

# Social Market: Combining Explicit and Implicit Social Networks

Davide Frey, Arnaud Jegou, and Anne-Marie Kermarrec

INRIA-Rennes Bretagne Atlantique, Rennes, France

**Abstract.** The pervasiveness of the Internet has lead research and applications to focus more and more on their users. Online social networks such as Facebook provide users with the ability to maintain an unprecedented number of social connections. Recommendation systems exploit the opinions of other users to suggest movies or products based on our similarity with them. This shift from machines to users motivates the emergence of novel applications and research challenges.

In this paper, we embrace the social aspects of the Web 2.0 by considering a novel problem. We build a distributed social market that combines interest-based social networks with explicit networks like Facebook. Our Social Market (SM) allows users to identify and build connections to other users that can provide interesting goods, or information. At the same time, it backs up these connections with trust, by associating them with paths of trusted users that connect new acquaintances through the explicit network. This convergence of implicit and explicit networks yields TAPS, a novel gossip protocol that can be applied in applications devoted to commercial transactions, or to add robustness to standard gossip applications like dissemination or recommendation systems.

## 1 Introduction

The advent of Online Social Networks (OSN) has shifted the core of Internet applications from devices to users. Explicit social networks like *Facebook*, or *LinkedIn* enable people to exploit real-world connections in an online setting. Collaborative tagging applications such as *delicious*, *CiteULike*, or *flickr* form dynamic implicit networks of users on the basis of their online activities, interest profiles, or search queries. Users can not only access and introduce new available content but they become themselves accessible through the online infrastructure. Existing online social networks can be grouped into two main categories: explicit and implicit. In explicit networks, users explicitly determine which other users they should be connected to. In *Facebook* or *MySpace*, they issue and accept friendship requests. In *Twitter*, they decide that they wish to follow the tweets of specific users. In all cases, the topology of the resulting network reflects the choices of users and often consists of links that already exist between real people. Explicit networks are therefore very useful in reinforcing and exploiting existing connections but provide little support for discovering new content [3, 1]. Implicit networks fill this gap by taking an opposite approach which allows users to discover new content, and acquaintances [4].

Specifically, implicit networks form dynamic communities by collecting information about collateral activities of users, such as browsing websites or tagging documents, URLs, or pictures. A given user may or may not be aware of the other members of her own communities depending on the target application. Other users should be clearly visible if the purpose of the application is to discover new people, while they may be hidden for the sake of privacy when they are simply being used as proxies to access new interesting content. In either case, the ability to establish new social connections is key to identifying new and useful data.

Recent years have seen the emergence of a significant number of research efforts and applications exploring the power of each of these two paradigms. Nonetheless, a lot more can be achieved if both approaches are combined into a single framework. Consider the following example. John, who lives in London, bought two electronic tickets for a classical-music concert in Paris, a concert version of *Berenice* by Handel, but an unexpected event makes him and his friend unable to travel to Paris to attend the concert. The concert is tomorrow and John would like to sell the tickets to someone who can actually attend the event. Unfortunately, while John has many friends interested in classical music, they are all based in the United Kingdom. He does know a few people in Paris, but they are mostly colleagues or friends he met while traveling and who do not share his musical tastes. He tries calling a few of them but his best bet is Joseph, who claims to have a friend whose parents often go to classical-music concerts. Unfortunately, this friend of Joseph's is out of town and Joseph does not know how to reach his parents. As a last resort, John posts a message on a French classical music forum, linking to an EBay ad. However, none of the classical music fans on the forum are responsive enough and some of them even become suspicious that the electronic ticket being sold by this new forum user is actually a fake.

John's problem would be very easy to solve if he was able to contact someone that, albeit not knowing him directly, was at the same time interested in the concert and would trust him enough to buy an electronic ticket from him. This is exactly what can be achieved by combining the discovery potential of implicit networks, with the real-world guarantees provided by trusted social links in explicit ones. While implicit networks do not convey any kind of trust, explicit links almost always carry some kind of trust properties resulting from being friends, coworkers and so on.

In our example, the implicit network allows John to identify people that could be interested in the concert. Among these, he discovers François, a music teacher from Paris who is trying to buy two tickets for one of his students and himself. A cross check on the explicit network then allows John to assess François's trustworthiness. He finds out that François is actually the cousin of a French colleague of his wife. This allows the two to gain confidence in each other and thus complete a safe transaction without external help.

Combining the discovery capabilities of implicit networks with the trust and confidence that are inherent in friendship relationships is useful not only in

the context of commercial transactions but also in applications like information dissemination, or recommendation. Consider a distributed news dissemination system [6]. Users receiving an unusual news item will be more likely to be interested in it if the source or the user that forwarded it is associated with some level of trust. Similarly, recommendation systems can take into account the trust of users in other people's opinions.

The power of combining explicit and implicit social networks, however, has a cost. First, the enormous amount of information necessary to manage user profiles requires significant storage capacity as well as computing power to determine who the best users are for a given transaction. Second, the associated costs are equally enormous and can only be afforded by a few very large companies, and this, in turn, brings significant privacy problems. Modifications in the terms of service of websites like Facebook have already caused public uproars in multiple occasions. Most people are rightfully upset at the idea that their personal data may be collected by a company and sold to third parties for whatever reasons. The most promising way to continue to use personalized information is therefore to develop scalable decentralized solutions.

A class of protocols that appear to be particularly suited for this purpose are those based on the gossip paradigm. Initially introduced in the context of distributed databases, gossip protocols have rapidly shown their applicability in a large number of applications including data dissemination, overlay maintenance, and more recently social networks. In this paper, we extend existing work on interest-based gossip overlays [4] and propose *Social Market*, a solution for the identification of trusted social acquaintances.

Our main contribution in Social Market is TAPS (Trust-Aware Peer Sampling), a novel protocol that operates by directly incorporating trust relationships extracted from an explicit social network into the gossip-based overlay. This provides each user with a set of neighbors that are not only useful but that can also benefit from a high degree of trust. Through extensive simulations, we show that Social Market and TAPS achieve performances that are comparable to those obtained by protocols equipped with global system knowledge, while limiting the diffusion of sensitive trust information. This makes our solution directly applicable to situations like the social transaction example described above. Moreover, our results open new directions for making existing gossip-based applications more robust in the presence of unreliable users.

## 2 System Model and Problem Definition

Social Market (SM) is a novel distributed application enabling users to identify previously unknown social acquaintances that, at the same time, are similar to them and can be trusted through a chain of explicit social connections, the *trusted path*. Selecting similar users is crucial when searching for the right people for a given transaction, but also when building recommendation or data-dissemination systems. Trust enables the implementation of transactions without external help and increases users' confidence in recommendation results.

We consider a system consisting of a set of users equipped with interconnected computing devices that enable them to exchange information in the form of messages. Each user is associated with a *user profile* that characterizes her interest, her past behavior, her geographical location, and whatever other information the user wishes to add. The profile is essentially a vector of strings that can represent, for example, URLs, words, or phrases. We refer to each such string as a *keyword*. Each *keyword* in a profile is also associated with a counter, its *weight*, which counts how many times the keyword has been added to the profile. The weight basically measures how relevant a given keyword is with respect to the other keywords in the profile. Keywords can be added by the user, they can be extracted from her browsing history, as well as from her interaction with Social Market. To simplify notation, we refer to a user and her profile with the same symbol  $u \in U$ , where  $U$  is the universe of all user profiles. Profiles can be compared with each other using a standard similarity metric. Even though our solution can operate using any similarity metric, in the following, we make use of the well known *cosine similarity* [19], which measures normalized overlap.

$$Sim(u_1, u_2) = \cos(u_1, u_2) = \frac{u_1 u_2}{\|u_1\| \cdot \|u_2\|}$$

Users interact with Social Market by proposing items that they wish to exchange with other users. An item can be, for example, an object to sell, an object to buy, but it can also be a question that is being asked and that needs an answer. When a user  $u$  creates an item, she associates it with an item profile, similar to what is done in [2]. Structurally, an item profile is identical to a user profile. Upon creation, the system initializes the item profile to the corresponding user profile. The user then completes the creation by selecting which keywords from this profile clone should be kept, which should be removed, and which, if any, new keywords should be added.

In the example of Section 1, John creates an item for the Handel concert in Paris. Prompted with a profile that contains, among other things: *computer science*, *cycling*, *mountaineering*, *violin*, *rock*, and *classical music*, John decides to keep only *violin* and *classical music* in the item profile. He then adds two more keywords, *Paris* and *Handel*, and decides to keep only the latter in his user profile as he's not interested in being notified about other items associated with Paris. Once an item has been created, the goal of Social Market is to lead this item to *meet* other users who

- (i) are interested in the item,
- (ii) can be trusted and can trust the creator of the item,
- (iii) can be reached through a trusted path on the social network (Figure 1a).

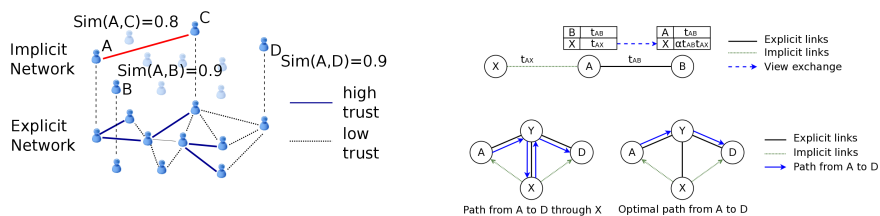
To make this possible, Social Market users can create explicit social links. While similar to friendship links in systems like Facebook, SM links also have an additional feature: *trust*. Upon establishing a link, users declare how much they trust each other by specifying a value in  $(0, 1]$ . The value of the trust link is similar to the degree of friendship/confidence specified in some existing social

networks such as CouchSurfing<sup>1</sup>. In particular, if user A assigns a value close to 0 to the link to a user B, it does not necessarily mean that A completely distrusts B, but it may simply mean that A does not know B enough to express a positive opinion.

In the following, we present our solution to address Social Market’s goals. In this version of our work, we assume that two explicit friends always agree on a trust value for the link they share. This yields an undirected social graph with arc weights between 0 and 1. Extensions to the directed case as well as mechanisms to guarantee high levels of privacy and resilience to attacks are outside the scope of this paper and will be the subject of our future work.

### 3 Social Market

Identifying users that, at the same time, are interested in an item and towards whom it is possible to identify a trusted path requires an effective protocol to group users according to these two conflicting requirements. Recent research on gossip-based protocols has shown their effectiveness in building overlays that cluster nodes or users according to some distance function or similarity metric. In this section, we extend this research by presenting a novel protocol capable to provide each user or item with a set of neighbors that have highly similar profiles and to which there is a trusted social path. Because users and items are treated identically by our protocol, we refer to both with the term *node*.



(a) Importance of a trusted path: A selects C, rather than B or D as a neighbor even if it has a lower similarity value because it is reachable through a trusted path (high-trust links).

(b) Exchange of trust information (top) and Short circuiting of trusted paths (bottom)

Fig. 1

#### 3.1 Background: Gossip-Based Implicit Networks

Protocols for gossip-based clustering generally consist of two layers. The bottom one is a random peer sampling protocol (RPS) [11] which provides each node with a continuously changing view of the network. The properties of the RPS are such that the resulting topology, that is the union of all the RPS views, can

<sup>1</sup> www.couchsurfing.org is a social community supporting the exchange of accommodation between its users.

be assimilated to a random graph. This makes the RPS effective in maintaining a connected overlay in the presence of disconnections and arrivals of new nodes. The top layer is also a gossip-based overlay maintenance protocol and is based on a variation of [20]. Specifically, at each gossip exchange, a node selects its neighbors by choosing those with the best similarity values.

Social Market exploits a similar protocol structure. Instead of a random peer-sampling layer, it applies our novel trust-aware peer-sampling protocol, TAPS. TAPS provides the clustering protocol with candidate nodes that not only have similar interests but that can also be trusted. The clustering protocol then uses this information to select those that offer the best compromise between trust and interest similarity as shown in Figure 1a.

To summarize, each node maintains three data structures: the explicit view, the TAPS view, and the CLUSTER view. The explicit view contains the node's explicit friends, while the other two are maintained by the TAPS and clustering protocols as described in the following.

### 3.2 Trust-Aware Peer Sampling

TAPS follows the general structure of a peer-sampling protocol described in [11]. Its goal is to populate the TAPS view with an ever-changing set of references to other nodes. Periodically, each node contacts a node selected from its TAPS view and the two exchange subsets of their respective views.

In a standard peer-sampling protocol, each view entry contains at least information on how to contact the corresponding node (eg. IP address and port), and an age or timestamp value indicating when the information in the entry was generated. In TAPS, we introduce additional fields. First, as in [4], we add a user profile. This makes it possible to cluster similar nodes together by computing the cosine similarity of their profiles. In addition, we include an *inferred trust value*, and a *trusted path*. The *inferred trust value* indicates the trust that a node can have in another node. If the other node is a friend in the explicit network, then the trust value corresponds to the one agreed upon when the friendship relationship was established. Otherwise, it is an inferred value that depends on the path that the trust information has taken during the gossip dissemination, the *trusted path*.

*Trust Propagation.* Each edge in a trusted path carries some amount of uncertainty about the trustworthiness of the target node even if all the nodes in the path fully trust each other. To model this, we define the *inferred trust* of a path as the product of the trust values of its edges, weighted by a *trust transitivity* coefficient,  $\tau$ , expressing how much a node values other nodes' recommendations. Given a path  $u_1, u_2, \dots, u_n$  with trust values  $t_{1,2}, t_{2,3}, \dots, t_{n-1,n}$ , the inferred trust between  $u_1$  and  $u_n$  is

$$\tilde{t}_{1,n} = \tau^{n-2} \prod_{i=1}^{i=n-1} t_{i,i+1},$$

Lower  $\tau$  values cause trust to decay faster with path length. For example, with  $\tau = 0.7$  trust decays from 1 to 0.49 in only three hops.

The ability to perform trust inference is what makes TAPS the perfect candidate for building our Social Market application. In the following we describe the mechanisms that enable trust inference to be implemented in the context of a gossip-based overlay protocol.

*Initialization and View Exchanges.* An inference process can only produce results if it starts from some reliable initial values. In TAPS, these values are those that have been agreed upon at the creation of explicit friendship relationships. We therefore initialize the TAPS view by inserting one entry for each explicit neighbor. During the course of the protocol, these entries are exchanged with entries received from other nodes. Initially nodes providing new entries will be the node's explicit friends, then the node's friends' friends, and so on.

As the gossip process evolves, nodes collaborate in computing inferred trust values. Consider a node  $A$  exchanging profiles with another node  $B$  as shown in the top diagram of Figure 1b.  $A$  sends  $B$  a subset of its view as well as the value it has for  $t_{AB}$ , the inferred trust between  $A$  and  $B$ .  $B$  uses this information to update the inferred trust values before adding the nodes to its own view. Specifically, let  $t_{AX}$  be the trust  $A$  has in  $X$ , then  $B$  computes its own trust for  $X$ ,  $t_{BX}$ , as follows. First it verifies if it already has a value for  $t_{AB}$ . If so, it keeps the highest value between the received one and its own. It then uses the selected  $t_{AB}$ , to compute  $t_{BX}$  as

$$t_{BX} = \tau t_{AB} t_{AX}.$$

During the course of dissemination, a node  $A$  will inevitably receive multiple references for the same node  $X$ . Each of these may arrive from a different neighbor and carry a different trust value. In the presence of multiple trust values for the same reference, a node always selects the largest. However, two references for the same node may also contain slight differences in the node's profile, and in this case, the node should keep the most recent information. To balance these two needs, when  $A$  receives a reference to a node  $X$ , it chooses the highest trust value between the one in its view and the one it received, and combines it with the most recent profile. Because  $\tau < 1$ , choosing the largest trust value guarantees that nodes always selects the most direct trust values, thus converging towards shorter and shorter trusted paths. Nonetheless, some nodes may still be unable to infer the trust of other nodes through the best social paths.

To enhance the chances of a successful trust inference, a node initiates gossip exchanges not only towards nodes in its TAPS view but also to those in its explicit neighbors. These additional gossip exchanges are executed every  $T_{exp}$  TAPS cycles and cause nodes to exchange entries from the unions of their TAPS and explicit views. Explicit-view exchanges increase the probability that nodes can infer trust through the shortest available paths.

*Managing Trust Paths.* When exchanging trust information with each other, nodes also update the associated path information. Specifically, when a node  $A$  receives a reference to node  $X$  from node  $B$ , it computes the associated path  $p_{AX}$  by concatenating  $p_{AB}$  and  $p_{BX}$ . Maintaining paths up to date is not only necessary to enable users to enforce correct transactions in the social market

application, but it is also useful to correct the degradation of trust values caused by possible loops in the inferred paths.

Consider the situation depicted in the bottom diagram of Figure 1b. Node  $X$  holds a reference to  $D$  with a trust value  $t_{XD}$  and a path going through  $Y$  and sends it to  $A$ . Node  $A$  should combine this with the trust value it has for node  $X$ ,  $t_{AX}$ , also obtained through  $Y$ . However, this would lead to a path that uselessly goes twice through node  $Y$  and once through  $X$ .

Node  $A$  can prevent this by short circuiting the path, thus computing a more accurate trust value. The problem is that the information received from  $X$  only contains the identifiers of the nodes in the path and not their trust values. Including such values in the path would allow the re-computation of trust, but at the cost of disclosing trust information to third parties.

We therefore replace the re-computation of trust with the computation of a lower bound. Specifically,  $A$  knows that the aggregated impact on trust of the segment  $YX$  cannot be greater than  $\tau^n$ ,  $n$  being the number of useless links in the path, each link being counted once for each time it is traversed ( $n=2$  in this case). It can therefore conclude that the trust value of node  $D$  as seen from  $A$ ,  $T_{AD}$  is at least:

$$t_{AD} \geq \frac{t_{XD}t_{AX}}{\tau^n}$$

This makes it possible to increase the accuracy of trust inference without disseminating private trust information to third parties.

### 3.3 Clustering Trusted Nodes

Our Trust-Aware Peer Sampling protocol provides each node with a continuously changing set of nodes along with their inferred trust values. This constitutes a source of information for our clustering protocol, a variation of the well-known Vicinity protocol [20]. This protocol maintains a `CLUSTER` view that collects the nodes that offer the best compromise between trust and similarity.

*Filling the Cluster View.* In order to fill its `CLUSTER` view, each node periodically selects the node with the oldest timestamp and exchanges the content of its view with it. Upon receipt of another node's view, a node  $X$  combines the received view, its own view, and its own TAPS view, and selects the entries that are associated with the best trade-off between similarity and trust. Specifically, for each entry  $N$ ,  $X$  computes a score  $s_{XN}$  as follows

$$s_{XN} = Sim(X, N)^{2-\epsilon} t_{XN}^\epsilon, \quad \epsilon \in [0, 2]$$

where  $\epsilon$ , the *trust weight*, determines the importance of trust in the trade-off.

To speed up convergence, nodes also update their `CLUSTER` views when they receive new information through the TAPS protocol. In this case, they simply combine the received TAPS view with the current `CLUSTER` view and extract the nodes with the best trade-off between trust and similarity. The selected nodes replace those that were previously in the `CLUSTER` view.

*Trust Verification.* The decentralized nature of our trust-inference protocol can allow nodes to cheat on their trust values when communicating with nodes that are not direct friends. For example, in Figure 1b (bottom), node D could try to enter A’s cluster by making A believe that it has a high-trust link to node Y.

To prevent this, each node verifies the trust values of the entries in its CLUSTER view, once they have remained in the view for at least  $c$  cycles.<sup>2</sup> To achieve this,  $A$  asks  $D$  to forward a *verification message* back to  $A$  along the trusted path. The message starts with a trust value of 1. Nodes along the path multiply the message’s value by  $\tau$  and by their trust for the node they received the message from.  $Y$  thus multiplies 1 by  $\tau t_{YD}$  in the example. In the absence of colluding nodes, this process causes the *verification message* to reach  $A$  with the correct value for  $t_{AD}$  thereby invalidating D’s cheating attempts.

## 4 Evaluation

We evaluated the effectiveness of our approach by means of extensive simulations on several data traces obtained from real social networks. In the following we first present the details of our setting, and then discuss our results.

### 4.1 Setting

We evaluated our protocol on real data traces consisting of 3000 users extracted from the Facebook and Digg social websites. The Facebook trace<sup>3</sup> contains friendship links and a list of social interactions. To obtain a treatable subset for our experiments, we first cleaned up the trace by removing all users that had only one friendship link, as they would be too isolated to benefit from our social platform. We then selected the user with the largest number of interactions and proceeded in a spiral fashion by selecting her friends, then the friends of her friends, and so on, until we reached 3000 users. We associated each of these users with a random user profile from the Digg social network. We obtained these profiles by crawling Digg in late 2010.

Friendship links in the Facebook trace and profiles in the Digg trace provided the base explicit links and profiles for our experiments. On top of them, we built several traces by varying the trust patterns between the nodes. We distinguish our traces into two groups: binary and multi-valued. In both, we made the assumption that the number of interactions between two nodes is a measure of trust (this assumption is not part of the protocol itself).

*Binary Traces.* In binary traces we assigned a binary trust value to each link in the data set. Specifically, we sorted the friends of each user according to the number of interactions she had with them. Then, for a user with  $|N|$  friends, we assigned a trust value of 1 to the  $\beta|N|$  directed links with the largest number of

<sup>2</sup> Similar to [4], we choose  $c = 5$ .

<sup>3</sup> Network A at <http://current.cs.ucsb.edu/facebook/index.html>.

interactions. Because, this process creates asymmetric trust values, we then set the symmetric trust value of each link as the logical OR of the two asymmetric values. These lead to the proportions of trusted links depicted in Figure 2a.

*Multivalued Traces.* While binary traces provide a simple experimental setting, reality tends to be more complex. Thus, we also considered traces with trust values of 1, 0.8, 0.5, and 0. Similar to the binary case, we sorted each user's friends by the number of interactions and assigned a trust value of 1 to the top  $\gamma_1|N|$ , a value of 0.8 to the following  $\gamma_{0.8}|N|$  and so on, leading to the three traces shown in Figure 2b.

## 4.2 Terms of Comparison

We compared the performance of our Social-Market solution against several alternatives. First, we considered *best*, an ideal systems that, powered with global knowledge, always provides each user with the set of neighbors that offer the best combination of similarity and trust. This allows us to assess TAPS's ability to reach similar results in a decentralized way. Then we considered *oracle*: this consists of a standard similarity-based implicit network [4], augmented with an oracle that provides each user with the best trusted path to her neighbors. *Oracle* therefore maximizes profile similarity at the cost of possibly lower trust values.

We also compared against two variations of our protocol. *Full-trust* is a version in which nodes exchange complete trust information along with trusted paths. This makes it possible to short-circuit long paths more accurately than as described in Section 3.2 at the cost of disclosing trust values. *Full-Trust* is only shown in multi-valued traces as it is equivalent to TAPS in binary ones. *TAPS no-bound* is instead a version of TAPS in which nodes do not attempt to short-circuit cycles thus yielding less accurate trust computation. Finally, unless otherwise specified, we ran our simulations using default values for  $\tau$  and  $\epsilon$ .  $\tau = 0.75$  provides a good balance between trust decay and path length (0.56 at 3 hops, 0.23 at 6).  $\epsilon = 1$  gives instead a fair tradeoff between trust and similarity.

## 4.3 Results

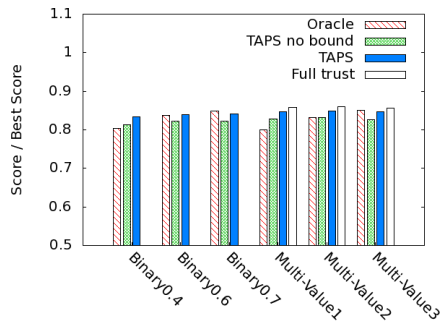
*Impact of Trust Density.* We start our performance comparison by examining the results obtained in the various traces with a trust transitivity of  $\tau = 0.75$ . Our results, depicted in Figure 2c, show the average score values in the CLUSTER views with TAPS as well as with its competitors as percentages of *best*'s scores. TAPS's performance is either comparable or better than that obtained by *Oracle* with the use of global knowledge. As expected, TAPS is particularly good whenever the social network has a limited proportion of high-trust links (binary-0.4 and multi-valued-1 and -2). This can be explained by observing that *Oracle* always selects the nodes with the best cosine similarity and is therefore penalized in networks where the density of high-trust links is lower. With smaller  $\tau$  values, we observed that TAPS outperforms *Oracle* in all the considered traces.

Name	$\beta$	% 1	% 0
Binary-0.4	0.4	53.5	46.5
Binary-0.6	0.6	71.3	28.7
Binary-0.7	0.7	80.7	19.3
Binary-0.8	0.8	89.3	10.7

(a) Binary Traces

Name	% 1	% 0.5	% 0.25	% 0
MultiValued-1	41.3	23.8	23.4	11.5
MultiValued-2	57.4	27.9	13.3	1.4
MultiValued-3	76.2	12.3	10.1	1.4

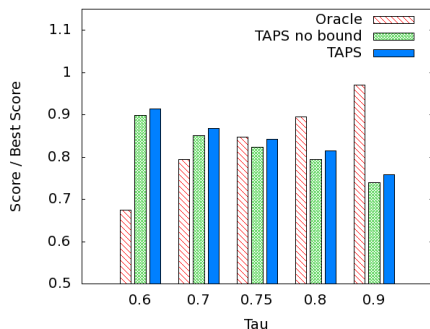
(b) Multi-valued Traces



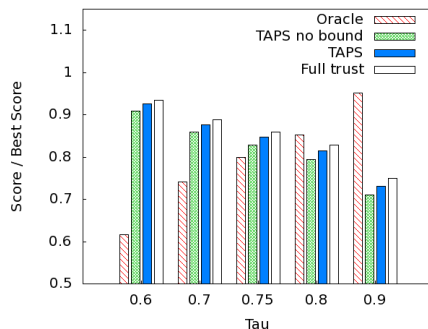
(c) Impact of trust density

Fig. 2: Trust values distribution for different traces (left), impact of trust density (right).

*Impact of Trust Transitivity.* Figure 3 confirms the above observation by examining the impact of trust transitivity,  $\tau$ , on performance in the binary-0.8 (left) and in the multi-value-1 traces. Results show that the performance of TAPS is particularly good when trust decays faster. A faster decay gives more importance to nodes that are closer in the social network even if they may have poorer similarity values. This suggests that a protocol like TAPS becomes more and more important when it is being used for important transactions and in situations in which people can tolerate only limited amounts of risk.



(a) binary



(b) multi-valued

Fig. 3: Impact of  $\tau$  in the binary (left) and multi-valued (right) traces.

With very high values of  $\tau$  trust decays more slowly. In this case, an protocol like *Oracle* that first finds the most similar nodes and then searches for a trusted path may be viable. However, it should be noted that *Oracle* achieves this through global knowledge. A distributed protocol to compute trusted paths would probably be either very costly or ultimately equivalent to TAPS in networks characterized by a high trust density. Moreover, such a protocol would remain inapplicable in situations where the density of trusted links is low, as shown in Figure 2c.

*Impact of Trust Weight.* Next, we evaluate the impact of the *trust-weight* factor on the performance of TAPS and its competitors. Figure 4 shows the result in the binary (left) and multi-valued (right) traces respectively. Both plots show that the benefits of a protocol like TAPS become more important as more weight is placed on the trust between nodes. With values of  $\epsilon$  above 1, TAPS performs better than *Oracle* even when considering its *no-bound* version.

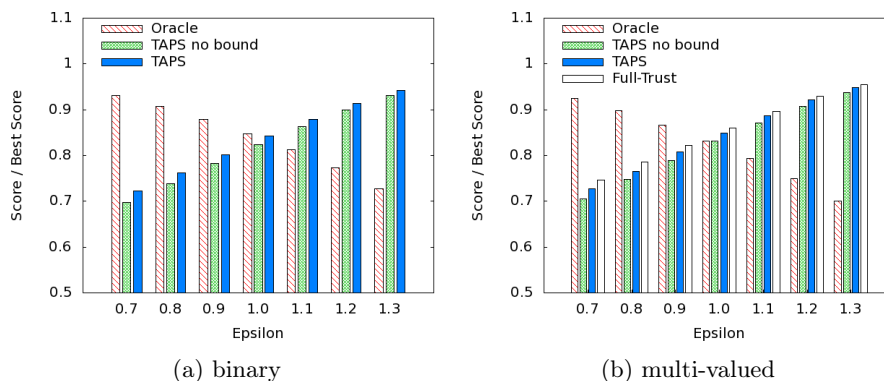


Fig. 4: Impact of  $\epsilon$  in the binary (left) and multi-valued (right) traces.

In addition, the plots show that the importance of short-circuiting cyclic paths is greater when the value of  $\epsilon$  is small. This is because smaller values make it possible for the protocol to select nodes that are farther away in the social network, which, in turn, makes the presence of cycles more likely.

*Graph Properties of TAPS.* To better understand the behavior of TAPS, we conclude our evaluation by examining the properties of the TAPS overlay from a graph-theoretical perspective. First we observe that convergence speed is comparable to that obtained with standard protocols: views reach 90% of their scores within 15 cycles and completely converge after 80. Then we examine our TAPS and clustered overlays in terms of clustering coefficient and in-degree distribution to assess how close they are, respectively, to a random and a clustered graph.

Figure 5a shows the cumulative distributions of the local clustering coefficients of the nodes in our TAPS, and CLUSTER views (TAPS and TAPS-cluster) and compares them to those of a standard peer sampling protocol (RPS) and a standard clustering protocol that does not consider trust (RPS-cluster). The plot (in logarithmic scale) shows that, according to expectations, the TAPS topology is slightly more clustered, and thus less uniformly random, than a standard RPS topology. Conversely, our CLUSTER topology is slightly less dense than the one based on pure similarity.

The in-degree distribution shown in Figure 5b also shows some differences with respect to traditional protocols. In this case, the difference is more accentuated between TAPS and the RPS than for the corresponding clustered overlays. The in-degree distribution of TAPS is in fact slightly skewed because nodes

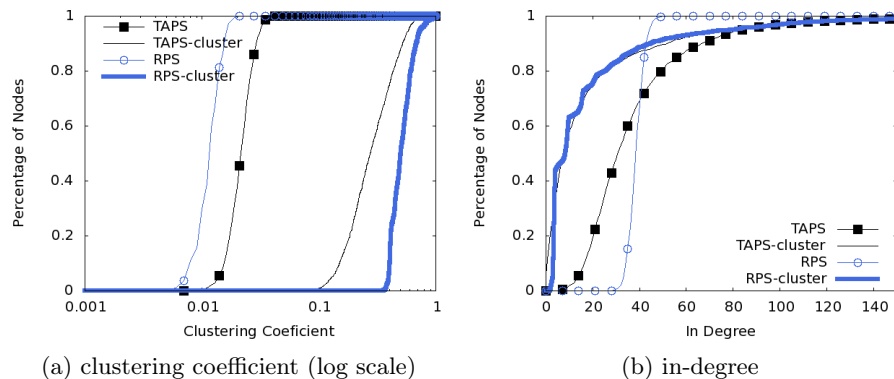


Fig. 5: Cumulative distributions of the local clustering coefficients (left) and of the in-degree (right) distributions for TAPS-based and standard protocols.

that are not trusted by many others tend to have fewer neighbors. This is indeed a desirable property as untrusted nodes could potentially harm the system through illicit behaviors. Moreover, while these differences are inherent in the trust-dependence of the TAPS overlay, their small absolute value suggests that TAPS could probably replace traditional protocols in a number of applications.

## 5 Related Work

The concept of trust in explicit social networks has been exploited in domains ranging from peer-to-peer security to recommendation systems. SybilGuard [23] and SybilLimit [22] propose protocols that exploit trust relationships between friends to protect peer-to-peer systems from sybil attacks. Reliable Email [9] uses a similar approach to build an email-whitelisting system based on friend-to-friend relationships, while Ostra [18] exploits social trust to limit the incidence of unwanted communication in messaging and content-sharing systems.

NABT [15] proposes the use of trust between friends to prevent freeriding behaviors using a more efficient form of tit-for-tat based on indirect trust relationships. NABT's credit-based approach can be viewed as a basic form of trust inference between friends of friends. A more advanced approach to trust-inference is adopted by SUNNY [14], a centralized protocol that takes into account both trust and confidence to build a Bayesian network. Even if SUNNY is centralized, its confidence-based idea could lead to interesting improvements for TAPS.

A number of research efforts have instead investigated the use of trust links to improve the performance of recommendation systems. [21] uses an approach similar to that of [14], while TrustWalker [10] combines trust and item-based collaborative filtering. TaRS [17] builds a recommendation system capable of operating both with global and with local trust metrics. Global trust metrics [7] predict a global reputation value for each node. Local trust metrics [16], on the other hand, take an approach similar to ours and compute trust values that are dependent on the target user.

Despite the mole of work on social trust, Social Market is, to the best of our knowledge, the first system to propose the use of trust relationships to build a decentralized interest-based marketplace. Similarly, TAPS is the first attempt to combine explicit and implicit social networks into a single gossip protocol. Existing research on gossip protocols has addressed a number of problems including data dissemination [5], aggregation [13], and overlay construction and maintenance [12, 20]. In this context, the two contributions that are most closely related to our work are [8], which uses gossip to disseminate news in explicit networks, and [4], which proposed the use of gossip for implicit ones.

## 6 Conclusions and Future Directions

We presented Social Market (SM), a novel distributed application designed to enable trusted collaborative actions between similar people in a social-network environment. We proposed a solution to the challenges posed by SM in the form a novel trust-aware peer-sampling protocol, TAPS, which creates an RPS-like overlay taking into account the mutual trust expressed by users when joining an explicit social network. Our experimental results show that, combined with a clustering protocol, TAPS is highly effective in providing users with high-quality implicit acquaintances that not only share similar interests but are also reachable through a verifiable trusted path. This makes Social Market a promising platform for the development of decentralized user-to-user transactions and warrants further investigation into the use of trust to secure existing gossip protocols.

Our promising results encourage us to extend Social Market and TAPS in a number of ways. First, we are examining the possibility of using asymmetric trust values as opposed to symmetric ones. This would make it possible to render trust information more private as users would not need to disclose to their friends the trust they have for them. Second, even though the gossip nature of TAPS makes it inherently self healing, we are considering solutions to maintain the quality of trust paths even during churn by having nodes rely on trusted peers to disseminate information while disconnected. Finally, we plan to explore the use of multiple redundant trusted paths as a way to limit the effects of colluding nodes, which cannot be tolerated by this version of the protocol. These improvements will enable us to integrate TAPS in our existing prototype applications as a way to reinforce the trust of users in disseminated information, recommendations, and ultimately in the implementation of a real-world social-market platform.

*Acknowledgments* This work is supported by the ERC Starting Grant GOSSPLE number 204742. We are grateful to Guang Tan for his initial contributions, as well as to the anonymous reviewers for their constructive suggestions.

## References

1. Ahn, Y.Y., Han, S., Kwak, H., Moon, S., Jeong, H.: Analysis of topological characteristics of huge online social networking services. In: WWW (2007)

2. Bai, X.: Personalized top-k processing: from centralized to decentralized systems. Ph.D. thesis, INSA Rennes, France (2010)
3. Bender, M., Crecelius, T., Kacimi, M., Miche, S., Xavier Parreira, J., Weikum, G.: Peer-to-peer information search: Semantic, social, or spiritual? TCDE 30 (2007)
4. Bertier, M., Frey, D., Guerraoui, R., Kermarrec, A., Leroy, V.: The gossple anonymous social network. In: *Middleware* (2010)
5. Birman, K.P., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., Minsky, Y.: Bimodal multicast. TOCS 17 (1999)
6. Boutet, A., Frey, D., Guerraoui, R., Kermarrec, A.M.: Whatsup: News, from, for, through, everyone. In: *P2P* (aug 2010)
7. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems* 30(1-7) (1998)
8. Datta, A., Sharma, R.: Godisco: selective gossip based dissemination of information in social community based overlays. In: *ICDCN* (2011)
9. Garriss, S., Kaminsky, M., Freedman, M.J., Karp, B., Mazières, D., Yu, H.: Re: Reliable email. In: *NSDI'06*. San Jose, CA (May 2006)
10. Jamali, M., Ester, M.: Trustwalker: a random walk model for combining trust-based and item-based recommendation. In: *KDD* (2009)
11. Jelasiy, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based peer sampling. TOCS 25(3) (2007)
12. Jelasiy, M., Babaoglu, O.: T-Man: Gossip-based overlay topology management. In: *ESOA'05* (Jul 2005)
13. Jelasiy, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.* 23 (August 2005)
14. Kuter, U., Golbeck, J.: Sunny: a new algorithm for trust inference in social networks using probabilistic confidence models. In: *AAAI*. AAAI Press (2007)
15. Liu, Z., Hu, H., Liu, Y., Ross, K.W., Wang, Y., Mobius, M.: P2p trading in social networks: the value of staying connected. In: *INFOCOM* (2010)
16. Massa, P., Avesani, P.: Controversial users demand local trust metrics: an experimental study on epinions.com community. In: *AAAI*. AAAI Press (2005)
17. Massa, P., Avesani, P.: Trust-aware recommender systems. In: *RecSys*. ACM, New York, NY, USA (2007)
18. Mislove, A., Post, A., Druschel, P., Gummadi, K.P.: Ostra: leveraging trust to thwart unwanted communication. In: *NSDI*. Berkeley, CA, USA (2008)
19. Salton, G., McGill, M.J.: *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc. (1986)
20. Voulgaris, S., van Steen, M.: Epidemic-style management of semantic overlays for content-based searching. In: *Europar* (2005)
21. Wang, Y., Vassileva, J.: Bayesian network trust model in peer-to-peer networks. In: *AP2PC* (2003)
22. Yu, H., Gibbons, P.B., Kaminsky, M., Xiao, F.: Sybillimit: a near-optimal social network defense against sybil attacks. *IEEE/ACM TON*. 18 (June 2010)
23. Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.D.: Sybilguard: defending against sybil attacks via social networks. *IEEE/ACM TON*. 16 (June 2008)