Epidemic-Style Management of Semantic Overlays for Content-Based Searching

Spyros Voulgaris and Maarten van Steen

Vrije Universiteit Amsterdam
Department of Computer Science
De Boelelaan 1081a, 1081HV Amsterdam
{spyros,steen}@cs.vu.nl

Abstract. A lot of recent research on content-based P2P searching for file-sharing applications has focused on exploiting semantic relations between peers to facilitate searching. To the best of our knowledge, all methods proposed to date suggest *reactive* ways to seize peers' semantic relations. That is, they rely on the usage of the underlying search mechanism, and infer semantic relations based on the queries placed and the corresponding replies received. In this paper we follow a different approach, proposing a *proactive* method to build a semantic overlay. Our method is based on an epidemic protocol that clusters peers with similar content. It is worth noting that this peer clustering is done in a completely implicit way, that is, without requiring the user to specify his preferences or to characterize the content of files he shares.

1 Introduction

File sharing peer-to-peer (P2P) systems have gained enormous popularity in recent years. This has stimulated significant research activity in the area of content-based searching. Sparkled by the legal adventures of Napster, and challenged to defeat the inherent limitations concerning the scalability and failure resilience of centralized systems, research has focused on *decentralized* solutions for content-based searching, which by now has resulted in a wealth of proposals for peer-to-peer networks.

In this paper, we are interested in those group of networks in which searching is based on grouping semantically related nodes. In these networks, a node first queries its semantically close peers before resorting to search methods that span the entire network. In particular, we are interested in solutions where semantic relationships between nodes are captured implicitly. This capturing is generally achieved through analysis of query results, leading to the construction of a local *semantic list* at each peer, consisting of references to other, semantically close peers.

Only very recently, an extensive study has been published on search methods in peer-to-peer networks, be they structured, unstructured, or of a hybrid form [1]. This study reveals that virtually all peer-to-peer search methods in semantic overlay networks follow an integrated approach towards the construction of the semantic lists, while at the same time accounting for changes occurring in the set of nodes. These changes involve the joining and leaving of nodes, as well as changes in a node's preferences.

The problem we are faced with is that the construction of semantic lists should result in highly clustered overlay networks. These networks excel for searching content

J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, LNCS 3648, pp. 1143–1152, 2005.

when nothing changes. However, to handle dynamics requires the discovery and propagation of changes that may happen *anywhere* in the network. For this reason, overlay networks should also reflect desirable properties of random graphs and complex networks in general [2, 3]. These two conflicting demands generally lead to complexity when integrating solutions into a single protocol.

Protocols for content-based searching in peer-to-peer networks should separate these concerns. In particular, we advocate that when it comes to constructing and using semantic lists, these lists should be optimized for search only, regardless of any other desirable property of the resulting overlay. Instead, a separate protocol should be used to handle network dynamics, and provide up-to-date information that will allow proper adjustments in the semantic lists (and thus leading to adjustments in the semantic overlay network itself).

In this paper we propose such a two-layered approach for managing semantic overlay networks. The top layer contains a gossip-based protocol that strives to optimize semantic lists for searching only. The bottom layer offers a fully decentralized service for delivering, in an unbiased fashion, information on new events, similar in nature to the peer-sampling service recently described in [4]. Again, this service is implemented using a gossip-based protocol (which, by the way, is very different from those described in [4]).

Our main contribution is that we demonstrate that this two-layered approach leads to high-quality semantic overlay networks. We substantiate our claims through extensive simulations using traces collected from the eDonkey file-sharing network [5].

The paper is organized as follows. We start with presenting our protocols in the next section, followed by describing our experimental setup in Section 3. Performance evaluation is discussed in Section 4, followed by an analysis of consumed bandwidth in Section 5. We conclude with a discussion in Section 6.

2 The Protocol

2.1 Outline

In our model each peer maintains a dynamic list of semantic neighbors, called its *semantic view*, of fixed small size ℓ. A peer searches for a file by first querying its semantic neighbors. If no results are returned, the peer then resorts to the default search mechanism.

Our aim is to organize the semantic views so as to maximize the hit ratio of the first phase of the search. We will call this the *semantic hit ratio*. We anticipate that the probability of a neighbor satisfying a peer's query is proportional to the semantic proximity between the peer and its neighbor. We aim, therefore, at filling a peer's semantic view with its ℓ semantically closest peers out of the whole network.

We assume the existence of a *semantic proximity function* $S(F_P, F_Q)$, which given the file lists F_P and F_Q of peers P and Q, respectively, provides a numeric metric of the semantic proximity between the two peers. The more semantically similar the file lists of P and Q are, the higher the value of $S(F_P, F_Q)$. We are essentially seeking to pick peers $Q_1, Q_2, \ldots, Q_\ell$ for peer P's semantic view, such that the sum $\sum_{i=1}^{\ell} S(P, Q_i)$ is maximized.

We assume that the semantic proximity function exhibits some sort of transitivity, in the sense that if P and Q are semantically similar to each other, and so are Q and R, then some similarity between P and R is likely to hold. Note that this transitivity does not consist a hard requirement for our system. In its absence, semantically related neighbors are discovered based on random encounters. If it exists though, it is exploited to dramatically enhance efficiency.

2.2 Design Motivation

From our previous discussion, we are seeking a means to construct, for each node, a semantic view from all the current nodes in the system. There are two sides to this construction.

First, based on the assumption of transitivity in the semantic proximity function S, a peer should explore the semantically close peers that its neighbors have found. In other words, if Q is in P's semantic view, and R is in Q's view, it makes sense to check whether R is also semantically close to P. Exploiting the transitivity in S should then quickly lead to high-quality semantic views.

Second, it is important that *all* nodes are examined. The problem with following only transitivity is that we eventually will be searching only in a single semantic cluster. Similar to the special "long" links in small-world networks [6], we need to establish links to *other* semantically-related clusters. Likewise, when new nodes join the network, they should easily find an appropriate cluster to join. These issues call for a randomization when selecting nodes to inspect for adding to a semantic view.

In our design we decouple these two aspects by adopting a two-layered set of gossip protocols, as can be seen in Figure 1. The lower layer, called CYCLON [7], is responsible for maintaining a connected overlay and for periodically feeding the top-layer protocol with nodes uniform randomly selected from the network. In its turn, the top-layer protocol, called VICINITY, is in charge of focusing on discovering peers that are semantically as close as possible, and of adding these nodes to the semantic views.

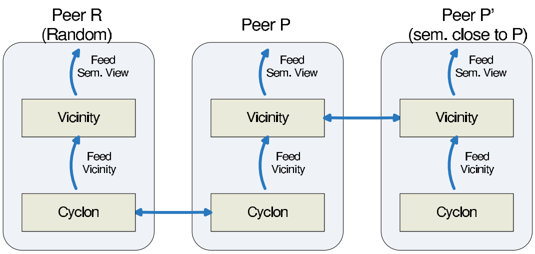

Fig. 1. The two-layered framework.

2.3 Gossiping Framework

All information exchange between peers is carried out by means of *gossip items*, or simply *items*. A gossip item created by peer P is a tuple containing the following three fields:

1. *P*'s contact information (network address and port)
2. The item's creation time
3. Application-specific data; in this case *P*'s file list

Each node maintains locally a number of items per protocol, called the protocol's *view*. This number is the same for all items, and is called the protocol's *view size* (c_v for VICINITY, and c_c for CYCLON).

Figure 2 presents a generic skeleton forming the basis for both VICINITY and CYCLON gossiping protocols. Each node runs two threads. An *active* one, which periodically wakes up and initiates communication to another peer, and a *passive* one, which responds to the communication initiated by another peer.

The functions appearing in boldface, namely `selectPeer()`, `selectItemsToSend()`, and `selectItemsToKeep()` form the three *hooks* of this skeleton. Different protocols can be instantiated from this skeleton by implementing specific policies for these three functions, in turn, leading to different emergent behaviors.

The number of items exchanged in each communication is predefined, and is called the protocol's *gossip length* (g_v for VICINITY, and g_c for CYCLON).

```
/*** Active thread ***/
// Runs periodically every T time units
q = selectPeer()
myItem = (myAddress, timeNow, myFileList)
buf_send = selectItemsToSend()
send buf_send to q
receive buf_recv from q
view = selectItemsToKeep()

/*** Passive thread ***/
// Runs when contacted by some peer
receive buf_recv from p
myItem = (myAddress, timeNow, myFileList)
buf_send = selectItemsToSend()
send buf_send to p
view = selectItemsToKeep()
```

Fig. 2. Epidemic protocol skeleton.

For VICINITY, we chose the policies shown in Figure 3(a). We note that the RANDOM protocol resembles T-Man [8]. The only difference is that in T-Man peers exchange their *whole* views, instead of just a subset of them. As we discuss below, AGGRESSIVELY BIASED will turn out to be an excellent choice for forming semantic clusters.

Note that `selectItemsToKeep()` takes into account CYCLON's cache too in selecting the best c_v items to keep. This is the default link between the two layers.

For CYCLON, we made the choices shown in Figure 3(b). CYCLON is a protocol we previously developed, and which is extensively described and analyzed in [7].

Effectively, what `selectItemsToSend()` and `selectItemsToKeep()` establish is an *exchange* of some neighbors between the caches of the two communicating

Hook	Description
`selectPeer()`	Select peer from the item with the oldest timestamp
`selectItemsToSend()`	
RANDOM	Randomly select g_v items
BIASED	Select the g_v items of nodes semantically closest to the selected peer
AGGRESSIVELY BIASED	Select the g_v items of nodes semantically closest to the selected peer from the VICINITY view *and* the CYCLON view
`selectItemsToKeep()`	Keep the c_v items of nodes that are semantically *closest*, out of items in its current view, items received, and items in the local CYCLON view. In case of multiple items from the same node, keep the one with the most recent timestamp.

(a)

Hook	Description
`selectPeer()`	Select peer from the item with the oldest timestamp
`selectItemsToSend()`:	
active thread	Select own item and randomly $g_c - 1$ others from the CYCLON view
passive thread	Randomly select g_c items from the CYCLON view
`selectItemsToKeep()`	Keep all g_c received items, replacing (if needed) the g_c ones selected to send. In case of multiple items from the same node, keep the one with the most recent timestamp.

(b)

Fig. 3. The chosen policies for (a) the VICINITY protocol and (b) the CYCLON protocol.

peers. In addition to that, the selected peer's item in the initiator's cache is always removed, but the initiator's (new) item is always placed in the selected peer's cache.

CYCLON creates an overlay with completely random, uncorrelated links between nodes, such that the in-degree (number of incoming links) is practically the same for each node. Importantly, it can achieve this property fairly quickly even when a small number of items (such as 3 or 4) is exchanged in each communication, even for large caches of several dozens of items. Therefore, it is ideal as a lightweight service that can offer a node a randomly selected peer from the current set of nodes.

3 Experimental Environment and Settings

All experiments presented here have been carried out with PeerSim [9], an open source simulator in Java for P2P protocols, developed at the University of Bologna.

To evaluate our protocol, we used real world traces from the eDonkey file sharing system [10], collected by Le Fessant et al. in November 2003 [5]. A set of 12,000 worldwide distributed peers along with the files each one shares is logged in these traces. A total number of 923,000 unique files is being collectively shared by these peers.

In order to simplify the analysis of our system's emergent behavior, we determined equal gossiping periods for both layers. More specifically, once every T time units each node initiates first a gossip exchange with respect to its bottom (CYCLON) layer, immediately followed by a gossip exchange at its top (VICINITY) layer. Note that even though nodes initiate gossiping at universally fixed intervals, they are not synchronized with each other.

Even though both protocols are asynchronous, it is convenient to introduce the notion of *cycles* in order to study their evolutionary behavior with respect to time. We define a cycle to be the time period during which *each* node has initiated gossiping exactly *once*. Since each node initiates gossiping periodically, once every T time units, a cycle is equal to T time units.

A number of parameters had to be set for these experiments, listed here.

Proximity Function S**.** We chose a rather simple, yet intuitive proximity function to test our protocol with. The proximity S between two nodes P and Q, with file lists F_P and F_Q respectively, is defined as the number of files that lay in both lists. More formally: $S(F_P, F_Q) = |F_P \bigcap F_Q|$. As stated in 2.1, the semantically closer two nodes are, the higher the value of S is. Note that our goal was to demonstrate the power of our gossiping protocol in forming a semantic network based on *a* proximity function. Even though much richer proximity functions could have been applied, it was out of the scope of this paper.

Semantic View Size ℓ**.** In all experiments the semantic view consisted of the 10 semantically closest peers in the VICINITY cache. As shown in [11], a semantic view size of $\ell = 10$ provides a good tradeoff between the number of nodes contacted in the semantic search phase and the expected semantic hit ratio.

Cache size. For the cache size selection, we are faced with the following tradeoff for both protocols. A large cache size provides higher chances of making better item selections, and therefore accelerate the construction of (near-)optimal semantic views. On the other hand, the larger the cache size, the longer it takes to contact all peers in it, resulting in the existence of older—and therefore more likely to be invalid—links. Of course, a larger cache also takes up more memory, although this is generally not a significant constraint nowadays.

Considering this tradeoff, and after a set of experiments that cannot be presented due to space limitations, we fixed the cache size to 100 as a basis to compare different configurations. When both Vicinity and Cyclon are used, they are allocated 50 cache entries each.

Gossip length. The gossip length, that is, the number of items gossiped per gossip exchange per protocol, is a crucial factor for the amount of bandwidth used. This becomes of greater consequence, considering that an item carries the file list of its respective node. So, even though exchanging more items per gossip exchange allows information to disseminate faster, we are inclined to keep the gossip lengths as low as possible, as long as the system's performance is reasonable.

Again, for the sake of comparison, we fixed the total gossip length to 6 items. When both Vicinity and Cyclon are used, each one is assigned a gossip length of 3.

Gossip period T**.** The gossip period is a parameter that does not affect the protocol's behavior. The protocol evolves as a function of the number of messages exchanged, or, consequently, of the number of cycles elapsed. The gossip period only affects how fast the protocol's evolution will take place in time. The single constraint is that the gossip period T should be adequately longer than the worse latency throughout the network, so that gossip exchanges are not favored or hindered due to latency heterogeneity. A typical gossip period for our protocol would be 1 minute, even though this does not affect the following analysis.

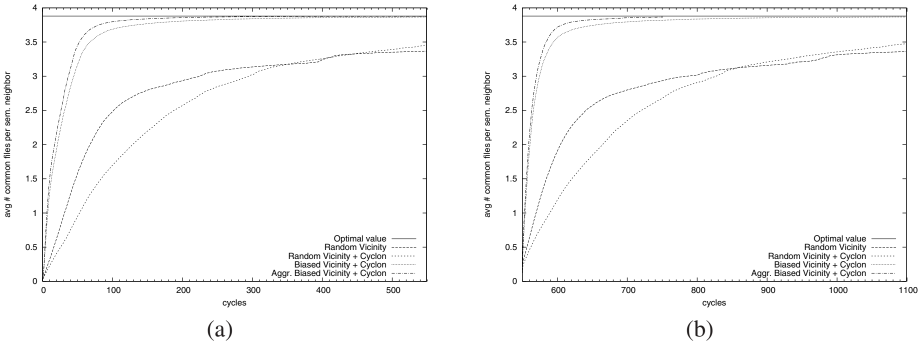

Fig. 4. (a) Convergence of sem. views' quality. (b) Evolution of semantic views' quality for a sudden change in all users' interests at cycle 550.

4 Performance Evaluation

4.1 Convergence Speed

To evaluate the convergence speed of our algorithm, we test how quickly it finds nodes having files in common. The proximity function's objective is for each node to discover the ℓ peers that have the most common files with it. Therefore, a good metric of the progress towards this goal is the average number of common files between a node and each one of its semantic neighbors. From our traces, we measured that in the optimal organization, this metric has a value of 3.88.

Figure 4(a) shows this metric as a function of the cycle for four distinct configurations. In favor of comparison fairness, the cache size and gossip length are 50 and 3, respectively, in each layer, for all configurations. The only exception is the first configuration, which has a single layer. In this case, the cache size and gossip length are 100 and 6, respectively. All experiments start with each node knowing 5 random other ones, simply to ensure initial connectivity in a single connected cluster.

In the first configuration, RANDOM VICINITY is running stand-alone. The progress of the semantic views' quality is rather steep in the first 100 cycles, but as nodes gradually concentrate on their very own neighborhood, getting to know new, possibly better peers becomes rare, and progress slows down.

In the second configuration, a two-layered approach consisting of RANDOM VICINITY and CYCLON is running. The slow start compared to stand-alone VICINITY is a reflection of the smaller VICINITY cache (3 as opposed to 6). However, the two-layered approach's advantage becomes apparent later, when CYCLON keeps feeding the RANDOM VICINITY layer with new, uniform randomly selected nodes, maintaining a higher progress rate, and outperforming stand-alone VICINITY in the long run.

In the third configuration, BIASED VICINITY demonstrates its contribution, as progress is significantly faster in the initial phase of the experiment. This is to be expected, since the items sent over in each BIASED VICINITY communication, are the ones that have been selected as the semantically closest to the recipient.

Finally, in the fourth configuration, AGGRESSIVELY BIASED VICINITY keeps the progress rate high even when the semantic views are very close to their optimal state. This is due to the broad random sampling achieved by this version. In every communication, a node is exposed to the best peers out of 50 *random* ones, in addition to 50 peers from its neighbor. In this way, semantically related peers that belong to separate semantic clusters quickly discover each other, and subsequently the two clans merge into a single cluster in practically no time.

4.2 Adaptivity to Changes of User Interests

In order to test our protocol's adaptivity to dynamic user interests, we ran experiments where the interests of some users changed. We simulated the interest change by picking a random pair of nodes and swapping their file lists in the middle of the experiment. At that point, these two nodes found themselves with semantic views unrelated to their (new) file lists, and therefore had to gradually climb their way up to their new semantic vicinity, and replace their useless links by new, useful ones.

Once again, we present the worst case —practically unrealistic— scenario, of *all* nodes changing interests at once, at cycle 550 of the experiment of figure 4(a). The evolution of the quality of the semantic views (using the metric introduced in 4.1) after the moment when all nodes change interests, is presented in figure 4(b). The faster convergence compared to figure 4(a) is due to the fact that views are already fully filled up at cycle 550, so nodes have more choices to start looking for good candidate neighbors.

Even though this scenario is very unrealistic, it demonstrates the power of our protocol in adapting to even massive scale changes. This adaptiveness is due to the priority given to newer items in `selectItemsToKeep()`, which allows a node's items with updated semantic information to replace older items of that node fast.

4.3 Effect on Semantic Hit Ratio

In order to further substantiate our claim that semantic based clustering endorses P2P searching, we conducted the following experiments. A randomly selected file was removed from *each* node, and the system was run considering proximity based on the remaining files. Then, each node did a search on that special file. We measured the semantic hit ratio to be over 36% for a semantic view of size 10.

Figure 5 presents the semantic hit ratio as a function of the cycle. Three experiments are shown, with gossip lengths for *both* layers set to 1, 3, and 5. Note that the hit ratio was autonomously computed in each cycle, without affecting the mainstream experiment's state.

5 Bandwidth Considerations

Due to the periodic behavior of gossiping, the price of having rapidly converging protocols may inhibit a high usage of network resources (i.e., bandwidth).

In each cycle, a node gossips on average twice (exactly once as an initiator, and on average once as a responder). In each gossip $2 \cdot (g_v + g_c)$ items are transferred to and from the node, resulting in a total traffic of $4 \cdot (g_v + g_c)$ items for a node per cycle. An item's size is dominated by the file list it carries. A single file is identified by its 128-bit (16-byte) MD4 hash value. Analysis of the eDonkey traces [5] revealed an average

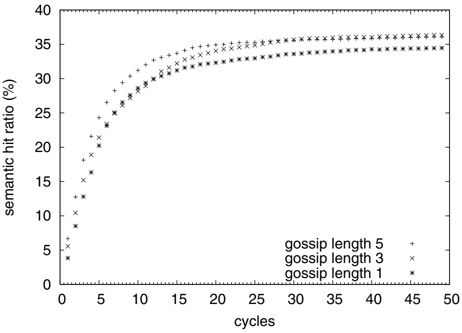

Fig. 5. Semantic Hit Ratio, for gossip lengths 1, 3, and 5 in each layer.

number of 100 files per node (more accurately, 99.35). Therefore, a node's file list takes on average 1,600 bytes. So, in each cycle, the total number of bytes transferred *to* and *from* the node is $6,400 \cdot (g_v + g_c)$.

For $g_v = g_c = 3$, the average amount of data transferred to and from a node in one cycle is 38,400 bytes, while for $g_v = g_c = 1$, it is just 12,800. Considering the gossip period T equal to 1 minute, this translates to an average bandwidth of 640 and 213 bytes per second, respectively. With $g_v = g_c = 3$ the system adapts a little faster to changes, but if bandwidth is of high concern, $g_v = g_c = 1$ can also provide very good results. Note that with a period of 1 minute, in the first 8 minutes we reach 85% of the optimal semantic hit ratio, having roughly 30% of all requests handled by the semantic neighbors.

We consider such a bandwidth consumption to be rather small, if not negligible compared to the bandwidth used for the actual file downloads. It is, in fact, a small price to pay for relieving the default search mechanism from about 35% of the search load.

6 Discussion

To the best of our knowledge, all earlier work on implicit building of semantic overlays relies on using heuristics to decide *which* of the peers that served a node recently are likely to be useful again in future queries [11–13].

However, all these techniques inhibit a weakness that challenges their applicability to the real world. They all assume a *static* network, free of node departures, which is a rather strong assumption considering the highly dynamic nature of file-sharing communities. Also, it is not clear how they perform in the presence of dynamic user preferences.

Regarding proximity-based P2P clustering, our work comes close to T-Man[8]. However, a key difference is that T-Man assumes *continuous* proximity metrics. That is, every node can point *any* other node to the right direction. This is not true in the problem we faced, i.e. in the case of completely unrelated peers. We dealt with it by harnessing CYCLON's randomness. This renders our solution more generic. Moreover, T-Man assumes a preconstructed almost random graph to start with. We make no such assumptions.

Another key difference is that T-Man aims at fixing an overlay's links to the optimal ones, that is, the ones that minimize a given energy function. Our work aims at continuously exchanging links, so that the optimal ones become known relatively soon to each node, but do not remain static links of this node.

Concluding, in this paper we introduced the idea of applying epidemics to build and dynamically maintain semantic lists in a large-scale file-sharing system. Specifically, we showed that using a two-layered approach combining two epidemic protocols is the appropriate way to build such a service. Finally, we presented a fast converging, highly adaptable, yet lightweight epidemic-style solution to this problem.

Acknowledgements

We would like to specifically thank Fabrice Le Fessant for providing us with the eDonkey2000 traces[5] he gathered in November 2003.

References

1. J. Risson and T. Moors. Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. Technical Report UNSW-EE-P2P-1-1, University of New South Wales, Sydney, Australia, September 2004.
2. Reka Albert and Albert-Laszlo Barabasi. Statistical Mechanics of Complex Networks. *Reviews of Modern Physics*, 74(1):47–97, January 2001.
3. M.E.J. Newman. Random Graphs as Models of Networks. In S. Bornholdt and H. G. Schuster, editors, *Handbook of Graphs and Networks: From the Genome to the Internet*, chapter 2. John Wiley, New York, NY, 2002.
4. M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations. In *Middleware 2004*, volume 3231 of *Lect. Notes Comp. Sc.*, Berlin, October 2004. ACM/IFIP/USENIX, Springer-Verlag.
5. Fabrice Le Fessant, S. Handurukande, Anne-Marie Kermarrec, and Laurent Massoulié. Clustering in peer-to-peer file sharing workloads. In *3rd International Workshop on Peer-to-Peer Systems (IPTPS)*, San Diego, USA, February 2004.
6. Duncan J. Watts. *Small Worlds, The Dynamics of Networks between Order and Randomness*. Princeton University Press, Princeton, NJ, 1999.
7. Spyros Voulgaris, Daniela Gavidia, and Maarten van Steen. Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, To appear in 2005.
8. Márk Jelasity and Ozalp Babaoglu. T-Man: Fast gossip-based construction of large-scale overlay topologies. Technical Report UBLCS-2004-7, University of Bologna, Department of Computer Science, Bologna, Italy, May 2004.
9. Peersim. http://peersim.sourceforge.net/.
10. edonkey. http://www.edonkey2000.com.
11. S. Handurukande, A.-M. Kermarrec, F. Le Fessant, and L. Massoulié. Exploiting semantic clustering in the edonkey p2p network. In *11th ACM SIGOPS European Workshop (SIGOPS)*, Leuven, Belgium, September 2004.
12. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM Conference*, 2003.
13. S. Voulgaris, A. Kermarrec, L. Massoulié, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004)*, Suzhu, China, November 2001.