GROUP: A Gossip Based Building Community Protocol*

Ranieri Baraglia[1], Patrizio Dazzi[1], Matteo Mordacchini[2], Laura Ricci[3], and Luca Alessi[3]

[1] HPC Lab, ISTI-CNR, Italy
{ranieri.baraglia,patrizio.dazzi}@isti.cnr.it
[2] Ubiquitous Internet Lab, IIT-CNR, Italy
matteo.mordacchini@iit.cnr.it
[3] Dept. Computer Science, University of Pisa, Italy
ricci@di.unipi.it, alessi.it@gmail.com

Abstract. The detection of communities of peers characterized by similar interests is currently a challenging research area. To ease the diffusion of relevant data to interested peers, similarity based overlays define links between similar peers by exploiting a similarity function. However, existing solutions neither give a clear definition of peer communities nor define a clear strategy to partition the peers into communities. As a consequence, the spread of the information cannot be confined within a well defined region of an overlay. This paper proposes a distributed protocol for the detection of communities in a P2P network. Our approach is based on the definition of a distributed voting algorithm where each peer chooses the more similar peers among those in a limited neighbourhood range. The identifier of the most representative peer is exploited to identify a community. The paper shows the effectiveness of our approach by presenting a set of experimental results.

1 Introduction

The exploitation of social information filtering solutions is becoming more and more relevant in a world where there is a growing need to rapidly access and be aware of many types of distributed resources like Internet pages, shared files, on line products and news. The definition of these solutions implies to face several challenges such as scalability and the management of highly dynamic information.

Several information filtering algorithms have been investigated for centralized architectures, while their definition is still a challenging issue for highly distributed systems, like the P2P ones. The basic block required for defining these algorithms is a protocol for spreading information over the P2P overlay. Among different proposals, the one exploiting the gossip ([6,7]) paradigm looks

* The research leading to these results has received funding from CONTRAIL (EU-FP7-257438),RECOGNITION (EU-FP7-257756) and SCUBE (EU-FP7-215483) EU projects.

S. Balandin et al. (Eds.): NEW2AN/ruSMART 2011, LNCS 6869, pp. 496–507, 2011.

very promising. These proposals stem from their ability to easy cast information among a large set of interconnected nodes, even if the nodes frequently join and leave the network. These protocols present good reliability, fault tolerance and scalability. Each node maintains a partial view of the other peers in the network, the protocol is highly adaptable and the exchange of gossip messages allows the natural removal of old information in favour of new one.

The gossip based approach has been exploited to build similarity based overlays [12,13], where a link between similar peers is defined by exploiting a point-to-point similarity function. Each peer exploits a random sampling layer to discover new nodes in the P2P network so that similar peers may get in touch and new links can be defined between them.

In the proposal we have presented in [8], each peer builds its own neighbourhood which is then exploited for exchanging information related to common interests. The neighbourhood of each peer is computed by considering a similarity function based on users' profiles, so that each peer may identify those characterized by similar interests. [8] exploits Cyclon [12] and Vicinity [13] protocols, where the former allows the random sampling of the network, while the latter is exploited to define the similarity overlay. However, [8] neither investigates the concept of community in a P2P network nor defines a strategy to partition the peers into a set of communities.

The task of detecting communities in a distributed environment is a complex issue because while each peer has a local vision of the overlay, the detection of communities requires the aggregation of peers which are not directly connected in the overlay. A community detection algorithm has to identify the boundaries of each community in the similarity overlay so that the spreading of the information may be confined within a well defined region of the overlay, i.e. the portion of the network including peers interested in that information. The algorithms proposed in the literature [6,7] do not enable the discovering and the characterization of similarity groups, i.e. large groups of peers with an high degree of similarity, they only links similar peers. In order to let these group to emerge, the collective knowledge of the peers in a network has to be exploited.

This paper proposes GROUP (**G**ossip-based peer-to-pee**R** c**O**mm**U**nity building **P**rotocol), an asynchronous protocol for building communities in P2P networks. GROUP exploits Vicinity to define the similarity based overlay and a distributed election algorithm to detect the communities over the overlay returned by Vicinity. The algorithm finds out the most suitable peer able to characterize a set of peers with similar interests. Each community is paired with a representative peer and its profile is exploited to represent the community. Peers can choose the community to adhere by comparing their profile against that of the representative. At the end of the protocol, each peer belongs to the community characterized by the highest profile similarity. Community profiles allow to have a more extensive representation of the more deep characteristics that link together nodes in the network and could be exploited for a more efficient and effective information diffusion among peers.

The paper is organized as follows. Section 2 describes solutions already present in the literature. Section 3 includes a more precise definition of the problem to be faced and describes the proposed solution. Section 4 discusses the experiments conducted to evaluate the proposed approach. Finally, Section 5 presents the conclusions.

2 Related Work

Several solutions have been proposed in the past for defining P2P overlays suitable for spreading and retrieving information in an efficient way. Such solutions exploit different techniques such as Gossip [4,8,6,7,2] or Semantic Overlay Network (SON) [1].

The GosSkip[4] system 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 a 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 aim of the study conducted in [1] is to reduce the search time of queries executed in peer-to-peer networks based on a random overlay, and at the same time to maintain a high degree of autonomy of the nodes. The authors propose to define node connections based on the content shared by the peers. Thus, each subset of semantically related nodes form a Semantic Overlay Network (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. The main disadvantages of this solution is the rigid predefined structure of the SON-based overlay network.

The authors of [9] 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 modelled through 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 neighbour selection mechanism. A critical aspect of this solution is the evolution of the interests of a peer. If this leads to changes in the peer's semantic concepts, it may triggers a distributed mechanism to reorganize the overlay network, involving all the neighbouring nodes belonging to the related SONs.

The protocol described in [10] 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 GROUP: Protocol Definition

We consider a scenario where each peer is associated with a network's user which is characterized by a profile including k terms. The k terms characterize the content to which the user is interested, for instance the documents that he/she has previously accessed. This set of terms can be built incrementally, for instance according to predefined thresholds that fix when a term has to enter/leave the profile of the user.

In this scenario the goal is to achieve a partition $\mathscr{P}_I = \{P_1, \ldots, P_s\}$ of the peers such that each P_i includes a subset of the peers in $\mathscr{P}_I$ which are similar in term of their profiles. Therefore each P_i represents a different community, which can be exploited to define more focused dissemination of the information in the P2P network.

Previous approaches (e.g. Vicinity) try to gather similar nodes by putting them in direct contact when they share similar interests. On the other way round, the local knowledge of the peers typically does not allow a good identification of the features which may characterize the communities of peers. Thus, a protocol based on the collaboration between peers has to be exploited in order to reach a collective agreement over the definition of the communities and of the communities representatives.

Our aim is to detect communities and to identify them so that each peer may characterize itself as member of a community. We have chosen to exploit the profile of a peer inside each community as the community identifier. Such peer is the one that is chosen by the community peers as their *best representative*.

In order to allow each peer to be aware of those peers belonging to the network, the Cyclon and the Vicinity protocols are exploited. While the former is exploited to define a random sampling protocol, the latter is used to put in contact each peer with the most similar ones. The paper does not present these gossip protocols as they are well defined in the literature [12,13].

Instead we focus on the definition of the community detection algorithm. Our approach for building communities is based on a distributed voting algorithm which is executed over the similarity based overlay returned by Vicinity. The algorithm leads to the election of a set of communities representatives peers. Each elected peer, together with the peers that have contributed to its election constitutes a community. The community is represented by the profile of the peer which have been elected.

The detection of the representative of each community is structured in the following stages: Similar Neighbours Detection, Potential Candidates Selection, Representative Election.

Algorithm 1. Similar Neighbours Detection

```
1: Order NEIGHBORS by similarity;
2: Let BESTN = Top-k(NEIGHBORS);
3: for all b ∈ BESTN do
4:     if d(p, b) ≤ (1 − neighbor_threshold) then
5:         Send Vote to b;
6:     else
7:         Break;
8:     end if
9: end for
```

Algorithm 2. Potential Candidates Selection

```
1: Let CANDIDATES = ∅;
2: for all n ∈ {NEIGHBORS ∪ {p}} do
3:     if VotesRcvd(n) ≥ REPRTHR then
4:         Let CANDIDATES = CANDIDATES ∪ {n}
5:     end if
6: end for
7: Let break=false;
8: if CANDIDATES = ∅ then
9:     Let CANDIDATES = NGHREPRS
10:    if CANDIDATES = ∅ then
11:        Let CANDIDATES = p ad interim
12:        Let break=true;
13:    end if
14: end if
15: if not(break) then
16:    Let R = min_{c ∈ CANDIDATES} d(p, c)
17:    Send ReprVote to R
18: end if
```

Algorithm 3. ActiveThread

```
1: while true do
2:
3:     if Timer t expired then
4:         Reset t;
5:         SimilarNeighbourDetection();
6:         wait(t′)
7:         PotentialCandidateSelection();
8:         wait(t″)
9:         RepresentativeElection();
10:    end if
11:
12: end while
```

Algorithm 4. PassiveThread

```
1: while true do
2:     Receive MSG as MESSAGE
3:     M-TYPE = MSG.type;
4:     M-TSTAMP = MSG.t-stamp;
5:     M-CONT = MSG.content;
6:     M-SENDER = MSG.sender;
7:     if M-TYPE = Vote then
8:         Add M-CONT to VQUEUE with M-TSTAMP
9:         Send VQUEUE.size to M-SENDER
10:    else if MTYPE = ReprVote then
11:        Add M-CONT to REPVQUEUE
12:                with M-TSTAMP
13:        Send REPVQUEUE.size to M-SENDER
14:    end if
15: end while
```

The three phases of the voting protocol are periodically executed by an active thread created by each peer. Since each phase implies the exchange of some messages with the neighbours, a corresponding passive thread whose main task is the reception of the messages is activated by each peer. Note that this is the typical behaviour of a gossip protocol.

Alg.3 and Alg.4 describe the active and the passive thread, respectively. In the active thread the different voting phases are separated by exploiting a set of timers. The goal of the first voting phase is to detect the most similar peers in the neighbourhood of a peer. This phase, detailed in Alg.1, consists in a preliminary voting procedure in which each peer votes its most similar peers, i.e. its best neighbours in term of profiles similarity. Each peer arranges its neighbours in decreasing order with respect to their similarity value, and gives a vote to, at most, the top-k neighbours, without considering the peers whose similarity is lower than a *neighbour threshold*. k defines the maximum number of neighbours that can be voted by a peer, while the neighbour threshold represents the peer

similarity value under which two peers are not considered similar, and therefore should not be selected to be voted. $d(p, n)$ denotes the distance in the similarity space between the peer p and its neighbour n. Each vote sent by a peer p to its neighbour n is received by the passive thread of n and it is inserted into a queue which is shared with the active thread of the same peer. Furthermore, the passive thread of n sends a reply message to p including the number of votes received by n in the current voting phase. This information is in turn received by a further passive thread of p which is not presented to make the presentation simple. Different similarity metrics can be exploited in this phase, for instance Jaccard or cosine similarity [6].

After this phase, each peer has notified to all its neighbours the number of votes it has received. At this point the second phase starts where each peer chooses, among its neighbours, its potential candidate to represent its community. The potential candidate is chosen among the neighbours which have received in the previous phase a number of votes (VotesRecvd(n)) higher than the representative threshold $ReprThr$, i.e. the minimum number of votes obtained by a peer to be voted as a potential candidate. Among the potential candidates, the neighbour characterized by the higher degree of similarity is chosen. If no neighbour has received a sufficient number of votes, the peer votes itself. By tuning 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 of peers. Alg.2 shows this voting phase. Again, the potential candidate vote is received by the passive thread which notifies to the sender the number of potential candidate votes received.

Finally, in the *Representative Election Phase*, each peer elects its actual representative by considering both the potential candidate votes which it has been received by itself and those received by its neighbours. We do not show the pseudo code of this phase because of its complexity.

If a peer p has collected the highest number of potential candidate votes with respect to its neighbourhood, it considers itself as its own representative. Otherwise, a potential representative R become actual representative if, among the peer's neighbour, R is the one that has received the highest number of potential candidates votes. 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. If p discovers that its voted representative R has not voted for itself, but it has chosen in turn as its own actual representative another peer R_0 that is within a distance ϵ from R, then p changes it choice by selecting R_0 as its actual representative. If R has chosen R_0 as its representative, but they are sufficiently separated ($d(R, R_0) > \epsilon$), p maintains its choice, i.e. it chooses R as its representative. The link between R and R_0 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, because it has received a few votes. In this case p asks to its neighbours for their actual representatives and it selects the one characterized by the highest similarity value. 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, the election procedure is repeated. 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 representatives, communities are dynamic entities subjected 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 ones. The only interaction is due to the exchange of information, for exchanging both gossip updates that includes peers' profiles and votes. When a proper defined time interval has elapsed, each peer independently starts a new voting turn 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 neighbourhood. Hence, when a new voting phase starts, new peers will be considered, and old or disconnected ones will be taken into account no more. If a high number of updates in p's neighbourhood occurs, 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 these mechanisms.

4 Evaluation

This section shows the evaluation of the our approach through a set of experiments which show its effectiveness. The implementation has been realized by the OverlayWeaver Peer-to-Peer framework [11]. In order to test the "real" effectiveness of our approach, we exploit a real dataset instead of using a synthetic one. More in detail, we have used a bio-* subset of the dataset released by Mendeley [5,3], a company that produces a publication management tool that run on the client side but takes advantages of a web storage where each user can store the most interesting publications he/she owns. The dataset released contains a set of anonymous 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 retrieved the content of the papers in his/her profile, we filtered out the stop-words and we extracted the most frequent terms that we used as user profile. The similarity metric used in the experiments is the Jaccard similarity measure [6]. The timer t value was set to a time equivalent to 2 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

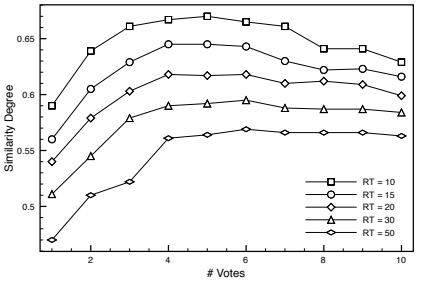

Fig. 1. Number of votes impact: Similarity Degree

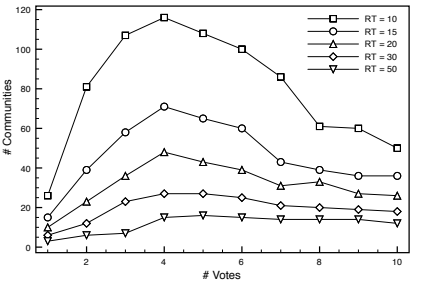

Fig. 2. Number of votes impact: Number of Communities

an algorithm for building up explicit communities composed by peers sharing a common interest, whose size should not be too small as well as too big and, possibly, independent from the network size. In order to evaluate the ability of our approach to address this goal we have measured the effectiveness of the results by considering the internal community similarity, as well as the total number of communities and, as a consequence, their mean size. To measure the mean internal community similarity we use the following Similarity Degree measure:

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

where N is the whole set 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 grade of similarity with their community members, thus enabling an effective communication on 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 analyse both these aspects in the following experiments.

Number of Votes and Representative Threshold impact: Figure 1 and Figure 2 show the effects of varying the number of votes a peer can give, with respect to the Similarity Degree and the number of communities. This is shown when different *representative_threshold* (RT in the figures) are used. The network size is fixed to 2800. Figure 1 shows that the highest values of similarity degree can be achieved when the *representative_threshold* is low. This is a quite expected behaviour. Indeed, a low threshold brings to the creation of a greater number of communities (as it is shown also in Figure 2). A higher number of communities means, in turn, a smaller number of peers per community, hence a (potential) greater internal homogeneity. Anyway, communities with too few members can be less useful when used to exchange information. It is interesting to note that

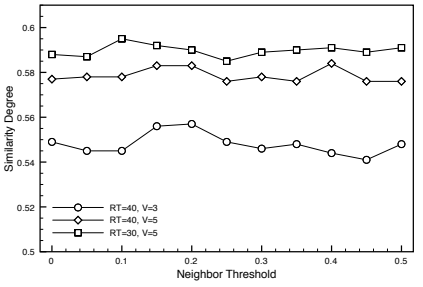

Fig. 3. Neighbor Threshold impact: Similarity Degree

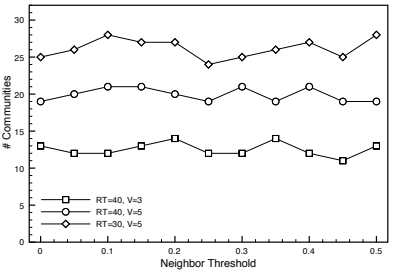

Fig. 4. Neighbor Threshold impact: Number of Communities

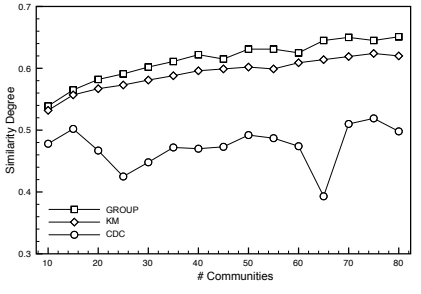

Fig. 5. Internal Community Cohesion: Similarity Degree

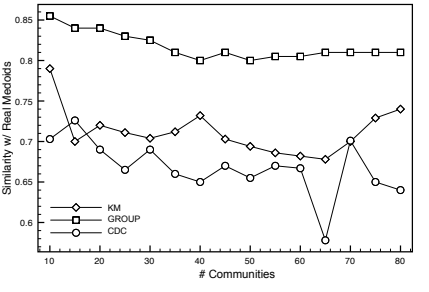

Fig. 6. Internal Community Cohesion: Similarity against the optimal medoid

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: Figure 3 and Figure 4 show the changes on the Similarity Degree and the number of communities obtained by using different values of the *neighbor_threshold* parameter. Even in this case the network size is fixed to 2800 nodes. Different *representative_threshold* are used as well as with different number votes (V in the figures) that can be expressed by a peer. As described above, it is easy to see how different representative thresholds brings up to the creation of very a different number of communities and a quite different result in terms of Similarity Degree, however both figures show how the different *neighbor_threshold* value has almost no relevant effects both in terms of communities number and in terms of Similarity Degree.

Internal Community Cohesion: Figure 5 and Figure 6 show a comparison among three different algorithms for grouping nodes, namely GROUP, the CDC algorithm[10] and K-Means, used as a centralized clustering algorithm. The network size is 2800 peers, the representative threshold is 30 and the number of votes is 4. The last two parameters are derived from the previous experiments.

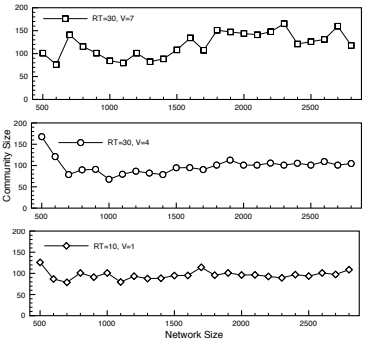

Fig. 7. Communities Mean Size

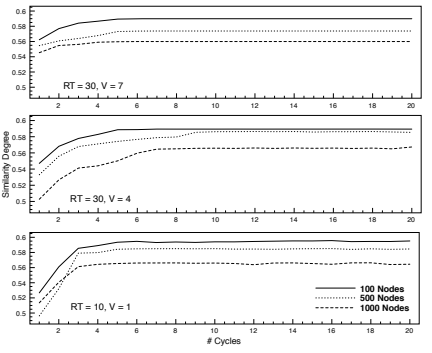

Fig. 8. Impact of Network Instability

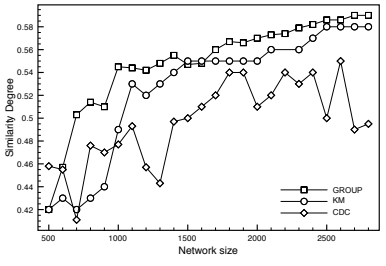

Fig. 9. Approach Scalability

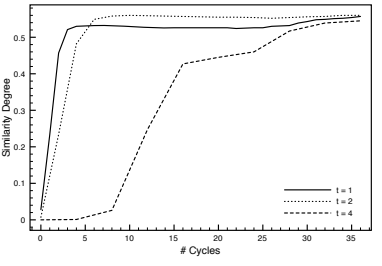

Fig. 10. Impact of different timers

The comparison has been conducted measuring both the Similarity Degree of communities and computing the distance between the community representative and the real medoid of the communities formed by each algorithm. It is easy to see that how our approach is able both to build communities characterized by a higher internal similarity degree and to elect better representatives, i.e. representatives that have near the "real" centers (in term of similarity) of the communities formed by GROUP.

Communities Mean size: Figure 7 shows the effect of the network size to the mean size of the communities. We used three different settings of thresholds and votes. This figure shows that the network size has basically little or no influence on the size of the communities. Indeed, the mean community size is about 100 elements. This is due to 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: Figure 8 shows the impact of instability in the network due to the presence of failing peers and the ability of our approach to re-build communities characterized by an high Similarity Degree. Namely, this figure shows the results achieved on a initial network of 2800 nodes by suddenly

(i.e. without any graceful departure mechanism) removing 100, 500 and 1000 peers respectively. We used three different settings of votes and thresholds pairs. Cycles can be seen 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 more nodes, the final community has a lower Similarity Degree, due to the fact that, probably, 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 lead 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 showing the robustness of the proposed approach.

Scalability: Figure 9 shows the scalability of our approach. The figure shows the 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 using a threshold of 30 votes and allowing 4 votes per peer. K-Means has run using the same number of communities that came out from GROUP. It is easy to note that our approach performs better than the other algorithms with almost all tested network sizes. In particular, it shows a clear trend to improve its performance when the network size increases.

Timer Impact: Figure 10 depicts the impact of choosing different values of the timer 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 be optimal only with an extra amount of cycles. A value that let the similarity layer to converge but leave a still small time from one wave to another, seems to achieve the best results.

5 Conclusions

This paper presents GROUP, a distributed P2P community building protocol whose goal is to identify a set of representatives peers characterizing user communities. A representative is a peer elected by a set of peers on the basis of their similarity with it. The approach is based on a set of P2P epidemic protocols. Several tests conducted using the Mendeley dataset have shown the effectiveness of our approach in comparison to a set of other ones. We are currently integrating *GROUP* with a locality sensitive DHT based on min wise independent permutations which indexes the communities. In this way a new peer joining group is able to efficiently retrieve and choose the more similar communities.

References

1. Aberer, K., Cudré-Mauroux, P.: Semantic overlay networks. In: Proc. of the Int. Conference on Very Large Data Bases, vol. 31, p. 1367. Citeseer (2005)
2. Bertier, Frey, Guerraoui, Kermarrec, Leroy: The Gossple Anonymous Social Network. In: Proc. of the ACM/IFIP/USENIX 11th Middleware Conference (Middleware 2010), pp. 814–829 (2010)
3. Jack, K., et al.: Mendeley's reply to the DataTEL challenge. In: Proc. of the 1st Workshop on Recommender Systems for Technology Enhanced Learning (2010)
4. Guerraoui, R., Sidath, B.H., Kermarrec, A.M., Le Fessant, F., Huguenin, K., Rivière, E.: Gosskip, an efficient, fault-tolerant and self organizing overlay using gossip-based construction and skip-lists principles. In: Sixth IEEE International Conference on Peer-to-Peer Computing, 2006 Ratnasamy (2001)
5. Henning, V., Reichelt, J.: Mendeley-A Last. fm For Research? In: IEEE Fourth Int. Conference on eScience, 2008, pp. 327–328. IEEE, Los Alamitos (2009)
6. Jelasity, M., Guerraoui, R., Kermarrec, A.M., van Steen, M.: The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 79–98. Springer, Heidelberg (2004)
7. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based peer sampling. ACM Trans. Comput. Syst. 25(3), 8 (2007)
8. Mordacchini, M., Baraglia, R., Dazzi, P., Ricci, L.: A p2p recommender system based on gossip overlays (prego). In: CIT, pp. 83–90 (2010)
9. Penzo, W., Lodi, S., Mandreoli, F., Martoglia, R., Sassatelli, S.: Semantic peer, here are the neighbors you want! In: Proc. of the 11th Int. Conf. on Extending Database Technology: Advances in Database Technology, pp. 26–37. ACM, New York (2008)
10. Ramaswamy, L., Gedik, B., Liu, L.: A distributed approach to node clustering in decentralized peer-to-peer networks. IEEE Transactions on Parallel and Distributed Systems, 814–829 (2005)
11. Shudo, K., Tanaka, Y., Sekiguchi, S.: Overlay weaver: An overlay construction toolkit. Computer Communications 31(2), 402–412 (2008)
12. Voulgaris, S., Gavidia, D., Van Steen, M.: Cyclon: Inexpensive membership management for unstructured p2p overlays. Journal of Network and Systems Management 13(2), 197–217 (2005)
13. Voulgaris, S., van Steen, M.: Epidemic-style management of semantic overlays for content-based searching. In: Cunha, J.C., Medeiros, P.D. (eds.) Euro-Par 2005. LNCS, vol. 3648, pp. 1143–1152. Springer, Heidelberg (2005)