A peer-to-peer recommender system for self-emerging user communities based on gossip overlays

Ranieri Baraglia\textsuperscript{a}, Patrizio Dazzi\textsuperscript{a}, Matteo Mordacchini\textsuperscript{b}, Laura Ricci\textsuperscript{a,c,*}

\textsuperscript{a} Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy
\textsuperscript{b} Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy
\textsuperscript{c} Dipartimento di Informatica, Università di Pisa, Pisa, Italy

\section{1. Introduction}

Automating the word-of-mouth \cite{1} is the aim of collaborative and social information filtering systems. Gossip \cite{2,3} is the term used to define a class of systems, especially a kind of Peer-to-Peer (P2P) networks. Although with different intents, these systems take inspiration from the human social behavior of spreading knowledge by exchanging information between people that are in direct contact. "Direct contact" in collaborative filtering and recommender systems means the selection of the most similar users of a given user in order to produce recommendations of potentially interesting items. In gossip-based P2P systems, the exchange of information between connected peers becomes a powerful tool to build up and maintain the network topology itself and to obtain a pervasive diffusion of the information associated with each node. In a world where there is a growing need to access and be aware of many types of distributed resources like Internet pages, shared files, online products, news and information, finding flexible, scalable and efficient mechanisms addressing this topic is a key issue, even with relevant social and economic aspects. In this paper, we propose the general architecture of a system whose aim is to exploit the collaborative exchange of information between peers in order to build a system able to gather similar users and spread useful suggestions among them.

\* Corresponding author.

E-mail addresses: ranieri.baraglia@isti.cnr.it (R. Baraglia), patrizio.dazzi@isti.cnr.it (P. Dazzi), matteo.mordacchini@iit.cnr.it (M. Mordacchini), ricci@di.unipi.it (L. Ricci).
the number of users and their diffusion can lead such systems to become inefficient and/or too much expensive. All the more, there is a growing need for users to aggregate information coming from the integration of different sources. Traditional approaches may provide aid only within their respective domains. Another concern about centralized system is their sustainability. Indeed, maintaining a large and complex infrastructure able to manage hundred of thousand of users could be very expensive. This, in turn, potentially leads to privacy and anonymity issues because the maintainer of the system, in order to economically sustain the platform, could “profile” users by analyzing their data either directly or by selling this information to third companies.

In this paper, we propose the general architecture of a system that tries to exploit the collaborative exchange of information between peers in order to overcome the above-mentioned problems by building a support able to gather similar users and spread useful suggestions among them. More precisely, we wish to push further the idea of exploiting collaboratively built recommender mechanisms, based on interest clustering and obtained through interactions among users. We propose to couple the two systems by exploiting gossip-style P2P overlay networks in order to ease the gathering of users with similar interests and then use the links established so far to spread recommendations among peers. The aim that we pursue is twofold. On one hand we want to build a flexible, adaptive system that allow the creation of communities of interests among users in a decentralized, distributed way. P2P approaches (and the gossip-based ones, in particular) scale well to large numbers of peers and deal gracefully with system dynamism, whereas centralized systems need expensive and complex techniques to ensure continuous operation under node and link failures. Moreover, the service is implemented through the collaboration of the peers without needing any centralized authority that would store all profiles and ratings of users and provide centralized-controlled recommendations.

On the other hand, our goal is to exploit such communities not only for sharing the knowledge of the interesting items, but also to overcome some traditional problems of recommender systems like that concerning the ability to recommend new items. In the system we are proposing, a neighbor of a peer $P$ will push to $P$ recommendations about items that it believes might be of potential interest for $P$. This decision is taken locally, when the neighbor of $P$ selects from its connections or becomes acquainted of the existence of a new item whose characteristics are related with one or more of its communities. It can then suggest this item to $P$ and its further neighbors of all the related communities. This mechanism would allow a more efficient and rapid diffusion of the information.

To this end, in this paper the efficiency of gossip-based approaches for the creation of communities of interest is examined using two proposals. First, we propose a pure, peer-centric similarity-based aggregation mechanism that allows to put similar peers in nearby areas of the logic overlay network. After that, we exploit and extend this mechanism for building public, explicitly recognized and identifiable communities of peers.

This paper is organized as follows: Section 2 reviews the literature; Section 3 presents the architecture of the proposed system. Section 4 gives an experimental evaluation of our approach. In Section 5, conclusions and further exploitations of our system are presented.

2. Related work

In the past several solutions have been proposed for distributing data over an overlay P2P network to enable resource tracings and retrieval. Such solutions exploit different techniques such as distributed indexing schemes [7–9], Gossip [10–12, 2,3,13] and Semantic Overlay Network (SON) [14].

[7] formalizes an indexing and retrieval model designed to achieve high performance, cost-efficient retrieval by exploiting Highly Discriminative Keys (HDKs) stored in a distributed global index maintained in a structured P2P network. HDKs are carefully selected terms that appear only in a small set of collected documents. The authors present both a theoretical analysis of the scalability of the model they propose, and experimental results achieved by using the HDK-based P2P retrieval engine. These results demonstrated that the total traffic generated exploiting the HDK approach is smaller than the one obtained with distributed single-term indexing strategies. The authors also performed a deep scalability analysis that demonstrated that the HDK approach is able to scale to large networks of peers indexing very large document collections.

In [9], the architecture of a P2P search system that supports full-text search in an overlay network with peer dynamics is proposed. This architecture, namely HAPS, consists of two layers of peers. The upper layer is a Distributed Hash Table (DHT) network interconnecting a set of super peers (which we refer to as hubs), Each hub maintains distributed data structures called search directories, which are used to guide the query and to control the search cost. The bottom layer consists of clusters of ordinary peers (called providers), which receives queries and return relevant results.

In [8] an approach to support semantic search on DHT overlays is proposed. The idea is to place, with high probability, indexes of semantically close files into the same peer node by exploiting information retrieval algorithms and locality sensitive hashing. This approach adds only index information to peer nodes, causing an acceptable storage overhead. Via network simulations, the authors show that their approach is viable because the number of nodes visited for a query is about 1020, independently from the overlay size.

In [14] a structured indexing solution is proposed to reduce the search time of queries executed in peer-to-peer networks that create the overlay network randomly, and at the same time to maintain a high degree of autonomy of the nodes. The authors propose node connections influenced by content. Thus, each subset of semantically related nodes form a SON. Queries are routed to the proper SON, increasing the chances to find matching files, and reducing the search load on nodes
that have unrelated content. One of the main disadvantages of this solution is that the SON-based overlay networks have a rigid predefined tree structure.

Structured indexing approaches like the one presented above may not deal well with very dynamic scenarios. This may refer to both the join/leave rate of the nodes in the network and to the information disseminated by the peers. Frequent changes involve frequent updates of the network and/or the index structures. Overlay networks with less requirements about the network organization may be more suited for those dynamic scenarios.

The GosSkip [10] systems is a self-organizing and fully distributed overlay that provides a support to data storage and retrieval in peer-to-peer environments. It is built using a gossip protocol that organizes peers so that they form an ordered double-linked list. Each peer is associated with a single item of data and it has a name that describes the semantics of the object to which is associated. These names follow a total and deterministic order. So, the position of an element is fully determined by its name. For the information distribution, its gossip protocol maintains $O(\log(N))$ peer states, and has a routing cost of $O(\log(N))$. The association of links to published object can lead to a very large number of connections. This is especially true in networks where the number of objects shared by each user is large. Furthermore, the use of GosSkip could be difficult when searching without knowing the exact name which identifies the item you are looking for.

The authors of [15] propose a solution for Peer Data Management Systems (PDMS). A PDMS consists of peers, viewed as autonomous and heterogeneous data sources, having their own content modeled upon schemas. The authors propose a strategy for clustering related peers through the maintenance of a multilayer network organization. Each layer represents different semantic concepts. Each peer takes part to its most semantically similar layers. Within each layer, it gathers with its most similar nodes using a fine-grained neighbor selection mechanism. A critical aspect of this solution is the evolution of the interests of a peer. If that leads to changes in the peer’s semantic concepts, it may trigger a distributed mechanism to reorganize the overlay network, involving all the neighboring nodes belonging to the related SONs.

The authors of [11] propose a layered, gossip approach for building SONs. They couple the Cyclon [16] protocol, used to select from the network a set of peers in a uniformly random way, with the Vicinity protocol they use for allowing each peer to build point-to-point connections with its most similar peers. One of the main drawbacks is the lack of a broader, more recognized measure of “similarity”. Each peer relies on its local view only. Thus, it is not able to determine whether a peer not included in its similarity-based neighborhood could be regarded as similar, with respect to the overall network organization.

In addition to the papers cited above, we are interested in works that investigate the characteristics of the users accesses to distributed data. In particular, in the context of this paper, we focus on the clustering of the graph that links users based on their shared interests, on the correlation between past and futures accesses of users (or groups of users) sharing similar interests, on the skewness of the distribution of interests per peer and of the distribution of accesses per data element. Skewness usually relates to Zipf-shape distributions, which are a feature of access behaviors of large groups of humans [17].

We first review the work related to the detection and the use of interest correlation between users in large-scale systems. The presence of communities amongst user interests and accesses can be exhibited in Web search traces [18,19], peer-to-peer file sharing systems [20] or RSS news feeds subscriptions [21]. The existence of a correlation of interests amongst a group of distributed users has been leveraged in a variety of contexts and for designing or enhancing several distributed systems. A sound approach to increase recall and precision of the search for peer-to-peer file sharing systems that include file search facilities (e.g., Gnutella, eMule, ...) is to group users based on their past search history or on their current cache content [22–24]. Interestingly, the small-world [25] aspects of the graph of shared interests linking users with similar profiles is observed and can be exploited not only for file sharing systems, but also in researcher communities or in web access patterns [20]. Another potential use of interest clustering is to form groups of peers that are likely to be interested in the same content in the future, hence forming groups of subscribers in a content-based publish-subscribe system [26].

Moreover, interest correlation can be used to help bootstrapping and self-organization of dissemination structures such as network-delay-aware trees for RSS dissemination [27]. Finally, user interest correlation can be used for efficiently prefetching data in environments where access delays and resource usage constraints can be competing [28], as it is an effective way of predicting future accesses of the users with good accuracy.

The correlation between the users past and present accesses has been exploited for user-centric ranking. In order to improve the personalization of search results, the most probable expectations of users are determined using their search histories stored on a centralized server [29,30]. Nevertheless, the correlation between users with similar search histories is not leveraged to improve the quality of result personalization, hence making the approach sound only for users with sufficiently long search histories. An alternative class of clustering search engines uses semantic information in order to cluster results according to the general domain they belong (not as in our approach which clusters users based on their interests). This can be seen as a centralized, server-side and user-agnostic approach which exploits the characteristics of distributed accesses to improve the user experience. The clustering amongst data elements is derived from their vocabulary and presents the user with results along different interest domains so that the user may disambiguate these results from a query covering several domains, e.g., the query word “apple” can relate to both food/fruits and computers domains.

---

1 Small-world aspects for the shared interests graph are: (i) a high clustering, (ii) a low diameter due to the existence of a small proportion of long links, i.e., links to exotic domains that are distant from the common interests of the node and its regular neighbors and that act as cross-interest-domain links, and (iii) the possibility to navigate through the graph of interest proximity amongst peers and effectively find short path between two interest domains based only on the one-to-one distance relationships amongst these domains, i.e., without global knowledge of the graph.
Interest communities.

EigenCluster [31], Grouper [32], SnakeT [33] or TermRank [34] are examples of such systems. Nonetheless, these systems simply modify the presentation of results so that the user may decide in which domain the interesting results may fall. Anyway, these results are not automatically tailored to the expectations of the user and do not consider the clustering of interests amongst users, but only the clustering in content amongst the data.

Further approaches cluster users on the basis of similarity between their semantics profiles. Approaches of this kind of systems includes GridVine [35], the semantic overlay networks approaches [14] and p2pDating [36]. These systems build a semantic overlay infrastructure based on a peer-to-peer access structure by relying on a logical layer storing data. In order to create links among peers they exploit schemas, and schema mappings and make use of heterogeneous but semantically related information sources. It is worth noticing that our approach does not rely on any kind semantic interpretation and, in principle, it enables a broader exploitation of more heterogeneous data sources.

Tribler [37], a P2P television recommender system is related to our proposal. Tribler is a system able to recommend, record and download television programs to/from its users by constructing a social P2P network. In contrast to our approach, neighbor lists (called buddy tables) can be directly filled in by the user herself by using an interface and topology or affinity property is considered. It is worth noticing that the gossip system we propose exploits past user activities but does not require user intervention to construct and maintain in rest groups of dynamic users.

The protocol described in [38] presents a distributed approach for discovering connectivity-based clusters in P2P networks. It discovers clusters based on the network graph connections around a given set of nodes using local peer knowledge. The main drawback is that the quality of the clusters highly depends on the initial choice of those peers. Moreover, this protocol needs an explicit management of joining and leaving peers.

3. The proposed approach

In this section we present the general principles for the construction and the use of a network of peers based on shared interest. A representation of a network of this kind is given in Fig. 1. Links between peers are established when they have been interested in the same content in the past. In general, they are considered to potentially show interests for the same content in the future. Thus, peers collaboratively exchange useful recommendations among themselves. In order to group similar users, the protocol exploits a clustering algorithm. First, each peer independently decides which are the peers it is linked with. These one-to-one relationships are chosen on the basis of an interest-based distance, amongst the peers it encounters. Every time it gets in touch with a new peer, it can learn of the existence of new potential neighbors (i.e. similar users) and then communicate with them so that it can become acquainted of further, potentially better neighbors.

An important point of our approach is that a peer is characterized by multiple interests. Hence, the process depicted above is conducted separately for each of the interests of a peer so that connections may be established and maintained separately for each distinct interest. Thus, a set of virtual different overlays are defined, where each peer participates in as many groups as it is required to cover all its interests. The situation is showed in Fig. 2. This approach allows each peer to identify a sort of “private” communities made of subsets of its neighbor list. Even if this “private” knowledge provides a support for the identification of the features needed to characterize network communities, it does not define a “public” identification of a community that can be used by each peer to characterize itself as member of a group and that can be further exploited to exchange information about communities it belongs over the network.

To this end it is fundamental to define a way to represent and identify the communities. A possibility consists in exploiting the profile of a peer inside a community as the community identifier. This approach allows, by fixing certain preference criteria, to choose the “best” representative for a community.
3.1. User profiles

Profiles of peers should be created by considering the users’ interests which can be represented by considering recently accessed resources, purchased items, visited pages, and so on. Such information has to be gathered and exploited in a proper way, since it forms the basis over which the whole network will be built up. Let \( \mathcal{I} \) be the set of items of the system and let \( \mathcal{I}_p \subseteq \mathcal{I} \) be the subset of items belonging to a peer \( p \). We consider the profile \( \pi \) of \( p \) as

\[
\pi_p = \{(i, C(i), R(i)) \mid i \in \mathcal{I}_p\}
\]

where \( i \) is an item of the set of \( \mathcal{I}_p \), \( C(i) \) is the content associated with \( i \) (potentially void) and \( R(i) \) is the rating given by \( p \) to \( i \). Moreover, \( p \) has a set \( I^p = \{I^p_1, \ldots, I^p_k\} \) of interests. Each item \( i \) in \( \pi_p \) may be associated with an interest \( I^p_j \). \( \pi_p \) is defined in the following way:

\[
\pi_p = \bigcup_{j=1}^{k} \pi_p(I^p_j)
\]

where \( \pi_p(I^p_j) \) is the set of items paired with the interest \( I^p_j \). Note that the set \( I^p \) is specific for each distinct peer \( p \). We do not assume any global labeling, categorization or subdivision of the objects in \( \mathcal{I} \). Each peer organizes its own subdivision of \( \mathcal{I}_p \) in the interests of \( I^p \) then it compares its objects partitioned by the interests with the sets of the other peers it will be put in contact.

The set of interests of the peers may be really large and sparse, anyway in real datasets it generally follows a power-law distribution [39]. It means that there exist sets of interests that, if identified, are able both to cover almost all the peers in the system and at the same time to discriminate peers on the basis of their preferences. In this scenario, the main goal of our approach is to exploit these sets of interests in order to build communities made of similar peers.

Given two peers \( p_1 \) and \( p_2 \), \( p_1 \) considers its local interest \( I^p_1 \) similar to the interest \( I^p_2 \) if it contains the most similar set of items among the other sets in \( I^p_2 \) with respect to the items in \( I^p_1 \).

Thus, defining a suitable method to code each user interests in the peer profiles, a proper similarity function \( sim : \Pi^2 \rightarrow \mathbb{R} \) should then be defined to compare profiles, where \( \Pi \) is the set of all possible profiles. This is a relevant point, since this function determines the relationships between peers on the basis of their interests. If each different interest is determined by different type of features, different similarity measures could be used to evaluate peers proximity with respect to each interest.

Several measures can be exploited to this end. For instance, one common way is to use a metric that takes into account the size of each profile, such as the Jaccard similarity, which has proved to be an effective similarity measure [23,27]. Given two peers \( p_1 \) and \( p_2 \) and two interests \( I^p_1 \) and \( I^p_2 \), their similarity is defined as

\[
sim(p_1, p_2) = \frac{|\pi_p_1(I^p_1) \cap \pi_p_2(I^p_2)|}{|\pi_p_1(I^p_1) \cup \pi_p_2(I^p_2)|}
\]

One possible use of this measure is shown in Fig. 3.

3.2. Detection of interest communities

Since each peer is able to compute its interest-based distance to any other peer, its objective is to group with other peers that have close-by interests, in order to put the basis for interests communities. The key step of this process is performed in a self-organizing and completely decentralized manner, using a gossip-style communication. Each peer knows a set of other
peers, namely its interest neighbors, and periodically tries to choose new neighbors that are closer to its interests than the previous ones. This is simply done by learning about new peers from some other peers, then retrieving their profiles, and finally choosing the nearest neighbors in the union of present and potential neighbors.

### 3.2.1. Private communities

When a peer $p$ enters the network, it is put in contact with one or more peers already taking part in the interest-proximity network. These use the profile similarity function to compute how similar they are. They compare one each other with respect to the interests they have. Moreover, the peers contacted by $p$ use the same similarity function to determine which are, among their neighbors, the most similar to $p$ and route the join request of $p$ toward them. All the peers that receive that request will react by using the same protocol described above. All the interactions are shown in Algorithms 1 and 2. This mechanism will lead $p$ to learn the existence of the most similar nodes in the network and allow $p$ to connect with them. In doing this process, the involved peers can only use their local knowledge to compare their respective profiles.

We couple this protocol with a random peer sampling protocol [2], that is a standard approach in gossip-based communications to guarantee that each peer is put in contact with any other peer of the network. To this end, the random peer sampling protocol maintains a local view of the network and randomly chooses a peer in this view. Afterwards, it exchanges a subset of its view with it. In this way, each peer is put into contact with new peers at each gossip iteration and it may continuously acquire knowledge about new, unknown peers.

---

**Algorithm 1** A peer $P$ gossip active process for each interest

1. Let $N(P)$ be the set of $P$’s actual neighbors
2. for all $P_i \in N(P)$ do
3. Get from $P_i$ a set NewPeers from its neighborhood
4. for all $P \in$ NewPeers do
5. if $P \notin N(P)$ then
6. connect with $P$
7. if $\text{Sim}(P, P') \geq \min_{P_j \in N(P)} \text{Sim}(P, P_j)$ then
8. add $P'$ to $N(P)$
9. end if
10. end if
11. end for
12. end for

**Algorithm 2** A peer $P$ gossip passive process for each interest

1. Let $CR(P')$ be a connection request from another peer $P'$
2. Let NewPeers = $\emptyset$
3. if $\text{Sim}(P, P') \geq \min_{P_j \in N(P)} \text{Sim}(P, P_j)$ then
4. Accept $CR(P')$
5. for all $P_i \in N(P)$ do
6. if $\text{Sim}(P_i, P') \geq \theta$ then
7. add $P_i$ to NewPeers
8. end if
9. end for
10. add $P'$ to $N(P)$
11. send NewPeers to $P'$
12. else
13. refuse $CR(P')$
14. end if
Once the process is stabilized, $p$ can consider its neighbors as the representatives of a “private” community of “friends” from which to request and to forward recommendations.

3.2.2. Public communities

The interest-proximity networks built by peers for set up their “private communities” can be exploited to build “public”, universally recognizable communities.

The goal of this approach is to address one of the most significant limitations of the previous solution: similarity of interests between nodes is computed in a point-to-point fashion. In other words, each peer individually computes which are the other nodes that may share similar content/interest with it. The “private communities” built so far are limited to each peer neighbor list. This “private” knowledge do not always allow to give a good identification of the features needed to characterize sets of similar peers, since it does not give any recognizable identifier for such communities.

Conversely, a “public” community is identified by the profile of one of the peers belonging to that community, in particular the one selected as the community representative. In order to select the representative peer for the public communities, we set up a voting procedure. Each peer votes by sending a message. Each vote is paired with a TTL (Time-To-Live), which determines when the vote has to be discarded. Each community is defined by the elected representative peer, together with the peers that have contributed to its election (and autonomously decide to join it). The voting procedure we propose is structured according to three stages:

1. Identification of the Representative Candidates,
2. Identification of the Potential Representatives,
3. The Representatives Election.

The algorithms used are shown from Algorithm 3 to Algorithm 7. Algorithm 3 and Algorithm 4 define the active and the passive thread of each peer, which define the gossip behavior of the system. The active thread (Algorithm 3), exploits a timer $t$ to separate the voting and election phases. Within this stage, phases are separated by using two other timers. The only constraint is that $t + t'' < t$.

**Identification of the Representative Candidates:** This phase aims to identify the potential representative candidates. It consists in a preliminary voting procedure in which each peer votes for carrying out its most similar peers, i.e. its best neighbors in term of profiles. This phase is mainly driven by two parameters:

- $n\_votes$: the maximum number of neighbors that a peer can vote for.
- $neighbor\_threshold$: the similarity value under which a peer is not considered similar to the voting one and therefore may not be selected to be voted.
Algorithm 5 CandSelectAndVote
1: Order Neighbors by similarity;
2: Let BestN = Top-k(Neighbors);
3: for all b ∈ BestN do
4:  if d(p, b) \leq (1 – neighbor_threshold) then
5:     Send Vote to b;
6:  else
7:     Break;
8: end if
9: end for

Algorithm 6 PotentialReprIdentify
1: Let Candidates = ∅;
2: for all n ∈ (Neighbors ∪ {p}) do
3:  if VotesRecvd(n) \geq ReprThr then
4:     Let Candidates = Candidates ∪ {n}
5: end if
6: end for
7: if Candidates = ∅ then
8:     Let Candidates = NghReprs
9:  if Candidates = ∅ then
10:     Let Candidates = p ad interim
11: end if
12: end if
13: Let R = min_c\in\text{Candidates} d(p, c)
14: Send ReprVote to R

Each voting peer arranges its neighbors in decreasing order with respect to the similarity value, and gives a vote to, at most, the n_votes ones with the highest similarity value, without considering the peers whose similarity is lower than \text{neighbor\_threshold}. At the end of this phase, a peer p has received a number of votes given by:

\[ \sum_{n \in Nbg_{in}} v_n(p) \quad \text{where} \quad v_n(p) = \begin{cases} 1 & \text{if } p \in \text{top(Nbg)} \text{ of } n \\ 0 & \text{otherwise} \end{cases} \]

Here, Nbg_{in} is the set of nodes for which p has an incoming link connection, Nbg is the usual set of neighbors (i.e. the neighbors with which there exists an outgoing link) and top(Nbg) is a function that selects the first n_votes most similar peers in Nbg. Clearly, for a node p the intersection between Nbg_{in} and Nbg is not void iff there exists at least another node p’ s.t. p ∈ Nbg(p’) and p’ ∈ Nbg(p). All the above operations are described in Algorithm 5.

Identification of the Potential Representatives: This phase is devoted to the identification of the potential representative candidates, each peer can give up to n\_representative votes. The representative votes are assigned to the most similar representative candidates. The potential representative candidates are the ones that received a number of votes higher than the representative\_threshold where representative\_threshold is the minimum number of votes for a peer to be considered as a representative candidate. Thus, usually a peer p identifies the potential representatives by selecting a set of candidates from its neighbors, defined as:

\[ \text{Candidates} = \left\{ n \in \text{Nbg}(p) \mid \text{VotesRecvd}(n) \geq \text{ReprThr} \right\} \]

Then, it votes a neighbor R as a potential representative by using the following function:

\[ V_R(p) = \begin{cases} 1 & \text{if } R = \min_c\in\text{Candidates} d(p, c) \\ 0 & \text{otherwise} \end{cases} \]

where d is a proper distance function used to compare both how much two peers are different and how much a peer is distant from its community representative. In order to face special cases that might occur, the above function needs to be modified as specified in Algorithm 6. As a consequence, by acting on the representative\_threshold parameter, it is possible to influence the total number of representatives, and, assuming to keep fixed the other parameters, the cardinality of each community.

The Representatives Election: In this phase each peer p elects its actual representatives. At this step p considers also the representative votes it has received from other nodes. A representative candidate RC is chosen by p in the following way:

\[ RC = \begin{cases} p & \text{if } p \text{ has the highest # of ReprVotesRcvd} \\ \min_{r \neq p, r \in \text{RCandidates}} d(p, r) & \text{otherwise} \end{cases} \]

where RCandidates is the set of potential candidates filtered by the votes expressed in the previous phase. When a peer p has collected the highest number of representative votes with respect to its neighborhood, it considers itself as its own
representative. Otherwise, a potential representative $R$ becomes actual representative if and only if, among the peer’s neighbors, $R$ is the one that has received the highest number of representative votes. With respect to the previous definition of $RC$, however, if there are two potential representatives receiving the same number of votes, $p$ chooses its most similar one. Anyway, there are further scenarios to be addressed. When $p$ discovers that its actual representative $R$ has chosen in turn as its own actual representative another peer $R'$ that is within a distance $\epsilon$ from $R$, then $p$ adopts $R'$ as its actual representative. When $R$ has chosen $R'$ as its representative, but they are enough distant ($d(R, R') > \epsilon$), $R$ is chosen as the representative of $p$. The link between $R$ and $R'$ represents a connection among their respective communities, augmenting the spreading of information inside the network. A further special case occurs when none of the potential representatives of $p$ can be considered its actual representative. In this case $p$ asks to its neighbors for their actual representatives and it selects the one sharing the highest similarity value. All the above behaviors are described in Algorithm 7. This process leads the peers in the network to spontaneously gather into communities, each community including all the peers that have chosen the same representative. As mentioned before, each vote has a limited life time, exactly as it happens in a democracy where a mandate expires after a certain amount of time. Thus, periodically, at predefined intervals of time, each peer contributes to the election of community representatives.

The continuous refresh of information is ensured by the activities performed by peers for building their interest-proximity networks that put in contact similar nodes. These activities are supported by an epidemic diffusion of the information, both as far as concerns the user profiles and the representative votes. As it happens to representations, communities are dynamic entities subject to changes. Beyond the joins and leaves due to peers churn, the community may be split or merged. It is worth noticing that all the operations described so far are performed by each peer individually, without any form of synchronization with other nodes. The only interaction is due to the exchange of information, both for exchanging votes and gossip updates that includes peers’ profiles, received votes and representative votes, actual community profiles, and so on. When a proper defined time interval has elapsed, each peer independently starts a new flow of votes and at the end of this phase it is able to cope with new situations, like the arrival or the departure of other peers. The underlying gossip mechanism allows a peer $p$ to obtain an up to date situation of similar peers in its neighborhood. Hence, when a new voting phase starts, new peers will be considered, and old (or disappeared) ones will be no more taken into account. If a large number of updates in $p$’s neighborhood occur, $p$ will choose a new community, possibly joining previously created communities or splitting its old community and defining a new group. Thus, no explicit mechanism to handle joins or splits of communities is required. The experimental section includes some highlights related to this mechanisms.

3.3. The recommender algorithms

The gossip protocol we have defined provides the support for the definition of a classical recommender systems. This is done in a distributed and adaptive way and the gossip protocol ensures a robust and constant maintenance of the similarity overlay and of the communities over time. Recommendations can then be requested to its neighborhood by $p$ which can, in turn, forward the new discovered items to its neighbors as defined by Algorithms 8 and 9.

Algorithm 8 describes a sort of pull recommendation behavior, where a peer seeking for recommendations explicitly requests it to one (or more) of its neighbors. Neighbors react by suggesting among their data, the items more closely related with the asking peer profile. On the other hand, Algorithm 9 presents a sort of push mechanism. In this case, a peer that has acquired or discovered a new item, decides to suggest it to the most interested nodes among its neighbors.
Algorithm 8 Recommendation response process
1: Let $N_p(I_j)$ be the set of $P$’s neighbors for the interest $I_j$
2: Receive a recommendation request from $p' \in N_p(I_j)$
3: for all $i \in \pi_p(I_j)$ do
  4: if $\text{Sim}(p', i) \geq \theta$ then
    5: recommend $i$ to $p'$
  6: end if
7: end for

Algorithm 9 Recommendation suggestion
1: Know about a new item $h$
2: Let $I_j$ be the interest $h$ is related to
3: Let $N_p(I_j)$ be the neighborhood of peers interested in $I_j$
4: for all $p' \in N_p(I_j)$ do
  5: if $\text{Sim}(p', h) \geq \theta$ then
    6: recommend $h$ to $p'$
  7: end if
8: end for

4. Experimental results

In order to evaluate the envisioned architecture and the algorithms presented in this paper, with respect to their ability to build a peer-to-peer system able to group users sharing common interests in a totally decentralized way, we developed a prototype implementing the gossip-based peer-to-peer protocols previously described.

The prototype has been developed and tested using the Overlay Weaver [40] peer-to-peer framework. Overlay Weaver is a framework implementing several P2P overlays which aims at separating high level services such as DHT, multicast and anycast from the underlying Key-Based Routing (KBR) level.

The routing layer architecture follows the KBR concepts but leaves behind the KBR monolithic approach by decomposing the routing layer into a set of independent modules, (e.g. communications, routing and query algorithms). The routing module is defined by three layers: the routing layer (bottom), the service layer and the application layer (top). The main advantages from the usage of Overlay Weaver are rapid prototyping (it does not require to deal with low-level distributed development issues, e.g. network socket programming, fault management), the possibility of simulating the behavior of a large scale network (even composed of thousands of nodes), and the possibility of defining a network observer able to monitor the status of the network in order to find out useful statistics.

We have conducted two different testbed, the first one devoted to the experimentation of the proposed solution to build the peer “private” communities. The second testbed deals with the solution provided for the identification of the “public” communities and their representatives. For the experiments we exploit two different datasets, the Movielens one and a dataset released by Mendeley, a company that develops a publications management tool.

4.1. Private communities

The data used for these experiments comes from the Movielens dataset [41]. The Movielens dataset is a movie recommendation dataset created by the Grouplens Research Project. The dataset consists of 1 million ratings for 3900 movies by 6040 users. Since each user only rates a few movies, the data matrix is very sparse. This makes this dataset a good benchmark for our approach because the construction of communities from a sparse dataset is a real hard task. The choice of using the Movielens dataset affected the structure of the user profiles and of their representation. In the current prototype each profile contains information about the movies seen by the corresponding user. Each peer stores the information about its neighborhood, i.e. the addresses and the profiles of its neighbors.

By using this dataset we have performed several experiments by varying:

1. the functions used for selecting both the peers to contact and the best peers among the ones in the neighborhood;
2. the neighborhood size;
3. the number of peers in the network;
4. the number of gossip-protocol iterations performed before measuring the similarity among a peer and its neighborhood;

Regarding the points 1–2, we have implemented three different gossip protocols in the simulator: Cyclon, Vicinity and Twinfinder. All of them have a behavior compliant with Algorithms 1 and 2. Cyclon and Vicinity are well known gossip algorithms, whereas Twinfinder is a customized version of Vicinity, we conceived and designed, where the sender transfers to each neighbor a potentially different set of profiles, in particular the ones considered the most similar to the receiver. Each protocol has been tested under several different conditions, varying the maximum number of neighbors a peer can store (in the range [1–20]) and the number of nodes in the networks (during this first experimental phase in the range [1000–3000]). The results we have achieved are shown in Fig. 4, this figure is in turn composed of eight subfigures organized in four rows and two columns. Each row of the first three ones represents a different protocol. The subfigures in the first
The second column reports the variance and the standard deviation of the results obtained by each peer in the network for each protocol implemented. Finally, the last row shows the comparison among the results provided by the three protocols for a network of 2000 peers, namely the half-way in the range [1000–3000].
It can be noticed that all the three algorithms achieved similar trends, it is due to the effect resulting by an increasing neighbor cache size. However, it is easy to see that the two protocols, Vicinity and Twinfinder, exploiting the peer profile similarity in the peer selection process achieve better results. Twinfinder, in particular, has obtained a sensible better value in terms of standard deviation.

These experiments give us a further confirmation that the gossip protocols, and our Twinfinder in particular, are suitable solutions to build links among similar peers, hence for building up Interest Communities. We have performed experiments for measuring the potential ability of gossip protocols for providing recommendations inside the Interest Communities as well. In order to do it we define the “Coverage” measure. Given the set of movie genres liked by a user, we defined her Coverage as the percentage of user genres potentially recommendable to her by its neighborhood. The left sub figure of Fig. 5 contains the Coverage achieved by the three protocols while the right sub figure shows the number of peer interactions with which that coverage is obtained in Twinfinder. The left sub figure shows that also in this case the similarity based protocols, and Twinfinder in particular, achieve better results in terms of Coverage with respect to Cyclon. The right sub figure shows that Twinfinder achieves a good coverage (about 90%) in a few number of peer interactions (about 4–5 neighbors). The algorithm convergence speed measurement is useful for evaluating the amount of time (or cycles) required by the gossip protocol to find a good set of neighbors.

The last testbed regards the ability of Twinfinder to provide good results for different network sizes. In this case we have measured the Coverage when the network size is 1000, 2000, 3000 and 5000 (see Fig. 6). The experiment shows the better results for networks including a large number of nodes and this is a further confirmation of the ability of the Twinfinder protocol to build up links with similar peers. Indeed, if we fix a certain user $U$ we can observe that on one side an increased network size increases the probability to find a user in the network with interests close to $U$.

### 4.2. Public communities

In order to evaluate the ability of the proposed approach to build good “public” communities, we have implemented our peer-to-peer representative election algorithm and we have conducted several different experiments as well. As we outlined before, in order to test the “real” effectiveness of our approach, we have decided to apply it to a real dataset. We have exploited a bio*- subset of the dataset released by Mendeley [42,43]. The dataset released contains a set of anonymized users, each one with a set of references indicating the papers owned by it. The subset we choose is made of 2800 users each of them with at least 20 papers. For each user we have retrieved the content of the papers in his/her profile, we have
filtered out the stop-words and extracted the most frequent terms that have been exploited to define the user profile. The timer $t$ value was set to a time equivalent to 2 simulation cycles, as justified later in this section. According to what we stated in the problem description section (Section 3), our goal consists in the definition of an algorithm for building up explicit communities composed by peers sharing a common interest, whose size should not be too small nor too big and, possibly, independent from the network size. In order to evaluate the ability of our approach to address these goals we have measured the effectiveness of the results by considering the internal similarity of the community, as well as the total number of communities and, as a consequence, their mean size. The mean internal community similarity is measured by the Similarity Degree which is defined as:

$$\frac{1}{N} \times \sum_{n \in N} d(n, R(n))$$

where $N$ is the number of peers belonging to the network, $R(n)$ is the representative of peer $n$ and $d$ is the distance function defined in the previous section. Since each peer autonomously chooses the community to join, this measure is useful to see whether its choice gives it a good representative. Good representatives ensure a sufficiently high degree of similarity with their community members, thus enabling an effective communication over the network. This factor has to be coupled with proper community sizes, in order to check that the system does not create communities made of too few members, thus breaking the network into small groups and vanishing the effect of peer gathering. We analyze both these aspects in the following experiments.

**Number of votes and Representative Threshold impact:** Fig. 7 and Fig. 8 show the effects of varying the number of votes a peer can give, with respect to the Similarity Degree as well as the number of communities. This is shown when different representative_threshold (RT in the figures) are used. The network size is fixed to 2800. Fig. 7 shows that the highest values of Similarity Degree can be achieved when the representative_threshold is low. This is a quite expected behavior. Indeed, a low threshold brings to the creation of a larger number of communities (as it is shown also in Fig. 8). A higher number of communities means, in turn, a smaller number of peers per community, hence a (potential) larger internal homogeneity.
Anyway, communities with too few members can be less useful when used to exchange information. It is interesting to note that both the number of communities and consequently the highest value of Similarity Degree are achieved when the number of votes a peer can express is 4 or 5, almost independently from the representative threshold value.

**Neighbor Threshold impact:** Fig. 9 and Fig. 10 show the Similarity Degree and the number of communities as a function of the values of the neighbor threshold parameter. Even in this case the network size is fixed to 2800 nodes. The results are shown when different representative threshold are used as well as with different number of votes (V in the figures) that can be expressed by a peer. As described above, it is easy to see how different representative thresholds (ReprThr) brings up to the creation of very a different number of communities. Moreover it leads also to a quite different result in terms of Similarity Degree. We recall that the representative threshold represents the minimum number of votes a peer has to receive to be considered as a potential community representative and that each community is identified by its representative. This naturally implies that a larger threshold corresponds to a smaller number of communities. Furthermore, a smaller number of communities implies larger communities and as a consequence, a smaller degree of similarity between the peers of each community. Note also that both the figures show how the different neighbor threshold value has almost no relevant effects both in terms of community number and in terms of Similarity Degree.

**Internal Community Cohesion:** Fig. 11 and Fig. 12 show a comparison of three different algorithms for grouping nodes, namely our approach (GROUP), the CDC algorithm [38] and K-Means, used as a centralized clustering algorithm. The network size is 2800 peers and the comparison has been conducted measuring both the Similarity Degree of communities and the distance between the community representative and the real medoid of the communities defined by each algorithm, for different community sizes. It is easy to see that our approach is able both to build communities characterized by a higher internal Similarity Degree and to elect better representatives, i.e. representatives that are close to the "real" centers (in term of similarity) of the communities formed by GROUP.

**Communities mean size:** Fig. 13 shows the mean size of the communities as a function of the network size. We have used three different settings of thresholds and votes. This figure shows that the network size has basically little or no influence
Fig. 11. Internal Community Cohesion: Similarity Degree.

Fig. 12. Internal Community Cohesion: similarity against the optimal medoid.

Fig. 13. Communities mean size.
Fig. 14. Impact of network instability.

on the size of the communities. Indeed, the mean community size is about 100 elements. This is due to the self-organization mechanism, that spontaneously avoids the creation of too big communities, while ensuring a sufficient number of members. This result is in line with what we stated in Section 3.

Network instability: Fig. 14 shows the impact of instability in the network due to the presence of failing peers and to the ability of our approach to re-build communities characterized by a high Similarity Degree. Namely, this figure shows the results achieved on an initial network of 2800 nodes by suddenly (i.e. without any graceful departure mechanism) removing 100, 500 and 1000 peers respectively. We have used three different settings of votes and thresholds pairs. Cycles are considered as time units. The figure shows that our approach is able to recover from these situations and to adapt almost immediately the communities. Clearly, when removing a larger number of nodes, the resulting community has a lower Similarity Degree. This is probably due to the fact that some of the most similar peers have left the network. Moreover, the effects of failing peers are more relevant with the lowest values of representative_threshold and votes. As noted in the first set of experiments, those settings leads to smaller communities. Thus, removing elements from them can easily disrupt their internal cohesion. Larger communities (highest values of representative_threshold and votes) suffer less in this scenario. All the settings converge in a very limited number of cycles (less than 10), thus demonstrating the robustness of the proposed approach.

Scalability: Fig. 15 shows the scalability of our approach. The figure shows a comparison among GROUP, CDC and K-Means regarding their ability to achieve a high Similarity Degree value when the network size changes. Our solution has been tested by using a threshold of 30 votes and allowing 4 votes per peer. K-Means was run by using the number of communities defined by GROUP. It is easy to note the better performance of our approach with respect to the other algorithms with almost all tested networks sizes. In particular, it shows a clear trend to improve its performance when the network size increases.

Timer impact: Fig. 16 shows the impact of choosing different values for the timer interval \( t \) that separates vote waves. We used three different values, 1, 2 and 4 cycles respectively. Results show that a longer time interval implies a slower convergence of the algorithm, as one may expect. Anyway, too frequent elections may be influenced by the partial convergence of the underlying similarity layer. It causes a fast achievement of suboptimal values, that tend to optimal only with an extra amount of cycles. A value allowing the convergence of the similarity layer while defining a low timer interval from one wave to another, seems to achieve the best results.

5. Conclusions and future work

The focus of this paper is to address the problem of clustering users in a purely decentralized way. This is particularly useful for enabling an automated creation of communities made from users sharing common interests. The paper presents mechanisms for building communities both in a simple way, where the communities are “privately” defined by each peer, and in a more complex way where communities are “publicly” identified by delegated peers named “representative” where
a "representative" is a peer elected by a set of other peers as their representative on the basis of their similarity with it. The election mechanism allows both the gathering of peers into communities and the determination of the community representatives. Starting from the "private" vision of each node, after successive interactions and exchange of information among peers, it allows to exploit each peer local knowledge to contribute to the identification of the community representatives.

Both the approaches are based on peer-to-peer epidemic protocols for spreading information about network nodes. Each peer is associated with a network user, and is characterized by a profile. In order to evaluate our community building solutions we have conducted several tests and compared our performance against the ones achieved by existing well known algorithms. Tests have been conducted by using a real datasets, namely the Movielens one and the one released by Mendeley, a company producing a publication management tool. The experiment results show that our solutions overcomes most of the issues related with the faced problem, and show very good results compared to the existing solutions.

A potential limit of our "public communities" approach is that a newly joining peer, in order to find communities it is interested in, has to wait that the stabilization of i) the similarity-based protocol and ii) the community building protocol. Anyway, communities may already exist in the network. A future research activity may investigate the exploitation of a companion index structure distributed among peers. The goal of this structure is to be coupled with the proposed approach in order to ease the registration and localization of already existing communities.

**Acknowledgments**

The authors acknowledge the support of the S-CUBE (EU-FP7-215483), CONTRAIL (EU-FP7-257438) and RECOGNITION (EU-FP7-257756) EU projects. The authors would like also to thank Luca Alessi and Matteo Fulgeri, who conducted several experiments.