Efficient Content Search in iShare, a P2P based Internet-Sharing System

Seyong Lee, Xiaojuan Ren, and Rudolf Eigenmann
School of ECE, Purdue University
West Lafayette, IN, 47907
{lee222,renx,eigenman}@purdue.edu

Abstract

This paper presents an efficient content search system, which is applied to iShare, a distributed peer-to-peer(P2P) Internet-sharing system. iShare facilitates the sharing of diverse resources located in different administrative domains over the Internet. For efficient resource management, iShare organizes resources into a hierarchical name space, which is distributed over the underlying structured P2P network. However, iShare's search capability has a fundamental limit inherited from the underlying structured P2P system's search capability. Most existing structured P2P systems do not support content searches. There exists some research that provides content search functionality, but the approaches do not scale well and incur substantial overheads on data updates. To address these issues, we propose an efficient hierarchical-summary system, which enables an efficient content search and semantic ranking capability over traditional structured P2P systems. Our system uses a hierarchical name space to implement a summary hierarchy on top of existing structured P2P overlay networks, and uses a Bloom Filter as a summary structure to reduce space and maintenance overhead. We implemented the proposed system in iShare, and the results show that our search system finds all relevant results regardless of summary scale and the search latency increases very slowly as the network grows.

1 Introduction

The proliferation of the Internet and related advances in networking technologies provides us with tremendous opportunities for utilizing the free resources available in Internet environments. One of the most promising but demanding topics in this area is to integrate computational and information resources scattered over the Internet into large-scale cohesive computing systems. iShare [10] is a distributed peer-to-peer (P2P) Internet-sharing system that facilitates the sharing of diverse resources located in different administrative domains over the Internet. For efficient resource management, iShare organizes resources into a hierarchical name space, which is distributed over the underlying structured P2P network. iShare uses Pastry [12] as underlying overlay network to exploit the efficiency, scalability, fault-tolerance, and autonomy. However, the underlying P2P network also sets a fundamental limit on iShare's search capability. In distributed P2P systems, advanced search functionality, such as content search and result-ranking, is difficult to build, because information about resources is also distributed.

In a content search function, the input is a set of keywords representing a user's interests and the output is a set of resources containing these keywords. In the content search context, resources represent text documents or metadata of *general resources*, such as software applications, computer platforms, or data volumes. Content search is useful when a user does not know the exact resource names of interests; this case is common in P2P-based searches as well as in web searches.

Several solutions have been proposed to build a content search system on top of distributed P2P systems. Most of the proposed solutions use an unstructured overlay network, where searches are done by forwarding query messages from node to node in a *flooded* manner, leading to poor scalability with respect to system size. On the other hand, structured P2P overlay networks, where resources are distributed based on hash functions, can provide improved scalability. Existing structured P2P systems such as Pastry [12] provide mappings from keys to locations in a distributed manner. Searches based on such mappings cause logarithmic latency and thus scale well as the P2P system grows. However, in spite of the high search efficiency of structured P2P systems, two challenges must be solved to provide efficient and accurate content search in such systems.

First, general structured P2P overlay networks only provide search capabilities with a single keyword due to the use of hashing. Simply searching for each of multiple keywords individually results in poor efficiency [11]. Moreover, up-

dating resource data is expensive, as multiple peer nodes containing the related keywords need to be notified.

The second challenge of building content search on structured P2P systems lies in the realization of a ranking capability, which orders the retrieved search results by their relevance to the queried keywords. Ranking techniques have been studied in the area of semantic searches for P2P overlay networks [15, 3, 13, 8]. In a semantic search, semantic information, such as a classification hierarchy, is used either to restrict the search scope with improved search efficiency or to rank the retrieved search results by their semantic relevance. To this end, many existing approaches proposed a semantic overlay network, where semantically related resources are closely clustered. However, most of these approaches provide a single semantic concept search only [8], i.e., they do not support content search capabilities. Some other approaches can not handle the update of semantic information, because they rely on either a static classification hierarchy [3, 13] or predefined dictionary space [15].

In this paper, we propose an efficient search system, built on structured P2P overlay networks. Our system addresses both issues of content search and semantic ranking. To provide high search efficiency and accuracy, we extend a *path-name-based* hierarchy system, which was proposed in iShare, to build a *summary hierarchy* on top of general structured P2P systems. A summary hierarchy is a tree-like structure, where an intermediate node contains the union set of keywords maintained by its child nodes. Each leaf node in the hierarchy contains the keywords for describing an individual resource. The proposed mechanism also allows semantic ranking when there is a semantic hierarchy (i.e., classification hierarchy), as in iShare. Moreover, flexibility in the path-name-based hierarchy makes our ranking system adaptable to changes of semantic information.

We deploy a Bloom Filter [2] as a space-efficient summary structure, so that the summary hierarchy scales well. The probabilistic property of a Bloom Filter allows our content search system to find all relevant results regardless of summary scale with an acceptable increase of search overhead. Moreover, by using Bloom Filters, the overhead of updating the hierarchical summary structure is much smaller than in other existing summary hierarchy systems [14].

We implemented a prototype for the proposed content search system in the iShare system. We evaluated our content search system in terms of search accuracy, search efficiency, and the effect of search-result caching, using iShare's built-in simulator. The rest of this paper is organized as following: Section 2 provides an overview of the iShare system and explains the background of path-name-based hierarchies and Bloom Filters. Section 3 presents the proposed summary-hierarchy-based content search system. Section 4 shows experimental results. Related work and conclusions are presented in Section 5 and Section 6, respectively.

2 Background

Our content search function builds on iShare's hierarchical name space to create a summary hierarchy on top of structured P2P overlay networks. A Bloom Filter summarizes a branch in the hierarchy. The leaves represent simple text documents or descriptions of metadata of general resources, such as software applications, computer platforms, or data volumes.

In this section, we provide an overview of the iShare system and describe the concepts of path-name-based hierarchies and Bloom Filters.

2.1 Overview of the iShare Internet Sharing System

iShare is an open Internet-sharing system, which includes resource management functionality for cycle-sharing systems, such as resource publishing, resource discovery, resource access, and administration. iShare's resource management is based on a decentralized P2P framework, where participants can play the roles of both providers and users. Resource providers can easily describe and publish their resources and usage policies; end users can easily browse and access published resources. These functionalities are provided as a set of tools in the user interface of the iShare system.

To enable the efficient search for resources with specific capabilities, iShare organizes resources into a hierarchical name space and distributes the name space to the underlying P2P network. More information about the path-name-based hierarchy is shown in the following section. This design enables resource discovery without knowledge of where the corresponding data items are stored. However, iShare's search system has a fundamental limit due to the underlying P2P network: more advanced search, such as content search and ranking, is difficult to realize in P2P-based search systems.

We propose a summary-hierarchy-based content search system to address these issues; the system is presented in Section 3.

2.2 Path-Name-based Hierarchy

Using a path name in a hierarchical name space, our system forms a hierarchy on top of an existing structured P2P system [12], where peer nodes are located in a flat ID space. In this hierarchical name space, each resource is represented by a node in a tree and the path from the root to this node is called the path name of the resource. If the name space is

configured as a semantic hierarchy (i.e., classification hierarchy), it can be used for general semantic search or ranking, where each node in the hierarchy tree represents a semantic concept or a category. In this system, the resources that are semantically related are stored in the same node or in sibling nodes in the semantic tree. Figure 1 shows a semantic hierarchy example where *Cyber Lab* is a root node and there are two categories: *Biology* and *Chemistry*. In this example, each category contains two resources; all intermediate nodes in the hierarchy represent semantic concepts or categories, and all leaf nodes represent resources.

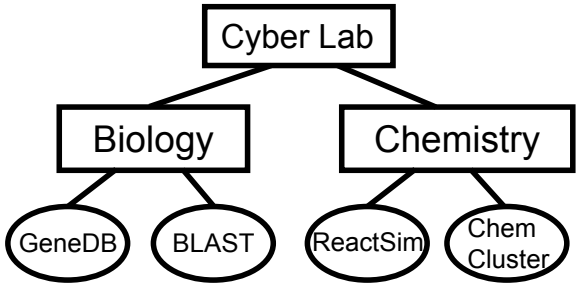


Figure 1. Semantic Hierarchy Example

Table 1. Messages needed to publish a resource whose path name is "*/Cyber Lab/Biology/GeneDB*"

Message	Key	Content
1	Cyber Lab	Biology
2	Biology	Biology/GeneDB
3	Biology/GeneDB	metadata of GeneDB

In a path-name-based P2P hierarchy, an item in the hierarchical space is mapped to a peer node, based on the hash value of the item's path name. A child's path name is kept in the parent. Hence, a resource can be searched incrementally through its hierarchical path name. For the hierarchical name space in Figure 1, Table 1 shows necessary messages to publish a resource, whose path name is "*/Cyber Lab/Biology/GeneDB*". The first message checks whether a node representing *Cyber Lab* has the *Biology* category as a child node, and if not, it creates the category. The second and third messages are used to publish a resource called *GeneDB* in the hierarchy. Each message is routed to the destination node, whose ID is numerically closest to the hash value of the message's key.

2.3 Bloom Filters

A Bloom Filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. Physically, a Bloom Filter is an array of m bits, where the value of each bit is decided by hash functions. When a Bloom Filter is created, all m bits are initialized to *0*. Given a keyword, a set of hash functions compute the corresponding bits in the Bloom Filter. The operation of inserting a keyword sets the selected bits to *1*. Due to hash functions, a Bloom Filter can perform insertion or membership-check in constant time.

One important property of a Bloom Filter is that it can give a positive answer even though a certain element or keyword is not contained in the set. However, it does not give false negatives; a Bloom Filter always answers yes if the set contains a keyword. Given the number of input keywords, both the array size m and the number of used hash functions decide the probabilistic error rate. In Section 4, we will show how different parameter settings for the Bloom Filter affect our search performance. Another intrinsic limit of a Bloom Filter is that deletion of a keyword is impossible. To delete a keyword, at least one of the bits set by the keyword must be reset to *0*. However, this has the side effect of removing any other keywords that set the bit to *1* and there is no way of determining whether any such keywords have been added. Hence, to delete a keyword from a set, the Bloom Filter must be regenerated from the beginning. In our system, rebuilding a Bloom Filter still causes negligible overhead, which will be explained in Section 3.1.

3 Summary Hierarchy based Content Search

In this section, we present an efficient summary-hierarchy-based content search system. We assume that each resource belongs to a category in some hierarchical name space. A hierarchy in the name space can be any predefined classification hierarchy, such as the ODP (Open Directory Project) category hierarchy [1] or a user-defined hierarchy, such as the one used in iShare. For example, iShare allows that each resource provider can either choose an existing category or create a new category, to which the resource will be published. Our system uses the hierarchy in the name space to form a summary hierarchy; each node (category) in the hierarchy contains summary information about its child nodes.

Figure 2 shows the overall system structure. In our search system, a large number of machines (peer nodes) constitute a structured P2P overlay network, where each resource is published using its path name as a hash key. Our system extends a path-name-based hierarchy to build a summary hierarchy. New categories are automatically published into the same ID space, where resources are published, if a new resource's path name contains categories that do not exist in the system. To summarize contents efficiently, we use a Bloom Filter.

Content search is performed by traversing the summary-

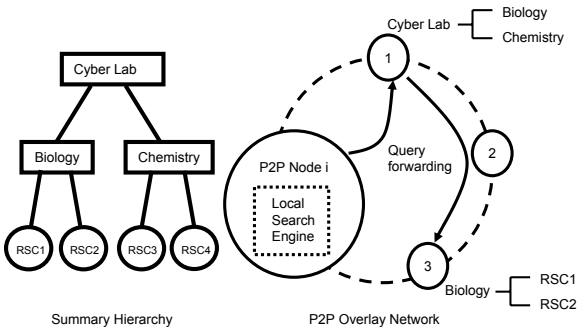


Figure 2. Overall search system. Each node in the P2P overlay network contains summary information of its child nodes in a path-name-based hierarchy.

hierarchy tree to find resources containing the keywords of the query. If a semantic hierarchy is used, various optimizations can be applied to our search system to improve efficiency and accuracy.

The following subsections describe design components of our search system. Section 3.1 explains how to build a summary hierarchy; Section 3.2 describes the overall mechanism of the proposed content search system; Section 3.3 shows a semantic ranking system as an application of our content search system and Section 3.4 discusses scalability.

3.1 Building Summary Hierarchy

To build a summary hierarchy, we extended the path-name-based hierarchy, such that each node in the hierarchy contains summary information of its children. When a resource is published, the summary information for the resource is also created. In our system, a summary is a set of keywords used in the description metadata of a resource. The keywords are stored in the form of a Bloom Filter. Because a Bloom Filter uses hash functions to represent keywords, summary data is generated with much less computational overhead than such techniques as the Vector Space Model (VSM) [16] and Latent Semantic Indexing (LSI) [9]. The summary data is included in a resource-publish-request message. It updates the summary information of the categories encountered along the path from the root to the leaf node representing the resource.

In a P2P overlay network, peer nodes can join and leave frequently. Therefore, corresponding summary data should be split or merged upon a peer node's join or leave. In our summary-hierarchy system, merging two sets of summary data is very easy. Because a Bloom Filter is implemented as a bit array, merging two existing summaries involves a simple bitwise OR operation. Considering that publishing new resources are more common than unpublishing (deleting) resources in a P2P system, such a lightweight merge operation makes the summary hierarchy efficiently adaptable to a dynamic P2P environment.

One intrinsic limit of a Bloom Filter-based summary hierarchy is that deleting partial summary information is not allowed, as mentioned in Section 2.3. To delete keywords from summary data, we should rebuild the summary using the remaining keywords from the beginning. In our summary hierarchy system, deleting keywords is needed when a resource is unpublished from the P2P overlay network or when a summary for a category must be split because of a new peer node's join. In these cases, however, rebuilding the summary hierarchy still does not cause large overhead. In terms of a summary hierarchy, a resource deletion or category split can be seen as deleting or splitting children of a node in the summary hierarchy, and thus, these operations can be done simply by doing bitwise OR operations on the summaries of the remaining child nodes. This summary change of a node is propagated up to parent nodes till it reaches the root node. For fault-tolerance, replicas of modified summary data are also updated.

In summary, the overhead for maintaining the summary hierarchy is tolerable, because it requires only simple bitwise OR operations in a branch of the hierarchy and a few messages to forward updated summary data. Further optimizations to reduce this overhead will be described in Section 3.4.

3.2 Hierarchy-based Content Search

Using the summary information, the search function proceeds as follows. First, a user's multiple-keyword query is sent to the root node in the summary hierarchy. The root node determines which child nodes contain related information, and forwards the query to these nodes. This forwarding is repeated from the root node to leaf nodes in the hierarchy, and when a leaf node receives the query, the node determines if it has the keywords of the query. If the leaf node, which represents one resource, contains these keywords, then it answers the query and replies to the user's local search engine. Next, the user's local search engine collects and ranks these replies. Methods for semantic rankings are discussed in Section 3.3.

One important issue in this search mechanism is how to efficiently find relevant nodes in the hierarchy, so that a query can be forwarded only to these nodes. For this goal, we use summary information. Each node in the hierarchy contains efficient summary information of its children as explained in Section 3.1. When one node receives a query message, this node examines the query against summary information of its children. The query is then forwarded only to the nodes containing the relevant keywords. By using

these summaries, we can reduce unnecessary message overhead and therefore, our search system scales well.

For a large-scale P2P system, space efficiency and computation complexity of a summary data structure are crucial factors for search efficiency. Space-inefficient summary data will cause significant memory overhead. Complex summary update operations are not suitable for general P2P systems, where nodes can join and leave frequently and the hierarchy can change dynamically. A Bloom Filter-based summary hierarchy addresses these issues.

Using a Bloom Filter as a summary, we can always find relevant results if they exist, because a Bloom Filter always answers positively if a keyword exists in the summary set. As mentioned in Section 2.3, however, a Bloom Filter may answer positively, even though a certain keyword is not stored in a set. A false answer incurs additional overhead, as the query messages will be forwarded to nodes that do not store the relevant resources. The probability of this false answering is a function of the number of hash functions, bit-array size, and the number of inserted keywords. Hence, if we can estimate the total number of keywords used in the target system, we can limit the probabilistic error rate by choosing an appropriate number of hash functions and bit-array size.

In our system, users may not have complete knowledge of the hierarchy because the hierarchy information itself is distributed over peer nodes and can change. However, if a user knows the hierarchy information, the user can restrict the search scope by initiating the query from an intermediate node in the tree, rather than from the root. If the hierarchy is a semantic hierarchy, this has the effect of restricting the search scope to certain semantic categories; this restriction can avoid the bottleneck of all requests going through the root node.

To reduce search latency, our system uses a search-result cache, where recently searched results are stored. Our search system first checks if search results are stored in the local cache. If valid (unexpired) cache results are found, the search system directly returns them. Otherwise, the search system initiates a request as usual.

3.3 Semantic Ranking using Summary Hierarchy

Semantic ranking is an important application of content search. Several ranking functions have been proposed; they assume underlying content search functionality. Here we describe possible ranking methods that build on our context search system.

One way of ranking is to employ the user's predefined preference. We call this semantic-distance-based ranking. In this ranking method, query users store semantic categories relevant to users' interests in their profiles, and when retrieved results are ranked, the ones closest to each user's preferred semantic categories are ranked first. Figure 3 shows one example where a user is interested in the *Chemistry* category and the search system found three results: *GeneDB*, *BLAST*, and *ReactSim*. Based on the relative distance between the user's interested category and each retrieved result, the shortest one is ranked first.

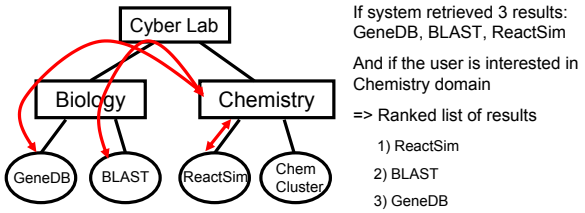


Figure 3. Example for Semantic-Distance-Based Ranking

Another method is a personalized-categorization-based ranking, which has been proposed for web search [7]. In this approach, the ranking system records the user's previous interests (i.e., semantic categories to which the user's previously selected results belong), and when a user makes a new query, semantic similarities between the user's query and categories representing the user's interests are calculated. Next, the categories (the user's interested domains) are ranked in descending order of similarity. This ranked list shows the user's interests implied by the query. The query results are then ranked according to this list; results belonging to the first category in the list will be ranked first. To calculate a semantic similarity, various similarity functions, such as Cosine Similarity or KL distance [5], which are commonly used in the Information Retrieval area, can be used.

3.4 Scalability Issue

As explained in Section 3.1, adding a new summary to our hierarchy is simple, but deleting a summary incurs overhead. In a large-scale system, where resources can be added or deleted frequently, this overhead can be significant. To reduce it, our system reconstructs the summary hierarchy only when the false answering rate exceeds a certain threshold. Avoiding delete operations in this manner does not reduce search accuracy, but it may cause extra message overhead. To strike a balance between search overhead and summary update overhead, measuring the false answering rate is key. If a node receives irrelevant query messages, this means that the Bloom Filter in its parent node answered a query with a false positive. By monitoring irrelevant query messages, each node can check this false answering rate. A

high rate indicates high search overhead; the summary hierarchy should be reconstructed. Triggering the summary hierarchy reconstruction in this way can control the balance effectively.

Another scalability issue is that the root node in our system can form a bottleneck, as it represents the entry node for all queries, unless a user restricts the search scope explicitly. To alleviate this hot-spot problem, our system explicitly replicates the root node; query messages are sent to a randomly selected replica. The number of replicas is a design parameter, which is chosen based on the expected P2P system size.

4 Evaluation

We have implemented a prototype of the proposed content search system, using the Pastry P2P overlay built in the iShare system. We evaluated our content search system in terms of search accuracy, search efficiency, and the effect of search-result caching. To measure these metrics in systems of different sizes, we simulated several iShare testbeds of various scales.

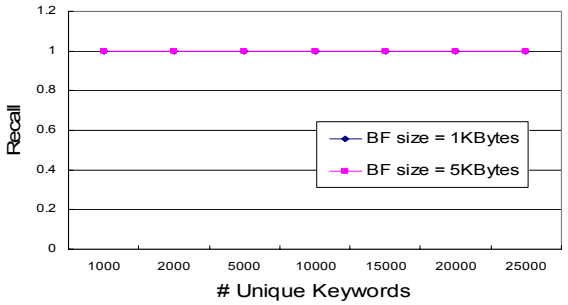
4.1 Methodology

We simulated our search system on a GT-ITM router network using the transit-stub model [17]. To test the scalability, we used several iShare testbeds with a number of nodes ranging from 100 to 10,000. Our simulations use the Reuters-21578 text categorization test collection [6]. The categories in the collection are used to build a summary hierarchy tree, and each text document is used as a resource published in the iShare system. For the summary structure, we used a Bloom Filter with 8 hash functions and bit array size varying from 1 KBytes to 5 KBytes. By changing the Bloom Filter's bit-array size, we examined the effect of summary data size on search accuracy and overhead.

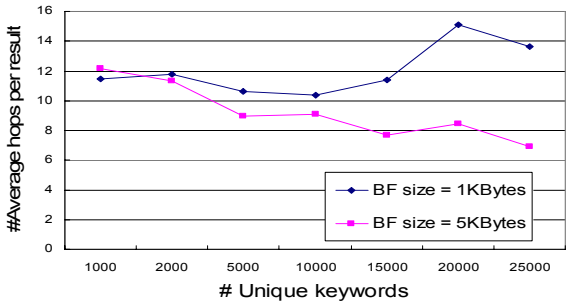
To measure the influence of a user's keyword search patterns on search latency, we examined a naive search system, which does not use caching, with three types of synthetic user search patterns: a random keyword selected from a collection of keywords, multiple random keywords with *AND* logical operators, and multiple random keywords with *OR* logical operators. For the last two types, the total number of keywords varies from *1* to *5*. Each search query starts from the root node in the hierarchy, and propagates to all relevant nodes. To understand the effect of search result caching, we compared search latency under different cache expiration times.

4.2 Results

Figure 4 shows the effect of summary data size (Bloom Filter size) when the number of unique keywords varies from 1000 to 25000. In this experiment, the iShare testbed consisted of 1000 nodes, and a set of resources were randomly published to the nodes. Table 2 shows the number of published resources and categories for each test. In this table, the number of keywords means the total number of keywords in all published resources. We tested for keyword searches using one random keyword selected from a keyword collection for each search query. For each search operation, one node in the P2P overlay network was randomly selected and the search request was initiated from the node.



(a) Recall



(b) Average hops per results

Figure 4. Keyword search accuracy and latency when number of keywords varies from 1000 to 25000. BF means Bloom Filter.

Figure 4 (a) shows the search accuracy for two summary sizes (1 KBytes and 5 KBytes). We use *recall values* as the metric for search accuracy. Recall is defined as follows:

$$Recall = \frac{the\ number\ of\ retrieved\ relevant\ resources}{the\ number\ of\ all\ relevant\ resources}$$

From the figure, we can see that our search system finds all relevant results regardless of the number of keywords stored in the system. This is expected, as a Bloom Filter

Table 2. Statistics of published resources and categories

Test	# published	# keywords	# unique keywords	# published categories
1	24	1556	1003	10
2	69	4056	2042	15
3	226	16093	5070	30
4	938	58759	10014	47
5	1992	126055	15002	52
6	3452	222304	20000	58
7	5561	367308	25003	281

does not return false negatives. On the other hand, Figure 4 (b) shows that, in certain cases, search overhead can increase, as the number of keywords increases. In Figure 4 (b), we can see that, when a Bloom Filter size is 1 KBytes, the average number of hops per result increases, as more keywords are added to the iShare testbeds. In a Bloom Filter-based summary, the error rate is proportional to the number of stored keywords, but inversely proportional to the size of the Bloom Filter. Error rate means the probability that the Bloom Filter gives false positives, as explained in Section 3.2; this property leads to the search query being forwarded to irrelevant child nodes, which increases the number of messages used in the search.

In the case of a Bloom Filter with the size of 5 KBytes, however, the number of average hops per result decreases as the number of keywords increases. This decrease can happen because our search system is tested with random keywords. If many relevant results are located within one or a small number of categories, the average number of used messages per result will be smaller than the opposite case where relevant results are scattered among several categories. From Figure 4 (b), we can see that Bloom Filter size of 5 KBytes is big enough for a system with total 25000 unique keywords.

Figure 5 presents how search efficiency scales with the system size. In these experiments, the three different user search patterns were used. For each search query, one random node in the P2P network was selected to initiate the search request. The results in Figure 5 show that the search latency increases very slowly with the total number of nodes, and the influence of different search query patterns is not significant.

In the last set of experiments, we measured the effect of caching search results with different normalized cache expiration time e (e = cache expiration time/average request period). In these experiments, a random keyword was chosen for each search query, and the query was initiated from a random node selected among five pre-selected nodes in the

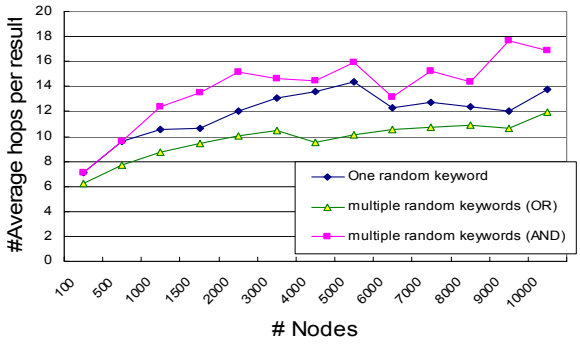


Figure 5. Average number of hops per result as a function of the number of iShare nodes

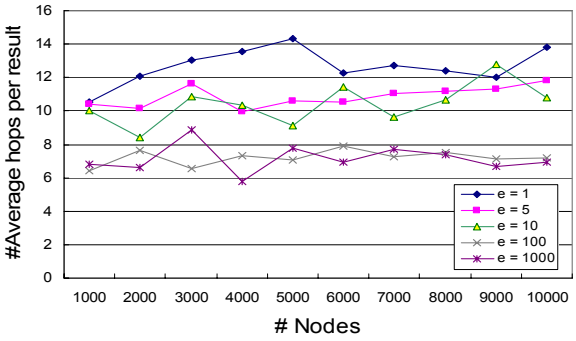


Figure 6. Average number of hops per result for random query model. e is the normalized cache expiration time (e = cache expiration time/average request period).

P2P networks. From Figure 6, we see that, with fair expiration time (e= 5), search response time is reduced by 14% on average, compared to the search without a local cache. Moreover, the figure indicates that the cache effect is not a function of system scale.

5 Related Work

Content search systems for unstructured P2P system are proposed in many papers [4], but these approaches incur substantial search overhead in unstructured P2P systems.

In [11], the authors proposed a multiple-keyword search technique using a distributed inverted index, which uses a Bloom Filter as a summary representation. The system in [11] supports only simple multiple-keyword searches, which do not use any semantic information. Moreover, the system incurs high maintenance overhead because, whenever a new resource is published or unpublished, all nodes containing the keywords of the resource must be updated.

There are several approaches to exploit semantic information for search or ranking: one approach [15] uses a semantic vector, which is generated by Latent Semantic Indexing (LSI) [9], as the key to store the document index in Content-addressable Networks (CAN); and [3, 13] use a predefined classification hierarchy or ontology to exploit a semantic relationship. In contrast to these approaches, [8] uses a path-name-based semantic hierarchy, which is similar to our system. However, [8] allows only an exact classification path or a statement for a query model, while our system allows general multiple-keyword searches.

The system proposed in [14] is the most closely related to our system. It uses a two-level hierarchical summary indexing structure for content searches. However, the proposed system in [14] works only on the unstructured Super-peer P2P Network, and the hierarchy used in the system does not have any semantic relationship. Semantic information is generated when an advanced index summary technique such as LSI is applied to Vector Space Model (VSM), which is a base summary technique in [14]. Even though LSI can discover an underlying semantic correlation among resources, the complexity of LSI generation and update, and the space requirement for VSM set limits on the system's scalability. Our system, however, uses a changeable semantic hierarchy realized by a Bloom Filter, achieving improved efficiency and scalability.

6 Conclusion

In this paper, we have presented a summary-hierarchy-based content search system. The proposed technique is realized in a general structured P2P overlay network. An extended path-name-based hierarchy allows us to build a summary hierarchy on top of the P2P system, and a Bloom Filter-based summary structure makes our system efficient and scalable without losing search accuracy. In this paper, we have focused on design and evaluation of the P2P content search system in our Internet-sharing system, iShare. The semantic hierarchy space of our system allows various optimizations, such as a semantic ranking. In future work, we will examine various ranking methods to explore their interaction with our content search system.

References

[1] Open directory project. [online]. available: http://dmoz.org/.

[2] B. H. Bloom. Bloom filter. [online]. available: http://en.wikipedia.org/wiki/bloom_filter.

[3] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. *Technical report, Computer Sicence Department, Stanford University*, October 2002.

[4] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. *Proc. HPDC*, June 2003.

[5] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[6] D. D. Lewis. Reuters-21578. [online]. available: www.daviddlewis.com/resources/testcollections/reuters21578/.

[7] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Trans. KDE*, 16-1:28–40, 2004.

[8] A. Loser, K. Schubert, and F. Zimmer. Taxonomy-based routing overlays in p2p networks. *Proc. IDEA*, pages 407–412, 2004.

[9] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. pages 159–168, 1998.

[10] X. Ren, A. Basumallik, Z. Pan, and R. Eigenmann. Open internet-based sharing for desktop grids in ishare. *1st Workshop on Large-scale, Volatile Desktop Grids*, March 2007.

[11] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. *Proc. Middleware*, June 2003.

[12] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Proc. Middleware*, pages 329–350, November 2001.

[13] C. Sangpachatanaruk and T. Znati. Semantic driven hashing (sdh): an ontology-based search scheme for the semantic aware network (sa net). *Proc. Peer-to-Peer Computing*, pages 270–271, August 2004.

[14] H. T. Shen, Y. Shu, and B. Yu. Efficient semantic-based content search in p2p network. *IEEE Trans. KDE*, 16-7:813–826, July 2004.

[15] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. *Technical Report, HP Labs*, November 2002.

[16] S. K. Wong, W. Ziarko, V. V. Raghavan, and P. C. Wong. On modeling of information retrieval concepts in vector spaces. *ACM Trans. TODS*, 12-2:299–321, 1987.

[17] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. *Proc. IEEE Infocom*, 2:594–602, March 1996.