Exploiting Semantic Proximity in Peer-to-Peer Content Searching

Spyros Voulgaris
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
spyros@cs.vu.nl

Anne-Marie Kermarrec
Microsoft Research
Cambridge, UK
annemk@microsoft.com

Laurent Massoulié
Microsoft Research
Cambridge, UK
lmassoul@microsoft.com

Maarten van Steen
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
steen@cs.vu.nl

Abstract

A lot of recent work has dealt with improving performance of content searching in peer-to-peer file sharing systems. In this paper we attack this problem by modifying the overlay topology describing the peer relations in the system. More precisely, we create a semantic overlay, linking nodes that are "semantically close", by which we mean that they are interested in similar documents. This semantic overlay provides the primary search mechanism, while the initial peer-to-peer system provides the fail-over search mechanism. We focus on implicit approaches for discovering semantic proximity. We evaluate and compare three candidate methods, and review open questions.

1. Introduction

The tremendous success of file sharing Peer-to-Peer systems such as Napster, Gnutella, E-Donkey and KaZaA, motivates the need to optimise the operation of such systems. One aspect which has recently received a lot of attention from the research community is the performance of content search mechanisms [11, 2, 6].

More specifically, Cohen and Shenker [6] identify document replication strategies that minimise the number of nodes queried before a document is found, assuming requests are submitted to random target nodes. Lv et al. [11] compare different search strategies for propagating queries throughout the system, namely flooding, "expanding ring" or gradual flooding, and random walks. They find that random walks perform better in minimising the number of queried nodes. Chawathe et al. [2] consider adapting the overlay topology so as to favour downloads from nodes with a high capacity connection to the network.

All three papers deal with search performance improvements which hold independently of any semantic structure, either in the document collections or in the successive searches made by individual nodes. In contrast, our aim in this paper is to exploit semantic structure present in document sharing systems in order to improve search performance.

This semantic structure can be used in several ways. One approach pursued by Crespo and Garcia-Molina [7] consists in explicitly identifying distinct semantic groups of documents, and building corresponding, possibly overlapping overlay networks for each group. A document request is then handled by the overlay to which this document presumably belongs. Unfortunately, classifying content turns out to be a difficult problem in practice, often requiring extensive manual intervention [7, 9].

Sripanidkulchai et al. [12] take an alternative approach, attempting to cluster nodes sharing similar interests, rather than similar documents. Another important difference is that in [12], clustering is performed in an implicit manner, i.e. without requiring the explicit identification of distinct groups of users. Nodes then try to obtain documents from their "semantic neighbours" first, before turning to other nodes.

The experimental results in [12] suggest that simple, light-weight techniques exploiting semantic structure in an implicit manner may yield significant gains in search performance. Our aim in the present paper is to push this statement further, and identify good candidate methods.

We consider the same architecture as in [12], namely each node maintains a list of semantic neighbours to which queries are submitted first, before turning to a default search mechanism if no semantic neighbour could answer the query. In order to investigate the effectiveness of different strategies for maintaining such semantic neighbours' lists,

Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04)
0-7695-2118-5/04 $20.00 © 2004 IEEE

IEEE COMPUTER SOCIETY

we develop a synthetic model of semantic structure linking nodes and documents in the system (Section 2). Section 3 describes our different strategies for maintaining lists of semantic neighbours. Section 4 discusses the performance evaluation results. In particular, a "list contamination" phenomenon is identified, and it is shown how the so-called POPULARITY strategy alleviates it. Before concluding in Section 6, we review in Section 5 three stimulating issues raised by the present work.

2. Modelling semantic structure

We now describe a simple synthetic model of request generation for documents by users[1]. This model features a semantic structure, which in turn induces locality of interest, a property observed in real data traces in [12].

We assume the existence of a number of semantic types, that are labelled by $n \in \{1, \ldots, N\}$, N denoting the number of such types. We assume that all documents in the system have an associated type. We let d_n denote the number of documents of type n. We also assume that users in the system can be associated to a semantic type. The number of users associated with type n is denoted by u_n.

Each user periodically generates a request. The target document satisfying that request is determined at random, according to a distribution which depends on the user's type only. This distribution is specified by the following two components: the probability $p_n(m)$ that a request generated by a type n-user will be targetted at a type m-document, and the probability $q_m(k)$ that a request for a document within class m will actually target the k-th document in that class.

Zipf's law is a natural candidate for the distribution $q_m(k)$ (see e.g. [10] for an extensive list of references on Zipf's law and its relevance to a number of fields, including document popularity) and we thus take $q_m(k)$ to be proportional to $1/k$. We also take the number of documents in class n, d_n, to be proportional to $1/n$ (within rounding error), thus imposing a Zipf's distribution on the class to which a document belongs.

The most critical component of our request model lies in the specification of the parameters $p_n(m)$. Our choice has been to select $p_n(m)$ so as to meet the following objectives:

(i) Ensure that the frequency of queries to documents from the whole collection (i.e., without class partition) also follows a Zipf's distribution.

(ii) Ensure that for each document class n, a proportion α of requests to these documents originates from class n-users, while other user classes all generate an equal proportion $(1-\alpha)/(N-1)$ of such requests.

An intuitive description of requirement (ii) above is that users of a given type will either place a query on a document in their preferred class, or place a general purpose, indifferentiated query on the global collection. The parameter α characterizes how strong is the specific interest of users for documents in their corresponding class: when $\alpha = 1/N$, users of all classes behave indifferently, while at the other extreme $\alpha = 1$, users of class n access only documents of class n. Thus α is naturally interpreted as measuring locality of interest in the system.

A specific choice which meets requirements (i) and (ii) is as follows:

$$\begin{cases} p_n(n) = Z_n^{-1}\,(\alpha/n)\sum_{k=1}^{d_n} k^{-1}, \\ p_n(m) = Z_n^{-1}\,\frac{1-\alpha}{m(N-1)}\sum_{k=1}^{d_m} k^{-1}, \quad m \neq n, \\ u_n \approx L\left[\alpha/n\sum_{k=1}^{d_n} k^{-1} + \sum_{m\neq n}\frac{1-\alpha}{m(N-1)}\sum_{k=1}^{d_m} k^{-1}\right], \end{cases}$$

where the parameter L controls the total number of users, and Z_n is a normalising constant chosen so that $\sum_m p_n(m)$ equals 1.

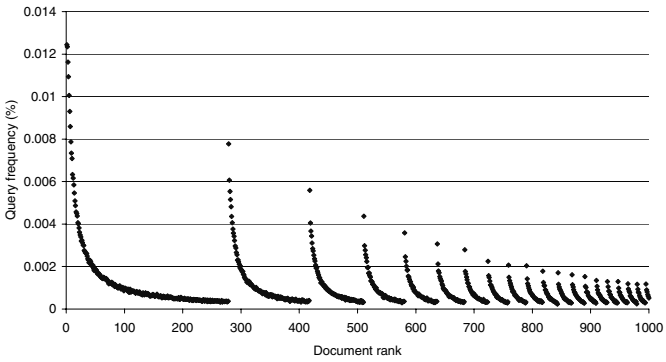

Figure 1. Document query frequency per class

Figure 1 shows the request frequencies per document, where documents are sorted by type. This distribution does not depend on the parameter α.

We are aware of the limitations of this model, and in particular of the fact that it captures only "pure" specific interests of users, whereas in reality one certainly expects users to have several specific interests. However we think it constitutes an adequate starting point, capturing in a parsimonious manner a number of important features.

3. Semantic overlay management

3.1. System model

Each user starts new document searches periodically, and the requests are generated according to the model of the previous section. We stress that neither user nor document types are exposed to the system. We assume that all

1 In the sequel we use the terms user and node interchangeably.

IEEE COMPUTER SOCIETY

users have a document cache of the same size, managed according to the Least Recently Used (LRU) policy[2]. We assume the availability of some peer-to-peer system supporting semantic–unaware searches. In our experiments we used expanding-ring search on an unstructured overlay built using the SCAMP protocol [8]. In addition, each user maintains a list of semantic neighbours. The basic search algorithm then consists in nodes first sending document requests to all semantic neighbours in their list[3]. If no semantic neighbour is able to serve the query, the semantic–unaware search algorithm is used.

3.2. Candidate strategies

All three strategies described below rely on the following basic feedback. After a node has placed a request, it is provided with a set of nodes possessing the requested document, either via the initial phase involving semantic neighbours only, or via the second, semantic–unaware phase.

LRU The most natural strategy consists in placing to the head of the semantic list that node from which the document has been obtained. As a result the semantic list contains the most recently used nodes, and evicts the least recently used one when the semantic search fails. This is essentially the method used in [12]. In our simulations, the document is downloaded from the first node to provide a positive answer. In a real-world scenario the choice could be made based on other factors (i.e. bandwidth, traffic, node load, etc.) so as to minimise download delays.

History As we shall see in the next section, semantic links created using LRU are mostly of two kinds, namely links to nodes of the same class as the requesting node, created upon querying a document of the corresponding type, and links to nodes who provided the requesting node with a popular document. The HISTORY approach aims at maintaining semantic links mostly of the first kind, as the latter are less useful. It requires a node to maintain a counter for each other node in the system. A node, say i, increments the counter maintained for another node, say j, whenever node i requests a document that node j could provide. The semantic list then consists of those nodes with the higher counter value.

This method has a number of drawbacks. It is expensive in terms of storage and of number of messages exchanged in order to update counter values. In addition, in an environment where user types may change over time, counters may take a long time to adapt to the new user preferences. For these reasons, it may not constitute a practical solution. However, it provides us with a benchmark for ideal performance, as it is very effective in creating semantic lists populated by nodes of the same semantic type only.

Popularity We now describe an approach which, like HISTORY, aims to create semantic lists populated mostly with nodes of the same type, while retaining the simplicity of LRU. This is based on the observation that document popularity can be inferred from the results of searches. In the expanding ring searches we are using for the failover mechanism, the number of nodes answering positively a query constitutes an estimate of the replication ratio of the corresponding document, which in turn reflects the popularity of that document.

The method works as follows. The entries in the semantic list have two additional fields: *numrep* and *lastreply*. *Numrep* gives the number of positive replies that were obtained for the document request for which this node entered the semantic list. *Lastreply* gives the last time the node offered to provide a requested document. *Lastreply* is used as a lease, when the gap beetwen *lastreply* and the current time is greater than a pre-defined threshold, the lease of the associated node is considered as expired. When a new request is submitted to the semantic neighbours, the *lastreply* entries of those who can answer the query are set to the current time. If none of them is able to serve the query, this is handled by the failover mechanism. In the latter case, let k denote the corresponding number of answers thus obtained. The node among those k from which the document is obtained will enter the semantic list of the requesting node, with *numrep* and *lastreply* set to k and the current time respectively, in the following circumstances:

(i) The semantic list is not fully populated yet;

(ii) Some node leases have expired. In that case, the node with the smallest *lastreply* is evicted from the list to make room for the new one;

(iii) The last two conditions are not met, but the largest *numrep* in the current list is larger than (or equal to) k, in which case the node with the largest *numrep* and the smallest *lastreply* is evicted from the list to make room for the new one.

4. Performance evaluation

Experimental setup Simulations, based on a discrete-event simulator, proceed as follows. First, a Scamp-based overlay is created. We then run a warm-up phase, during which requests are generated and both document caches and semantic lists are populated. The results we report are gathered during a subsequent phase of searches. Unless explicitly stated otherwise, all the results below are obtained for a system with 2000 nodes, 1000 documents, 20 semantic types, and a parameter α set to 0.8.

2 As a result, the degree of replication of a document should be roughly proportional to its popularity, that is the overall frequency at which it is requested; see e.g. [6].

3 Alternatively, semantic neighbours could be contacted sequentially until an answer is obtained.

IEEE COMPUTER SOCIETY

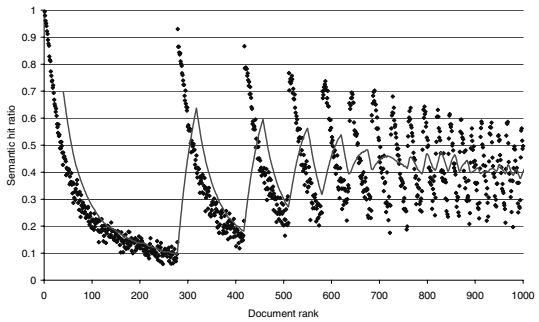

Figure 2. LRU semantic hit ratio

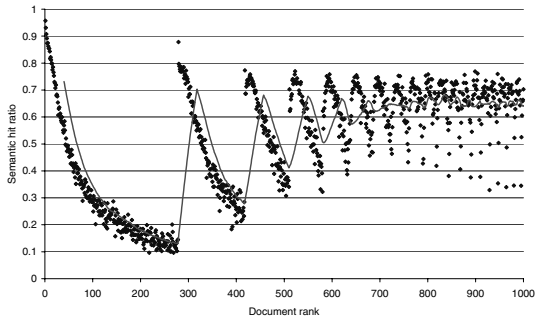

Figure 3. HISTORY semantic hit ratio

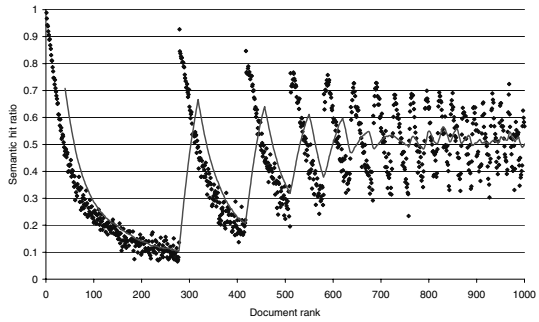

Figure 4. POPULARITY semantic hit ratio

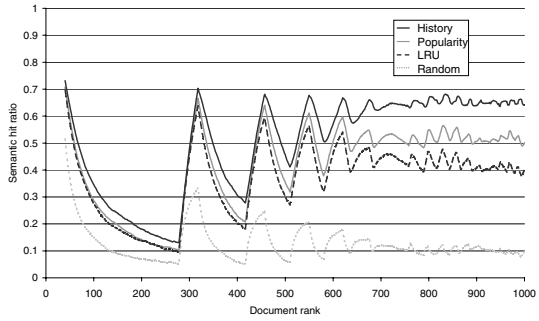

Figure 5. Summary

On each node, the cache size is set to 20 documents and the semantic list size is limited to 10 neighbours. The three candidate strategies are compared along two metrics: the semantic hit ratio *i.e.* the hit ratio obtained using only the semantic links and the *quality* of the semantic links, *i.e* the proportion of semantic links between nodes belonging to the same type.

Semantic hit ratio The distributions of semantic hit ratios per document for LRU, HISTORY and POPULARITY are depicted on Figures 2, 3 and 4 respectively. For the sake of comparison, we also ran experiments with semantic lists populated by nodes chosen uniformly at random from the whole population. Moving averages, averaged over 40 documents, of hit ratios for such random lists, as well as LRU, HISTORY and POPULARITY are reported on Figure 5. The semantic hit ratio averages for LRU, HISTORY, POPULARITY and Random are 40%, 65%, 50% and 10% respectively. Note that these are unweighted averages, taken over the whole document collection. Such values are indicative of the relative reduction in numbers of queries performed by the failover search mechanism. Indeed, while the number of times a document is requested is proportional to its popularity, for our replication strategy the number of failover queries before a document is located is inversely proportional to its popularity [11], and the two biases cancel out.

We also ran experiments with α set to $1/20$, which amounts to making all node types equivalent, and thus removes any locality of interest from the corresponding semantic structure. The corresponding hit ratios were very close to those resulting from random neighbours, as one would expect.

All three strategies LRU, HISTORY and POPULARITY were able to exploit locality of interest when it was present, and thus improve upon fail-over search mechanisms. We also note that for all three strategies, the improvement upon random lists was more pronounced for documents from the less popular types. This might be explained by the fact that there are fewer documents of such types, and thus the likelihood of finding them in the document caches of nodes with the same semantic type is increased. the POPULARITY strategy achieves a good trade-off between the simplicity of LRU and the performance of HISTORY.

Semantic link quality The semantic link quality averages per node type and per strategy are plotted on Figure 6. Recall that we define the quality of a semantic link to be 1 if it is between nodes of the same type, and zero otherwise. As previously mentioned, we note that LRU semantic links are contaminated by peers servicing very popular documents. The HISTORY strategy captures much more accurately the semantic similarities between nodes, and clusters almost ex-

IEEE COMPUTER SOCIETY

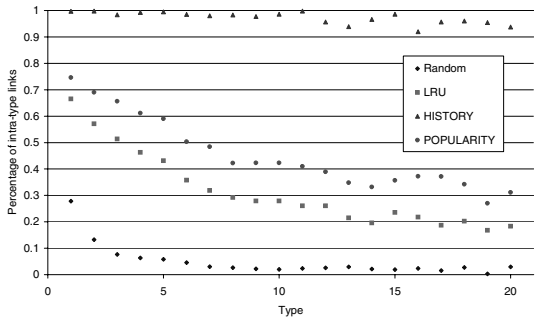

Figure 6. Semantic links quality

clusively nodes belonging to the same class. The POPULARITY strategy improves the links quality over LRU at almost no extra cost.

This data provides additional insight into the hit ratio results we just described. The poor semantic ratio for unpopular documents of popular classes despite a good clustering of nodes for the popular classes is due to two factors. First, the replication of these documents remains low as they are hardly queried. Second, the number of semantic neighbours is small compared to the size of popular classes. Semantic search in such classes could be improved by increasing the radius of the semantic search, rather than using the semantic overlay to query one-hop remote neighbours only. How to set this radius constitutes an interesting problem, to which we have no satisfactory answer yet.

5. Open issues

Semantic list ordering In the context of our request model, we found that the POPULARITY algorithm managed lists of semantic neighbours efficiently. A challenging issue consists in finding algorithms for managing such lists which perform well in arbitrary environments.

Such an arbitrary environment can be described by the following model, which is essentially equivalent to the model proposed by Cohen et al. [5, 4]. A given node may place queries (for documents) to any one of n other nodes, labelled by $i \in \{1, \ldots, n\}$. For each document search, there exists a corresponding subset of such nodes which may answer the query. It is assumed that the subsets corresponding to successive searches are random, independent and identically distributed, drawn from an unknown distribution. The list ordering problem then consists in finding an ordering $\pi(1), \ldots, \pi(n)$ of the nodes, so as to support efficient search. Assuming we form a semantic list of a given size s with the first s nodes in the list, and as before forward queries to all of them, a natural objective consists in maximizing the probability that queries may be served by either of these nodes. Alternatively, we may place queries sequentially on the nodes in the list until an answer is obtained, in which case a natural objective consists in minimising the expected number of queries thus generated.

Cohen et al. [5] consider a GREEDY strategy, which consists in selecting $\pi(i)$ so that the probability that $\pi(i)$ answers a query, given it has not been answered by $\pi(1), \ldots, \pi(i-1)$, is maximised. They show that the probability of a query being answered by any of the first s nodes under GREEDY is always within $(1 - 1/e)$ of the optimal probability, while for for sequential search the expected number of queries performed under GREEDY is within a factor of $4e/(e-1)$ of the expected number of queries under the optimal ordering.

An important issue is then to design list adaptation strategies which achieve similar performance guarantees, while being more reactive and less expensive than the GREEDY strategy of [4]. In the case of sequential queries, the Move To Front rule, in which the first answering node is placed at the head of the list, achieves an expected number of queries within $\pi/2$ of the optimal, in the special case where for each query there is only a single node which can treat it (see [3]). However, when multiple nodes can answer queries, it can perform arbitrarily badly: we can exhibit instances where Move To Front has a cost of order $O(n)$ queries while the optimal cost is of order $O(1)$. A natural candidate would be to augment the Move To Front rule with a learning step for each request, consisting in placing an extra query to a randomly selected node, and moving it to the front of the list as well if it also answers the query.

Loose coordination of document caches So far we have only discussed maintenance of the list of pointers to other nodes. Search performance may actually be enhanced by modifying the document cache management. In the previous experiments we were applying LRU for such caches. However, if we assume that the semantic peering relationships will remain stable over a long period of time, we may modify the cache replacement policy to take advantage of this fact. More specifically, when a node uploads a document to one of its semantic neighbours, it therefore learns that the corresponding document is being replicated there, and is likely to be accessible from there at a later time. We suggest that such information could be used to improve coordination between document cache management at distinct nodes. For instance, the uploading node could move the uploaded document to the back of its document cache, effectively freeing cache space. This strategy, which may be described as joint Move To Front / Move To Back, may lead to further improvements on hit ratios. We are planning to investigate this in a quantitative manner.

Guided search Although our main goal is to exploit semantic properties of requests in a completely implicit manner, it is certainly true that explicit treatment of semantic information should allow performance improvements by guiding

IEEE COMPUTER SOCIETY

content search. We now describe a light-weight approach for doing this (see [13] for a related method).

When a node i obtains a document d from another node j, node i adds j to the front of its list of "semantic neighbours" and tags j's entry in this list with some semantic information describing d. Tags can be used afterwards to guide the search based on semantic similarities between tags and queries. When a node places a request for a document d', the query is forwarded to one of its neighbours, chosen with a probability inversely proportional to the semantic distance between the query and the corresponding tag (see [1] for a description of two semantic distances in common use in the information retrieval community). Queries get forwarded in the same manner until they are served. The question is then to evaluate the performance of such random walk searches biased by semantic hints. Another important problem is to understand in what circumstances such semantic hints are going to bring a significant improvement compared to the implicit methods described previously.

6. Conclusions

It has been observed that peer-to-peer file sharing traces exhibit locality of interest [12]. In this paper we investigate implicit techniques for clustering nodes with similar interests in order to improve content search in such file sharing systems. Each node, based on the history of query/responses creates a list of *semantic neighbours* to which queries are forwarded first before using as a failover the standard search algorithm.

We consider three candidate strategies for managing semantic lists and evaluate them in the context of a synthetic request model which captures locality of interest. We find that the lists generated using the LRU strategy can be *contaminated* by nodes having served popular documents, which have no common interest with the requesting nodes. We investigate HISTORY and POPULARITY strategies to fix this issue. Our evaluations suggest that the POPULARITY strategy provides a good trade-off between complexity and efficiency: it approaches the performance of the expensive HISTORY strategy while retaining the simplicity of the LRU mechanism. These conclusions are arrived at in the context of our specific request model, and we plan to experiment with real traces shortly in order to confirm, infirm or refine our observations. In particular, we will try to assess the impact on these results of dynamically changing user interests, document collections, and user populations.

As a debate is currently taking place in the research community about the relative merits of structured versus unstructured overlays for content searching [2], it is worth noting that the semantic list approach described here is equally applicable in both contexts, and irrespective of whether the search is based on random walks or flooding techniques. POPULARITY requires an estimate of a requested document's popularity. With flooding techniques, the number of answers to a query provides such an estimate; when random walks are used instead of flooding, the number of steps walked before the query can be served also provides an estimate of the (reciprocal of the) requested document popularity.

References

[1] A. Asvanund, S. Bagla, M. Kapadia, R. Krishnan, M. smith, and R. Telang. Intelligent club management in peer-to-peer networks. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[2] Y. Chawathe, S. Ratnasamy, L. Bresbau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM*, 2003.

[3] F. Chung, D. Hajela, and P. Seymour. Self-organizing sequential search and hilbert's inequality. In *17th Annual Symposium on the Theory of Computing*, 1985.

[4] E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer-to-peer networks: Harnessing latent semantics. In *INFOCOM*, 2003.

[5] E. Cohen, A. Fiat, and H. Kaplan. Effi cient sequences of trials. In *14th ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[6] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *SIGCOMM*, 2002.

[7] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. http://www-db.stanford.edu/ crespo/publications/, 2003.

[8] A. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2), February 2003.

[9] H. Kung and C. Wu. *Content Networks: Taxonomy and New Approaches*. Oxford university press, 2002.

[10] W. Li. Zipf's law. http://linkage.rockefeller.edu/ wli /zipf/.

[11] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th ACM International Conference on SuperComputing*, New York, USA, 2002.

[12] K. Sripanidkulchai, B. Maggs, and H. Zhang. Effi cient content location using interest-based locality in peer-to-peer systems. In *INFOCOM Conference*, 2003.

[13] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *SIGCOMM*, 2003.