

Exploiting Semantic Clustering in the eDonkey P2P Network

S. B. Handurukande[†], A.-M. Kermarrec[‡], F. Le Fessant^{*} & L. Massoulié^{*}

[†]Distributed Programming Laboratory, EPFL, Switzerland

[‡]INRIA, Rennes, France

^{*} INRIA-Futurs and Lix, Palaiseau, France

^{*} Microsoft Research, Cambridge, UK

Abstract

Peer-to-peer file sharing now represents a significant portion of the Internet traffic and has generated a lot of interest from the research community. Some recent measurements studies of peer-to-peer workloads have demonstrated the presence of semantic proximity between peers. One way to improve performance of peer-to-peer file sharing systems is to exploit this locality of interest in order to connect semantically related peers so as to improve the search both in flooding- and server-based systems. Creating these additional connections raises interesting challenges and in particular (i) how to capture the semantic relationship between peers (ii) how to exploit these relationships and (iii) how to evaluate these improvements. In this paper, we evaluate several strategies to exploit the semantic proximity between peers against a real trace collected in November 2003 in the eDonkey 2000 peer-to-peer network. We present the results of this evaluation which confirm the presence of clustering in such networks and the interest to exploit it.

1 Introduction and background

File sharing peer-to-peer systems have significantly grown the past decade to the extent that they are now the highest consumers of Internet bandwidth [11, 14, 16]. As expected, this has generated a lot of interest from the research community to understand these applications, measure their workloads, propose new architectures [15, 20] and improve existing ones [7].

Peer-to-peer file sharing systems usually either rely on flat overlays such as Gnutella [3] in which the primary search algorithm is based on flooding, or hierarchical overlays composed of a set of servers indexing peers contents and in charge of

redirecting requests. Examples of such hierarchical systems are KaZaA [4] and eDonkey [1]. Besides, number of research works have been done to improve the search mechanism by optimizing replication strategies [13] or switching from flooding to random walks [7]. Recent works [6, 12] propose to build Gnutella-like systems in combination with structured peer-to-peer overlays.

More recently, another class of research aims at capturing and exploiting semantic proximity or interest-based proximity to improve the search in peer-to-peer file sharing systems. Semantic proximity between peers is defined as the similarities between their cache contents or download patterns. The idea behind these works is that semantically related peers are more likely to be useful to each other (e.g., than random peers). Therefore, it might be useful to connect semantically related peers in the overlay. These peers, called semantic neighbours in the rest of this paper, are solicited first in the search process because the probability that they are able to satisfy a request might be quite high. In other words, in Gnutella style search schemes, the semantic neighbours are queried first; if this first phase fails the normal gossip style phase is performed. In hierarchical schemes like in KaZaA, the first phase can be used to bypass the servers (aka super peers) thus alleviating their load.

There are several ways of capturing the semantic relationship between peers. One approach is to explicitly identify distinct semantic groups of documents, using a predefined classification and to build separate overlays for each separate semantic relationship [8]. Precise classification is always a difficult task and the permanent classification may lead to a static configuration which cannot adapt to changes in peers preferences or recover from a wrong classification. At the other end of the spectrum, in [19], semantic shortcuts are dynamically created to link peers sharing some interests. The semantic relationship between peers is captured

implicitly based on the most recent downloads. In [21], alternative ways of creating shortcuts are proposed and evaluated in the context of a synthetic workload.

In a recent paper [9], we presented a clustering analysis of peer-to-peer file sharing traces that we obtained crawling the eDonkey network during a three day period. This preliminary study demonstrated that there is a significant overlap between peers contents that one can consider as semantic proximity between peers. In this paper, we evaluate several strategies using *implicit* or *explicit* semantic relationships. We evaluate these approaches using a trace obtained in November 2003 on the eDonkey 2000 peer-to-peer network.

The rest of this paper is organized as follows; in Section 2, we briefly present the experimental setting including the description of the trace; in Section 3, we present the evaluation of three simple strategies (i) implicit semantic; (ii) semantic overlap, and (iii) light explicit semantic, before concluding in Section 4.

2 Experimental setting

2.1 Peer-to-peer workload

One of the main challenges to validate improvements in peer-to-peer systems is to be able to use realistic input data. In the last few years some measurements studies have been carried out [17, 18, 5, 10] mainly concerned by peers requests, connectivity and availability. Therefore, the associated traces did not include content of the peers (the offered files to the other peers) that we need to carry out our evaluations.

In a recent study [9], we collected and analyzed a peer-to-peer file sharing application trace, focusing on the clustering properties of the peers. To this end, we actively probed a community of the eDonkey 2000 clients, the main competitor of KaZaA, recently referred as ahead of KaZaA in Europe [22]. We obtained a trace of 12,000 clients, sharing 923,000 documents, distributed worldwide with a majority in Europe. We obtained a trace containing for each client, its list of cache contents. More details about the trace are available in [9]. In our analysis, we measured files popularity as the number of replicas over the peers' caches. An interesting observation is that the popularity distribution of the files is similar whether it is related to the number of requests [10] or the number of replicas. Besides, the study presented in [10] concluded on a *fetch once behavior*: peers tend to download files only once and therefore do not exhibit temporal

locality in their access patterns.

Given this combination of observations, we concluded that a list of cache contents can reasonably be used as a list of requests as explained below.

2.2 Simulation environment

We wrote a simple discrete-event simulator composed of n nodes to simulate the file exchange between peers and search only using semantic neighbours in a peer-to-peer file sharing system¹. We assigned nodes randomly to the eDonkey clients of the trace. The simulator maintains the global list of the files shared in the system. Each client is associated with a list of files according to the real trace. The simulation consists in, for each client, requesting sequentially each file of its list. Two situations may occur when a node n requests a file f : (i) either f has already been requested in the system and is replicated in other peers caches. In that situation, n downloads f from one of those peers and updates its semantic neighbour list accordingly (as we will expand in Section 3). If more than one peer has the file, all those peers are taken into account when updating the list. (ii) n is the first peer requesting f and we then consider that n maintains the initial replica of f .

3 Simulating semantic links

3.1 Implicit semantic neighbours

In the first set of experiments, we evaluated two of the strategies described in [21], namely LRU and History. To this end, we computed the hit ratio due to the use of semantic neighbours as opposed to random peers. We implemented both the History and LRU strategies. In the experiments, each peer maintains a list of semantic neighbours, updated during the simulation as follows:

LRU. The most natural strategy to capture recent history (which proved to work well in data management systems, such as caches for example) is to order the semantic neighbours such that the most recent uploader² peer is placed at the top of the list and to remove the least recent one if needed. This method is mostly used in [19] and in [21]. Upon a request, the x (x is a system parameter) first neighbours of the list are requested

¹The search algorithm is orthogonal to the way we use the semantic neighbours so we did not implement the search algorithm.

²*uploader* is a peer who offered a desired file to another peer

first. The standard search mechanism is then used in case of a miss.

History. This strategy has first been introduced in [21]. This approach aims at capitalizing on a larger period of time than LRU. Where the LRU strategy selects the x most recent useful peers, History selects the x most useful peers over a time frame. To this end, a peer i maintains an entry for a set of peers depicting how many times the peer i has downloaded a document from other peers with whom peer i previously interacted with. In other words, each peer (with whom peer i previously interacted) is associated with a counter; the peer i increments the counter of peer j whenever i downloads a file from j . The implementation of this strategy presents several issues. First of all, the data structure associated with each peer for this purpose grows linearly with the number of uploaders. However, such a data structure might be leveraged by other mechanisms such as incentives [2]. Secondly, deciding on a relevant period is not straightforward: if the period is too long, the strategy will not be able to adapt to change of preferences or interests; on the other hand, if the period is too short, the benefit of using History over LRU is not clear.

Figure 1 displays the hit rate according to the number of semantic neighbours contacted upon request (as specified by the parameter x). For the purpose of comparison, the hit rate when choosing the same number of random neighbours is also shown. We observe that the results are very encouraging: even with as few as 5 peers detected as semantically related, based on recent downloads, the hit rate is close to 30%. Note that the experiment starts with an empty semantic neighbour lists. On the contrary, the hit rate when neighbours are selected randomly is very low and re-

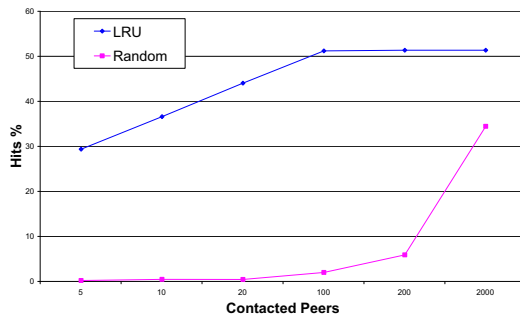


Figure 1: Hit rate using the LRU strategy to manage semantic neighbour lists

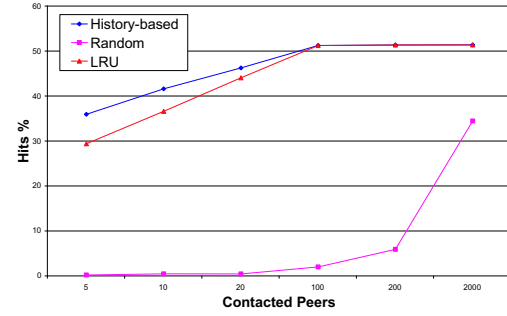


Figure 2: Hit rate using the History strategy to manage semantic neighbour lists

mains low even with a significant number of peers.

Figure 2 shows the hit ratio for the same configuration using History (for the purpose of comparison we also plot the hit ratio of LRU). Results show that History gives a slightly better hit ratio in the context of this trace. However, the fact that in our trace we do not capture any notion of time, we do not consider dynamic changes in user preferences.

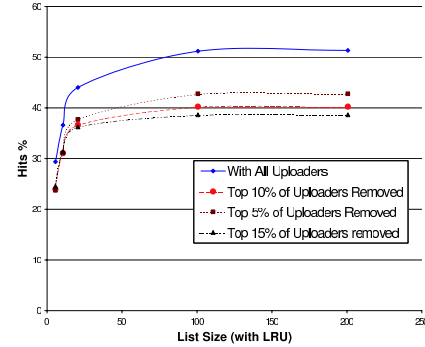


Figure 3: Impact of the removal of $u\%$ of the most generous uploaders

The observed improvement in the hit ratio suggests that there is indeed some semantic proximity between peers implicitly captured and exploited by the LRU and History strategies. However, the presence of generous uploaders may bias the semantic relationships. If a few peers provide most of the contents, which has been usually observed in measurements studies, the semantic neighbour lists might end up being mostly populated with these peers. To evaluate the impact of generous peers, we conducted a complementary experiment where we removed from the semantic neighbour lists the $u\%$ most generous uploaders, where we ex-

perimented with $u=5\%, 10\%, 15\%$. Figure 3 shows the results. We observed that the hit ratio drops roughly by 6%. This is an indication that the hit ratio is slightly influenced by the presence of very generous uploaders who contribute a large amount of files. In other words, the semantic neighbour lists contain the IDs of very generous peers. However, as seen in Figure 3, when the percentage of u is increased from 5 to 15 the reduction of hit ratio keeps unchanged for smaller list sizes (e.g., 5, 10, 20). This indicates that despite the *generous peers syndrome*, the semantic clustering truly exists. Otherwise the hit ratio would have been decreased linearly as the value of u is increased. On the other hand, even if the semantic lists contain the IDs of very generous peers, contacting those peers allows to bypass the standard search mechanism (either gossip-based or server-based) which is generally very expensive due to flooding or due to heavily loaded servers.

3.2 Semantic overlay

In the second set of experiments, we investigated the transitivity of the semantic relationship. In other terms, *can the semantic neighbours of my semantic neighbours help me when searching files?* To this end, we extended the search to two hops away, using the semantic neighbours. More precisely, first a peer searches using its semantic neighbours; failing that, the peer uses the semantic neighbours of its semantic neighbours by forwarding the search query on an overlay network composed of semantic neighbours, a semantic overlay. Our aim is not to evaluate the amount of clustering between a peer and its 2 hops away semantic neighbours but to observe the improvements that can be achieved in terms of search using a given number of such neighbours. Results are depicted on Figure 4. For comparison, we plot the results obtained according to the list size (as specified by x) of semantic neighbours for both direct semantic neighbours and 2nd level semantic neighbours which are 2 hops away. In other words, the amount of memory used to keep the information about semantic neighbours at each peer is fixed for the 2 schemes; but for a given size of the list (or x) the number of nodes contacted while searching is much higher (in the worst case scenario) when the 2nd level of semantic neighbours are also used. For example, when the list size is 5, in the worst case scenario, 25 peers are contacted in the 2nd level semantic search.

Results do not exhibit particularly better results for 2nd level semantic search, if we consider that

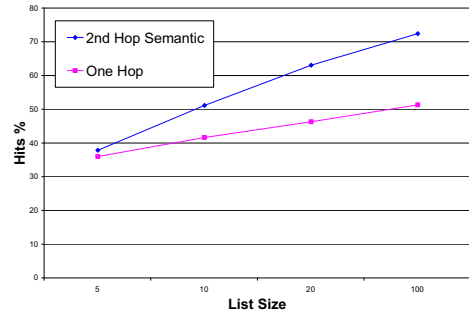


Figure 4: Semantic overlay

the important factor is the number of peers contacted. However, the fact that 2nd-level semantic search provides similar results as when the same number of semantic neighbours peers are contacted during the first hop, shows that semantic links tend to automatically cluster semantically-related peers. Other results related to the clustering properties of the trace are available in [9].

3.3 Explicit semantic

As we observed in previous experiments, simple heuristic enables to capture implicit semantic relationships between peers (without any additional information on the content exchanged or the peer profiles) and to cluster peers together. On the other hand, the information about the type of file being searched and peer profiles (such as the content they store) is available and can be used to make the semantic search more efficient.

We evaluated one such light explicit semantic scheme. To that end, instead of maintaining a unique list of semantic neighbours, each peer maintains several lists (according to the LRU or History policy), for example one per type of the file (audio, video, software etc). If, for example, peer i provides some music files to peer j , peer j will keep peer i in a separate list dedicated to music. In other terms, each type is associated with a separate list of semantic neighbours and a peer contacts only the neighbours who are in the relevant list for a given search. eDonkey, as well as most of its other competitors, associates with each file some meta information about the file such as the name, size, file extension and the kind of file (audio, video etc). These meta information helps when devising these explicit semantic schemes.

In this last set of experiments, we considered only a list for audio files since they represent the largest number of files ³ in the trace [9] and there-

³Audio files represent approximatively 50% of the files

fore the largest number of requests in our simulations. We investigated the use of already available information in most peer-to-peer file sharing systems and Figure 5 depicts the hit ratio obtained for audio files using both implicit and explicit semantics. More precisely, we show the performance improvement that can be achieved by having a separate list for a given file type: the hit ratio for audio files is evaluated by having a common list for all file types and a separate list for audio files as shown in Figure 5. In a real setting a peer will maintain number of such list for each file type (audio, software etc). These results show that the hit ratio is impacted when maintaining a peer profile per type of file.

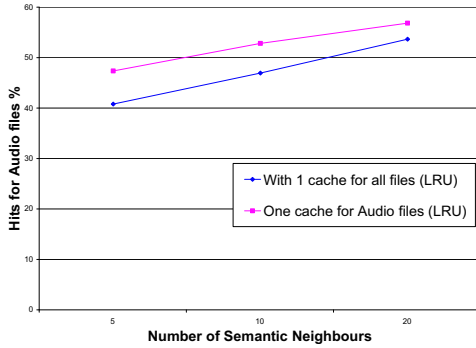


Figure 5: Explicit versus implicit

Another important point to note is that in the implicit scheme, while searching files that are in different categories of interests, a peer might lose (e.g., due to pruning of lists) some semantic neighbours just because those neighbours have no relevance to the current search or interests. This can be avoided by having separate lists for each interest group.

One could argue that the number of links maintained is much higher. This is true but that the important factor in terms of load is the number of neighbours which are contacted on a request rather than the number of pointers maintained on each node⁴. In addition, given the fact that a peer can have number of interest categories and different peers will act as uploaders for different categories, it is natural to think that a peer may exhibit semantic proximity with different peers depending on the type of information.

in our trace.

⁴maintaining semantic peers does not involve a heavy heartbeat-based scheme.

4 Conclusion

In this paper, we evaluated some simple strategies that can easily capture and exploit the semantic relationships observed between peers in peer-to-peer file sharing systems. The main issue in semantic proximity is how to capture the semantic relationship between two peers without explicitly involving peers themselves or structuring or grouping peers in the overlays into a static configuration which 1) does not evolve well as the interests of peers change and 2) generally needs significant amount of manual intervention by the users when structuring the peers.

We evaluated two different policies (LRU and History based) of maintaining semantic neighbours according to implicit scheme and present a simple explicit scheme. As was shown, both schemes produce good results in terms of hit ratio, demonstrating that exploiting the observed clustering between peers may lead to improvements in the search process. One important property of such improvements is that the approach may be used in unstructured (flooding-based), semi-structured (super-peers-based) or structured (DHT-based) peer-to-peer file sharing systems. We are currently investigating the implementation in these various contexts. Future work is also needed to evaluate these approaches in dynamic traces.

References

- [1] edonkey. <http://www.edonkey2000.com/index.html>.
- [2] Emule. <http://www.emule-project.net/>.
- [3] Gnutella. <http://www.gnutella.com>.
- [4] KazAa. www.kazaa.com.
- [5] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *IPTPS'03*, Feb. 2003.
- [6] M. Castro, M. Costa, and A. Rowstron. Should we build gnutella on a structured overlay? In *HotNets 2003*, Boston, MA, USA, Nov 2003.
- [7] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM'03*, 2003.
- [8] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, Stanford University, 2003.

- [9] F. L. Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in peer-to-peer file sharing workloads. In *The 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, 2004.
- [10] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *SOSP'03*.
- [11] <http://bitconjurer.org/BitTorrent/>. Bittorrent.
- [12] B. T. Loo, R. Huebsch, I. Stoica, and J. Hellerstein. The case for a hybrid p2p search infrastructure. In *The 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, 2004.
- [13] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, 2002.
- [14] D. Plonka. Napster traffic measurement. Technical report, University of Wisconsin-Madison, 2000.
- [15] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.
- [16] S. Saroiu, K. P. Gummadi, R. Dunn, S. D. Gribble, and H. M. Levy. An analysis of Internet content delivery systems. In *OSDI'02*, Dec. 2002.
- [17] S. Saroiu and S. G. P. Krishna Gummadi. A measurement study of peer-to-peer file sharing systems. In *MMCN'02*, Jan. 2002.
- [18] S. Sen and J. Wong. Analyzing peer-to-peer traffic across large networks. In *SIGCOMM'02 Workshop on Internet measurement*, 2002.
- [19] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM'03*.
- [20] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM 2001*, San Diego, USA, Aug. 2001.
- [21] S. Voulgaris, A.-M. 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)*, China, May 2004.
- [22] G. Wearden. eDonkey pulls ahead in European P2P race. http://news.com.com/2100-1025_3-5091230.html, 2003.