

PROXSEM: Interest-based Proximity Measure to Improve Search Efficiency in P2P Systems

Yann Busnel
IRISA / ENS Cachan
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Yann.Busnel@irisa.fr

Anne-Marie Kermarrec
IRISA / INRIA
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Anne-Marie.Kermarrec@irisa.fr

Abstract

Peer-to-peer (P2P) file sharing systems are now at the origin of most of Internet traffic. Improving the performance of the query mechanism of such systems has generated a lot of interest both in industry and academia. In a P2P system, peers are connected to a subset of other peers with which they can communicate. Each peer maintains a cache and makes available its contents to the rest of the system. Connecting peers sharing similar interest in the context of a given application has recently been identified as a sound basis to improve the search efficiency.

Nevertheless, capturing such interest-based (or semantic) proximity patterns is a difficult task. Most of current approaches measure this proximity between peers as the overlap between their cache contents. Given the well-known popularity patterns of peer-to-peer file sharing systems, the overlap between cache contents of two peers may not reflect accurately their semantic proximity.

In this paper we propose PROXSEM, a refined proximity measure taking into account peer generosity and file popularity. We evaluated the proposed solution by simulation against a real peer-to-peer file sharing system (eDonkey) workload and results show the effectiveness of the proposed approach. While peers generosity can easily be computed locally, file popularity may require a global knowledge of the system. We also propose in this paper an epidemic algorithm to compute in a fully decentralised fashion an estimation of files popularity.

1. Introduction and background

Peer-to-peer file sharing systems Peer-to-peer (P2P) overlay networks have recently proved to be efficient to support a large spectrum of large-scale distributed applications. P2P overlay networks and their applications have

generated a lot of interest in the research community in the past five years spanning from the overlays networks themselves [10, 20, 23] to streaming, archival, voice over IP applications, etc. [5, 8, 12]. Nevertheless, the main P2P applications deployed on Internet today are the P2P file sharing systems, which consume most of Internet bandwidth [21].

File sharing systems may be implemented through unstructured, structured or hierarchical overlay networks. The query mechanism is dependent upon the structure of the underlying network. In unstructured overlays, such as, for example, the early version of Gnutella [2], the query approach is implemented by flooding the system. On the other hand, hierarchical overlays are composed of a set of super peers to which clients are connected. Super peers are in charge of indexing the clients cache contents and redirect the requests towards relevant peers. KaZaA [3] and eDonkey [1] networks rely on such hierarchical models. Structured overlays provide an efficient exact-search mechanism by providing distributed hash table (DHT) functionality [4]. Many approaches have been proposed to improve search mechanisms in file sharing systems by optimising the replication strategies [18], using random walk instead of flooding to improve the load-balancing [10] or combining Gnutella-like systems with some structured overlays [7, 17].

Semantic proximity While generic P2P infrastructures have been optimised to take into account physical locality [9] early on, some recent work relies on capturing and exploiting other forms of proximity such as interest-based, or semantic, proximity¹. For example, similarities in download patterns and/or similar cache contents may be exploited to define a semantic proximity measure and to improve the cost and efficiency of the query mechanism. These approaches are motivated by the fact that semantically related peers are more likely to be useful to each other than peers picked at random [24].

¹We stick to the term semantic proximity in the remaining of this paper.

Detecting semantic relationships Various approaches can be used to capture semantic proximity between peers. Some approaches relies on a predefined ontology (semantic classification) [11]. Unfortunately, classifying items is not easy and the classification may vary over time to reflect the changes of semantic profiles. Another approach is to add some semantic shortcuts (*i.e.* additional links) between peers that share some interest [14, 22]. These links are created dynamically between peers, based on the set of most recent downloads, for instance. Such a mechanism is very reactive to evolving download patterns. Nevertheless, the non-intrusive nature of this approach does not allow to exploit further available information such as the overlap between caches for example.

Detecting and measuring semantic proximity is a difficult task. In [13], an analysis of clustering based on cache overlap in peer-to-peer file sharing system traces has shown the existence of semantic clustering. In addition to recent download patterns, the overlap between cache contents may be used to measure the semantic proximity between peers [25]. In [24], an evaluation of several strategies to capture semantic proximity has been conducted. More specifically, simple strategies based on past requests behaviour are compared. One observation is that assessing semantic proximity this way may lead to biased measurements. This is mostly due to the presence of generous peers and popular files which tend to hide genuine locality [13]. taking into account in the proximity measure these factors is the main goal of this paper.

Gossip-based protocols for tracking semantic proximity

The approach presented in [25] relies on a two layer gossip-based protocol to explicitly detect proximity between peers, measured as the overlap between peers cache contents. The underlying overlay is then updated accordingly. The bottom layer of the protocol ensures connectivity [16] while the top one is used to improve semantic search. We will use this gossip-based approach as a basis in this paper to evaluate the proposed solution.

Contribution In this paper, we propose PROXSEM a refined semantic proximity measure capturing peer generosity and file popularity in addition to overlap between cache contents. Quite a few approaches have been proposed to use semantic proximity. Nevertheless, to the best of our knowledge, none takes into account those issues. We assume that unpopular files are more representative than popular ones. Indeed, owning a rare file specify your profile better than a file owns by everyone. As demonstrated in [13], generous peers are chosen several times. With PROXSEM, we balance the peer in-degree by taking into account generosity, and increase the hit ratio by handling file popularity.

We also propose a decentralised gossip-based algorithm

to assess file popularity. We evaluate the improvement of our metric using a eDonkey 2000 [1] network trace obtained in November 2003 [13]. This paper is organised as follows: in Section 2, we present the design rationale of the proposed approach; in Section 3, we define PROXSEM; Section 4 presents an evaluation of the proposed measure; in Section 5, we introduce a decentralised protocol to estimate file popularity, and finally conclude in Section 6.

2. System model

Almost all query mechanisms, whether they rely on flooding, DHT, or super-peers based systems, may be improved by querying semantically related peers (called semantic neighbours in the remaining of this paper) first. If the request is not satisfied by semantic, the standard query mechanism may be used secondly.

Although, the proposed approach may be used in a wide range of peer-to-peer file sharing systems, we present it in the context of an unstructured network. We believe that unstructured P2P overlay networks are particularly relevant as they are flexible. In the rest of the paper, we focus on the capture of semantic relationship between peers with PROXSEM.

Using interest-based proximity to improve file-sharing applications requires to be able to: capture, evaluate and exploit semantic proximity between peers.

Capture Reorganising the overlay may be done either by adding or switching links. In this paper, we switch links as in [19] as soon as a better candidate is encountered. Each peer maintains a list of semantic neighbours called its semantic view and denoted χ_i . Neighbours are selected according to a semantic proximity metric. The way a peer comes across potential new neighbours is due to the implementation of the gossip-based protocol. The algorithm is given in Figure 1 and described below. As in [25], a bottom layer ensures connectivity.

Evaluation Each peer needs to evaluate its semantic “distance” to other encountered peers and order them using this measure as a ranking function. The *closest* known peers according to PROXSEM are selected as neighbours. We present this measure in Section 3.

Exploitation Semantic neighbours are first solicited in a search request. Even with a single hop search, studies have shown that for typical P2P file sharing systems workload, we obtain a good hit ratio [14, 24]. If the search based on semantic neighbours fails, the standard one is used.

```

(a) active thread
at each cycle, do
  q ← GETNEIGHBOUR()
  send  $\chi_p$  to q
   $\chi_q$  ← receive(q)
   $\chi_p$  ← UPDATEVIEW( $\chi_p, \chi_q$ )

```

```

(b) passive thread
do forever
   $\chi_q$  ← receive(*)
  send  $\chi_p$  to sender( $\chi_q$ )
   $\chi_p$  ← UPDATEVIEW( $\chi_p, \chi_q$ )

```

Figure 1. Gossip-based protocol executed by peer p

The two threads presented in Figure 1 constitute the gossip-based standard structure. At each round, each peer in the system executes the following steps (*active thread*):

1. chooses one of its semantic neighbours, picked randomly in its view ² (GETNEIGHBOUR());
2. sends to this selected neighbour the content of its own view. The selected neighbour sends back its own view;
3. upon receipt of its neighbour's view, the peer merges the new neighbours known in this exchange with its own view;
4. ranks the peers in the obtained set according to PROXSEM (First part of UPDATEVIEW());
5. keeps the x semantically closest neighbours (Last part of UPDATEVIEW()).

3. PROXSEM: a semantic proximity measure

We introduce a number of proximity measure notations:

- A is the local peer;
- B is a distant peer distinct from A ;
- $\xi_A(B)$ is PROXSEM: distance from A to B ;
- χ_A (resp. χ_B) is the view of peer A (resp. B);
- κ_A (resp. κ_B) is the cache of peer A (resp. B):
i.e. its set of shared files;
- $\sigma_{A,B}$ is the overlap of κ_A and κ_B :
i.e. $\sigma_{A,B} = \kappa_A \cap \kappa_B$;
- $\tau_{A,B}$ is the number of popular files in $\sigma_{A,B}$:
i.e. $\tau_{A,B} = |\{f | f \in \sigma_{A,B} \wedge f \text{ is popular}\}|$;

For the sake of clarity, we start from a basic semantic measure, and present successive refinements as the section goes.

²The view of a peer is the set of its neighbours.

3.1. Cache overlap

A basic idea, used in several approaches [14, 15, 25], consists in measuring the semantic proximity between peers as the size of the overlap between their caches. The greater this value, the semantically closer the peers. We first normalise this value according to A 's cache in order to keep the same definition region.

$$\xi_A^1(B) = \frac{|\sigma_{A,B}|}{|\kappa_A|} \quad (1)$$

The more files A and B have in common, the closer to 1 the measure. Note that at this stage, neither the file popularity nor B 's generosity are taken into account.

3.2. Ignoring generous peers

A generous peer is defined as a peer which owned a large file sharing cache.

As introduced in [15], generous peers have a greater probability to share several files with other peers. Therefore, they are more likely to be chosen as semantic neighbours and this may mask genuine semantic proximity. Although, this may have a positive effect on performance (obviously a generous peer is able to serve many requests), this may lead to an unbalance in the system. Therefore, PROXSEM should take peer generosity into account. Figure 2 presents possible configurations between two peers with varying cache size. In each configuration, the cache size of peers A and B are represented. The overlap $|\sigma_{A,B}|$ is constant for all configurations except the 10th one. These configurations must be ordered so that configuration 1 is the best configuration and 9 the worse one. In configuration 10, A and B do not have any file in common. Note that the ordering may not be symmetric depending on which peer measures the semantic proximity.

The semantic proximity measure given in Equation 1 can be adapted to implement the requested order, represented in Figure 2. The two first lines of Equation 2 infer the partial order between configurations 1, 2 and 3 and 1, 4 and 7 as well as 2, 5 and 8 and further.

To ensure the total order (the order between 2 and 4; 3, 5 and 7; 6 and 8 is missing) depicted on this figure, α should be lower than β . The theoretical analysis is provided in [6].

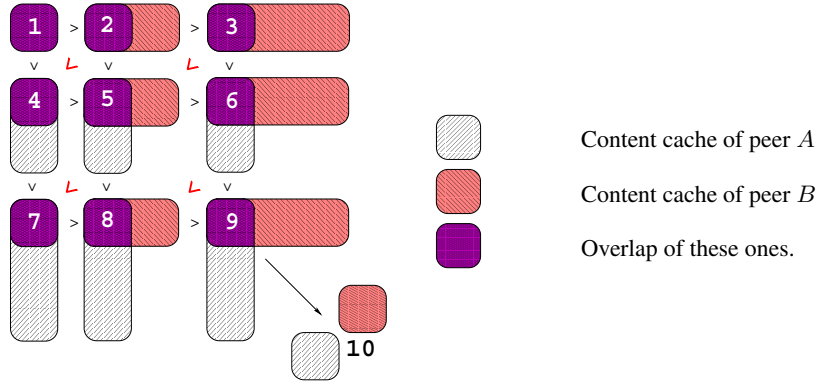


Figure 2. Ordering according to generosity with same overlap size (considered from A's point of view)

Strategies	Queries sent to peers
Random	Chosen at random
Overlap	Selected according to Equation 1
Overlap + Generosity	Selected according to Equation 2: PROXSEM with $\alpha = 1/3, \beta = 2/3$ et $\gamma = 0$
Overlap + Popularity	Selected according to Equation 3: PROXSEM with $\alpha = 1, \beta = 0$ et $\gamma = 1$
Total (Gen. + Pop.)	selected according to Equation 4: PROXSEM with $\alpha = 1/3, \beta = 2/3$ et $\gamma = 1$

Table 1. Summary

We decided to favour configuration 2 over configuration 4: from A's point of view, we assume that this order is more relevant than the other for the following reason. In configuration 2, B has a larger cache than the overlap, therefore B has a higher probability to be useful to A in the future. Conversely, 4 represents a setting in which A might be more useful to B. This relation is not symmetric.

To match the previous chosen ordering, a refined definition of the semantic proximity measure is:

$$\begin{cases} \xi_A^2(B) = \alpha \cdot \frac{|\sigma_{A,B}|}{|\kappa_A|} + \beta \cdot \frac{|\sigma_{A,B}|}{|\kappa_B|} \\ \alpha + \beta = 1 \\ \alpha < \beta \end{cases} \quad (2)$$

3.3. Handling file popularity

The file popularity is defined as the number of replicas of a file in the system.

The study in [15] shows that the popularity of files may significantly impact the semantic proximity. Consider two peers, B and C, having the same size overlap with A content cache (i.e. $|\sigma_{A,B}| = |\sigma_{A,C}|$) and the same generosity (i.e. $|\kappa_B| = |\kappa_C|$) so, $\xi_A^2(B) = \xi_A^2(C)$. Yet, B and C might not have the same "utility" for A. For example, B may have more popular files in common with A, but C may have more

rare files in $\sigma_{A,C}$. Non popular files (and rare files³ *a fortiori*) are more representative of a peer's semantic profile. Therefore, the distance between A and C should be greater than the distance between A and B : $\xi_A(B) < \xi_A(C)$.

Equation 2 has to be refined to take this into account. We decided to apply a multiplicative factor to the previous measure, in order to be conservative and have the same definition region. We use the last introduced notation. $\tau_{A,B}$ represents the number of popular files in the set of overlapping files between two peers. File popularity is a parameter of the system.

$$\lambda = \left(1 - \frac{\tau_{A,B}}{|\sigma_{A,B}|}\right)^\gamma \quad (3)$$

where γ is an exponent enabling to tune the importance of this factor.

Computing file popularity requires a global knowledge of the system whereas peers only know a subset of the overlay network. We present a mechanism to compute an estimation of the popularity of a file in a distributed way in Section 5.

³A file is considered as a rare file if the number of this file's replicas in the system is lower than a predefined threshold.

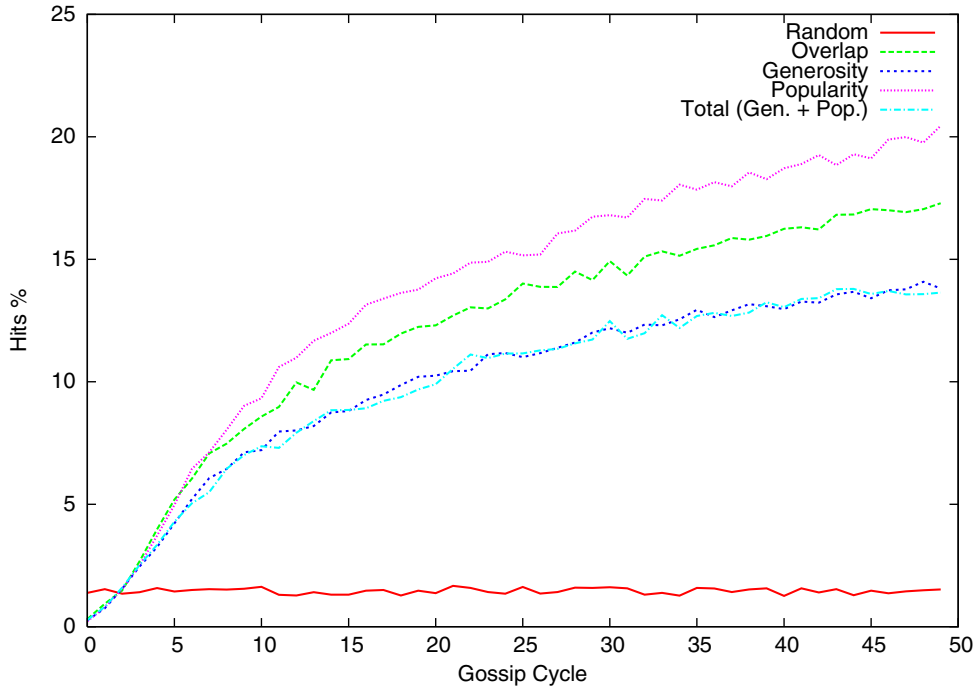


Figure 3. Hit ratio

3.4. Summary: PROXSEM complete

By merging Equation 2 and Equation 3, we obtain PROXSEM, presented in Equation 4, a full semantic proximity measure, which can be used in the gossip-based protocol presented in Section 2.

$$\xi_A(B) = |\sigma_{A,B}| \cdot \left(\frac{\alpha}{|\kappa_A|} + \frac{\beta}{|\kappa_B|} \right) \cdot \left(1 - \frac{\tau_{A,B}}{|\sigma_{A,B}|} \right)^\gamma \quad (4)$$

$$\text{where } \begin{cases} \alpha + \beta = 1 \\ \alpha < \beta \\ \gamma \geq 0 \end{cases}$$

In the remaining of the paper, we compare these variants, summarised in Table 1.

4. Performance evaluation

In order to evaluate the performance of PROXSEM, we wrote a discrete-event simulator in which the behaviour of n peers are simulated. The following sections describe the experimental set up and the simulation results.

4.1. Experimental setting

To evaluate the accuracy of the proposed measure, we used a real trace collected from the eDonkey file sharing

system [1] in November 2003. This trace has also been used to evaluate semantic-based systems in [13] and [25]. Using the same workload enables to compare directly the results. A set of 11,291 world-wide distributed peers with the files each one shares, is logged in this trace. A total number of 1,268,536 unique files are being collectively shared by these peers. We used that trace as a workload for our simulation the same way as in [25] for comparison.

We assigned peers randomly to the eDonkey clients of the trace. The simulator maintains the global list of files shared in the system and the popularity of each file⁴. Each client is associated with its list of files according to the real trace and maintains a set of semantic neighbours in its semantic view.

The list of semantic neighbours of each peer is initialised with a set of x random peers. The results presented in this paper are obtained with a set of 20 random peers. For each peer, the simulation consists in executing the active thread of the protocol described in Section 2 and Figure 1.

This results in a semantic overlay network. The approaches are compared along two metrics: the hit ratio for rare files and the load on each peer. At each cycle, each peer asks to all its semantic neighbours one of the files picked at random among rare files⁵ they owned. A file is considered as a rare file if the number of this file's replicas in the sys-

⁴For now, we consider that the popularity of a file is known *a priori*.

⁵Note that the rareness of a file here is globally determined.

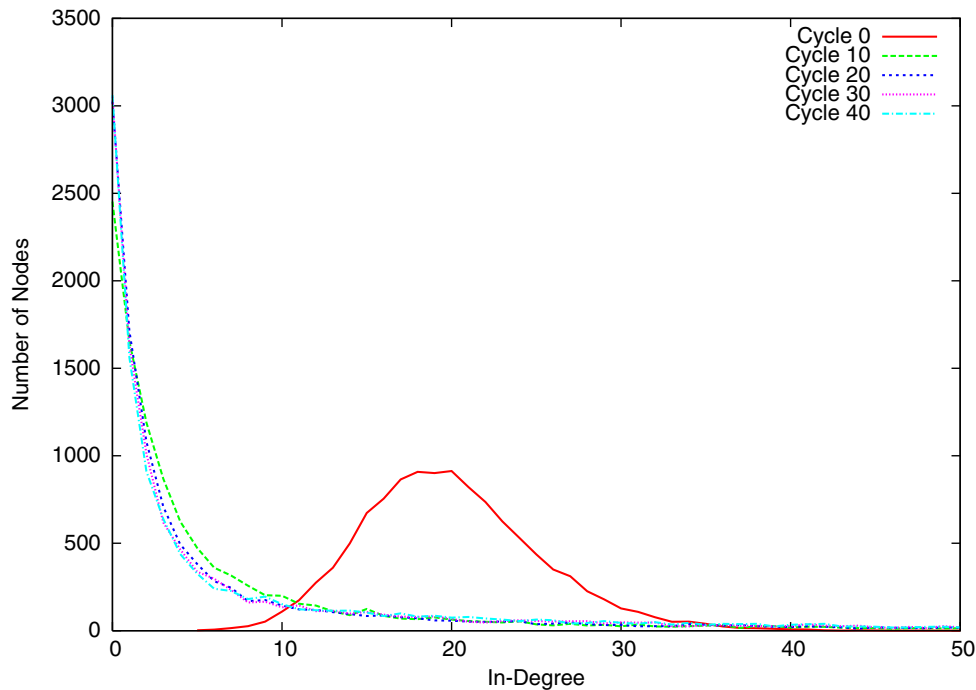


Figure 4. Peer Distribution for Several Cycles in the Gossip-based Protocol

tem is lower than a predefined threshold (here, 10 replicas). This threshold can be selected statically at the application running, or dynamically. After a complete round⁶, we compute the average hit ratio (for all peers). Rare files are chosen in priority since they are more representative and it is more difficult to locate them than popular ones. To evaluate the impact of generosity, we also compute the load of each peer, measured as the number of occurrences of a peer in the lists of semantic neighbours.

As a basis for comparison, we also use a random search: each peer sends a request for a rare file to x random peers, where x is equal to the number of semantic neighbours. Furthermore, we compare all results with the simulation in which the overlay network is randomly totally rewired at each cycle.

4.2. Simulation results

4.2.1 Hit ratio results

Figure 3 presents the average hit ratio depending on the cycle of the gossip-based protocol for the four considered strategies.

We observe that the hit ratio of the random approach does not exceed 1%. We observe that when taking into account popularity (popularity plot), the hit ratio reaches 21%

⁶A complete round is done when all peers have executed the active thread.

instead of 17% in the simple overlap case. When peer generosity is taken into account (Equation 2), the average hit rate decreases slightly as expected. The goal of this strategy is more to balance the load in the network than to improve on the hit ratio performance. We observe that when all factors are integrated in the semantic proximity measure, results are close to the ones obtained with generosity. Generosity and popularity have actually opposite impacts in terms of hit ratio.

4.2.2 Peer distribution

Figure 4 represents the in-degree distribution of peers (*i.e.* the number of occurrences of a peer in all the other peers view) depending on the gossip-based protocol cycle. At cycle one, the network is randomly initialised so that, the plot is a Gaussian-like, centered around 20 (the same value than the view size chosen for the experiment). For all other cases, from the cycle 10, 90% of peers have got an in-degree lower than 10. All strategies except the random one have the same plot appearance.

4.2.3 Impact of peer generosity

The goal of the proximity measure capturing peer generosity (Equation 2) is to improve load balancing in the system. Table 2 and 3 depict the maximum value of the nodes' in-degree after 50 cycles. Table 2 shows the maximum load

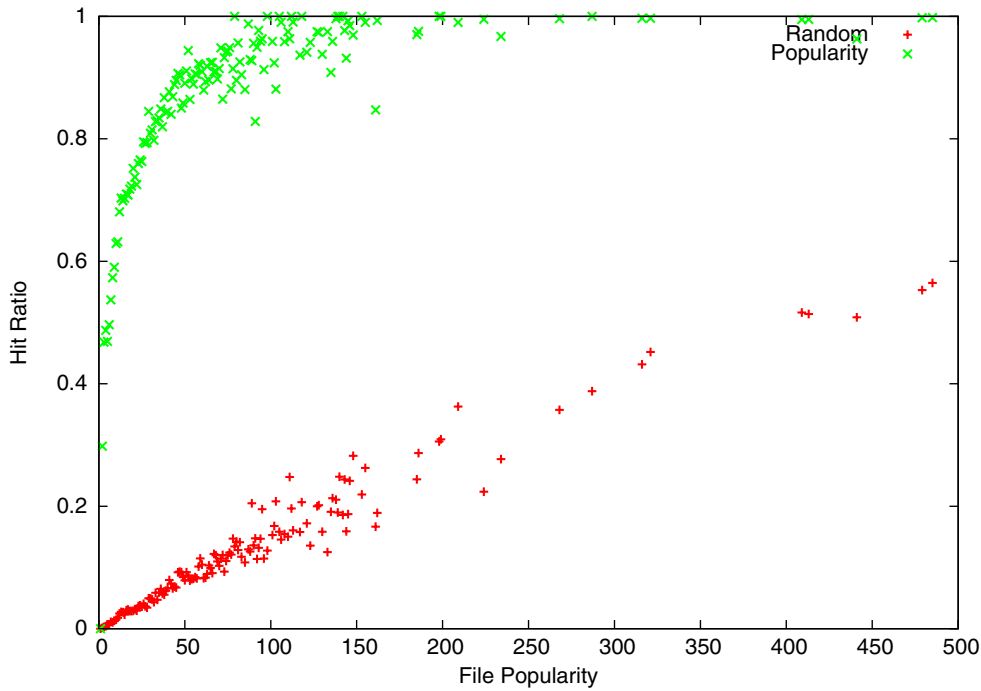


Figure 5. Hit Rate function of File Popularity

of most solicited peers in the system. These results confirm that taking into account generosity helps to limit the maximum in-degree.

Strategy	Maximum In-degree
Overlap	1702
Generosity	990

Table 2. Maximum In-Degree for different strategies

Table 3 shows the influence of α and β parameters on the load balancing. We also observe that the closer α and β , the better the load balance. The average hit rate is improved as well (14.53% for $\alpha = 1/2.1$ and $\beta = 1.1/2.1$ instead of 12.23% for $\alpha = 1/51$ and $\beta = 50/51$ at cycle 50). Complete plots and results can be found in [6].

4.2.4 Impact of file popularity

Figure 3 shows that the average hit ratio is greatly improved when popularity is taken into account as opposed to a simple overlap strategy.

To further evaluate this impact, we measured the hit ratio for every file across all peers. To this end, at cycle 50, each peer sends a request for all the files included in its cache

(α, β)	Maximum In-degree
$(\frac{1}{51}, \frac{50}{51})$	1210
$(\frac{1}{3}, \frac{2}{3})$	990
$(\frac{11}{21}, \frac{10}{21})$	898

Table 3. Maximum In-Degree for several gaps behind α and β

to all of its neighbours. Then, we calculated and associated the hit ratio for each file in the system. Figure 5 and 6 report the average hit ratio according to file popularity. Figure 5 shows that the less popular the files, the greater the impact. On Figure 6, the difference between the different values for rare files is increase up to 12%. The result for popular files is obviously the same for each case, except for Random strategy.

4.2.5 Summary

The objective of evaluation is to measure the impact of taking into account file popularity and peer generosity with PROXSEM. Our conclusion is:

- Hit ratio improves consistently by at least 23.5 % (from 1.5% to 21 % as compared to the random strategy and from 17% to the overlap strategy);

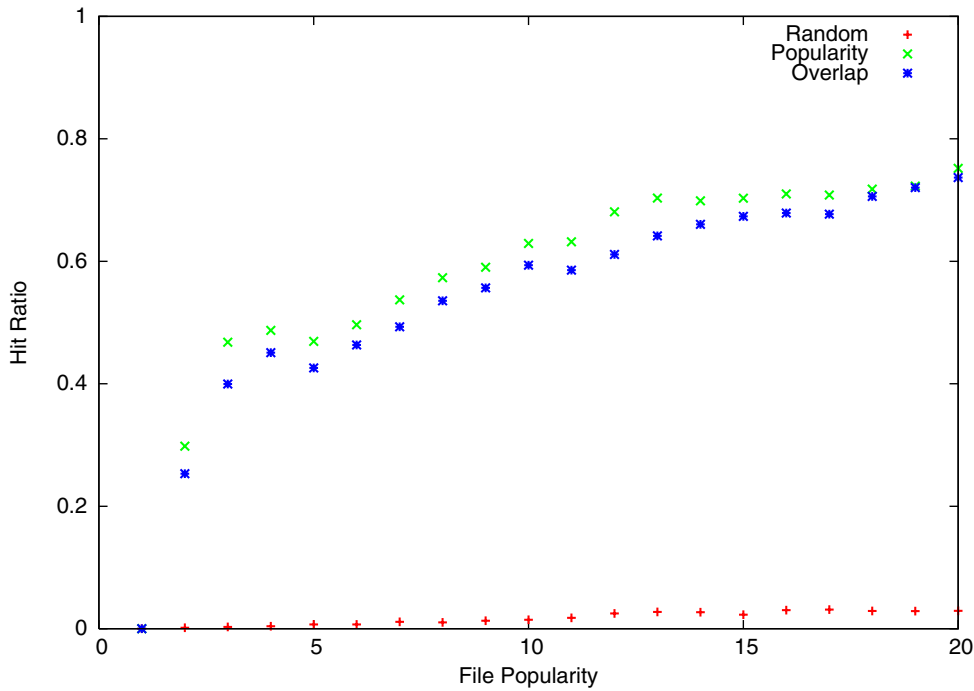


Figure 6. Hit Rate according to Unpopular Files

- Peer generosity improves load balancing;
- File popularity has a greater impact for searching rare files.

5. Tracking file popularity in a distributed way

Peer generosity can be easily computed locally as communicating peers only require to exchange their cache size. Nevertheless, computing file popularity, measured as the number of replicas in the system, requires to probe the entire system. To determine the $\tau_{A,B}$ value of the proximity measure, each peer requires the file popularity of each file of its content cache. The goal of the algorithm presented in this section is to compute file popularity in a fully decentralised way.

We use a gossip-based protocol to collect some information across the system. We use the same protocol structure as in Figure 1. At each cycle, the information exchanged between peers is related to the popularity value of all files they know about. When a peer joins the system, it sets this value to 1 for each file it owns. It sets this value to 0 for each discovered file in a gossip cycle. Then, each peer keeps the average of each popularity they exchange. For each file, a peer computes implicitly the following limit:

$$pop_f = \frac{\text{number of replicas in the network}}{\text{number of peers in the network}}$$

without knowing the size of the network.

Using the same experimental setting, the evaluation compares the estimate value of popularity files and the value computed by the simulator based on a global knowledge of the system. Representative peers⁷ are randomly chosen among the subset of peers.

The gossip-based protocol has a reasonable network overhead: at each cycle, every peer sends one message containing a file ID and an associated popularity value to only one other peer. Figure 7 shows the results of the simulation. At cycle 6, 60% of the values are well ordered and after cycle 20, 80% of the values are well aligned.

The main overhead of this approach is the memory consumption: each peer needs to maintain information about each file it encounters. We evaluate the maximum memory load at 10 MBytes on each peer, for 1,000,000 files in the system.

6. Conclusion and future work

Semantic proximity has been identified as a relevant metric to improve search in peer-to-peer file sharing systems. However, so far, very simple, and yet efficient approaches, have been proposed relying mostly on the recent history of

⁷A representative peer has a large diversity cache content, including several popular and rare files

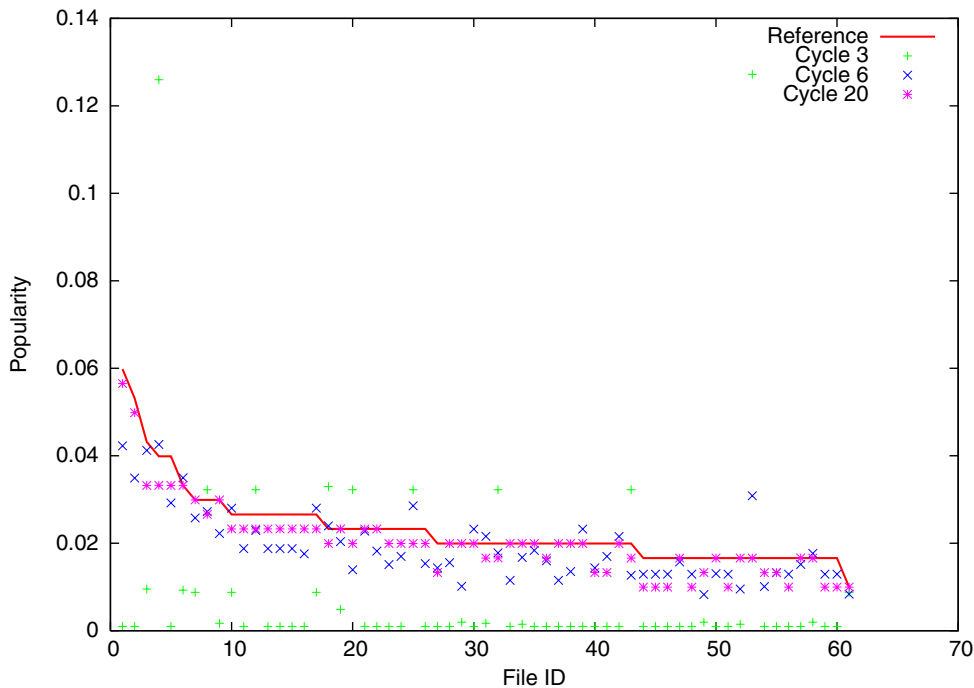


Figure 7. Detection with a Gossip-based Protocol

requests. In this paper, we evaluated PROXSEM, a finer-granularity semantic proximity measure to capture and exploit the semantic relationships observed between peers in a file sharing system. The goal of PROXSEM was to take into account both file popularity and peers generosity in the semantic measure as those factors have been previously identified as potential biases. We integrated the resulting measure in a gossip-based protocol, where links between peers were set according to their semantic proximity measure. We also proposed a fully decentralised algorithm to compute file popularity.

Based on simulation results, we observed that considering the peer generosity greatly improves the load balancing. Considering popularities of shared files improves the average hit ratio and the localisation of rare files.

References

- [1] The eDonkey 2000 project. <http://www.edonkey2000.com/>.
- [2] The Gnutella project. <http://www.gnutella.com/>.
- [3] The KaZaA project. <http://www.kazaa.com/>.
- [4] The Overnet project. <http://www.overnet.com/>.
- [5] The Skype project. <http://www.skype.com/>.
- [6] Y. Busnel and A.-M. Kermarrec. Integrating file popularity and peer generosity in proximity measure for semantic-based overlays. Research Report RR-5731, INRIA, IRISA, Rennes, France, October 2005.
- [7] M. Castro, M. Costa, and A. Rowstron. Should we build Gnutella on a structured overlay? In *The 2nd Workshop on Hot Topics in Networks (HotNets-II)*, MIT, Cambridge, MA, USA, Nov. 2003.
- [8] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *19th ACM Symposium on Operating Systems Principles (SOSP'03)*, The Sagamore, Bolton Landing, NY, USA, Oct. 2003.
- [9] M. Castro, P. Drushel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in peer-to-peer overlay network. Technical report, Microsoft Research, Cambridge, UK, 2003.
- [10] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *Special Interest Group on Data Communications (ACM SIGCOMM'03)*, San Diego, CA, USA, Aug. 2003.
- [11] A. Crespo and H. Garcia-Molina. Semantic overlay networks for P2P systems. Technical report, Database Group, Stanford University, Stanford, CA, USA, Sept. 2002.
- [12] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *The 18th ACM Symposium on Operating Systems Principles (SOSP'01)*, Lake Louise, AL, Canada, Oct. 2001.
- [13] S. Handurukande, A.-M. Kermarrec, F. L. Fessant, and L. Massoulié. Clustering in peer-to-peer file sharing workloads. In *The 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, San Diego, CA, USA, Feb. 2004.
- [14] S. Handurukande, A.-M. Kermarrec, F. L. Fessant, and L. Massoulié. Exploiting semantic clustering in the edonkey

P2P network. In *The 11th ACM SIGOPS European Workshop (SIGOPS'04)*, Leuven, Belgium, Sept. 2004.

- [15] S. Handurukande, A.-M. Kermarrec, F. L. Fessant, L. Massoulié, and S. Patarin. Peer sharing behaviour in the eDonkey network, and implication for the design of server-less file sharing systems. In *EuroSys'06*, Leuven, Belgium, Apr. 2006.
- [16] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In *ACM/IFIP/USENIX 5th International Middleware Conference (Middleware'04)*, Toronto, Ontario, Canada, Oct. 2004.
- [17] B. T. Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein. The case for a hybrid p2p search infrastructure. In *The 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, San Diego, CA, USA, Feb. 2004.
- [18] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *The 16th international conference on Supercomputing (ICS'02)*, June 2002.
- [19] A. Montresor, M. Jelasity, and O. Babaoglu. Robust aggregation protocols for large-scale overlay networks. In *Proceedings of The 2004 International Conference on Dependable Systems and Networks (DSN)*, pages 19–28, Florence, Italy, 2004. IEEE Computer Society.
- [20] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *The IFIP/ACM International Conference on Distributed Systems Platforms – Middleware (IFIP'01)*, Heidelberg, Germany, Nov. 2001.
- [21] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy. An analysis of internet content delivery systems. In *The 15th Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, USA, Dec. 2002.
- [22] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'03)*, San Francisco, CA, USA, Apr. 2003.
- [23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Special Interest Group on Data Communications (ACM SIGCOMM'01)*, San Diego, CA, USA, Aug. 2001.
- [24] S. Voulgaris, A.-M. Kermarrec, L. Massoulié, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *The 10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS'04)*, Suzhou, China, May 2004.
- [25] S. Voulgaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. In *The EuroPar'05*, Lisboa, Portugal, Sept. 2005.