The Gossple Anonymous Social Network *

Marin Bertier[1], Davide Frey[2], Rachid Guerraoui[3]
Anne-Marie Kermarrec[2], and Vincent Leroy[1]

[1] INSA de Rennes, Rennes, France
[2] INRIA-Rennes Bretagne Atlantique, Rennes, France
[3] EPFL, Lausanne, Switzerland

Abstract. While social networks provide news from old buddies, you can learn a lot more from people you do not know, but with whom you share many interests. We show in this paper how to build a network of anonymous social acquaintances using a gossip protocol we call Gossple, and how to leverage such a network to enhance navigation within Web 2.0 collaborative applications, à la LastFM and Delicious. Gossple nodes (users) periodically gossip digests of their interest profiles and compute their distances (in terms of interest) with respect to other nodes. This is achieved with little bandwidth and storage, fast convergence, and without revealing which profile is associated with which user. We evaluate Gossple on real traces from various Web 2.0 applications with hundreds of PlanetLab hosts and thousands of simulated nodes.

1 Introduction

Context: The Web 2.0 has radically changed the way people interact with the Internet: this has turned from a read-only infrastructure to a collaborative read-write platform with active players. Content is no longer generated only by experts but pretty much by everyone. Web 2.0 applications, such as LastFM, Flickr, CiteULike and Delicious [1], contain a goldmine of information. Yet, matching a specific query in such a mine might rapidly turn out to be like looking for a needle in a haystack, for the content of the Web is not indexed with a controlled vocabulary, e.g, *ontology*. Instead, freely chosen keywords are typically used to *tag* billions of items, i.e., with a *folksonomy* (folk + taxonomy). In short, the freedom left to the users to express their interests underlies the success of the Web 2.0 but it is also an impediment to navigation.

This paper proposes to improve navigation in such systems through implicit personalization: we associate every user with a network of *anonymous acquaintances* that are interested in similar items, independently of how they expressed their interests, e.g., which keywords they used to tag those items. These acquaintances are then implicitly used to guide and refine users' search operations.

Insight: To illustrate the benefit of *implicit personalization*, consider the following real example. After living for several years in the UK, John is back

* This work is supported by the ERC Starting Grant GOSSPLE number 204742

to Lyon in France. To maintain his kids' skills in English, he is looking for an English speaking baby-sitter who would be willing to trade baby-sitting hours against accommodation. There is no doubt that such an offer would be of interest to many of the young foreigners living in a big city such as Lyon. Yet, John's Google request *"English baby-sitter Lyon"* does not provide anything interesting for the term *baby-sitter* is mainly associated with *daycare* or local (French) baby-sitting companies.

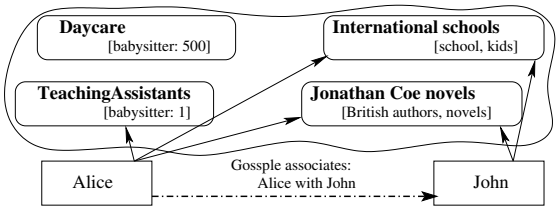

Fig. 1. Associating John to Alice enables John to leverage the unusual association between the keywords teaching assistants and baby-sitter.

None of John's Facebook buddies in Lyon or the UK can help either as none has ever looked for an English speaking baby-sitter in Lyon. Yet, Alice living in Bordeaux after several years in the US, and who was looking for a similar deal with her kids, has been lucky enough to discover that teaching assistants in primary schools are a very good match. Clearly, John could leverage Alice's discovery if only he knew about it. Should a system be able to capture the affinity between Alice and John, through their *common interest* in international schools and British novels for example (Figure 1), John could use Alice's information.

Indeed, consider a collaborative tagging system through which Alice has annotated the *teaching assistant* URL with *baby-sitter* (Figure 1). Assume both Alice and John have expressed their interest in international schools and British novels by tagging related URLs. A personalized search could leverage these affinities to return Alice's *teaching assistant* URL first instead of the millions of URLs of standard (non English speaking) baby-sitter related URLs. Likewise, a personalized query expansion could expand John's query appropriately with tags derived from Alice's activities on the Web and make it easy to solve by any reasonable search engine. The crucial aspect here is the unusual (personal) association between baby-sitter and teaching assistant, which is relevant to a niche community (the one gathering Alice and John) while baby-sitter is dominantly associated with (non English speaking) daycare. Discovering such specific affinity is very rewarding for the users, yet challenging to automatically capture.

Challenges: First, capturing associations between tagging profiles is difficult with millions of users, each with a dynamic variety of interests. Meaningful associations that help retrieve appropriate matching results, without drowning these within tons of useless information, should be derived from a large amount of information about proximity between user profiles. Such information grows not only with the number of users, but also with the number of interests per user. To illustrate this, consider our previous example. John, besides being interested in a baby-sitter, does also search for music over the Internet. Despite their proximity as far as kids and English are concerned, Alice and John have opposite tastes

music-wise. The fact that Alice and John are identified as acquaintances should not prevent John from benefiting from relevant music information using other acquaintances.

Second, discovering social acquaintances might be hampered by the resistance of users to publicize their tagging behavior. In fact, the apparent eagerness of companies to benefit from user-generated content might already dissuade users from generating new content and making their interests explicit[4]. A decentralized solution is appealing to address both the scalability and *big-brother* issues but poses nontrivial maintenance and efficiency issues, besides the fact that users might still be reluctant to reveal their interests to other unknown users.

Contributions: This paper presents GOSSPLE, a system that takes up these challenges, in a pragmatic way, to *automatically* infer *personalized* connections in Internet-scale systems. GOSSPLE nodes (users) continuously gossip digests of the tagging profiles (of their corresponding users) and locally compute a personalized *view* of the network, which is then leveraged to improve their Web navigation. The view covers *multiple interests* without any explicit support (such as explicit social links or ontology) and without violating *anonymity*: the association between users and profiles is hidden. In the context of the Alice-John example above, GOSSPLE leverages the very fact that John's request can benefit from Alice's tagging profile, without knowing who the profile belongs to. Basically, every GOSSPLE node has a proxy, chosen randomly, gossiping its profile digest *on its behalf;* the node transmits its profile to its proxy in an encrypted manner through an intermediary, which cannot decrypt the profile.

To reduce bandwidth consumption, the gossip exchange procedure is *thrifty*: nodes do not exchange profiles but only Bloom filters of those until similarity computation reveals that the two nodes might indeed benefit from the exchange. To limit the number of profiles maintained by each node, while encompassing the various interests of the user associated with the node, we introduce a new similarity metric, we call the *set cosine similarity*, as a generalization of the classical *cosine similarity* metric [2, 3], as well as an effective heuristic to compute this new metric.

While GOSSPLE can serve recommendation and search systems as well, we illustrate the effectiveness of GOSSPLE through a query expansion application for collaborative tagging systems. We believe this application to be interesting in its own right. We compute scores between tags using GOSSPLE and apply ideas from ranking Web pages (PageRank) to leverage the relative centrality of the tags through an algorithm we call *GRank*. [5]

Evaluation: We evaluated GOSSPLE with a wide-range of Web 2.0 application traces, including Delicious, CiteULike, LastFM and eDonkey, up to 100,000 users through simulation and in a real distributed system of 223 PlanetLab nodes.

[4] See recent events with privacy threats of Facebook.

[5] Classical file sharing applications could also benefit from our approach: our experiments with eDonkey (100,000 nodes) provided very promising results.

We first show that our multi-interest similarity metric improves on state-of-the-art ones by up to 70% in the considered datasets. Our gossip exchange procedure effectively builds a GOSSPLE network from scratch in less than 20 cycles, and maintains a baseline bandwidth of 15kbps. This is achieved with good anonymity guarantees through the use of onion-routing-like techniques.

We also evaluate our query expansion application independently and show how we retrieve items that state-of-the-art search systems do not (recall, also called completeness) whilst improving precision (accuracy) at the same time. We show, among other things, how GOSSPLE addresses the Alice-John baby-sitter situation above. For instance, a query expansion of size 20 retrieves 40% of unsuccessful original queries while improving the precision by 58% with respect to the originally successful queries (against resp. 36% and 24% for competitors [4]) on a Delicious trace. This is achieved in a thrifty manner: 10 acquaintances are enough to achieve effective query expansion in a $50,000$-user system, while exchanging gossip messages and profile digests of approximately 12.9KBytes and 603Bytes respectively.

In summary: The contributions of this paper are threefold: (i) an *anonymous*, *thrifty* gossip protocol; (ii) a *set cosine similarity* metric to compute semantic distances; and (iii) a *query expansion* application.

Roadmap: Section 2 details the GOSSPLE protocol. Section 3 reports on its evaluation. Section 4 describes the application of GOSSPLE to query expansion. Section 5 surveys the related work. Section 6 highlights some of the limitations of GOSSPLE.

2 The Gossple Protocol

GOSSPLE is a fully decentralized protocol aimed at building and maintaining dynamic communities of anonymous acquaintances. Such communities differ from networks of *explicitly* declared friends (e.g. Facebook) which are often not the most suited to enhance the search procedure as reported in [5].

2.1 Overview

More specifically, GOSSPLE provides each user with *GNet*, a network of semantically close anonymous interest profiles. Building and maintaining a node's *GNet* goes first through determining a *metric* to identify which profiles exhibit similar interests. This is particularly challenging for the goal is to capture the whole range of a node's interests while limiting the number of profiles in the *GNet*. Another challenge is to devise an *effective* communication procedure for profile exchange while limiting the amount of generated network traffic and hiding the association between nodes and profiles to preserve *anonymity*.

We detail in the following how GOSSPLE addresses these challenges. We assume, for presentation simplicity, a one-to-one mapping between a user and a machine: we refer to a node henceforth. From an abstract perspective, we model the profile of a node as a set of items $I = \{i_1, \ldots, i_n\}$. Each item can be described

by a potentially large amount of meta-information, such as tags describing it in the case of a folksonomy. Depending on the target application, this set may represent downloaded files, a summary of published content, etc. In the following, we first present a novel *multi-interest (set item cosine similarity)* metric; then, we describe our *thrifty gossip-based* protocol to establish and maintain connections between nodes (i.e., compute *GNet*) using this metric. For presentation simplicity, we present our protocol in a modular manner: we present first how a node communicates with other nodes to review a large set of profiles, how it encodes a profile, and finally how it preserves its anonymity.

2.2 Selecting acquaintances

Selecting relevant acquaintances requires nodes to rate each-other's profile.

Rating individuals. One way to build a node's *GNet* is to individually rate other nodes and select the ones that score the highest. Cosine similarity [6] is widely used in the area of data mining. The score between two nodes increases when interests are similar, specific overlapping of interests being favored over large profiles. Our preliminary experiments have shown indeed that cosine similarity outperforms simple measures such as the number of items in common. Thus we implemented cosine similarity as a reference in our experiments. More specifically, let I_{n_i} be the set of items in the profile of node n, the item cosine similarity between two nodes is defined as follows: $ItemCos(n_1, n_2) = \frac{|I_{n_1} \cap I_{n_2}|}{\sqrt{|I_{n_1}| \times |I_{n_2}|}}$.

Individual rating, and thus cosine similarity, selects nodes having the most similar profiles. Yet, this may lead to consider only the dominant interest, ignoring minor, potentially important, ones. In a large-scale system where a node can only keep track of a limited number of profiles, individual rating cannot capture emerging interests until they represent an important proportion of the profile, which they might never. Consider Bob whose tagging actions reveal that 75% of his interests are in football while 25% are in cooking. Selecting the profiles of the closest nodes might lead to only selecting those interested in football: the cooking topic will not be represented enough to provide interesting information. In Figure 2, the individual rating selects a view of 8 nodes interested only in football (users n_1, n_2, ... n_8).

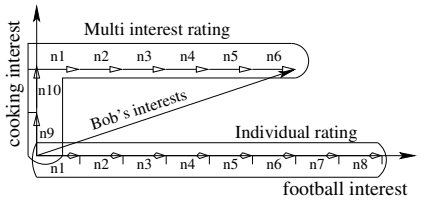

Fig. 2. Multi-interest rating *vs.* Individual rating

On the contrary, the *GNet* should preserve the distribution of 75% of football and 25% of cooking. We achieve this by rating a *set of profiles* as a whole rather than each profile independently. This is crucial to achieve scalability and limit the size of the *GNet wrt* the overall size of the network. In the example, *multi-interest* scoring would lead to selecting 2 nodes interested in cooking: n_9 and n_{10} and 6 in football, covering Bob's interest in cooking as well.

Rating sets. Rating a set of profiles as a whole involves a balance between the extent to which nodes in the set share interests with a given node n, and how well the distribution of the interests in the set matches the one in n's profile. This is precisely what the GOSSPLE *item set cosine similarity* metric achieves.

Let $IVect_n$ be the vector that represents the items in the profile of n. If item i is in the profile of node n $IVect_n[i] = 1$, otherwise 0. Following the same principle, $SetIVect_n(s)$ builds an item vector that represents the distribution of items in a set of nodes s with respect to the node n. Each value of this vector is computed as follows: $SetIVect_n(s)[i] = IVect_n[i] \times \sum_{u \in s} \frac{IVect_u[i]}{\|IVect_u\|}$. The rationale behind this metric is to represent the distribution of interests in *GNet* while normalizing the contribution of each node to favor specific interests. The items that are not present in the profile of node n are discarded since their distribution should not impact the score of *GNet*. Following the cosine similarity between nodes, a node n computes a score for a set of nodes as follows:

$$SetScore_n(s) = (IVect_n \cdot SetIVect_n(s)) \times cos(IVect_n, SetIVect_n(s))^b$$

The first part of the formula sums all the values in the vector representing the distribution of items in s, while the second represents how well the distribution of the items in s matches the one in n's profile, i.e. how fair the contribution to all the interests of n is. b is the *balance* between the amount of shared interests and a fair distribution that does not favor any item. The set that scores the highest forms the node's *GNet*. It is important to notice that for $b = 0$ (no cosine impact), the distribution is not considered and the resulting *GNet* is exactly the same as the one obtained from the individual rating.

2.3 Discovering acquaintances

GOSSPLE's multi-interest metric is key to selecting the best set of profiles to fill a given node's *GNet*. This would however be of little use without an effective mechanism to review a large set of candidate profiles while ensuring rapid convergence to the ideal *GNet*. GOSSPLE achieves this through a *gossip* protocol to establish and maintain connections between nodes. The GOSSPLE protocol relies on two sub-protocols (Figure 3): a *Random Peer Sampling protocol* (RPS) [7] and a multi-interest clustering protocol (*GNet* protocol). Each node maintains two corresponding data structures, *a random view* and the *GNet*.

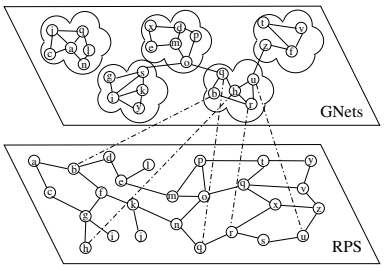

Fig. 3. GOSSPLE network

RPS. In the RPS, each node maintains a *view* of a random subset of network nodes. Periodically, it selects a node from this view and they exchange parts of their views. This provides each node with a random, changing, subset the network used for the bootstrap and the maintenance of the *GNet* protocol. Each entry in the random view contains *(i)* the IP address and the GOSSPLE ID of the node; *(ii)* a digest of its profile in the form of a Bloom filter (discussed in Section 2.4) and, *(iii)* the number of items in the profile (used for normalization).

Algorithm 1: GNet protocol

```
Do once for each T time units
begin
    g = oldest node in GNet_n
    Send GNet_n ∪ ProfileDigest_n to g
    Receive GNet_g from g
    compute GNet_n using
    (GNet_n ∪ GNet_g ∪ RPS_n)
    increment timestamp of GNet_n
    foreach u ∈ GNet_n do
        if u ∈ GNet_n for K cycles &
        u's full profile has not been
        fetched then
            fetch u's profile
end
```

Algorithm 2: Scoring heuristic

```
Input: set of nodes candidateSet
Output: a view of size viewSize
bestView = {}
for setSize from 1 to viewSize do
    foreach candidate ∈ candidateSet do
        candidateView =
        bestView ∪ {candidate}
        viewScore =
        SetScore(candidateView)
    bestCandidate = candidate that got the
    highest viewScore
    bestView = bestView ∪ {bestCandidate}
    candidateSet− = {bestCandidate}
Result: bestView
```

GOSSPLE relies on Brahms [8], a byzantine resilient RPS, which provides the building blocks to build GOSSPLE's anonymity (presented in Section 2.5).

GNet protocol. *GNet* contains c entries composed of the same fields as the random view entries with the addition of a timestamp representing the last time the entry was updated in the view. c is a parameter of the system that selects the trade-off between the amount of available information and its personalization degree. In addition, each entry also contains the full profile of nodes that have been chosen as acquaintances. We denote by $GNet_n$ the *GNet* of a node n.

The *GNet* protocol (Algorithm 1) is also gossip-based and maintains a list of implicit acquaintances (nodes). Each node n periodically selects the oldest (according to the timestamp) node g in its *GNet*, or a node picked from the RPS view if $GNet_n$ is empty. Then n sends the descriptors of c nodes in $GNet_n$ to g and g does the same. Upon receiving the other node's *GNet*, n and g update their own *GNet* structure. It first computes the union of $GNet_n$, $GNet_g$ and its own RPS view, then it selects the c nodes that maximize the GOSSPLE metric described in Section 2.2. Selecting the best view consists of computing the score of all possible combinations of c nodes out of $3c$ nodes. Because the corresponding running time is exponential in c, our protocol uses a heuristic (Algorithm 2) as an approximation. It incrementally builds a view of size c from an empty view by adding at each step the node that provides the best view score. This heuristic has complexity $O(c^2)$ and turns out to be very appropriate in our evaluations.

2.4 Gossiping digests

In order to keep the bandwidth consumption of GOSSPLE within reasonable bounds, nodes do not send their full profiles during gossip. Instead, they exchange a Bloom filter [9]: a compact representation of the set of items in the profile. The Bloom filter provides a reasonably good approximation of a node's profile that can be used to compute GOSSPLE's metric with a negligible error margin (as we discuss in Section 3.3).

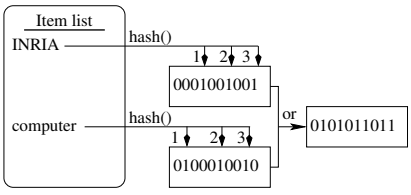

Fig. 4. The Bloom filter of a profile

The Bloom filter, as depicted in Figure 4, is an array of bits representing a set. When an item is added to the set, h hash functions are used on the item to obtain h positions in the array: these are set to 1. In order to query for the presence of an item in the set, one uses the same hash functions and checks if all the bits at the h indexes are set to 1. In the case of several additions to the Bloom filter, the request can return true even if the item was never added to the set. The rate of false positives depends on the size of the set, h, and the number of items in the system. However, a Bloom filter never returns a false negative.

In GOSSPLE, the Bloom filter is used as a first approximation of a node's profile. If a node remains in the *GNet* for K gossip rounds, it is considered as a good candidate and the entire profile is requested ($K = 5$ in our experiments). Once the full profile of the node has been retrieved, it is used to compute an exact similarity score. This prevents the expensive transfers of useless entire profiles of nodes that will reveal distant according to the GOSSPLE metric. For example, the profile of a Delicious user is on average 12.9KBytes large, while the corresponding Bloom filter is only 603Bytes. This leads to a 20-fold improvement in bandwidth usage.

Since a Bloom filter can return false positive results, some nodes considered as acquaintances through their Bloom filters can be discarded when the exact score is computed with the profile. However, a node that should be in the *GNet* will never be discarded due to a Bloom filter approximation. Hence, an entry for a node in the *GNet* contains either the full profile of the node or a Bloom filter of the profile with a counter incremented at each cycle. When the counter reaches the value of K, it triggers the retrieval of the corresponding full profile. The pseudo-code of the *GNet* protocol is presented in Algorithm 1.

2.5 Preserving anonymity

The decentralized nature of GOSSPLE somehow inherently preserves some level of anonymity, as compared to a central entity which would control and store all personal data. We go a step further by observing that, while personalized applications may benefit from the profile information contained in *GNets*, they need not know which nodes are associated with which profiles. In the Alice-John example, what matters is that John can leverage the information provided by Alice: he does not need to know Alice. This observation underlies our *gossip-on-behalf* approach: each node n is associated with a proxy p that gossips on its behalf. Since P2P networks are subject to churn, p periodically sends snapshots of $GNet_n$ to n so that n can resume the gossip protocol on a new proxy without losing any information. This *anonymity by proxy* setup is achieved by an encrypted two-hop communication path *à la* onion routing [10]. Therefore p receives n's profile and its updates but does not know n's identity while the nodes relaying the communication cannot decrypt the profile. GOSSPLE relies on the Byzantine resilience properties of the RPS protocol to prevent colluders from manipulating the selection of the proxy and the relays. Furthermore, we assume that the system is protected against Sybil attacks through a certificate mechanism or a detection algorithm [11]. Finally, we also consider that it is a

user's responsibility to avoid adding very sensitive information to her profile. In that case, the profile alone would be sufficient to find the identity of the user as it was the case in the famous anonymized AOL query-log dataset. GOSSPLE ensures anonymity deterministically against single adversary nodes and with high probability against small colluding groups.

3 Gossple Evaluation

We report on the effectiveness and cost of GOSSPLE by means of both simulation (100,000 nodes) and PlanetLab deployment (446 nodes). The former provides insights about GOSSPLE's behavior with a very large number of participants while the latter about the inherent overhead of GOSSPLE in a real distributed setting.

3.1 Setting and methodology

We evaluated the quality of GOSSPLE's *GNet*, its convergence speed, as well as its bandwidth consumption and behavior under churn. We considered various Web 2.0 datasets: *(i)* a trace crawled from Delicious (social URL bookmarking) in January 2009; *(ii)* a trace from CiteULike (reference management system) available on October 9, 2008; *(iii)* a LastFM (music recommender system) trace crawled in Spring 2008 and composed of the 50 most listened-to artists for each user; and *(iv)* an eDonkey (P2P file-sharing) trace [12]. Table 5 provides the figures for each trace. We configured GOSSPLE's *GNets* size to 10 and the gossip cycle length to 10 seconds. We deliberately select a small *GNet* size in order to stress the system and highlight the impact of the selection of each acquaintance.

In this section, we evaluate the quality of a node's *GNet* through its ability to provide the node with interesting items. This measures the basic usefulness of GOSSPLE's *GNets* in search applications and recommender systems. We remove a subset (10%) of the items in the profile of each node, which we call the *hidden interests* of the node. GOSSPLE uses the remaining part of the profile to build the *GNets*. We express the quality of a *GNet* in terms of *recall*, i.e. the proportion of hidden interests of a node n that are present in the profile of at least one node in n's *GNet*. The better the *GNet*, the more hidden interests it contains. Each hidden interest is present in at least one profile within the full network: the maximum recall is always 1. Recall is enough to evaluate quality here because n contacts a fixed number of nodes. We further evaluate precision in Section 4.

We evaluate the overall quality by computing the recall for the whole system: we divide the sum of the number of hidden interests retrieved by GOSSPLE for each node and divide it by the sum of the number of hidden interests of each node.

3.2 Metric (the quality of GNets)

Our item set cosine similarity (multi-interest) metric enables GOSSPLE to cover the diversity of user interests in a large scale system by assessing the quality

of the whole *GNet* at once, instead of rating profiles individually. Parameter b in the definition of *SetScore* (*cf.* Section 2.2) represents the weight of minor interests *wrt* major ones. We evaluate the impact of b using the hidden-interest test described above. Intuitively, too small a value of b will achieve poor results because the *GNet* will fail to provide items related to minor interests. On the other hand, too high a value of b might select profiles with too little in common with the node to provide relevant items. Figure 6 presents the normalized recall achieved by *GNets* on the datasets with increasing values of b. The score is normalized to take as a reference the case when $b = 0$, equivalent to individual rating. As expected, the performance initially increases with increasing values of b, but it decreases for greater values of b. Multi-interest improves recall from 17% (LastFM) to 69% (Delicious). For all datasets, GOSSPLE significantly improves the quality of *GNets* over traditional clustering algorithms illustrated by the $b = 0$ configurations. The exact recall values are given in Table 5. We also observe that the improvement of multi-interest has more impact when the base recall is low. Further experiments, omitted due to space constraints, revealed that this is because GOSSPLE is particularly good at finding rare items. Finally, we observe that the maximum performance is not obtained for a specific value of b, but for a full range of values, $b \in [2, 6]$, across all datasets. This demonstrates GOSSPLE's effectiveness on a wide variety of datasets without requiring prior fine tuning of parameters.

	Delicious	CiteULike	LastFM	eDonkey
Nb users	$130k$	$34k$	$1,219k$	$187k$
Nb items	$9,107k$	$1,134k$	$964k$	$9,694k$
Nb tags	$2,214k$	$237k$		
Avg profile size	224	39	50	142
Nb nodes	$50k$	$10k$	$100k$	$100k$
Recall $b = 0$	12.7%	33.6%	49.6%	30.9%
Recall GOSSPLE	21.6%	46.3%	57.6%	43.4%

Fig. 5. Dataset properties

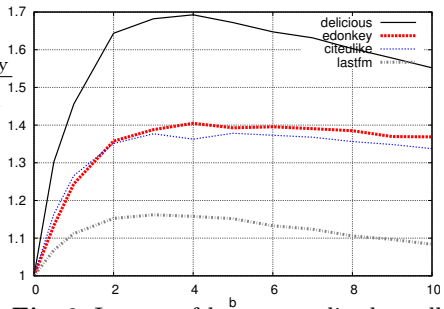

Fig. 6. Impact of b on normalized recall

3.3 Convergence time

We evaluate GOSSPLE's ability to build and maintain a useful *GNet* for each node. First, we consider the time required to build a network of GOSSPLE acquaintances from empty *GNets*. Then we consider the maintenance of this network by evaluating convergence in a dynamic scenario where nodes join an existing stable network.

Bootstrapping. We consider a simulation of $50,000$ nodes and a real-world PlanetLab deployment with 446 nodes on 223 machines. Figure 7 plots the hidden-interests' recall (*cf.* Section 3.2) during the construction of the GOSSPLE network. It is normalized by the value obtained by GOSSPLE at a fully converged state.

As expected, the multi-interest clustering ($b = 4$) constantly outperforms the standard one ($b = 0$), although it requires slightly longer to reach a stable state.

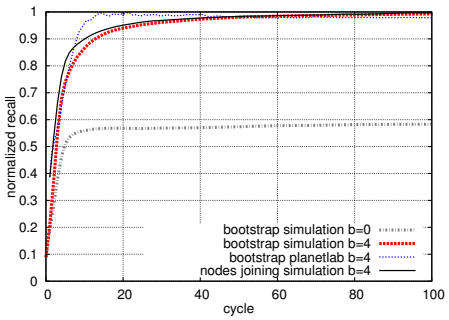

Fig. 7. Recall during churn

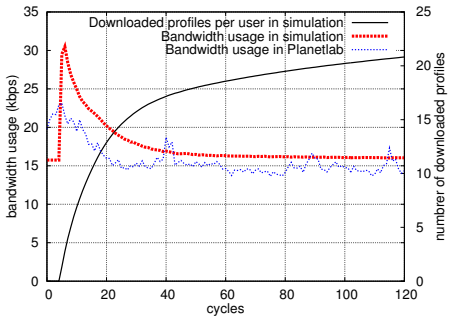

Fig. 8. Bandwidth usage at cold-start

The difference is minimal, GOSSPLE reaches 90% of its potential after only 14 gossip cycles in simulation in our Delicious traces for instance. This convergence is extremely fast given the size of the network (50,000) and the small size of the *GNet* and RPS (10). The measures conducted on LastFM confirm the scalability of the protocol: for twice as large a network, only 3 more cycles are needed to reach the same convergence state. The PlanetLab deployment confirms the simulation results. The smaller scale of the experiments causes GOSSPLE's *GNets* to reach 90% of their potential after an average of 12 cycles and stabilize after 30. This confirms the scalability of our protocol by demonstrating convergence times that grow very slowly with the size of the network.

Maintenance. Bootstrapping represents a worst-case scenario for the convergence of our protocol. It is, in fact, a one-time event in the life of a GOSSPLE network. During normal operation, the system will instead experience perturbations that cause it to deviate from a stable state in which each node has converged to its best possible *GNet*. Examples of such perturbation include variations in the interests of users, or the presence of nodes that join and leave the network.

To evaluate the impact of these perturbations, we consider a scenario where 1% of new nodes join an existing GOSSPLE network at each gossip cycle. We measure the hidden-interest recall of these new nodes to see how many gossip cycles are needed for them to reach a quality of *GNet* equivalent to the one of the previously converged nodes (i.e. 1 on the normalized score). Joining an existing network is indeed faster than bootstrapping, as 9 cycles are enough to reach a 90%-quality *GNet*. Clearly, this represents an upper bound on the time required for convergence in the presence of dynamic profiles or nodes that leave the network. In both of these cases, nodes that are required to converge only need to partially reconstruct their *GNets* as they already have some good neighbors to rely on. Moreover, the removal of disconnected nodes from the network is automatically handled by the clustering protocol through the selection of the oldest peer from the view during gossip exchanges as discussed in detail in [13].

3.4 Bandwidth

The bandwidth consumption of GOSSPLE when the *GNets* are built from scratch is depicted in Figure 8. This cost is the result of: *(i)* the exchange of profile digests upon gossip; *(ii)* the download of the full profiles of nodes in the *GNet*; *(iii)*

the extra communication required to maintain anonymity. Each node gossips through the RPS and the *GNet* protocols every $10s$: each message containing respectively 5 and 10 digests. This results in a maximum bandwidth consumption of 30kbps in the most demanding situation, that is during the burst at the beginning of the experiment. This is because no profile information has yet been downloaded.[6] As soon as the *GNets* start to converge, the rate of exchange of full profiles decreases, as shown by the line depicting the total number of downloaded profiles per user. This causes bandwidth consumption to decrease to the fixed cost of gossiping digests, 15kbps, a clearly irrelevant value for almost any user on the network today.

If GOSSPLE did not use Bloom filters, on the other hand, profile information would be exchanged continuously, thereby increasing bandwidth consumption. In the considered data trace, replacing Bloom filters with full profiles in gossip messages makes the cost 20 times larger. Finally, we observe that, while GOSSPLE's anonymity protocol continuously sends keep-alive messages, the only ones that impact bandwidth are those sent when new profile information needs to be exchanged.

4 Gossple at Work: Query Expansion

A query expansion system seeks to extend a query with additional keywords to improve the results. We now describe how to use GOSSPLE to expand queries within a collaborative tagging system, such as Delicious or CiteULike, where every node is associated with a tagging profile. As we show, GOSSPLE significantly improves the completeness (recall [14]) and accuracy (precision [14]) of the results, *wrt* the state-of-the-art centralized personalized approach, namely Social Ranking [4].

4.1 Overview

We use GOSSPLE *GNets* to compute a data structure we call *TagMap* (Figure 9), a personalized view of the relations between all tags in a node's profile and in its *GNet*. This is updated periodically to reflect the changes in the *GNet*. A query from the node is then expanded using the *TagMap* of that node through a *centrality* algorithm we call *GRank*, which we derived from the seminal *PageRank* [14] algorithm. While PageRank computes the relative importance of Web pages (eigenvector centrality [15]), *GRank* computes the relative importance of tags on a given node: we refer to this notion as the *tag centrality* from now on. *GRank* estimates the relevance of each tag in the *TagMap wrt* the query and assigns a score to the tag. We then recursively refine the correlation between tags by computing their distance using random walks, along the lines of [16].

[6] The is burst is slightly longer on PlanetLab due to the lack of synchronization between nodes.

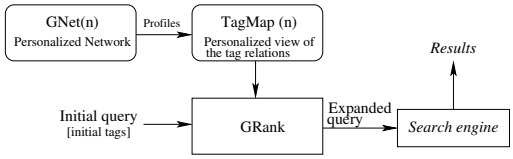

Fig. 9. Expanding queries with GOSSPLE

	Music	BritPop	Bach	Oasis
Music	1	0.7	0.1	0
BritPop		1	0	0.7
Bach			1	0
Oasis				1

Fig. 10. Example of a TagMap

4.2 TagMap

The *GNet* of a node n contains a set of profiles covering its interests. These profiles are used to compute the *TagMap* of a node n, namely a matrix $TagMap_n$, where $TagMap_n[t_i, t_j]$ is a score that reflects the distance between tags t_i and t_j as seen by node n. We denote by IS_n the information space of node n, namely its profile and the profiles in its *GNet*; T_{IS_n} and I_{IS_n} denote the set of all the tags and items in IS_n. The *TagMap* uses item-based cosine similarity to compute a score between the tags. The information needed to fill the *TagMap* is, for each tag in T_{IS_n}, the number of occurrences of the use of that tag per item, i.e., for all $t \in T_{IS_n}$, a vector $V_{n,t}$ of dimension $|I_{IS_n}|$ is maintained such that $V_t[item_i] = v$, where v is the number of times the item $item_i$ has been tagged with t in IS_n. More precisely: $TagMap_n[t_i, t_j] = cos(V_{n,ti}, V_{n,tj})$. Table 10 depicts an example of a node's *TagMap*. In this example $TagMap_n[Music, BritPop] = 0.7$.

4.3 GRank

The *TagMap* contains a personalized view of the scores between pairs of tags. This information can be directly used to expand queries as in [4]. In this approach, called *Direct Read (DR)*, the number of tags q added to the initial query is a parameter of the query expansion algorithm. With DR, a query is expanded with the q tags scoring the highest. More precisely: $DRscore_n(t_i) = \sum_{t \in query} TagMap[t, t_i]$. While this approach seems intuitive, it gives bad results when the item sparsity of the system is high. This is very likely to happen for niche content: with a very large number of items, the relationships between tags might not always be directly visible in the *TagMap*. To illustrate the issue, consider the *TagMap* presented in Table 10 and a query on *Music* with $q = 2$. The *TagMap* exhibits a high score between *Music* and *BritPop* (based on a given set of items). In addition, there is a low score between *Music* and *Bach* suggesting that the node is more interested in *BritPop* than in *Bach*. However *BritPop* and *Oasis* have also a high score in the same *TagMap* (gathered from a different set of items), DR will never associate *Music* and *Oasis* whilst this association seems relevant for that node. In fact, DR would instead expand the query with *Bach*, increasing the result set size and reducing the precision of the search (Figure 11).

Our approach is based on the observation that, by iterating on the set of newly added tags, more relevant tags can be selected for the query. We capture this idea through an algorithm inspired by *PageRank* [14], which we call *GRank*[7]. In short, *GRank* runs a personalized PageRank on the tag graph

[7] PageRank provides information about the importance of vertices in a graph. Personalized PageRank (PageRank with priors) compute centrality *wrt* a set of vertices.

extracted from the *TagMap* and assigns the set of priors to the tags in the query. More specifically, considering the weighted graph provided by the *TagMap*, all the tags in T_{IS_n} are vertices and, for each non-null score in the *TagMap*, we add an edge weighted by the score. These scores affect the transition probabilities when generating the query expansion. As opposed to PageRank, where each link is chosen with equal probability, in *GRank*, the transition probability (TRP) from one tag to another depends on the edge weight provided by the *TagMap*: $TRP_n(t_1, t_2) = \frac{TagMap_n[t_1,t_2]}{\sum_{t \in T_{IS_n}} TagMap_n[t_1,t]}$

To limit the computation time of a score (which can be prohibitive in a large graph), instead of running an instance of *GRank* per query, we divide the computation per tag in the query, forming partial scores. These are approximated through random walks [16] and cached for further use whenever the same tag is used in a new query. Still, a centralized server computing *GRank* for all nodes would not scale. It can only be applied in the context of GOSSPLE where each node provides its processing power to compute its own *GRank*.

Expanding a query using *GRank* then simply consists in adding to the original query the q tags scoring the highest. All tags receive a score which is transmitted to the companion search engine.

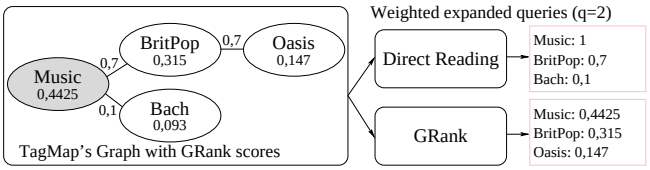

Fig. 11. Example of query expansion

4.4 Evaluation

We evaluated GOSSPLE's ability to achieve a complete and accurate query expansion. We describe our experimental setup, report on the evaluation of the GOSSPLE query expansion mechanisms on Delicious and CiteULike traces, and finally, we consider synthetic traces modeling the baby-sitter example given in the introduction, as well as the behavior of a mad tagger trying to disrupt GOSSPLE's operation.

Evaluation setup. **Workload.** Each node is associated with a user (and her profile) from a real trace (Delicious or CiteULike). The profiles drive the generation of requests: let I_n be the set of items in n's profile, n generates a query for each item $i \in I_n$ such that at least two users have i in their profiles. The initial query consists of the tags used by n to describe item i since they are also likely to be the ones n would use to search for i. We evaluate GOSSPLE's query expansion using the generated queries. Given a query from node n on an item i, we first remove i from n's profile so that n's *GNet* and *TagMap* are not built with it, then we determine if i is indeed an answer to the expanded query.

Search engine. Although any search engine could be used to process a query expanded by *GRank*, for the sake of comparison, we consider the search engine

and ranking method used in [4]. The search engine runs on the set of items available in the real trace. An item is in the result set if it has been tagged at least once with one of the tags of the query. To rank the items, the score of an item is the sum, for each tag in the query, of the number of users who made an association between the item and the tag, multiplied by the weight of the tag.

Evaluation criteria. The query expansion mechanism is evaluated along the two classical and complementary metrics: *recall* (completeness) and *precision* (accuracy). Recall expresses the ability of the query expansion mechanism to generate a query that includes the relevant item in the result set. In these experiments, we are interested in the item i for which the query has been generated. A query succeeds when i is in the result set: recall is 1 if i is in the result set, 0 otherwise. Note that the size of the result set increases with the length of the query expansion, potentially reducing the visibility of the relevant item. To balance this effect, we also assess the quality of the query expansion by evaluating *precision*. We consider the absolute rank (based on the score provided by the search engine) of the items in the result set as a metric for precision: namely, the precision is defined as the difference between the rank with query expansion and the rank with the initial query.

Recall is evaluated on the queries that do not succeed without query expansion. This comprises 53% of the queries in the CiteULike trace and 25% in the Delicious one. These results show that a significant proportion of items have been tagged by several users with no tags in common and highlights the importance of query expansion. Precision is evaluated on the queries that are successful even without query expansion since they provide a reference rank for the item.

Impact of GOSSPLE*'s personalization.* Our first results, in Figure 12, isolate the impact of personalization through a *TagMap* based on a small *GNet wrt* to a global *TagMap* based on all users' profiles. Figure 12 shows the extra recall obtained with query expansion sizes from 0 to 50 on the Delicious[8] traces with *GNet* sizes from 10 to 2,000. We compare these results with those of Social Raking, i.e. the case where *GNet* contains all the other users. The figure presents the proportion of items that were found with the query expansion out of the ones that were not found without. For example, the point $(x = 20, y = 0.37)$ on the GOSSPLE 10-neighbor curve says that 37% of the requests not satisfied without query expansion are satisfied with a 20-tag expansion based on a *GNet* of 10 nodes.

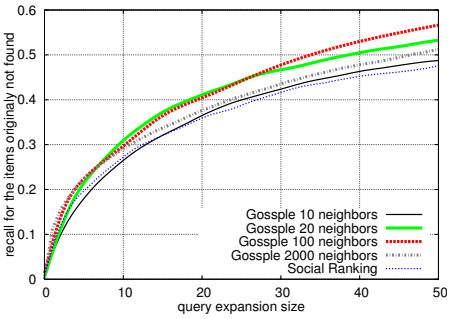

Fig. 12. Extra recall (Delicious)

[8] Experiments on the CiteULike trace lead to the same conclusions but the results are not presented for space reasons.

The figure highlights the benefit of personalization over a *TagMap* that would involve all users such as Social Ranking. Even though increasing the size of the *GNet* up to 100 has a positive impact on recall, larger sizes degrade performance. With a 30-tag query expansion, a 10-node *GNet* has a recall of 43%; recall goes up to 47% in a 100-node network and drops to 42% in the case of Social Ranking. As the number of node profiles integrated in the *TagMap* increases, relevant tags are gradually *swallowed* by less relevant tags or popular ones. This is clearly an illustration of how the baby-sitter/teaching assistant association in the example presented in Section 1 could be diluted if we had considered all users.

Impact of GOSSPLE*'s centrality.* We evaluate here the impact of computing the relative importance of tags (*GRank*) over the DR query expansion. As mentioned previously, personalization is mostly responsible for increasing recall. However, computing the relative importance of tags (i.e., tag centrality) significantly improves the precision of the results. This is clearly illustrated in Figure 13, which compares our results against those of the non personalized DR query expansion, i.e. Social Ranking.

On Figure 13(right), as the query expansion size increases, the recall for items which were not found initially, improves. With a 20-tag query expansion, we observe a 37% recall of the items which were not originally found. Yet, this is accompanied by a significant drop in the precision for 71% of the items originally found. *GRank* however with a 20-, resp. 50-tag query expansion, increases the recall of items not found initially up to 40%, resp. 56%, while increasing the ranking of approximately 58, 5% resp. 40% of the items which were returned to the initial query. This is a clear illustration of the impact of *GRank* on the precision. Interestingly enough, GOSSPLE improves the precision for approximately 50% of the items when using a query expansion of 0. This is due to the fact that in GOSSPLE the tags' weights reflect their importance.

To summarize, while the direct use of the *TagMap* improves the recall over existing approaches, mostly through personalization, the precision is significantly improved through the centrality provided by *GRank*. The latter is of utmost importance as the better precision enables us to expand queries with more tags.

Synthetic traces. We also assessed the benefit of GOSSPLE, on specific cases, through two synthetic traces. Due to space limitations, we only give an overview of these experiments. The first trace was generated to demonstrate the ability of GOSSPLE to address the baby-sitter example of the introduction with query

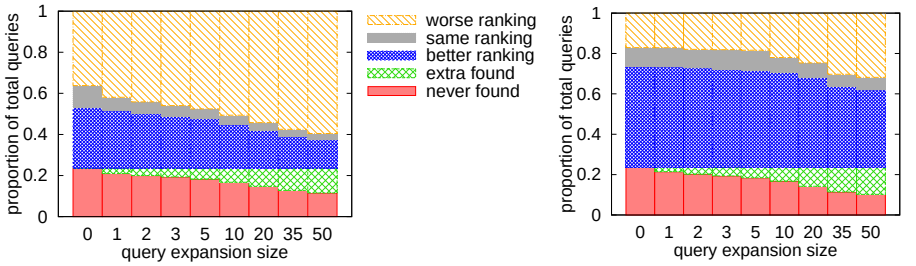

Fig. 13. Overall performance (Delicious) for Social Ranking (left) and GOSSPLE (right)

expansion. The measures show that the personalized view, provided by GOSSPLE and leveraged in the GOSSPLE query expansion system, efficiently clustered users into interest communities enabling users interested in international schools and English novels to expand babysitter with teaching assistant. The second trace shows that the GOSSPLE personalization of *TagMap* limits the impact of a user trying to force an association between tags. We call this situation a GOSSPLE *bombing*, in reference to *Google bombing*. If an attacker builds a profile with very diverse items, then no node adds the attacker to its *GNet* and the attack fails. If the attacker targets a specific community, then it can have an impact on their query expansion, but the number of users affected is very limited.

5 Related Work

5.1 Semantic overlays

Explicit approaches. Many user-centric approaches [17, 18, 5] consider an explicit (predefined) social network, typically derived from systems such as Facebook, LinkedIn, LastFM or MySpace, to improve the search on the Web. The assumption here is that the explicit, declared, relationships between users reflect their shared interests [19] and can help their search. In many cases, however, and as pointed out in [5, 20], the information gathered from such networks turns out to be very limited in enhancing the navigation on the Web. A couple of alternative approaches exploit the unknown acquaintances of a user to improve the search further [21]. These require the user to explicitly declare a list of topics of interest and assume a strict topic classification, rendering it impossible to dynamically detect new communities. GOSSPLE goes a step further and automatically assigns acquaintances to users solely based on their common items: GOSSPLE's associations naturally evolve over time.

Implicit approaches. Some approaches form semantic acquaintances by associating users with those that successfully answered their queries in the past [12, 22–26]. In file sharing communities, these approaches gradually transform a random overlay into weakly structured communities that improve the success ratio of queries. A major drawback is the need for a warm-up phase to establish the network based on a reasonable sample of queries: the first queries are sent to random users, leading to poor results. Furthermore, because the acquaintances of a user only reflect her last few queries, queries on new topics are inefficient. GOSSPLE actively locates the acquaintances of a user independently of her past queries but based on her (full interest) profile instead. In order to avoid the warm-up phase, some (active) clustering approaches rely on gossip. [13] uses the number of shared files in common as a proximity metric. While this approach improves search, it overloads generous nodes that share many files. [27] considers the cosine similarity of the users as a metric to penalize non-shared interests. This gives better performance than simple overlap and underlies our GOSSPLE metric. These approaches also tend to choose uniform acquaintances that only reflect the primary interest of a user, while GOSSPLE spreads the acquaintances

among all the interests of a user. As discussed in [28], a user has usually several areas of interests, typically non-correlated: on average, a node requires three clusters to find 25% of the data it is looking for. [29] uses gossip to select semantic acquaintances for RSS feed transmission. The selection algorithm increases the score of nodes that provide items not covered by other acquaintances. GOSSPLE's multi-interest metric can be considered a generalization of this idea.

In [30], a centralized ontology-based analysis of all the items in the system assigns a type to each one of them and infers users' interests. This procedure is centralized and relies on an external source of knowledge for the classification, while GOSSPLE is fully distributed and only uses the item interest pattern of users as a source of information. In [31], the attributes of the items generate a navigable overlay. This approach is centered on items rather than users, and does not scale with the number of items a user shares. It is also limited to items which carry a lot of metadata, like music or text files. The system proposed in [32] is fully distributed: it clusters items in order to obtain semantic coordinates. The node joins the unstructured community responsible for most of its items and obtains small-world links to other communities. Although the node has to advertise the items which are not part of its main community, this approach scales quite well with the size of the user profile. However, it suffers from the same drawback as [31] and requires items with a lot of metadata for clustering.

5.2 Query expansion

Personalized systems. Personalized query expansion has so far considered two approaches. The first consists in leveraging information about users' past behavior. In [33], the query is expanded by tags already used in previous queries. In [34], the query is expanded using the information available on the user's local computer. The tags are chosen with a local search engine and completed by user feedback. In both cases, no information is inferred from other users. The second approach [17, 18, 5] consists in enhancing the queries using an explicit social network of the user. As already discussed, this is not always appropriate.

Global approaches. Centralized Web search engines often rely on handwritten taxonomies (Wordnet, Open Directory Project, Yahoo! Directories) to expand queries. Instead, we only consider knowledge that is automatically extracted from the user's profile. [35] proposes a query-expansion framework that uses social relations to rank results. However, only the scoring model is personalized. The query-expansion mechanism exploits the co-occurrence of tags in the full document collection leading to a non-personalized query-expansion output. Social Ranking [4] is somehow similar but relies on the cosine similarity of tags to infer their proximity. This is, in a sense, the closest to our query expansion technique. We presented our comparison in Section 4.4.

6 Concluding Remarks

We presented GOSSPLE, an Internet-scale protocol that discovers connections between users and leverages them to enhance navigation within the Web 2.0.

With little information stored and exchanged, every GOSSPLE node (user) is associated with a relevant network of anonymous acquaintances. Nodes can join and leave the system dynamically and the profile of a node that left the system is eventually automatically removed from all personalized networks. No central authority is involved and there is no single point of failure. Decentralization makes it furthermore possible to effectively perform certain tasks, such as computing the *TagMap* and the *GRank*, which would have been computationally prohibitive in a centralized system. Interestingly, GOSSPLE naturally copes with certain forms of free-riding: nodes do need to participate in the gossiping in order to be visible and receive profile information. As we have also shown, the impact of arbitrary tagging, or even individual malicious tagging, is very limited.

Yet, GOSSPLE has some limitations and many extensions might need to be considered. Our gossip-on-behalf approach is simple and lightweight, but users may require additional security even by enduring a higher network cost. It would be interesting to devise schemes where extra costs are only paid by users that demand more guarantees. It would also be interesting to explore the benefits of a social network of explicit friends. For instance, GOSSPLE could take such links into account as a ground knowledge for establishing the personalized network of a user and automatically add new implicit semantic acquaintances. This would pose non-trivial anonymity challenges.

References

1. Web 2.0. www.citeulike.org www.delicious.com www.flickr.com www.last.fm.
2. Yildirim, H., Krishnamoorthy, M.S.: A random walk method for alleviating the sparsity problem in collaborative filtering. In: RecSys. (2008) 131–138
3. Cattuto, C., Benz, D., Hotho, A., Stumme, G.: Semantic analysis of tag similarity measures in collaborative tagging systems. In: OLP3. (2008) 39–43
4. Zanardi, V., Capra, L.: Social ranking: uncovering relevant content using tag-based recommender systems. In: RecSys. (2008) 51–58
5. Bender, M., Crecelius, T., Kacimi, M., Miche, S., Xavier Parreira, J., Weikum, G.: Peer-to-peer information search: Semantic, social, or spiritual? TCDE **30**(2) (2007) 51–60
6. McGill, M., Salton, G.: Introduction to modern information retrieval. (1986)
7. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based peer sampling. TOCS **25**(3) (2007)
8. Bortnikov, E., Gurevich, M., Keidar, I., Kliot, G., Shraer, A.: Brahms: byzantine resilient random membership sampling. In: PODC. (2008) 145–154
9. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. CACM **13**(7) (1970) 422–426
10. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: SP. (1997) 44–59
11. Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.D.: Sybilguard: defending against sybil attacks via social networks. TON **16**(3) (2008) 576–589
12. Handurukande, S.B., Kermarrec, A.M., Le Fessant, F., Massoulie, L., Patarin, S.: Peer Sharing Behaviour in the eDonkey Network, and Implications for the Design of Server-less File Sharing Systems. In: EuroSys. (2006) 359–371

13. Voulgaris, S., van Steen, M.: Epidemic-style management of semantic overlays for content-based searching. In: Europar. (2005) 1143–1152
14. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30**(1-7) (1998) 107–117
15. Abiteboul, S., Preda, M., Cobena, G.: Adaptive on-line page importance computation. In: WWW. (2003) 280–290
16. Fogaras, D., Rácz, B., Csalogány, K., Sarlós, T.: Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. Internet Mathematics **2**(3) (2005) 333–358
17. Amer-Yahia, S., Benedikt, M., Lakshmanan, L., Stoyanovich, J.: Efficient network aware search in collaborative tagging sites. In: VLDB. (2008) 710–721
18. Mislove, A., Gummadi, K.P., Druschel, P.: Exploiting social networks for internet search. In: HotNets. (2006)
19. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: IMC. (2007) 29–42
20. Ahn, Y.Y., Han, S., Kwak, H., Moon, S., Jeong, H.: Analysis of topological characteristics of huge online social networking services. In: WWW. (2007) 835–844
21. Khambatti, M., Ryu, K.D., Dasgupta, P.: Structuring peer-to-peer networks using interest-based communities. In: DBISP2P. (2003) 48–63
22. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient content location using interest-based locality in peer-to-peer systems. INFOCOM **3** (2003) 2166–2176
23. Cholvi, V., Felber, P., Biersack, E.: Efficient search in unstructured peer-to-peer networks. In: SPAA. (2004) 271–272
24. Chen, G., Low, C.P., Yang, Z.: Enhancing search performance in unstructured p2p networks based on users' common interest. TPDS **19**(6) (2008) 821–836
25. Sedmidubsky, J., Barton, S., Dohnal, V., Zezula, P.: Adaptive approximate similarity searching through metric social networks. In: ICDE. (2008) 1424–1426
26. Eyal, A., Gal, A.: Self organizing semantic topologies in p2p data integration systems. In: ICDE. (2009) 1159–1162
27. Jin, H., Ning, X., Chen, H.: Efficient search for peer-to-peer information retrieval using semantic small world. In: WWW. (2006) 1003–1004
28. Le-Blond, S., Guillaume, J.L., Latapy, M.: Clustering in p2p exchanges and consequences on performances. In: IPTPS. (2005) 193–204
29. Patel, J.A., Rivière, ı., Gupta, I., Kermarrec, A.M.: Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. Computer Networks **53**(13) (2009) 2304–2320
30. Crespo, A., Garcia-Molina, H.: Semantic overlay networks for p2p systems. In: AP2PC. (2004) 1–13
31. Banaei-Kashani, F., Shahabi, C.: Swam: a family of access methods for similarity-search in peer-to-peer data networks. In: CIKM. (2004) 304–313
32. Li, M., Lee, W.C., Sivasubramaniam, A., Zhao, J.: Ssw: A small-world-based overlay for peer-to-peer search. TPDS **19**(6) (2008) 735–749
33. Carman, M., Baillie, M., Crestani, F.: Tag data and personalized information retrieval. In: SSM. (2008) 27–34
34. Jie, H., Zhang, Y.: Personalized faceted query expansion. In: SIGIR. (2006)
35. Bender, M., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J.X., Schenkel, R., Weikum, G.: Exploiting social relations for query expansion and result ranking. In: ICDE Workshops. (2008) 501–506