2013 IEEE 27th International Symposium on Parallel & Distributed Processing

WHATSUP: A Decentralized Instant News Recommender

Antoine Boutet*, Davide Frey*, Rachid Guerraoui†, Arnaud Jégou* and Anne-Marie Kermarrec*†
*INRIA Rennes, France
Email: antoine.boutet,davide.frey,arnaud.jegou,anne-marie.kermarrec@inria.fr
†EPFL, Switzerland
Email: rachid.guerraoui@epfl.fr

***Abstract*—We present WHATSUP, a collaborative filtering system for disseminating news items in a large-scale dynamic setting with no central authority. WHATSUP constructs an implicit social network based on user profiles that express the opinions of users about the news items they receive (*like-dislike*). Users with similar tastes are clustered using a similarity metric reflecting long-standing and emerging (dis)interests. News items are disseminated through a novel heterogeneous gossip protocol that (1) biases the orientation of its targets towards those with similar interests, and (2) amplifies dissemination based on the level of interest in every news item.**

We report on an extensive evaluation of WHATSUP through (a) simulations, (b) a ModelNet emulation on a cluster, and (c) a PlanetLab deployment based on real datasets. We show that WHATSUP outperforms various alternatives in terms of accurate and complete delivery of relevant news items while preserving the fundamental advantages of standard gossip: namely, simplicity of deployment and robustness.

***Keywords*-recommendation system; social networks; epidemic protocols**

I. INTRODUCTION

The stream of news items we are exposed to is huge and keeps growing exponentially. This calls for automatic techniques to filter the right content for every one, alleviating the need to spend a substantial amount of time browsing information online. Explicit subscription-based approaches (*e.g.* RSS, pub/sub, online social networks) are not always relevant in this context: they either filter too much or not enough. *Personalized news recommender systems*, based on so-called social or collaborative filtering (CF) [1], are much more appropriate for they operate in a dynamic and fine-grained manner to automate the celebrated *word-of-mouth* pattern by which people recommend useful items to each other. However, CF approaches require the maintenance of huge amounts of information as well as significant computation resources, especially in the context of continuous streams of news items that must be instantly delivered to users that potentially change interests over time.

The motivation of this work is to determine whether a CF instant news system is feasible in a completely decentralized manner. Intuitively, a P2P approach is attractive because it naturally scales and circumvents a central entity that controls all user profiles potentially exploiting them for commercial purposes. Yet, the absence of a central authority with global knowledge makes the filtering very challenging and calls for CF schemes that need to cope with partial and dynamic interest profiles.

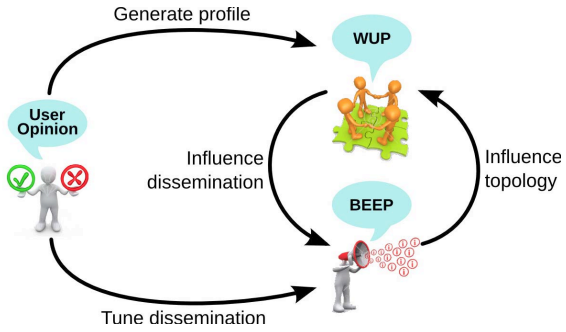


Figure 1: *Interactions between (1) user opinion; (2)* WUP*: implicit social network; (3)* BEEP*: news dissemination protocol*

We present in this paper WHATSUP: the first decentralized instant news recommender system. WHATSUP consists of a simple user interface and two distributed protocols: WUP and BEEP (Figure 1), which are key in providing an *implicit publish-subscribe* abstraction. They enable users to receive published items without having to specify explicit subscription filters. The user interface captures the opinions of users on the news items they receive through a simple *like/dislike* button. A user *profile* collects the resulting implicit interests in a vector associating news items with user opinions. This provides the driving information for the operation of WUP and BEEP.

The axiom underlying WHATSUP, as for any CF scheme [1], is that users who have exhibited similar tastes in the past are likely to be interested in the same news items in the future.

WUP maintains a dynamic implicit social network, a directed graph linking nodes (reflecting users) with similar interests. WUP periodically samples the network by gossiping profiles and connects similar users by mixing randomization to seek completeness (or *recall*), and similarities to seek accuracy (or *precision*). The similarity metric we consider accounts for the ever-changing interests of users and prevents the formation of isolated islands of interest.

News items are disseminated using BEEP (Biased EpidEmic Protocol), a novel heterogeneous epidemic protocol

1530-2075/13 $26.00 © 2013 IEEE
DOI 10.1109/IPDPS.2013.47

obeying the *explore-and-exploit* principle. The protocol biases dissemination towards nodes that are likely to have similar tastes (*exploit*), while introducing enough randomness and serendipity (ability of making fortunate discoveries while looking for something unrelated) to tolerate the inherent unreliability of the underlying network as well as to prevent interesting news items from being isolated within specific parts of the network (*explore*). If a user likes a news item, BEEP forwards it along the (implicit) social-network topology constructed using WUP. Otherwise, BEEP gives the item a chance to visit other parts of the network. The news-dissemination process generates a wave of profile updates, in turn potentially impacting the WUP network topology. Unlike classical gossip protocols [2], which aim at delivering news items to all users, BEEP targets specific subsets of users determined dynamically for each news item. To our knowledge, it is the first gossip protocol to provide *heterogeneity* along multiple dimensions: *amplification* and *orientation*. *Amplification* tunes the number of times a node gossips a news item (its *fanout*). This acts as a social filter based on the opinions of the users exposed to news items. *Orientation* biases the choice of gossip targets towards users with similar tastes.

We fully implemented WHATSUP and we extensively evaluated it both by simulation and by deploying it over a cluster as well as on PlanetLab. Specifically, our results compare the performance of WHATSUP against various alternatives, including social cascades, a traditional topic-based pub/sub system, as well as distributed CF schemes. We show that WHATSUP outperforms competitors in terms of precision (*i.e.* accuracy), recall (*i.e.* completeness) and their harmonic mean (*i.e.* F1-Score) on several datasets: a synthetic one, a real one crawled from Digg, and a news survey we conducted ourselves. For instance, we show on the survey dataset that WHATSUP improves the precision of the dissemination by up to 5% upon traditional gossip protocols whilst requiring only less than half the messages and preserving the robustness of gossip protocols, *e.g.*, more than 20% message loss in our cluster experiment has a negligible impact. Our similarity metric increases the F1-Score by 10% on average compared to the traditional cosine similarity metric [3] for an equivalent cost on the network (*i.e.* number of messages). Finally, WHATSUP decreases the quality of the dissemination by only 5% when compared to its centralized version with global knowledge.

To summarize, this paper presents two contributions, each, we believe, interesting in its own right: a clustering protocol, WUP, integrating a proximity metric, and BEEP, a heterogeneous dissemination protocol. The integration of these protocols within the same coherent system, of which we provide an implementation and extensive evaluation can also be viewed as an actual contribution. The system is available at http://www.irisa.fr/asap/whatsup.

II. WUP

WUP builds and maintains an implicit social network and is itself based on two gossip protocols. The lower-layer random-peer-sampling (RPS) protocol [4] ensures connectivity by building and maintaining a continuously changing random topology. The upper-layer clustering protocol [5] uses this overlay to provide nodes with the most similar candidates to form their WUP social network.

At each node n, each protocol maintains a *view*, a data structure containing references to other nodes: the RPS neighbors and the WUP neighbors. Each entry in each view is associated with a node and contains *(i)* its IP address, *(ii)* its node ID, *(iii)* its *profile* (as defined in Section II-B), as well as *(iv)* a timestamp specifying when the information in the entry was generated by the associated node. Periodically, each protocol selects the entry in its view with the oldest timestamp [4] and sends it a message containing its profile with half of its view in the case of the RPS (typical parameter in such protocols), or its entire view for WUP (the view sizes and frequencies of each protocol are given in Section IV).

In the RPS, the receiving node renews its view by keeping a random sample of the union of its own view and the received one. The union of the RPS views represents a continuously changing random graph [4]. In WUP, instead, the receiving node selects the nodes from the union of its own and the received views whose profiles are closest to its own according to a similarity metric. This metric is an asymmetric variation of the well-known cosine similarity [3]: it seeks to maximize the number of items that were liked in both profiles being compared. It also strives to minimize spam by discouraging a node, n, with profile P_n, from selecting a neighbor, c, with profile P_c, that explicitly dislikes the items that n likes. We achieve this by dividing the number of liked items in common between the two profiles by the number of items liked by n on which c expressed an opinion. We define $sub(P_n, P_c)$ as the subset of the scores in P_n associated with the items that are present in P_c. By further dividing by the number of items liked by c (as in cosine similarity), we then favor neighbors that have more restrictive tastes. The asymmetric structure of this metric is particularly suited to push dissemination (*i.e.* users choose the next hops of news items but have no control on who sends items to them) and improves cold start with respect to cosine similarity as explained in Section V-A.

$$Similarity(n, c) = \frac{sub(P_n, P_c) \cdot P_c}{\|sub(P_n, P_c)\| \, \|P_c\|}$$

A. News item

A news item consists of a title, a short description, and a link to further information. The source of an item (the user publishing it) associates it with a timestamp indicating its creation time and a dislike-counter field initialized to zero

that sums the number of dislikes obtained by the item. The WUP algorithm also uses an 8-byte hash as the identifier of the news item. This hash is not transmitted but computed by nodes when they receive the item.

B. Profiles

WUP records information about interest for items in profile data structures. A profile is a set of triplets: identifier, timestamp, and score; $P \in \{< id, t, s > | id \in \mathcal{N}, t \in T, s \in [0,1]\}$. Identifier and timestamp are defined as above, and each profile contains only a single entry for a given identifier. The score, instead, represents the level of interest for an item: 1 meaning interesting, and 0 not interesting.

WUP associates each node with a profile, the *user profile* ($\tilde{P}$), which contains information about the node's own interests. The scores associated with this profile are integer values (like-dislike). To disseminate news items, nodes employ an additional profile structure, the *item profile* (P^I). Unlike a user profile, the item profile is associated with a news item. Its score values are real numbers and are obtained through the aggregation of the profiles of the users that liked the item along its dissemination path. As a result, two copies of the same item along two different paths will have different profiles. This causes an item profile to reflect the interests of the portion of the network it has traversed. The item profile can also be viewed as a community profile expressing the interests of an implicit social network of nodes.

C. Updating profiles

Updating user profiles ($\tilde{P}$): A node updates its profile whenever it expresses its opinion on a news item either by clicking the *like* or the *dislike* button (line 5 or 7 in Algorithm 1), or when generating a new item (line 14). In either case, the node inserts a new tuple containing the news item's identifier, its timestamp, and a score value of 1 if it liked the item and 0 otherwise.

Updating item profiles (P^I): The item profile of an item I records the interests of the users who like I by aggregating their profiles along I's path. This works as follows. Let n be a node that likes I. When n receives I for the first time, it first updates its own user profile, $\tilde{P}$, as described above. Then, it iterates through all the tuples in $\tilde{P}$ (line 3). Let id be the identifier of one such tuple and let s^n be its score. Node n checks if I's item profile already contains a tuple for id (`addToNewsProfile` function). If so (line 20), let s be the tuple's score value in I's item profile; n replaces s with the average between s and s^n—the score in n's user profile. This averaging gives the same weight to both scores, s and s^n: it thus personalizes I's item profile according to n's interests. If I's item profile contains no tuple for id, node n inserts the tuple from its own user profile into I's item profile (line 22). When a new item is generated (function `generateNewsItem` in Algorithm 1), the source initializes the corresponding item profile by integrating its own user profile (line 15).

Algorithm 1: WUP: receiving / generating an item.

```
1  on receive (item < id^I, t^I >, profile P^I, dislike counter d^I) do
2      if iLike(id^I) then
3          for all < id, t, s > ∈ P̃
4              addToNewsProfile(< id, t, s >, P^I)
5          add < id^I, t^I, 1 > to P̃
6      else
7          add < id^I, t^I, 0 > to P̃
8      for all < id, t, s > ∈ P^I
9          if t older than profile window then
10             remove < id, t, s > from P^I
11     BEEP.forward((< id^I, t^I >, P^I, d^I))
12 function generateNewsItem(item id^I)
13     P^I ← ∅;   d^I ← 0;   t^I ← currentTime
14     add < id^I, t^I, 1 > to P̃
15     for all < id, t, s > ∈ P̃
16         addToNewsProfile(< id, t, s >, P^I)
17     BEEP.forward((< id^I, t^I >, P^I, d^I))
18 function addToNewsProfile(< id, t, s^n >, P^I)
19     if ∃s| < id, ∗, s > ∈ P^I then
20         s ← (s + s^n)/2
21     else
22         P^I ← < id, t, s^n >
```

D. Initialization

A node, n, that is joining the system for the first time (*cold start*) contacts a random node, and inherits its RPS and WUP views. It then builds a fresh profile by selecting and rating the 3 most popular news items from the profiles of the nodes in its the selected RPS view. This process results in a profile and in a WUP view that are very unlikely to match n's interests. However, it provides n with a way to enter the WUP social network. Because the WUP metric takes into account the size of user profiles, nodes with very small profiles containing popular items such as joining nodes are more likely to be part of the WUP views of other nodes and quickly receive additional news items. This allows them to fill their profiles with more relevant content, thereby acquiring closer neighbors.

E. Profile window

The information stream is continuously evolving. In order to take into account only the current interests of users and to dynamically connect similar users, all profiles are cleaned of old items. Specifically, each node periodically purges its user profile of all the tuples whose timestamps are older than a profile window. Similarly, nodes purge item profiles of non-recent items before forwarding items to BEEP for dissemination (lines 8 to 10). The value of this profile window defines the reactivity of the system with respect to user interests as discussed in Section IV-D.

It is important to note that the profile window also causes inactive users who have not provided ratings during the current window to have empty profiles, thus being considered as new nodes. Yet, as in the case of initialization, the WUP metric allows these users to reintegrate quickly as soon as they connect and resume receiving news items.

III. BEEP

BEEP is a novel gossip-based dissemination protocol embodying two mechanisms: *orientation* and *amplification*, both triggered by the opinions of users on news items. Orientation leverages the information provided by WUP to direct news items towards the nodes that are most likely to be interested in them. Amplification varies the number of dissemination targets according to the probability of performing a useful forwarding action. Orientation and amplification make BEEP the first user-driven gossip protocol to provide heterogeneity in the choice as well as in the number of dissemination targets, achieving differentiated delivery. BEEP follows the well-known SIR (Susceptible, Infected, Removed) [2] model. A node receiving a news item for the first time updates the item's profile as described in Section II-C. Then, it forwards the item to fanout (f) other nodes chosen according to its opinion on the item, as described in the following. A node receiving an item it has already received simply drops it.

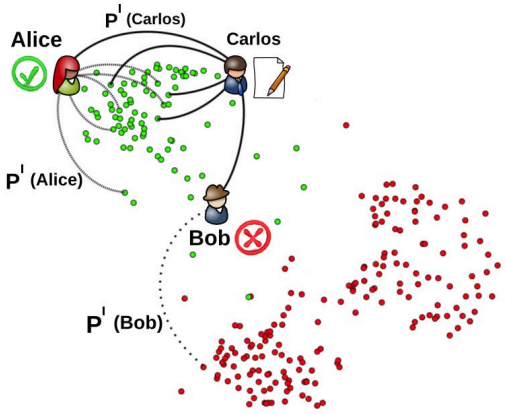


Figure 2: *Orientation and amplification mechanisms of Beep*

A. Forwarding a disliked item

With reference to Algorithm 2 and Figure 2, consider Bob, who does not like item I sent by Carlos. BEEP first verifies if the dislike-counter field of the item has already reached the prescribed TTL (line 25). If it has, it drops the item. Otherwise it increments its value, and achieves orientation by identifying the node from Bob's RPS view whose user profile is closest to the item's profile (line 27) and forwards the item to it (line 34). The item profile allows BEEP's orientation mechanism to identify a target that is reasonably close to someone who liked the item, even if its topic falls outside Bob's interests. The use of a fanout of 1, instead, accounts for unexpected interests and addresses serendipity by giving news items the chance to visit portions of the overlay where more interested nodes are present. At the same time, it also prevents non-interesting items from invading too many users.

Algorithm 2: BEEP: forwarding a news item.

```
23 function forward((< id^I, t^I >, profile P^I, dislike counter d^I))
24     if ¬ iLike(id^I) then
25         if d^I < TTL then
26             d^I ← d^I + 1
27             N ← selectMostSimilarNode(P^I, RPS)
28         else
29             N ← ∅
30     else
31         N ← selectRandomSubsetOfSize(WUP, f_LIKE)
32     if N ≠ ∅ then
33         for all n ∈ N
34             send < id^I, t^I > with associated P^I and d^I to n
```

B. Forwarding a liked item

Consider now Alice (Figure 2), who instead finds item I interesting. BEEP achieves orientation by selecting dissemination targets from her social network (WUP view). Unlike the profiles in the RPS view, those in the WUP view are relatively similar to each other. However, to avoid forming too clustered a topology by selecting only the closest neighbors, BEEP selects its targets randomly from the WUP view (line 31 in Algorithm 2). Moreover, since the targets' interests are expected to be similar to those of the node, BEEP amplifies I by selecting a relatively large subset of f_{LIKE} (*like fanout*) nodes instead of only one node, thus giving I the ability to reach more interested nodes.

IV. EXPERIMENTAL SETUP

In this section, we provide the experimental setup of WHATSUP's evaluation: the workloads, the competitors we compared against, WHATSUP's parameters, and the evaluation metrics we use to assess the performance of WHATSUP.

A. Datasets

We evaluate WHATSUP using several datasets: *(i)* a 3180-user synthetic trace derived from Arxiv, *(ii)* a Digg dataset crawled in late 2010, and *(iii)* a survey conducted in our lab providing a real set of WHATSUP users. Table I summarizes the figures of the workloads used in our evaluations.

Synthetic dataset: To validate WHATSUP without the artifacts of real datasets, we identified distinct groups among the 5242 users in the Arxiv dataset (covering scientific collaborations between authors [6]) using a community-detection algorithm [7]. This allows us to deal with clearly

Name	Number of users	Number of news
Synthetic	3180 Arvix Users	2000
Digg	750	2500
WHATSUP Survey	480	1000

Table I: Summary of the workloads

defined communities of interest, thus enabling the evaluation of WHATSUP's performance in a clearly identified topology. The resulting dataset contains 21 communities ranging in size from 31 to 1036, for a total of 3703 users. For each community, we use a random subset of nodes as sources to disseminate 120 news items (for a total of about 2000).

Digg dataset: Digg is a centralized social-news website designed to help users discover and share content. It disseminates news items along the edges of an explicit social network (*i.e.* cascading). Relying on explicitly declared friends, as in Digg, is known to limit the content that can be received [8] by substantially influencing decision making [9]. Basically, users are only exposed to the content forwarded by their friends, while other items may be of interest to them. To remove this bias, we extracted for each user, u, the categories of the news items she generates. We then defined user u's interests by including all the news items associated with these categories. We collected traces from Digg over 3 weeks in 2010. The resulting dataset consists of 750 users and 2500 news from 40 categories.

Dataset from a WHATSUP *user survey:* We also conducted a survey on 200 news items involving 120 colleagues and relatives. We selected news randomly from a set of RSS feeds illustrating various topics (culture, politics, people, sports, ...). We exposed this list to our test users and gathered their reactions (like/dislike) to each news item. This provided us with a small but *real* dataset of WHATSUP users exposed to exactly the same news items. To scale our system, we generated 4 instances of each user and news item in the experiments. Yet, the resulting bias affects both WHATSUP and the state-of-the-art solutions we compare against.

B. WHATSUP *Competitors*

In order to demonstrate the effectiveness of WHATSUP, we evaluate it against the following alternatives:

Explicit cascading: Cascading is a dissemination approach followed by several social applications, e.g., Twitter, Digg. Whenever a node *likes* (tweets in Twitter and diggs in Digg) a news item, it forwards it to all of its explicit social neighbors. We compare WHATSUP against cascading in the only dataset for which an explicit social network is available, namely Digg.

Complete explicit pub/sub: WHATSUP can be seen as an *implicit publish/subscribe* (pub/sub) system turning interests into implicit subscriptions. Typically, pub/sub systems are explicit: users explicitly choose specific topics [10], [11]. Here, we compare WHATSUP against C-Pub/Sub, a centralized topic-based pub/sub system achieving complete dissemination. C-Pub/Sub guarantees that all the nodes subscribed to a topic receive all the associated items. C-Pub/Sub is also ideal in terms of message complexity as it disseminates news items along trees that span all and only their subscribers. For the sake of our comparison, we extract explicit topics from keywords associated with the RSS feeds in our survey. Then we subscribe a user to a topic if she likes at least one item associated with that topic.

Decentralized collaborative filtering: In a decentralized CF scheme based on nearest-neighbor technique, when a node receives a news item it likes, it forwards it to its k closest neighbors according to some similarity metric. We implemented two versions of this scheme: one relying on the same metric as WHATSUP (CF-WUP) and one relying on cosine similarity [3] (CF-Cos). While it is decentralized, this scheme does not benefit from the orientation and amplification mechanisms provided by BEEP. More specifically, it takes no action when a node does not like a news item.

Centralized version of WHATSUP*:* We also compare WHATSUP with a centralized system (C-WHATSUP) gathering the *global knowledge* of all the profiles of its users and news items. C-WHATSUP leverages this global information (vs a restricted sample of the network) to boost precision using complete search. When a user likes a news item, the server delivers it to the f_{LIKE} closest users according to the cosine similarity metric. In addition, it also provides the item to the f_{LIKE} users with the highest correlation with the *item's profile*. When a user does not like an item, the server presents it to the f_{DISLIKE} nodes whose profiles are most similar to the item's profile (up to TTL times).

C. Evaluation metrics

We consider two types of metrics in our evaluation. User metrics measure the quality of WHATSUP's dissemination and its ability to filter content. They are important for users to decide whether to adopt WHATSUP as a system. In contrast, system metrics are transparent to users but are crucial to assessing the effectiveness of our solution.

User metrics: We evaluated WHATSUP along the traditional metrics used in information-retrieval systems: *recall* (*i.e.* completeness) and *precision* (*i.e.* accuracy). Both measures are in $[0, 1]$. For an item, a recall of 1 means that all interested users have received the item. Yet, this measure does not account for spam since a trivial way to ensure a maximum recall is to send all news items to all users. This is why precision is required. A precision of 1 means that the news item has reached only the users that are interested in it. An important challenge in information retrieval is to provide a good trade-off between these two metrics. This is expressed by the F1-Score, defined as the harmonic mean of precision and recall [12].

$$Precision = \frac{|\{interested\ \ users\} \cap \{reached\ \ users\}|}{|\{reached\ \ users\}|}$$

$$Recall = \frac{|\{interested\ \ users\} \cap \{reached\ \ users\}|}{|\{interested\ \ users\}|}$$

$$F1 - Score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

System metrics: To evaluate the behavior of WHATSUP from a systems perspective, we first consider the network traffic it generates. For simulations, we compute the total number of *sent messages*. For our implementation, we instead measure the average consumed *bandwidth*. Throughout our evaluation, we examine results obtained over a wide range of fanout values by plotting the F1-Score against the fanout, and against the number of generated messages. The F1-Score for corresponding fanout values makes it possible to understand and compare the behavior of WHATSUP and its competitors under similar conditions. The F1-Score for corresponding numbers of messages, instead, gives a clearer picture about the trade-offs between recommendation quality and cost. Two different protocols operating at the same fanout, in fact, do not necessarily generate the same amount of traffic.

D. WHATSUP *system parameters*

The operation of WHATSUP is controlled by a number of system parameters. The first two parameters we consider are the WUP view size (WUP_{vs}) and the BEEP-I-like fanout (f_{LIKE}). Clearly, the former must be at least as large as the latter. As a node forwards a liked news item to random neighbors among its WUP view, a WUP_{vs} close to f_{LIKE} boosts precision while a large WUP_{vs} compared to f_{LIKE} increases recall. We set the value of WUP_{vs} to the double of f_{LIKE} as experiments provide the best trade-off between precision and recall for these values.

The third important parameter is the RPS view size. It directly impacts the potential of WUP to discover new nodes. We set its value to 30 to strike a balance between the need to discover information about nodes, the cost of gossiping, and the need to retain some randomness in the selection of WUP neighbors. Too large values would lead the WUP view to converge too fast, hampering the ability to address non-foreseeable interests (serendipity). Nonetheless, we verified that our protocol provides good performance with values between 20 and 40 in the considered traces.

The BEEP TTL controls WHATSUP's *serendipity*, but it should not be too large in order not to hamper precision. We therefore set it to 4, and examine its impact in Section V-B.

Finally, the size of the profile window determines WHATSUP's ability to adapt to dynamic and emerging interests of users. We set its value to 13 gossip cycles, corresponding to $1/5$ of the experiment duration, according to an analysis of its influence on the F1-Score. A size between $1/5$ and $2/5$ of the whole period gives the best F1-Score, while smaller or larger values make WHATSUP either too dynamic or not enough. For practical reasons, our simulations use the duration of a gossip cycle as a time unit to represent the length of the profile window. Yet, the actual duration of a gossip cycle is important and determines the dynamic response of our system. We discuss this parameter and its impact when evaluating our deployment (Section V-F).

Parameter	Description	value
RPS_{vs}	Size of the random sample	30
RPS_{f}	Frequency of gossip in the RPS	1h
WUP_{vs}	Size of the social network	$2f_{\text{LIKE}}$
Profile window	News item TTL	13 cycles
BEEP TTL	Dissemination TTL for dislike	4

Table II: WHATSUP *parameters - on each node*

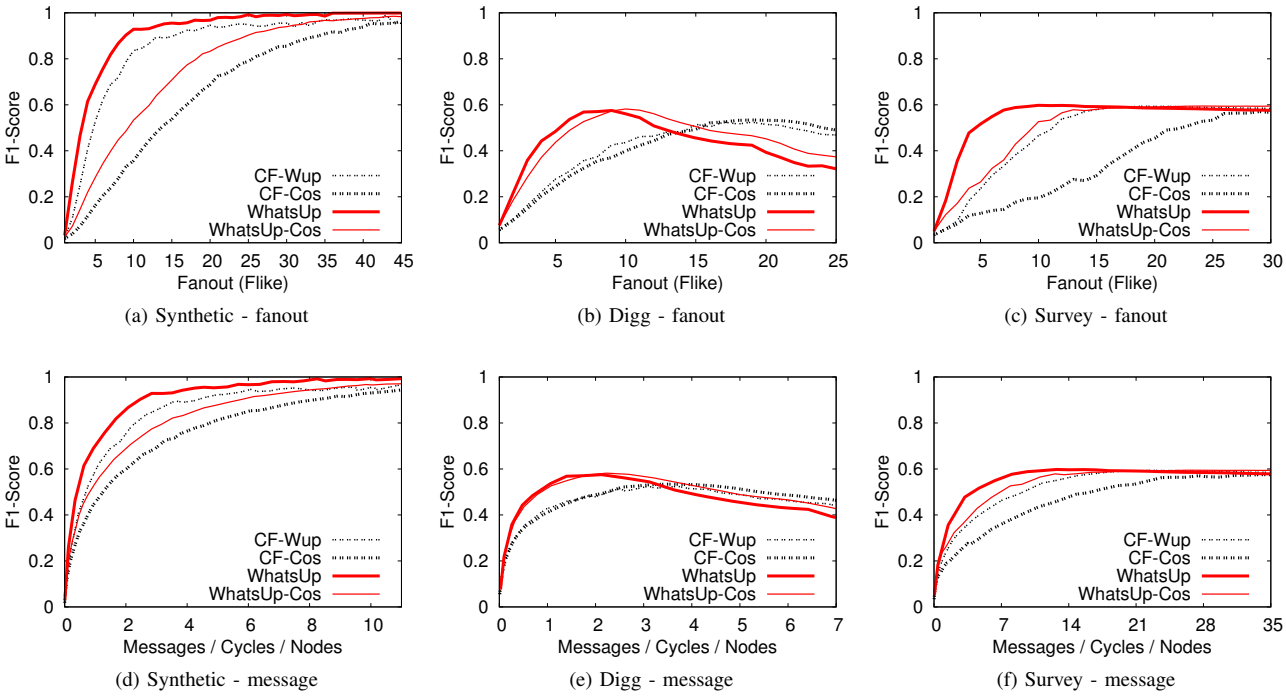
V. IMPLEMENTATION AND EVALUATION

WHATSUP is written in Java and available at http://www.irisa.fr/asap/whatsup. It consists of a Web user interface and a lightweight application server running on client nodes. Users only require a browser to interact with WHATSUP. The user interface is a fully dynamic Web widget that can be integrated in both dashboards and Web pages. The local application server continuously updates the widget with a stream of news items received from other nodes. To make this possible, it combines the implementations of WUP and BEEP with a lightweight local database containing user-profile information. The underlying network library, designed for the management of gossip-based overlays [13], also provides support for peers operating behind NATs.

We carried out an extensive evaluation of WHATSUP by simulation and by deploying its implementation on PlanetLab and on a ModelNet-based [14] cluster. All parameters, based on observations on a wide range of experiments on all datasets, are summarized in Table II. We present the results by highlighting each important feature of WHATSUP.

A. Similarity metric

We start by evaluating the effectiveness of the WUP metric. Figures 3a-3f compare two CF approaches and two versions of WHATSUP based, respectively, on cosine similarity (CF-Cos and WHATSUP-Cos) and our WUP metric (CF-WUP and WHATSUP). Our metric consistently outperforms cosine similarity in all datasets. Table III conveys the fact that it achieves this by improving recall over cosine similarity (by 30% for CF approaches and 15% for WHATSUP in the survey dataset with lower message cost in both cases). Moreover the relatively high precision of cosine similarity is partly an artifact of its low recall values resulting from highly clustered topologies. As a result, approaches using cosine similarity require a much larger fanout and message cost to provide the same quality of recommendation. The WUP metric generates instead topologies with a lower clustering coefficient by avoiding node concentration around hubs (an average clustering coefficient of 0.15 for WUP metric compared to 0.40 for cosine similarity in the survey



(a) Synthetic - fanout (b) Digg - fanout (c) Survey - fanout

(d) Synthetic - message (e) Digg - message (f) Survey - message

Figure 3: *F1-Score depending on the fanout and message cost*

Algorithm	Precision	Recall	F1-Score	Mess./User
Gossip ($f = 4$)	0.35	0.99	0.51	4.6k
CF-Cos ($k = 29$)	0.50	0.65	0.57	5.9k
CF-Wup ($k = 19$)	0.45	0.85	0.59	4.7k
WHATSUP-Cos ($f_{\text{LIKE}} = 24$)	0.51	0.72	0.60	4.3k
WHATSUP ($f_{\text{LIKE}} = 10$)	**0.47**	**0.83**	**0.60**	**2.4k**

Table III: *Survey: best performance of each approach*

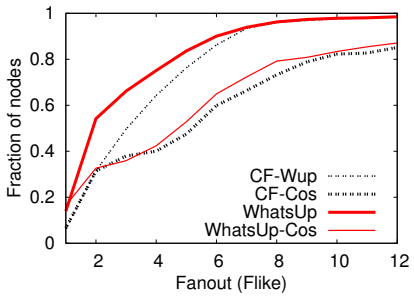


Figure 4: *Survey: Size of the LSCC depending on the approach*

dataset). In addition, the WUP metric avoids fragmenting the topology into several disconnected parts. Figure 4 shows the fraction of nodes that belong to the largest strongly connected component (LSCC) with increasing fanout values. Once all users are part of the same connected component, news items can be spread through any user and are not restricted to a subpart of the network. This corresponds to the plateaus in the F1-Score values visible in Figure 3c. The WUP metric reaches this state with fanout values around 10 both in CF-WUP and WHATSUP. This is a lot earlier than cosine similarity, which only reaches a strongly connected topology with fanout values above 15. Additional results, not plotted for space reasons, also show that the fragmentation induced by the WUP metric is consistently lower than that associated with cosine similarity even for smaller fanout values. With a fanout of 3, for instance, WHATSUP's and CF-WUP's topologies contain respectively an average of 1.6 and 2.6 components, while WHATSUP-Cos's and CF-Cos's contain respectively an average of 12.4 and 14.3.

B. Amplification and orientation

Comparing WHATSUP with CF schemes allows us to evaluate the impact of amplification and orientation. The results in Figures 3a-3f show that WHATSUP consistently outperforms CF, reaching higher F1-Score values with lower fanouts and message costs. Table III shows that it achieves recall values much higher than those of CF, with less than two thirds the message cost. This is a direct result of the amplification and dislike features, which allow an item to reach interested nodes even after hitting uninterested ones. This observation is confirmed by comparing Figure 3c with Figure 4. Even if approaches adopting the same metric result in similar topologies as conveyed by Figure 4, the performance of those that employ amplification and dislike is consistently higher for corresponding fanout values.

Table IV further illustrates the impact of the *dislike* feature

by showing, for each news item received by a node that likes it, the number of times it was forwarded by nodes that did not like it. For instance, we can see that 31% of the news items liked by nodes were forwarded exactly once by nodes that did not like them. This conveys the benefit of the dislike feature and the importance of (negative) feedback from users in giving items a chance to reach interested nodes across the entire network.

Number of dislikes	0	1	2	3	4
Fraction of news	54%	31%	10%	3%	2%

Table IV: News received and liked via dislike

Figure 5 shows the impact of the TTL value on the performances. Too low a TTL mostly impacts recall; yet values of TTL over 4 do not improve the quality of dissemination. Finally, Table III also includes the performance of a standard homogeneous gossip protocol, which achieves the worst F1-Score value of 0.51 with almost twice as many messages as WHATSUP.

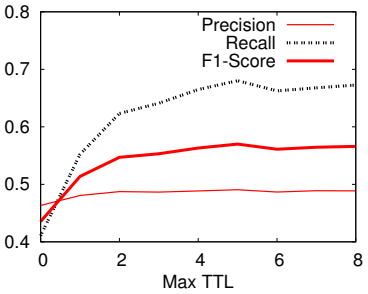


Figure 5: *Survey: Impact of the dislike feature of* BEEP

Figure 6 shows hows nodes at increasing distances from the source of a news item contribute to dissemination. We observe from the bell-shaped curve that most dissemination actions are carried out within a few hops of the source, with an average around 5. This is highly beneficial because a small number of hops leads to news items being disseminated faster.[1] Finally, the plot also confirms the effectiveness of the dislike mechanism with a non-negligible number of infections being due to dislike operations.

C. Implicit nature of WHATSUP

Next, we evaluate WHATSUP's reliance on implicit acquaintances by comparing it with two forms of *explicit* filtering: *cascading* over explicit social links, and the ideal pub/sub system, C-Pub/Sub.

The first set of results in Table V shows that WHATSUP achieves a higher F1-Score with respect to cascading. More specifically, while both approaches provide almost the same

[1]A precise analysis of dissemination latency would require knowledge of the response time of users. Such an analysis is outside the scope of this paper and is subject of ongoing work.

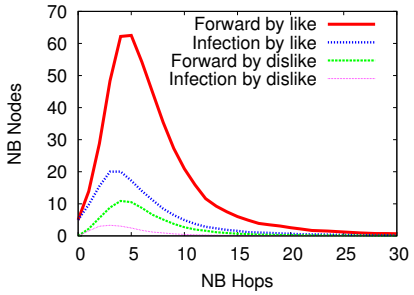


Figure 6: *Survey ($f_{\text{LIKE}} = 5$): Impact of amplification of* BEEP

level of precision, WHATSUP outperforms (by more than six times) cascading in terms of recall. The very low recall of cascading highlights the fact that the explicit social network does not necessarily connect all the nodes interested in a given topic. The low number of messages of cascading is a result of its small recall. The network traffic per infected user generated by WHATSUP is, in fact, 50% less than that of cascading (2.57K messages vs 5.27K).

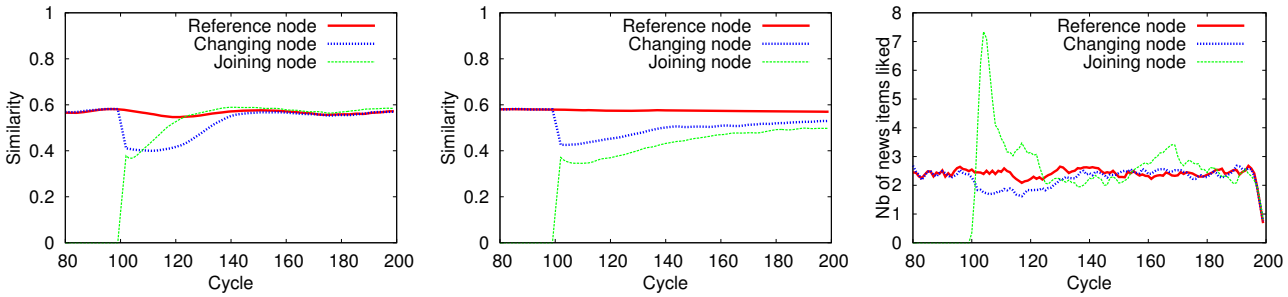
Dataset	Approach	Precision	Recall	F1-Score	Messages
Digg	Cascade	0.57	0.09	0.16	228k
	WHATSUP	**0.56**	**0.57**	**0.57**	**705k**
Survey	C-Pub/Sub	0.40	1.0	0.58	470k
	WHATSUP	**0.47**	**0.83**	**0.60**	**1.1M**

Table V: WHATSUP *vs C-Pub/Sub and Cascading*

The second set of results in the table compares WHATSUP with C-Pub/Sub. As discussed in Section IV-B, C-Pub/Sub disseminates news items to all subscribers with a minimal number of messages. Its recall is therefore 1 while its precision is only limited by the granularity of its topics. In spite of this, WHATSUP improves C-Pub/Sub's accuracy by 12% in the survey dataset with a little more than three times as many messages while conserving a good recall. This results in a better trade-off between accuracy and completeness as indicated by its higher F1-Score.

Another important advantage of WUP's implicit approach is its ability to cope with interest dynamics. To measure this, we evaluate the time required by a new node joining the network and a node changing of interests to converge to a view matching its interests both in WHATSUP (Figure 7a) and in WHATSUP-Cos (Figure 7b).

For the joining node, we select a reference node and introduce a new joining node with an identical set of interests. We then compute the average similarity between the reference node and the members of its WUP view and compare it to the same measure applied to the joining node. We repeated the experiment by randomly choosing 100 joining nodes and averaged the results. The WUP metric significantly reduces the number of cycles required by the joining node to rebuild a WUP view that is as good as that of the reference node (20 cycles for WHATSUP vs over 100 for WHATSUP-Cos).



(a) Similarity in WUP view (WHATSUP) (b) Similarity in WUP view (WHATSUP-Cos) (c) Reception of liked news items (WHATSUP)

Figure 7: *Cold start and dynamics in* WHATSUP

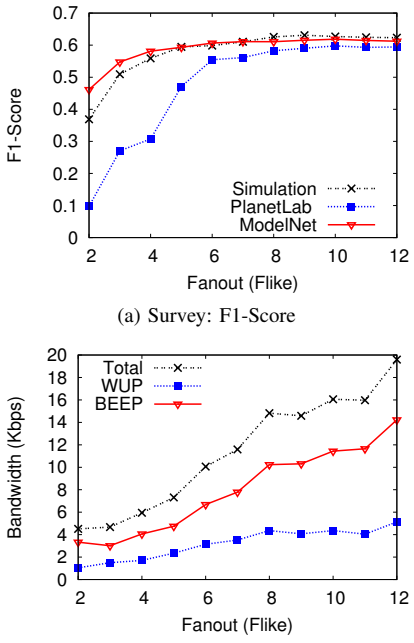
Yet, the node starts receiving news quickly as shown in Figure 7c with the peak in the number of interesting news received as soon as the node joins. This is a result of both our cold start mechanism (Section II-D) and our metric's ability to favor nodes with small profiles. Once the node's profile gets larger, the number of received news per cycle stabilizes to values comparable to those of the reference node. Nonetheless, the joining node reaches 80% of the reference node's precision after only a few cycles.

For the changing node, we select a pair of random nodes from the survey dataset and, at 100 cycles into the simulation, we switch their interests and start measuring the time it takes them to rebuild their WUP views. Figure 7 displays results obtained by averaging 100 experiments. Again, the WUP metric causes the views to converge faster than cosine similarity: 40 cycles as opposed to over 100. Moreover, the values of recall and precision for the nodes involved in the change of interests never decrease below 80% of the reference node's values. These results are clearly tied to the length of the profile window, set to about 40 cycles in these experiments. Shorter windows would in fact lead to an even more responsive behavior. We are currently evaluating this aspect on the current WHATSUP prototype. Moreover, while it may seem surprising that switching interests takes longer than joining a network from scratch, this experiment is an unlikely situation that provides an upper bound on the impact of more gradual interest changes.

Finally, the implicit nature of WHATSUP and the push nature of BEEP also make WHATSUP resilient to basic forms of content bombing. Unless a spammer node has enough resources to contact directly a large number of nodes, it will be unable to flood the network with fake news. The dislike mechanism, with its small fanout and TTL values will, in fact, limit the dissemination of clearly identified spam to a small subset of the network.

D. Simulation and implementation

We also evaluate the performance obtained by our implementation in two settings: *(i)* a 170 PlanetLab node testbed with 245 users, and *(ii)* an emulated network of 245 nodes (machines and users) deployed on a 25-node cluster equipped with the ModelNet network emulator. For practical reasons we consider a shorter trace and very fast gossip and news-generation cycles of 30sec, with 5 news items per cycle. These gossip frequencies are higher than those we use in our prototype, but they were chosen to be able to run a large number of experiments in reasonable time. We also use a profile window of 4min, compatible with the duration of our experiments (1 to 2 hours each).



(a) Survey: F1-Score

(b) Bandwidth in planetlab

Figure 8: *Implementation: bandwidth and performance*

Figure 8a shows the corresponding results obtained on the survey and compares them to those obtained through simulation on the same 245-user dataset with increasing fanout values. ModelNet results confirm the accuracy of our simulations. The corresponding curves closely match each other except from some fluctuations with small fanout

values. PlanetLab results, on the other hand, exhibit a clear decrease in performance with small fanouts. To understand this behavior, we can observe that in simulation and ModelNet, recall reaches scores above 0.50 with fanout values as small as 3. In PlanetLab, it only achieves a value of 0.18 with a fanout of 3, and goes above 0.50 only with fanouts of at least 6. The difference in recall with small fanout values can be easily explained if we observe the message-loss rates in the PlanetLab setting. With a fanout of 3, we recorded that nodes do not receive up to 30% of the news that are correctly sent to them. This is due to network-level losses and to the high load of some PlanetLab nodes, which causes congestion of incoming message queues. The impact of these losses becomes smaller when the fanout increases because BEEP is able to produce enough redundancy to recover from the missing messages.

E. Message loss

To understand the impact of lost messages, we experiment in the ModelNet network emulator with increasing loss rates affecting both BEEP and WUP messages and ranging from 0 to a huge value of 50%. Table VI shows that both protocols preserve the reliability properties of gossip-based dissemination. With a fanout of 6, the performance in terms of F1-Score is virtually unchanged with up to 20% of message loss, while it drops only from 0.60 to 0.45 when half of the messages are lost by the network layer. With a fanout of 3, the impact of message loss is clearly more important due to the smaller amount of redundancy. 20% of message loss is sufficient to cause the F1-Score to drop from 0.54 to 0.47. This explains the differences between PlanetLab and ModelNet in Figure 8a. These drops are almost uniquely determined by the corresponding recall. With a fanout of 3 and a loss rate of 50%, recall drops to 0.07, causing an artificial increase in precision, and yielding an F1-Score of 0.12, against the 0.45 with a fanout of 6.

Loss Rate	0%		5%		20%		50%	
Fanout	3	6	3	6	3	6	3	6
Recall	0.63	0.82	0.61	0.82	0.46	0.80	0.07	0.45
Precision	0.47	0.48	0.47	0.47	0.47	0.46	0.55	0.44

Table VI: *Survey: Performance versus message-loss rate*

F. Bandwidth consumption

Increasing fanout has a cost, which is highlighted by our bandwidth analysis in Figure 8. The number of times each news item is forwarded increases linearly with fanout values, causing an equally linear increase in the bandwidth consumption of BEEP. The bandwidth used by WUP also shows a slight increase with fanout due to the corresponding increase in the sizes of the WUP social networks. Nonetheless, the cost of the protocol is dominated by news. This highlights the efficiency of our implicit social-network maintenance. These experiments on a very fast trace with a gossip cycle every 30sec lead to a bandwidth consumption of about 4Kbps for WUP's view management. Our prototype is characterized by significantly lower gossip frequencies, on the order of 5min per gossip cycle. This results in a much lower average bandwidth consumption of about 0.4Kbps.

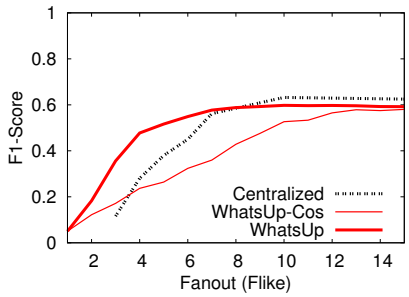


Figure 9: *Survey: centralized vs decentralized*

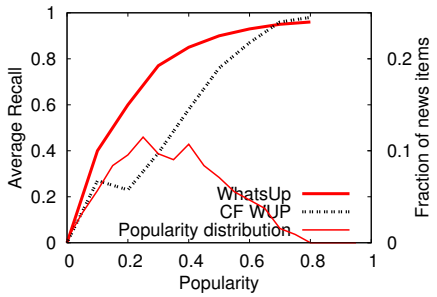


Figure 10: *Survey: recall vs popularity*

G. Partial information

To understand the impact of decentralization, we compare WHATSUP with a centralized variant, C-WHATSUP, that exploits global knowledge to instantaneously update node and item profiles. Figure 9 shows that WHATSUP provides a very good approximation of this variant (a 5% decrease of the F1-Score). More precisely, global knowledge yields better precision (17%) but slightly lower recall (14%).

H. Sociability and popularity

An additional interesting aspect is the impact of the popularity of items and the sociability of users. Figure 10 depicts the distribution of news-item popularity in the survey dataset together with the corresponding recall for WHATSUP and CF-WUP. WHATSUP performs better across most of the spectrum. Nonetheless, its improvement is particularly marked for unpopular items (0 to 0.5). This is highly desirable as popular content is typically much easier to manage than niche content. Recall values appear to converge for very popular items. However, each point in the plot represents an average over several items. An analysis of the data distribution (not shown for space reasons), instead, highlights how CF-WUP exhibits much higher variance leaving some items almost completely out of the dissemination.

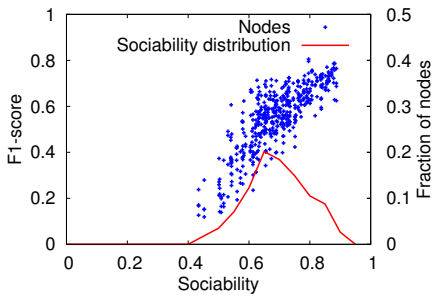


Figure 11: *Survey: F1-Score vs sociability*

WHATSUP provides instead good recall values across all items thanks to the effectiveness of its *dislike* feature.

Figure 11 instead examines how the F1-Score varies according to the sociability of users in the survey dataset. We define sociability as the ability of a node to exhibit a profile that is close to others, and compute it as the node's average similarity with respect to the 15 nodes that are most similar to it. Results confirm the expectations. WHATSUP leverages the similarity of interests between users and provides relevant results for users with alter-egos in the system. The more sociable a node the more it is exposed only to relevant content (improving both recall and precision). This acts as an incentive: the more a user exhibits a consistent behavior, the more she will benefit from the system.

VI. Related work

Gossip: Epidemic protocols [2] are well known to be simple, efficient, and robust means to disseminate information in large-scale systems. So far, they have been mainly homogeneous with respect to fanout and target selection. While some consider an adaptive fanout to control the infection patterns in the network [15], [16], their goal is for messages to reach all nodes, unlike in WHATSUP. Some approaches leverage the explicit social structure of the network to achieve selective dissemination. GoDisco [17] disseminates information through gossip in an explicit social network enriched with bridges between communities. Yet, it relies on explicit interest classification and node categorization, both requiring an upfront analysis of content. The Friendship-Interests Propagation model [18] leverages explicit social networks to filter messages. While this typically works well for structural attributes (years, location, etc.) [19], it does not in the dynamic context of news dissemination (as shown by our cascading results).

Collaborative filtering: This is an appealing approach to provide users with recommendations on items [1]. While content-based recommenders use item descriptions to associate items with users (*e.g.* PersoNews [20], [21] or [22]), content-agnostic approaches are a better match for settings where content characterization is not always possible. Determining the most similar users to every user is computationally expensive and usually impossible in real-time for the information stream is huge and changes quickly. Instead, it is typical to cluster users rather than fully leverage the user-centric personalization potential [23]. In this sense, WHATSUP can be seen as a CF scheme producing user-centric recommendations at a small cost through local (P2P) computation and information exchange.

Similarity metrics: Several metrics have been used to compute the similarity between user profiles e.g. Pearson correlation coefficient, cosine similarity, Jaccard Index [3]. An evaluation of the performance of several metrics on the Orkut social network concluded that cosine similarity shows the best empirical results [24]. In the context of news dissemination, we showed that WHATSUP's metric outperforms cosine similarity. In [25], a topic-diversification approach highlights the importance of serendipity and shows that user satisfaction does not always correlate with high recommender accuracy. WHATSUP's orientation mechanism addresses this issue by balancing precision and recall.

Decentralized recommenders: Research on decentralized recommender systems is still modest despite their clear scalability advantage [26], [27], [11]. Most of these approaches are applied to much less dynamic contexts than instant news. While [28] proposes a Chord-based CF system to decentralize the recommendation database on a P2P infrastructure, it is unclear if it can cope with frequent profile changes and huge continuous streams of items. On the data-sharing front, the fear of the Big-Brother syndrome has also led to decentralized initiatives [29]. However, none of them exploits an implicit social network.

VII. Concluding remarks

This paper contributes to convey the feasibility of a fully decentralized collaborative filtering instant news system providing an implicit publish-subscribe abstraction. We did devise and implement such a system: WHATSUP. Our exhaustive experiments show that WHATSUP, while relying only on partial knowledge, achieves a good trade-off between the accuracy and completeness of dissemination. We had to make several design choices to preserve the simplicity of the system and enable its easy deployment, leaving aside complex or heavyweight alternatives. Yet, leveraging the keywords within news items or ranking them according to users' interest profiles may help refining the filtering. Another observation from our results is the very fact that WHATSUP performs best when user communities are disjoint. While real datasets do not exhibit such communities, an interesting avenue of research would be to investigate solutions that somehow separate communities, potentially allowing nodes to be part of several ones in the form of virtual instances. This is particularly challenging when no explicit classification is available or desirable.

While privacy concerns were out of the scope of this paper, they might be an issue for users who do not want

to disclose their profiles to other users. Integrating a mechanism to protect user profiles from curious users while conserving efficient online personalized dissemination is arduous. Yet, we did actually explore obfuscation mechanisms to hide the exact tastes of users as well as a proxy-based solution inspired by Onion routing [30] to anonymize both the exchange of user profiles and news dissemination. In short, while obfuscation provides a trade-off between the accuracy of recommendation and the disclosure of personal data, the proxy-based solution provides unchanged recommendation quality at the cost of increased bandwidth consumption. Clearly, the design of lightweight solutions capable of providing strong privacy guarantees constitutes an interesting research direction.

ACKNOWLEDGMENT

This work is part of the ERC SG GOSSPLE project 204742 (http://gossple.fr).

REFERENCES

[1] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, 2009.

[2] M. Draief and L. Massoulié, *Epidemics and rumours in complex networks*. Cambridge University Press, 2009.

[3] P. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Addison Wesley, 2005.

[4] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Steen, "Gossip-based peer sampling," *ACM TOCS*, 2007.

[5] S. Voulgaris and M. v. Steen, "Epidemic-style management of semantic overlays for content-based searching," in *Euro-Par*, 2005.

[6] Stanford Network Analysis Platform http://snap.stanford.edu.

[7] M. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, 2004.

[8] F. Wu, B. Huberman, L. Adamic, and J. Tyler, "Information flow in social groups," *Physica A: Statistical and Theoretical Physics*, 2004.

[9] T. Kameda, Y. Ohtsubo, and M. Takezawa, "Centrality in sociocognitive networks and social influence: an illustration in a group decision-making context," *Journal of Personality and Social Psychology*, 1997.

[10] P. T. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *Computing Surveys*, 2003.

[11] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication," in *DEBS*, 2007.

[12] C. J. van Rijsbergen, *Information retrieval*. Butterworth, 1979.

[13]

[14] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kosti, J. Chase, and D. Becker, "Scalability and accuracy in a large-scale network emulator," in *OSDI*, 2002.

[15] D. Frey, R. Guerraoui, A.-M. Kermarrec, B. Koldehofe, M. Mogensen, M. Monod, and V. Quéma, "Heterogeneous gossip," in *Middleware*, 2009.

[16] S. Verma and W. Ooi, "Controlling gossip protocol infection pattern using adaptive fanout," in *ICDCS*, 2005.

[17] A. Datta and R. Sharma, "Godisco: selective gossip based dissemination of information in social community based overlays." in *ICDCN*, 2011.

[18] S. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha, "Like like alike - joint friendship and interest propagation in social networks," in *WWW*, 2011.

[19] A. Mislove, B. Viswanath, P. K. Gummadi, and P. Druschel, "You are who you know: inferring user profiles in online social networks," in *WSDM*, 2010.

[20] E. Banos, I. Katakis, N. Bassiliades, G. Tsoumakas, and I. P. Vlahavas, "Personews: a personalized news reader enhanced by machine learning and semantic filtering," in *ODBASE*, 2006.

[21] M. Agrawal, M. Karimzadehgan, and C. Zhai, "An online news recommender system for social networks," in *SIGIR-SSM*, 2009.

[22] G. Giuffrida and C. Zarba, "A recommendation algorithm for personalized online news based on collective intelligence and content," in *ICAART*, 2011.

[23] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *WWW*, 2007.

[24] E. Spertus, M. Sahami, and O. Buyukkokten, "Evaluating similarity measures: a large-scale study in the orkut social network," in *KDD*, 2005.

[25] C. Ziegler, S. McNee, J. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *WWW*, 2005.

[26] B. N. Miller, J. A. Konstan, and J. Riedl, "Pocketlens: toward a personal recommender system," *TOIS*, 2004.

[27] Tribler http://www.tribler.org.

[28] P. Han, B. Xie, F. Yang, and R. Shen, "A scalable p2p recommender system based on distributed collaborative filtering," *Expert Systems with Applications*, 2004.

[29] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Privacy-preserving p2p data sharing with oneswarm," *SIGCOMM Computer Communication Review*, 2010.

[30] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *USENIX Security Symposium*, 2004.