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
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, US
callan@cs.cmu.edu

ABSTRACT

Hierarchical *peer to peer* networks with multiple directory services are an important architecture for large-scale file sharing due to their effectiveness and efficiency. Recent research argues that they 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 the network.

This paper studies the robustness, effectiveness and efficiency of content-based federated search in hierarchical *peer to peer* networks when directory services fail unexpectedly. Several recovery methods are studied using simulations with varying failure rates. Experimental results show that quality of service and efficiency degrade gracefully as the number of directory service failures increases. Furthermore, they show that content-based search mechanisms are more resilient to failures than the match-based search techniques.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models, search process, selection process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*distributed systems, performance evaluation (efficiency and effectiveness)*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*data sharing*

General Terms

Algorithms, Design, Experimentation, Performance

*This work was done when the author was a Visiting Graduate Research Assistant at the Language Technologies Institute, School of Computer Science, Carnegie Mellon University.

CIKM'04, November 8–13, 2004, Washington, DC, USA.

Keywords

Peer to Peer, Hierarchical, Search, Retrieval, Content-based, Robustness

1. INTRODUCTION

Peer to peer (*P2P*) networks have emerged in recent years as a popular method of sharing information, primarily popular music, movies, and software, among large numbers of networked computers. Computers ('nodes') in *P2P* networks can be information providers and information consumers, and may provide network services such as message routing and regional directory services. A single node may play several roles at once, for example, posting queries, sharing files, and relaying messages.

Information providers in *P2P* networks can be viewed as digital libraries. In many of the well-known operational *P2P* networks, for example, Gnutella [5], KaZaA [10], or LimeWire [13], the digital libraries are relatively small, ranging from a few dozen to a few thousand files. The search services they offer tend to be based on exact or partial match of file names that are known or easily guessed (e.g., names of popular songs, movies, or software) and/or small amounts of manually-assigned metadata.

Search for information in *P2P* networks is essentially *federated search* (or 'distributed information retrieval' [22]) across a possibly large set of digital libraries. Recent *P2P* research has focused on how to efficiently and effectively determine which digital libraries have desired information. The two major approaches are highly-organized networks (*e.g.*, Tapestry [8], OceanStore [12], Pond [20], Pastry [21], Chord [24]), in which content location can be inferred easily due to well-defined data placement policies; and hierarchical *P2P* networks with local directory services. The latter approach has became the most popular architecture for widely-deployed *P2P* applications (e.g., Gnutella 0.6 [5], KaZaA [10], Morpheus [15], Napster [16], Project JXTA [17]).

Our research interest is in using *P2P* networks to provide federated search of *text-based* digital libraries. For example, *P2P* protocols could be an effective and convenient method of providing federated search across the various document repositories of a medium-sized company. A *P2P* architecture is appealing in this environment because there is no need to collect and maintain centralized repositories or indexes. *P2P* protocols could also be an effective method of providing federated search of digital libraries that can be searched but not crawled by a Web crawler (so called "Hidden Web" repositories).

Prior research demonstrated that a hierarchical *P2P* network architecture can provide effective and efficient access to several thousand small text-based digital libraries [14]. In a hierarchical *P2P*

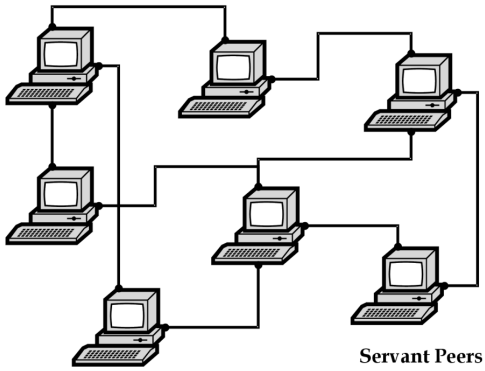

Figure 1: *Pure P2P* Network.

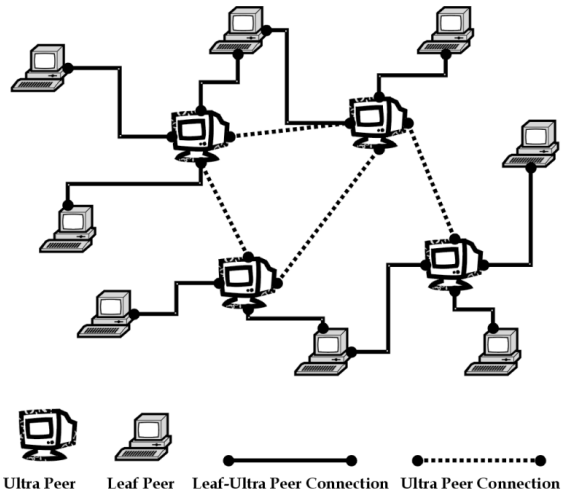

Figure 2: *Hierarchical P2P* Network.

network some nodes ('hub' nodes) provide regional or topical directory services that improve the accuracy and efficiency of the network by directing messages appropriately. Our interest in this paper is to study the stability and robustness of such networks when directory services fail. There is ample 'real world' evidence that hierarchical *P2P* networks such as KaZaA and Morpheus are stable as directory services ('ultra peers') enter and exit the network, but their stability relies on massive sharing of information. Our goal is to discover whether an architecture that gives digital libraries greater control over their intellectual property is as stable in the presence of directory service failure.

The next section of this paper describes some prior work related to our research. Section 3 provides more details about the motivation for our research. Section 4 describes the search mechanisms we used. Section 5 presents the behaviour of the search protocol under dynamic conditions. Section 6 describes the data and methodology adopted for the experiments. Section 7 reports and discusses the results obtained. Finally, Section 8 concludes.

2. PEER-TO-PEER SYSTEMS

P2P applications for file sharing have become widespread in recent years, which has focused considerable research attention on this area, particularly on how to improve *P2P* search. *P2P* search methods can be classified into three broad categories, based on how they manage the search process. We discuss these categories, and related prior research, below, with an emphasis on well-known solutions and solutions likely to be useful for federated search of text-based digital libraries.

2.1 Centralized Search (One Directory Service)

Napster, one of the first *P2P* systems, was essentially a single, network-wide directory service.[1] When a new digital library ('node') wanted to join the network, it provided a catalog of its current holdings to Napster. Napster handled search requests, returning lists of matching files and the addresses of the digital libraries providing them.

The Napster architecture was efficient and effective, but not robust because of the central and unique point of failure. Although Napster was highly influential, most later systems adopted decentralized search.

[1]For efficiency reasons, this single, logical directory service was actually implemented as several distinct, slightly different directory services.

2.2 Decentralized Search Without Directory Services

One reaction to the vulnerability of *P2P* networks with a single directory service was *P2P* networks with no directory services, sometimes called 'pure' or 'flat' *P2P* networks (Figure 1). Gnutella 0.4 [5] is an early example. It adopted unstructured search to find content: A peer sends a query to its neighbors (*'flooding'*), which attempt to satisfy it, and which also relay it to their neighbors, until the request has traveled a maximum distance from the initiating node (TTL, *Time to Live*). The Gnutella 0.4 protocol is simple, robust, and easy to apply to any type of content, but it is not scalable because the message overhead is high, and because messages do not propagate to distant parts of the network (which may have the best matching files).

A more focused *interest-based locality* approach [23] is based on collaborative filtering: If peer A is looking for document d and peer B owns it, A creates an *interest shortcut* to B because B might have other documents A is interested in. The shortcuts provide a loose overlay structure on top of Gnutella 0.4. To avoid flooding, the protocol first looks for content through shortcuts; when all of a node's shortcuts fail, it reverts to Gnutella 0.4. Shortcuts seem to have good load distribution properties, and help reduce the number of peers touched by each query. Prior experimental results were based on relatively short simulations (one hour), where the system was always running in optimal conditions [23], so the generality of the experimental results are not yet clear.

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 [24] (see below), but it exploits local knowledge and does not require central coordination or global knowledge, as Chord does. The P-Grid's search has good robustness and scalability characteristics, due to the replication distributed over all the peers.

Other approaches [9, 26] select which nodes to search based

on specific heuristics (past behaviour, past number of results, past query response time, etc.). These methods are generally efficient, but the heuristic-based selection of nodes can fail to find relevant documents.

2.3 Decentralized Search With Directory Services

The advantages of the centralized search model were its efficiency and global view of the network contents. The disadvantage was its single point of failure. A variety of recent *P2P* solutions maintain the directory service model, but distribute it across the network to improve reliability.

Gnutella 0.6 [4], KaZaA [10], and Morpheus [15] distinguish between two types of nodes: digital libraries that provide content ('leaf peers') and local directory services that route messages ('super nodes', 'ultra peers'). Each *Ultra Peer* provides centralized directory services to a subset (region) of *Leaf Peers*; the set of *Ultra Peers* covers the whole network of *Leaf Peers* (Figure 2), 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 to this as ***hierarchical* P2P *search***. Hierarchical *P2P* architectures with multiple directory services have become one of the dominant forms of deployed *P2P* network due to their effectiveness and efficiency. Their widespread use in large music file-sharing networks demonstrates that they can be robust ([10, 15]).

Recent research demonstrates that hierarchical *P2P* architectures can be used to provide federated search of text-based digital libraries. Techniques developed for *distributed information retrieval* [22] are easily adapted to provide multiple, overlapping directory services for a *P2P* network [14]. Experimental results comparing a variety of different resource selection and document retrieval algorithms demonstrate that in such networks full-text search is more accurate and more efficient than the simple name-based search mechanisms that are more common in *P2P* research. Furthermore, the use of resource selection, both at *Leaf Peer* and *Ultra Peer* levels, improves the efficiency of query routing without degrading effectiveness. However, the experimental results reported were based on static simulations in which the system is always assumed to operate under normal conditions.

Project JXTA [17] is an ad-hoc, multi-hop adaptive *P2P* network. *Relay peers* are a type of *Ultra Peer* nodes that 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.

An alternative to explicit directory services is the directory service implied in *distributed hash table* (*DHT*) architectures. 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.

Many systems rely on the *DHT* functionality to efficiently find data in *P2P* networks in a small number of hops (e.g., Tapestry [8], OceanStore [12], CAN [18], Chord [24]). *DHT*-based systems are scalable and efficient, and effective for name-based searches.

Although DHTs are appealing due to their simplicity and efficiency, DHTs have several disadvantages as a solution for federated search of text-based digital libraries. Name-based retrieval is generally not effective for text retrieval. For example, readers of this paper do not know the name of the file that contains it. DHTs can be effective if documents are described by controlled vocabulary terms ('metadata') instead of filenames, but people don't like assigning controlled vocabulary terms and have difficulty assigning them consistently. DHTs also define a very rigid overlay structure on the network, thus limiting peer autonomy. Finally, their performance, especially in terms of retrieval accuracy, in dynamic *P2P* conditions (e.g., peers leaving and joining the network) is unknown [19].

3. MOTIVATION

Prior research demonstrated that hierarchical *P2P* search is an effective method of providing decentralized full-text federated search of several thousand small text-based digital libraries [14]. However, it also made several assumptions about the *P2P* search environment that differ significantly from popular file sharing applications. It assumed that the contents of digital libraries were generally disjoint, and that widespread redistribution of contents provided by other digital libraries was generally unacceptable. These choices have the advantage of providing greater protection of intellectual property, but by reducing replication of files in the network, they raise concerns about the robustness of the approach. In particular, this type of *P2P* network may be much more sensitive to disruptions in directory services.

Hierarchical *P2P* networks such as KaZaA are effective in finding results even when directories fail or exit the network. However, when directory services are based on name-based retrieval, they are tiny, and thus easy to replicate or rebuild. KaZaA nodes also tend to redistribute files that they download, so popular files become available from many nodes within the network. The robustness of file sharing systems like KaZaA is due at least in part to massive file and directory service replication.

If we assume that nodes do not redistribute files that they have downloaded, there is essentially no replication of content in the network. Replication of directory services is possible, but directories that support full-text resource selection are much larger than directories for name-based resource selection, making them more costly to replicate.

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 directory service providers; these directory services are a form of regional centralization that creates potentially critical failure points. We do not consider failures of individual digital libraries (i.e., 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 effect 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: *resource selection* at the *Ultra Peer* (directory service) level and *document retrieval* at the *Leaf Peer* (digital library) level. *Ultra Peers* do not own/share data, but provide regionally-centralized services for a portion of the network (a mix of *Leaf Peer* and *Ultra Peer* nodes). In order to provide services they have to maintain information about the content of their *Leaf Peer* and *Ultra Peer* neighbors. *Resource descriptions* of neighboring nodes are acquired by each *Ultra Peer* and used to route queries

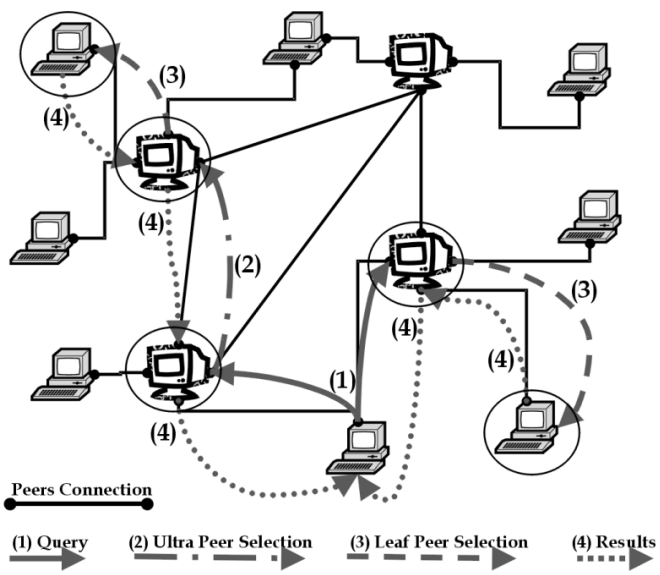

Figure 3: Search Mechanism: (1) Query Submission; (2) *Ultra Peer* Selection; (3) *Leaf Peer* Selection; (4) Results.

to a selected subset of *Leaf Peer* and *Ultra Peer* neighbors, *i.e.* those that are likely to contain relevant documents (*resource selection*). The regions served by each *Ultra Peer* can be organized along a variety dimensions, for example geographic ("Italy"), temporal ("the 15^{th} century"), and subject ("medicine"), and all of these could exist simultaneously in the network. Since any individual digital library can be a heterogeneous information source, it could be referenced by multiple directory services, 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).

We assume that a search client relays ('floods') its queries to all of its neighboring *Ultra Peers* (presumably this is a small number). Each *Ultra Peer* uses resource descriptions of neighboring nodes to select which *Ultra Peers* and/or *Leaf Peers* to relay the query to. We describe below how an *Ultra Peer* selects the *Ultra Peer* neighbors to which it forwards queries (Section 4.1), how an *Ultra Peer* selects the *Leaf Peer* neighbors to which it forwards queries (Section 4.2), and finally how a *Leaf Peer* performs document retrieval (Section 4.3).

4.1 *Ultra Peer* Selection

How best to represent the set of *Leaf Peers* served by a single *Ultra Peer* is an open problem. The *Ultra Peer* description could be represented, by the union of the descriptions of *Leaf Peers* it serves, for example as in hGlOSS [7]. However, our experience is that term frequency information relayed from one *Ultra Peer* to another becomes difficult to normalize as it propagates through the network, and thus difficult to compare with information from other nodes.

Our approach for this research is for each *Ultra Peer* to use query learning to acquire models of neighboring *Ultra Peers*. An *Ultra Peer* represents a neighboring *Ultra Peer* by the terms in the queries it has satisfied in the past. This approach produces small *Ultra Peer* resource descriptions that are effective for routing queries among *Ultra Peers*.

4.2 *Leaf Peer* Selection

There are several methods of acquiring resource descriptions of a digital library. In the research reported here, we adopt the STARTS protocol [6]: *Leaf Peers* provide their resource descriptions to *Ultra Peers* upon request.

We adopt two approaches to *Leaf Peer* selection.

Match-Based Selection: The resource description is the set of all unique terms present in the document collection the *Leaf Peer* provides. In this case the *Ultra Peer* simply forwards the query to those *Leaf Peer* neighbors that contain (match) the query terms.

Content-Based Selection: The resource description is the set of all unique terms that appear in the document collection the *Leaf Peer* owns *and their corresponding collection frequencies*. In this case the *Ultra Peers* measure how well each *Leaf Peer* description predicts a query with a Kullback-Leibler (*K-L*) divergence-based algorithm, an information theoretic metric used to measure how well one probability distribution predicts another [11]. Prior research has shown that the K-L method works well for collection selection [22, 25]. 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.

We use match-based retrieval as the baseline for our experiments. Match-based retrieval can be viewed as an enhancement of the simple filename-based retrieval that is used in most *P2P* file sharing systems.

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

5. NETWORK 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 represent the part in the hierarchical *P2P* network we are studying.

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*" if 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);

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: *Ultra Peer* node IDs, ranked by message load in content-based and match-based experiments.

- 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 [14]. 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 *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 $50,000$ available, to reduce simulation running time.

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

$$Precision = \frac{\#\ RelRet}{\#\ TotRet} \quad (1)$$

$$Recall = \frac{\#\ RelRet}{\#\ TotRel} \quad (2)$$

$$F - Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (3)$$

We adopt the same measure of relevance used in [14]. The results report the average recall computed over all queries and the average precision computed over the queries with at lease one returned document.

Efficiency is measured by the average number of messages exchanged for each query, and the average number of nodes reached by each query (*Query Scope*).

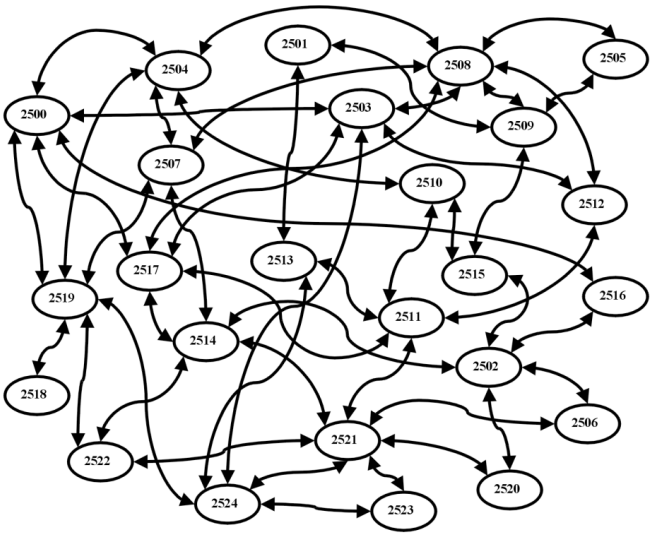

Figure 4: *Ultra Peer* connections.

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 without *Ultra Peer* deactivations.

EXPERIMENTAL SETUP. In the *Leaf Peer* selection/routing phase (Section 4.2), 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 [14] that the value of 1% for *Leaf Peer* selection gives satisfactory retrieval performance in hierarchical *P2P* networks (even in terms of recall).

The sizes of *Ultra Peer* resource descriptions (Section 4.1) were 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 effectiveness. To allow the insertion of new terms, when an *Ultra Peer* language model reaches the size limit, terms with lower frequency are removed.

Since each *Ultra Peer* has 4 *Ultra Peer* neighbors on average, an *Ultra Peer* was allowed to forward the query to up to 1 *Ultra Peer*, *i.e.*, at most 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 *Ultra Peer* 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 experiments increasing the number of *Ultra Peers* deactivated.

For each N deactivations, $N \in \{1, 2, 3, 6, 9\}$ (up to 36% of the *Ultra Peers*), 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 the Gnutella 0.6 protocol [5] and to support match-based and content-based search in hierarchical *P2P* networks. The simulator supported all of the dynamic events mentioned above, such as connections and disconnections between existing peers, peer deactivations and activations, and different types of network reconfiguration when *Ultra Peers* failed.

7. RESULTS

Table 2 reports efficiency and effectiveness results for the baseline experiments. Content-based search was more efficient than match-based search (only 77 messages per query and 38 nodes reached by each query, compared with 424 and 145, respectively). Touching such a smaller portion of the network, the content-based search mechanism did not always reach all the nodes with relevant documents, thus slightly affecting the recall. However, the accuracy of content-based search was not compromised: comparable values of F-Score indicate that these two search mechanisms have similar retrieval accuracy. In content-based search, this is due to the high precision of the results.

The standard deviation of recall and precision across the query set was high, as is common with full-text search [3].

Figures 5–11 report the results (efficiency and effectiveness) 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 are 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*

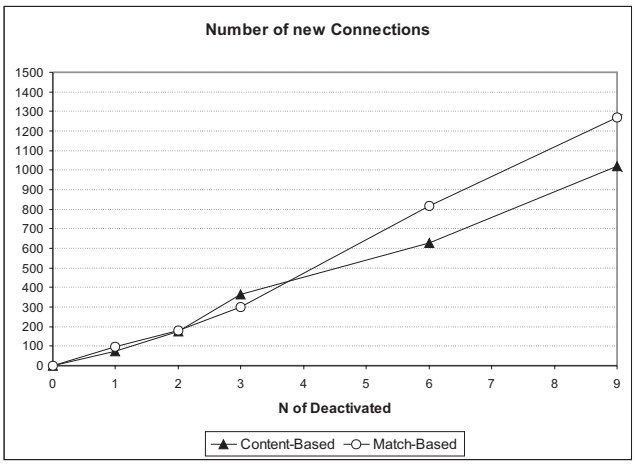

Figure 5: Number of new peer connections for Content-Based and Match-Based search mechanisms.

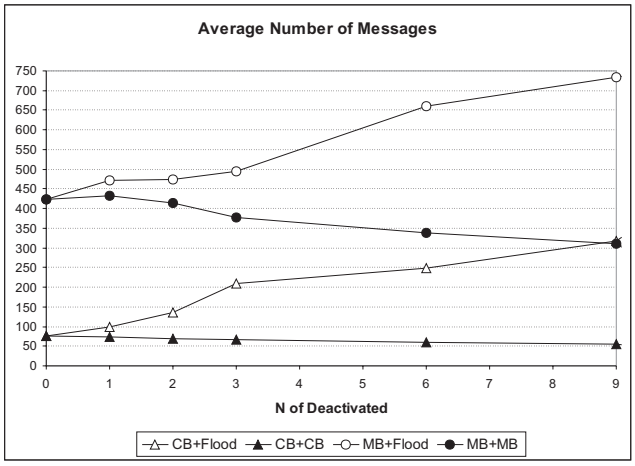

Figure 6: Average number of messages per query for the different search mechanisms.

2503 for match-based experiments (Table 2); hence experiments that deactivate 3 or more *Ultra Peers* produce different numbers of new connections in the network. As expected, the higher the number of deactivations, the higher the number of new connections.

In a proportional way, the number of messages exchanged among the peers per 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.

Among 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 main-

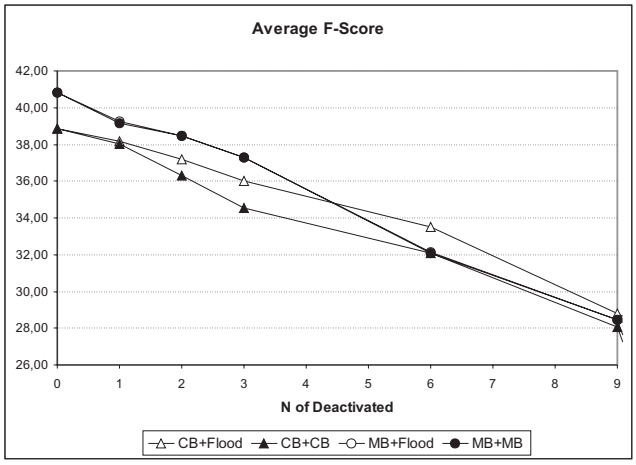

Figure 7: Average F-Score for the different search mechanisms.

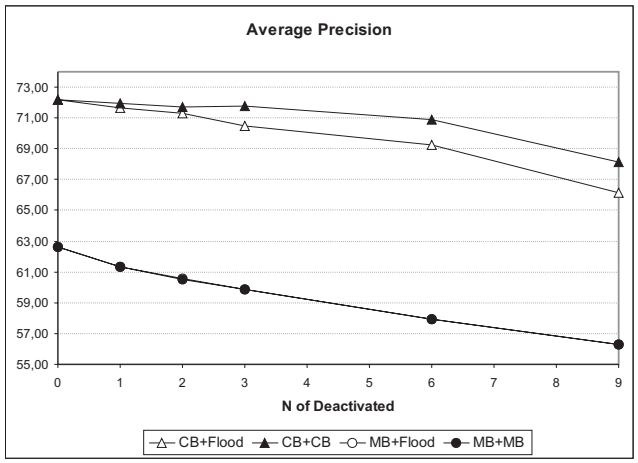

Figure 8: Average Precision for the different search mechanisms.

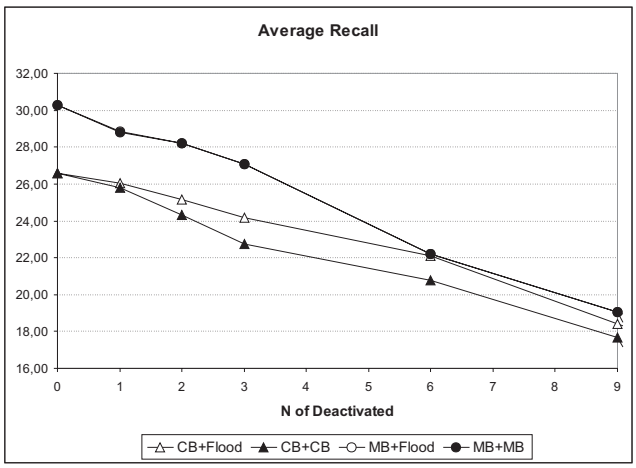

Figure 9: Average Recall for the different search mechanisms.

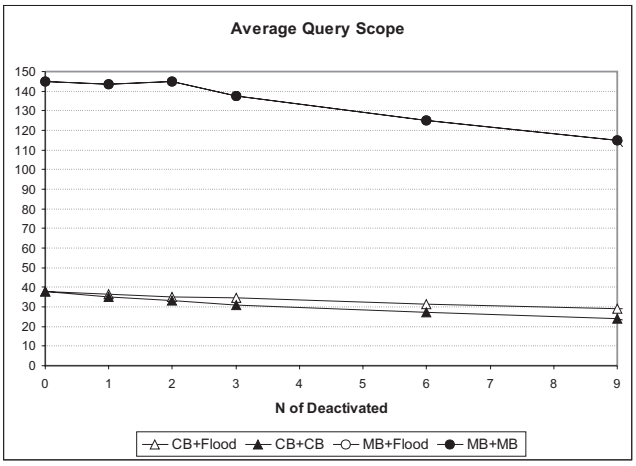

Figure 10: Average Query Scope for the different search mechanisms.

tain the same relative performance observed in the baseline experiments, in terms of precision (Figure 8), while CB+FLOOD performs better (*i.e.*, degrades less) 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, more than 75% of *Ultra Peers*) confirm the behaviour of these two search mechanisms, both in terms of effectiveness and efficiency.

For the search mechanisms based on the match-based paradigm (MB+FLOOD and MB+MB), the accuracy 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.

Among the content-based search methods (CB+FLOOD and CB+CB), the precision of CB+CB degrades less than that of CB+FLOOD, while the situation is reversed for the recall. This is because when the orphan *Leaf Peers* and *Ultra Peers* migrate to different regions of the network they have less chance 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 accuracy 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

This paper describes research on 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 provide 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 considered 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 accuracy 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 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

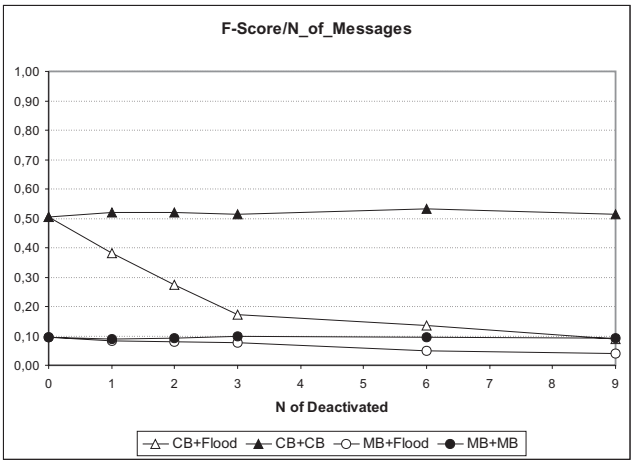

Figure 11: F-Score / Number of messages for the different search mechanisms.

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.

Acknowledgements

This material is based on work supported by NSF grants IIS-0096139 and IIS-0240334. Any opinions, findings, conclusions or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsor.

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] Gnutella Web Site: `http://www.gnutella.com`, WWW.

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

[6] L. Gravano, K. C. Chang, H. García-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] L. Gravano, H. García-Molina, and A. Tomasic. GlOSS: Text-Source Discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] Limewire Web Site: `http://www.limewire.com`, WWW.

[14] 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.

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

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

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.