the past. In this way, the size of the model representing each *Ultra Peer* description is limited.

4.3 Document Retrieval

When a *Leaf Peer* receives a query, it matches the query terms against the content of the documents in the collection it owns, producing a ranking of the documents based on the K-L divergence retrieval algorithm. Then the *Leaf Peer* sends a $QUERY\ HIT$ message in response to the *Ultra Peer* neighbor from which it received the $QUERY$ message. The $QUERY\ HIT$ message contains information about the documents that match the query terms.

Since the *Ultra Peer* selection part and the document retrieval part do not change, in the following we will distinguish each search mechanism based only on the *Leaf Peer* selection, *i.e.* content-based search and match-based search.

5. SEARCH PROTOCOL BEHAVIOUR UNDER DYNAMIC CONDITIONS

In a *P2P* network there can be different type of events:

1. the messages exchange among different peers, such as the request of information ($QUERY$) and the reply to a given request ($QUERYHIT$), the message used to actively discover new peers in the network ($PING$) and the response to this ($PONG$);
2. the file transfer among two peers;
3. the new connection between two existing peers in the network;
4. the joining of a new peer in the network ($ACTIVATION$);
5. the interruption of a file transfer among two peers;
6. the disconnection between two existing peers in the network;

7. the leaving, temporary or permanent, of an existing peer from the network ($DEACTIVATION$).

While the message exchanges and the file transfers can be seen as "normal" events in *P2P* networks, we can call all the others "dynamic" events; in particular, the last ones (file transfer interruption, peer disconnection and peer deactivation) are events that happen when the network, or a part of it, is not running under normal conditions. After introducing the *P2P* model and the search mechanisms used, we have now to describe how the different search mechanisms react in presence of dynamic events. As described in Section 3, in this paper we consider only *Ultra Peer* failures (*Ultra Peer Deactivations*) because they may represent a critical part in the hierarchical *P2P* network we are taking into account.

An *Ultra Peer* can disappear from the network at any time, and be re-activated later, if the failure is temporary. When an *Ultra Peer* is deactivated (temporarily or permanently), we have to take into account what happens to its neighbors (*Leaf Peers* and *Ultra Peers*). In fact, it could happen that some (or all) of its neighbors remain without *Ultra Peer* connections in the network.

If a *Leaf Peer* remains without *Ultra Peer* connections, it cannot submit requests through the network and its collection is invisible to the rest of the system. If an *Ultra Peer* remains without *Ultra Peer* connections, the network is partitioned and the *Leaf Peer* region(s) controlled by this *Ultra Peer* may be unreachable. To solve this problem these "orphan" peers can create new connections to the *Ultra Peer* neighbors of the failed node. In order to do this, each peer needs to know who are the *Ultra Peer* neighbors of the *Ultra Peers* it is connected to; these serve as "backup *Ultra Peers*" if the primary *Ultra Peer* should fail. We assume that *Ultra Peers* periodically notify the *Leaf Peers* they serve about the *Ultra Peers* that are currently adjacent and capable of serving as "backup *Ultra Peers*" should the primary *Ultra Peers* fail.

When a node previously deactivated joins the network again, it re-connects to the old neighbors (*Leaf Peers* and *Ultra Peers*).

When an *Ultra Peer* U gets new neighbors, it needs to update its resource selection models to include the new peers. If in the neighborhood of U there are new *Ultra Peers*, U starts to flood new queries to all the new *Ultra Peer* neighbors, learning, query after query, their content language models. If in the neighborhood of U there are new *Leaf Peers*, U can: (a) flood the query to all the new *Leaf Peer* neighbors (unstructured selection/routing); (b) acquire the vocabulary of the new *Leaf Peer* neighbors (match-based selection/routing); (c) acquire the description of the new *Leaf Peer* neighbors (content-based selection/routing); or, (d) learn the new *Leaf Peer* neighbors model from the queries they answer (query learning-based selection/routing).

Since we developed both match-based and content-based mechanisms for the selection of the new *Leaf Peers*, we decided to perform experiments with hierarchical *P2P* networks using the following search configurations:

- content-based search without deactivations (CB-BASELINE);
- match-based search without deactivations (MB-BASELINE);
- content-based search with *Ultra Peer* deactivations, applying the flooding to all the new *Leaf Peer* neighbors (CB+FLOOD);
- match-based search with *Ultra Peer* deactivations, applying the flooding to all the new *Leaf Peer* neighbors (MB+FLOOD);
- content-based search with *Ultra Peer* deactivations, applying the content-based selection to all the new *Leaf Peer* neighbors (CB+CB); and
- match-based search with *Ultra Peer* deactivations, applying the match-based selection to all the new *Leaf Peer* neighbors (MB+MB).

6. EXPERIMENTAL METHODOLOGY

In the following, we briefly describe the data (documents, queries, network topology), the evaluation methodologies, the experimental setup and the characteristics of the simulator used for our simulations.

DATA SET. To build a hierarchical *P2P* network, we need to define the *Leaf Peers* with their contents, the *Ultra Peers*, and the region each *Ultra Peer* serves.

Prior research used widely available $TREC$ data to define a *P2P* testbed consisting of $2,525$ nodes ($2,500$ *Leaf Peers* and 25 *Ultra Peers*), 1.4 million documents, and $50,000$ queries [12]. In the research reported here we used the same testbed because, to the best of our knowledge, it is one of the largest publicly available[1] *P2P* network definitions, and the only one we know of that is suitable for research on content-based retrieval in *P2P*networks.

In [2] it was shown that content-based clustering, based on interest similarity among peers, can significantly increase the efficiency and effectiveness of searching *P2P* networks. We decided to follow this strategy to obtain the regions covered by each *Ultra Peer*, clustering the *Leaf Peers* based on their content.

A subset of $15,000$ queries was randomly selected from the $50,000$ queries available, to reduce the running time of simulations.

EVALUATION METHODOLOGY. To measure the effectiveness and the QoS of each search mechanism implemented, we compute:

- the proportion of the retrieved documents actually relevant (*Precision*), given by:

$$Precision = \frac{\#\ RelRet}{\#\ TotRet} \tag{1}$$

- the proportion of the relevant documents actually retrieved (*Recall*), given by:

$$Recall = \frac{\#\ RelRet}{\#\ TotRel} \tag{2}$$

- and the retrieval accuracy given by the *F-Score*, defined as:

$$F - Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3}$$

We adopt the same measure of relevance used in [12]. The results report the average recall computed over all queries and the average precision computed over the queries with at lease one returned document.

To measure the efficiency of each search mechanism implemented we compute:

- the average number of messages exchanged for each query; and

[1] `http://hartford.lti.cs.cmu.edu/callan/Data/#P2P`

RANK	CB	MB	RANK	CB	MB	RANK	CB	MB
1	2500	2500	10	2521	2509	19	2511	2511
2	2507	2507	11	2519	2523	20	2518	2518
3	2513	2503	12	2523	2517	21	2510	2510
4	2517	2501	13	2514	2519	22	2512	2520
5	2503	2513	14	2522	2521	23	2520	2516
6	2508	2522	15	2505	2515	24	2516	2512
7	2509	2508	16	2504	2514	25	2506	2506
8	2524	2524	17	2502	2502			
9	2501	2505	18	2515	2504			

Table 1: The *Ultra Peer* IDs ranking based on the message load in Content-Based and Match-Based baseline experiments.

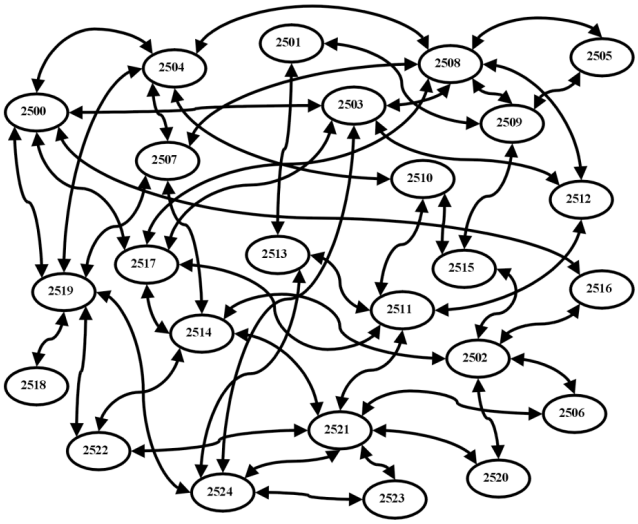

Figure 4: *Ultra Peer* connections.

- the average number of nodes reached by each query (*Query Scope*).

EXPERIMENTAL SETUP. In the *Leaf Peer* selection/routing phase (Section 4.1), to reduce the number of $QUERY$ messages through the network, an *Ultra Peer* can send the query to up to 1% of its *Leaf Peer* neighbors. Prior research demonstrated [12] that the value of 1% for *Leaf Peer* selection gives satisfactory retrieval performance in hierarchical *P2P* networks (even in terms of recall).

For the *Ultra Peer* selection phase (Section 4.2), an *Ultra Peer* learns the *Ultra Peer* neighbors description, storing the terms of the queries the *Ultra Peer* neighbors have satisfied in the past. To limit the information stored for each *Ultra Peer*, the size of the model is fixed at 750 terms. This value was chosen because, by preliminary simulations, we verified that increasing the *Ultra Peer* description size beyond this value does not improve the QoS. To allow the insertion of new terms, when the *Ultra Peer* language model reaches the limit size, the terms with lower frequency are removed.

Since each *Ultra Peer* has 4 *Ultra Peer* neighbors on average, we decided that an *Ultra Peer* can forward the query to up to 1 *Ultra Peer*, *i.e.*, at most the 25% of the *Ultra Peers* on average are touched by the forwarded query. This way, the number of messages is considerably reduced as compared to the simple flooding.

The TTL of each $QUERY$ message is set to 4 at the beginning, and decreased each time the query touches a peer. When an *Ultra Peer* sends a query to a *Leaf Peer*, the $QUERY$ message TTL is set to 1, because the *Leaf Peers* cannot forward the query to any peer. When a *Leaf Peer* sends a $QUERY\ HIT$ message to an *Ultra Peer* replying to a $QUERY$ message, it provides information at most on the top 50 ranked documents that predict the query.

Besides the CB-BASELINE and MB-BASELINE experiments (without *Ultra Peer* deactivations), for each search mechanism implemented (CB+FLOOD, MB+FLOOD, CB+CB, MB+MB), we performed a set of experiments increasing, for each simulation, the number of *Ultra Peers* deactivated.

For each N deactivations, $N \in \{1, 2, 3, 6, 9\}$, we report the average results on at least 3 different simulations, in which, respectively, the N *Ultra Peers* most loaded, the N *Ultra Peers* least loaded, and the N *Ultra Peers* with a medium load are deactivated. We defined the *Ultra Peer* "*message load*" as the number of messages passed through the node under normal network conditions (*i.e.*, without *Ultra Peer* deactivations). In Table 1 we report the *Ultra Peer* identifiers ordered by message load, both for Content-Based and Match-Based experiments. In Figure 4 we depict the connections at the *Ultra Peer* level we have in the network. Comparing Table 1 and Figure 4, we observe that an *Ultra Peer* load is strictly bounded to the number of *Ultra Peer* connections it has.

SIMULATOR. The experiments were conducted using a network simulator extended to simulate *P2P* networks based on Gnutella 0.6 protocol [5] and to support match-based and content-based search in hierarchical *P2P* networks. We modified the simulator to support all the dynamic events mentioned above, such us connections and disconnections between existing peers, peer deactivations, and peer activations, and, in particular, to model the behaviour of each search mechanism when *Ultra Peer* failures occur in the network.

7. RESULTS

In Table 2 are reported the performance results (efficiency and QoS) for the baseline experiments, both for the content-based and the match-based search mechanism. Comparing these results, we can observe that the efficiency of content-based search is higher than the match-based one (only 77 messages per query and 38 nodes reached by each query, against 424 and 145, respectively). Touching such a smaller portion of the network, the content-based search mechanism does not always reach all the nodes with relevant documents, thus slightly affecting the recall. However, the QoS of content-based search is not compromised: comparable values of F-Score indicate that these two search mechanisms have almost the same retrieval accuracy. In content-based search, this is due to the high precision of the results.

It may appear surprising that the standard deviation is so high, but this is in fact quite common in context-based search [3].

Figures 5–11 report the results (efficiency and QoS) for the experiments performed while increasing the number of *Ultra Peer* failures, both for content-based and match-based search mechanisms.

Figure 5 reports the number of new peer (*Leaf Peer* and *Ultra Peer*) connections, due to the *Ultra Peer* deactivations. Note that the new peer connections are the same for all content-based experiments (CB+FLOOD and CB+CB), as well as for all match-based experiments (MB+FLOOD and MB+MB). The differences between the *match-based* and *content-based* experiments is due to the differing message traffic generated by the two different search processes. For example, the $3rd$ most heavily loaded *Ultra Peer* is 2513 for content-based search experiments, but it is *Ultra Peer* 2503 for match-based experiments (Table 2); hence experiments that deactivate 3 or more *Ultra Peers* produce different numbers

METHOD	CB-BASELINE	MB-BASELINE
Number of nodes	2,525	2,525
Number of queries	15,000	15,000
Average # of messages per query	77	424
Average Query Scope	38	145
Maximum Query Scope	77	2226
Minimum Query Scope	1	2
Average Precision	72.20	62.63
Standard deviation	33.63	38.44
Maximum Precision	1	1
Minimum Precision	0	0
Average Recall	26.58	30.29
Standard deviation	30.06	33.35
Maximum Recall	1	1
Minimum Recall	0	0
Average F-Score	38.86	40.83
Total # of deactivated nodes	0	0
Total # of new connections	0	0

Table 2: The performance of Content-Based and Match-Based search mechanisms without *Ultra Peer* deactivations.

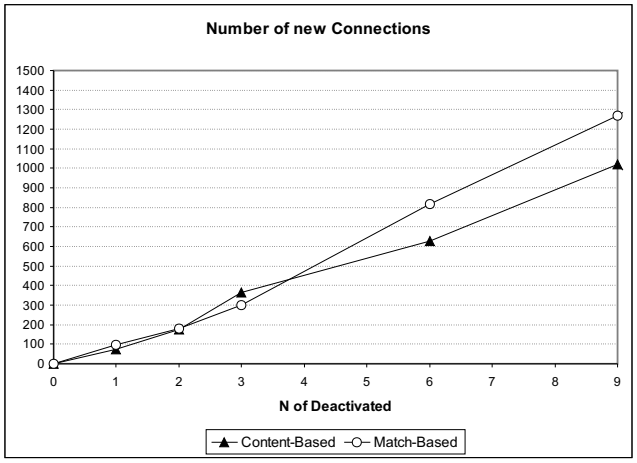

Figure 5: Number of new peer connections for Content-Based and Match-Based search mechanisms.

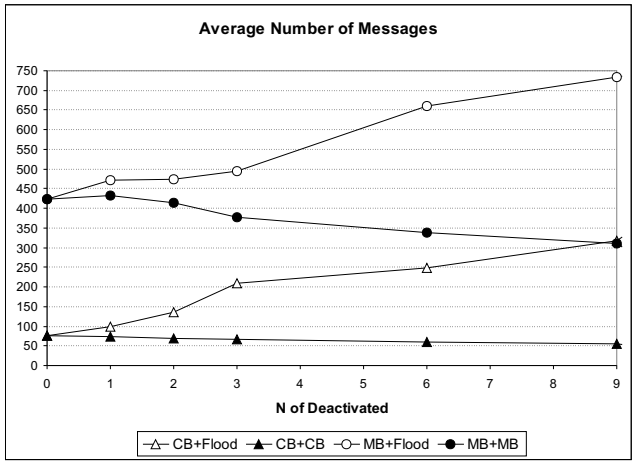

Figure 6: Average number of messages per query for the different search mechanisms.

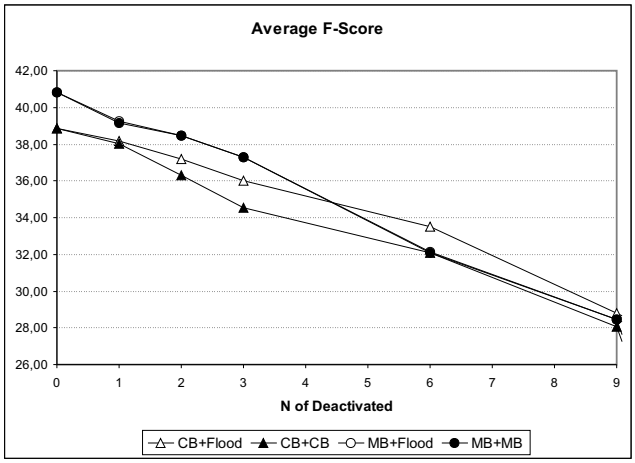

Figure 7: Average F-Score for the different search mechanisms.

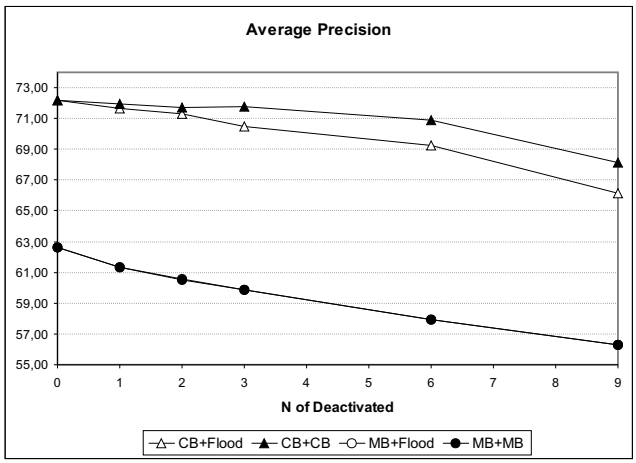

Figure 8: Average Precision for the different search mechanisms.

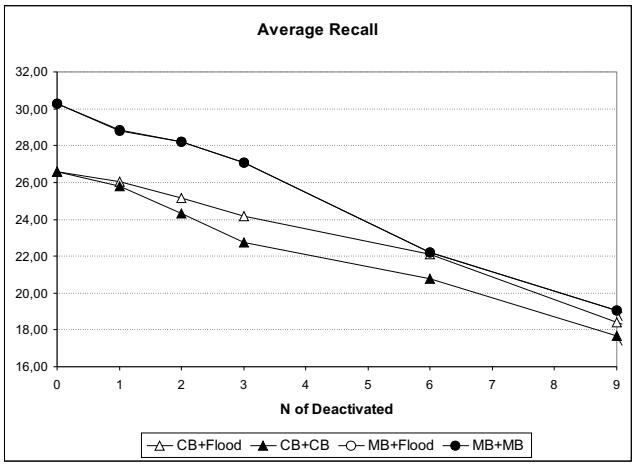

Figure 9: Average Recall for the different search mechanisms.

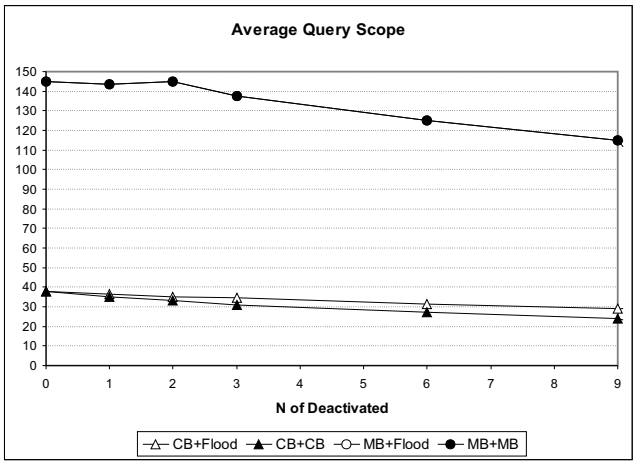

Figure 10: Average Query Scope for the different search mechanisms.

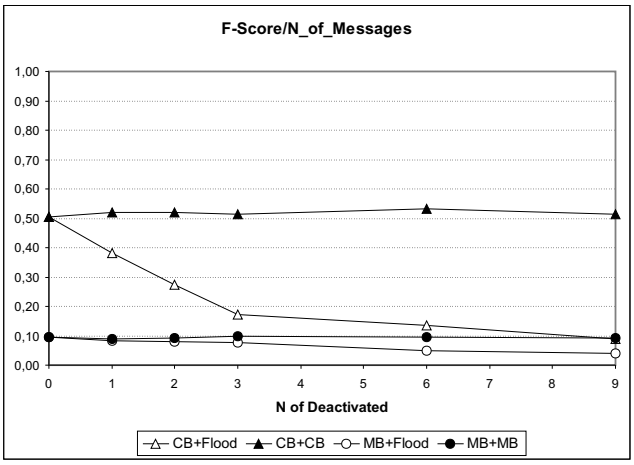

Figure 11: F-Score / Number of messages for the different search mechanisms.

of new connections in the network. As expected, the higher is the number of deactivations, the higher is the number of new connections.

In a proportional way, the number of messages exchanged among the peers per each query (Figure 6) grows for those search mechanisms that flood the query to all the new *Leaf Peers* (CB+FLOOD and MB+FLOOD). It is somewhat surprising that the number of messages actually drops for the two search mechanisms that attempt to learn models for newly assigned *Leaf Peers* (CB+CB and MB+MB); this drop reflects the time delay between when a *Leaf Peer* is assigned to a new *Ultra Peer* and when neighboring *Ultra Peers* begin noticing that the *Ultra Peer* is able to satisfy a broader range of queries; during this period of learning, fewer messages are exchanged.

Retrieval accuracy is reported in Figures 7, 8, and 9. The two match-based algorithms (MB+FLOOD, MB+MB) are nearly indistinguishable in these figures.

Looking at the methods that adopt the flooding to the new *Leaf Peers* (*i.e.*, CB+FLOOD and MB+FLOOD), we can observe that in presence of *Ultra Peer* failures CB+FLOOD and MB+FLOOD maintain the same relative performance observed in the baseline experiments, in terms of precision (Figure 8), while CB+FLOOD performs better in terms of recall as the number of *Ultra Peer* failures increases (Figures 9). Further, CB+FLOOD maintains its relative advantage over MB+FLOOD also in presence of failures both for the number of messages (Figure 6) and the query scope (Figure 10). Preliminary experiments with a higher number of *Ultra Peer* failures (up to 18) confirm the behaviour of these two search mechanisms, both in terms of QoS and efficiency.

For the search mechanisms based on the match-based paradigm (MB+FLOOD and MB+MB), the QoS and its degradation is almost the same, while the number of messages exchanged in the MB+MB search is smaller. This means that the two selection methods adopted to send the query to the new *Leaf Peers* (flooding to the new *Leaf Peers*, *i.e.* no selection, for MB+FLOOD, and *Leaf Peer* selection based on the matching between the query terms and the *Leaf Peer* vocabulary for MB+MB) obtain the same results, but the MB+MB is slightly more efficient in terms of number of messages exchanged.

For the search mechanisms based on the content-based paradigm (CB+FLOOD and CB+CB), the precision of the two methods degrades almost in the same way; quite surprisingly, also the recall of CB+CB degrades in the same way as the number of failures increases. This means that when the orphan *Leaf Peers* and *Ultra Peers* migrate to different regions of the network they have less chances to be selected (i) because the new *Ultra Peer* neighbors have to update their model with the new *Leaf Peer* description and learn the model of the new *Ultra Peers*; (ii) a *Leaf Peer* may be connected to an *Ultra Peer* being in its borderline, in terms of content. In fact, looking at the average number of nodes reached by the queries (Figure 10) with, for example, 6 failures, we have only 25 nodes reached in CB+CB against 31 in CB+FLOOD. For both the content-based search mechanisms studied, the QoS degrades gracefully as the number of deactivations increases, but the efficiency level of CB+CB is better. The retrieval accuracy of the two methods is similar, both in terms of recall and precision.

Even if the flooding to the portion of the network affected by the *Ultra Peer* failures is the simplest recovery way, it is not very efficient (Figure 6). CB+CB is the more efficient of the methods analyzed, both in terms of number of messages and query scope (Figures 6 and 10), maintaining a good level of retrieval accuracy (Figures 7, 8, 9).

Figure 11 reports the ratio between the F-Score and the number of messages for the different search mechanisms. As seen from the figure, CB+CB performs best with respect to this metric.

8. CONCLUSIONS AND FUTURE WORK

In this paper we have studied the robustness, effectiveness and efficiency of large-scale content-based federated search of text-based digital libraries organized in a hierarchical *P2P* network with multiple directory services. In such a *P2P* architecture, thanks to the *Ultra Peers* that provides centralized services to a portion of the network in a decentralized manner, the search mechanism can combine the advantages of centralized and decentralized search: the efficiency of the former, the autonomy and the robustness of the latter. However, the directory service failures must be taken into account to guarantee the robustness of the system.

In hierarchical *P2P* networks based on name-based retrieval, such as KaZaA, the directory services are easy to replicate or rebuild. Hence, these systems are robust even when the directories fail thanks to the massive file/directory replication. We have investigated the robustness issue in a network with no replication. The robustness demonstrated in our experiments is due to the directory

services and a simple set of network reorganization protocols that cope with unexpected failures of directory services.

As the number of failures at the *Ultra Peer* level increases, the results show that content-based search (CB+FLOOD and CB+CB) is more efficient than match-based search (MB+FLOOD and MB+MB) of a hierarchical *P2P* network, and equally effective. The simplest and most effective way to recover from one or more *Ultra Peer* failures is flooding to the portion of the network affected by the deactivations. In fact, the CB+FLOOD is a good way to obtain a graceful degradation of the QoS in case of failures. However, flooding is not very efficient. CB+CB maintains a good level of efficiency and effectiveness, but the results obtained in terms of recall suggest that there is room for improvement in (i) how orphan peers are reconnected to the network, and (ii) how quickly the directory service models are adjusted to include the contents of newly connected peers.

We are currently investigating the performance of search mechanisms that include:

- the use of pruning techniques for content-based selection, in order to reduce the information that must be acquired for each *Leaf Peer*; and
- content-based and match-based search with *Ultra Peer* deactivations, applying the query-learning selection to all the new *Leaf Peer* neighbors (CB+QL, MB+QL).

Studying the case in which the *Ultra Peers* may fail, we encountered more general problems; in particular, how to re-organize the directory services when a new *Leaf Peer* is connected to them. The work reported here takes into account a network where the maximum number of nodes is fixed. Currently, we are studying a network where new *Leaf Peers* and, if necessary, new directory services can be added at any time. To do this, we are studying a model for the directory services that is able to understand, based on the information maintained over the neighborhood, if a newly connected *Leaf Peer* could be served better by another *Ultra Peer* and, in this case, allowing/guiding the connection between these two peers.

Our long-term goal is a hierarchical *P2P* networks with large numbers of directory services that can self-organize and reorganize as the network changes, in order to perform content-based federated search of complex digital libraries, and to maintain high levels of effectiveness and efficiency under a wide variety of conditions. The research reported here demonstrate that *P2P* networks are a robust platform for large-scale federated search even when the digital libraries and search processes involved are more complex than those typically used for music filesharing.

9. REFERENCES

[1] K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. In *Proc. of the International Conference Cooperative Information Systems*, pages 179–192. Lecture Notes in Computer Science, 2001.

[2] A. Asvanund, S. Bagla, M. H. Kapadia, R. Krishnan, M. D. Smith, and R. Telang. Intelligent club management in Peer-to-Peer networks. In *Proc. of the Workshop on Economics of Peer to Peer Systems*, 2003.

[3] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In *Proc. of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR-00)*, pages 33–40, 2000.

[4] N. Daswani, H. Garcia-Molina, and B. Yang. Open problems in data-sharing Peer-to-Peer systems. In *Proc. of th 9th International Conference on Database Theory*, 2003.

[5] Gnutella Protocol Development: `http://rfc-gnutella.sourceforge.net/developer`, WWW.

[6] L. Gravano, K. C. Chang, H. Garcia-Molina, and A. Paepcke. Starts: Stanford proposal for internet meta-searching (experience paper). In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 207–218. ACM Press, 1997.

[7] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao. Distributed object location in a dynamic network. In *Proc. of the 14th ACM Symposium on Parallel Algorithms and Architectures*, pages 41–52, 2002.

[8] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for Peer-to-Peer networks. In *Proc. of the 11th Int. Conf. on Information and Knowledge Management (CIKM-02)*, pages 300–307. ACM Press, 2002.

[9] KaZaA Web Site: `http://www.kazaa.com`, WWW.

[10] J.C. Keegel, J.H. Kullback, and S. Kullback. Topics in statistical information theory. *Lecture Notes in Statistics*, 42, September 1987.

[11] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proc. of the ACM ASPLOS*, pages 300–307. ACM Press, 2000.

[12] J. Lu and J. Callan. Content-based retrieval in hybrid Peer-to-Peer networks. In *Proc. of the 12th Int. Conf. on Information and Knowledge Management (CIKM-03)*. ACM Press, 2003.

[13] Morpheus Web Site:. `http://www.musiccity.com`, WWW.

[14] Napster Web Site: `http://www.napster.com`, WWW.

[15] Project JXTA: `http://www.jxta.org`, WWW.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proc. of the ACM SIGCOMM*, 2001.

[17] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. In *Proc. of the International P2P Workshop*, 2002.

[18] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: The Oceanstore prototype. In *Proc. of the USENIX File and Storage Technologies (FAST03)*, 2003.

[19] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale Peer-to-Peer systems. In *Proc. of the International Conference on Distributed Systems Platforms (IFIP/ACM)*, pages 329–350, 2001.

[20] L. Si and J. Callan. The effect of database size distribution on resource selection algorithms. In *Proc. of the SIGIR 2003 Workshop on Distributed Information Retrieval*, 2003.

[21] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in Peer-to-Peer systems. In *Proc. of the IEEE INFOCOM03)*, San Francisco, CA USA, 2003.

[22] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-to-Peer lookup service for internet applications. In *Proc. of the ACM SIGCOMM*, pages 149–160. ACM Press, 2001.

[23] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proc. of the 22nd Annual International ACM SIGIR on Research and Development in Information Retrieval*, pages 254–261, 1999.

[24] B. Yang and H. Garcia-Molina. Improving search in Peer-to-Peer networks. In *Proc. of the 22nd International Conference on Distributed Computing Systems (ICDCS02)*, 2002.