$\mathcal{DS}^4$: Introducing Semantic Friendship in Distributed Social Networks

Paraskevi Raftopoulou[1], Christos Tryfonopoulos[1], Euripides G.M. Petrakis[2], and Nikos Zevlis[1]

[1] University of Peloponnese, Tripoli, Greece
[2] Technical University of Crete, Chania, Greece
{praftop,trifon,cst06027}@uop.gr
petrakis@intelligence.tuc.gr

Abstract. Existing social networks are based on centralised architectures that manage users, store data, and monitor the security policy of the system. In this work, we present $\mathcal{DS}^4$, a Distributed Social and Semantic Search System that allows users to share and search for content among friends and clusters of users that specialise on the query topic. In $\mathcal{DS}^4$, nodes that are semantically, thematically, or socially similar are automatically discovered and logically organised into groups. Content retrieval is then performed by routing the user query towards social friends and clusters of nodes that are likely to answer the query. In this way, search receives two facets: the social facet, addressing friends, and the semantic facet, addressing nodes that are semantically close to the query. In this work, we present the main components of our architecture, the protocols and services that regulate social and semantic interactions, and an extensive experimental evaluation both on simulated scenarios and on an implemented prototype. Our experiments are the first in the literature to demonstrate that searching only among friends is not effective in distributed social networks (returns only 10% of the relevant content), and showcase the necessity and importance of semantic friendship. By utilising the semantic friendship our design achieves a recall up to 80%, while message traffic is kept low.

1 Introduction

In recent years a number of social networking services have been developed to offer users a new way of sharing, searching, and commenting on user-generated content. Following the development of such services, people have shown great interest in participating in "social" activities by generating and sharing vast amounts of content, ranging from personal vacation photos to blog posts, or comments and like/agree/disagree tags. All these social networking services are typically provided by a centralised site, where users need to upload their content, thus giving away access control and ownership rights as a requirement to making it available to others. This centralised administrative authority may sometimes utilise the content in any profitable way, from selling contact details to marketing firms to mining of user information for advertising purposes. Furthermore, the rate of growth of both content and user participation in such services raises concerns about the scalability of the centralised architectures used, as they are called to serve millions of users and gigabytes of content every day.

Idea and Challenges. In this work, we present a *distributed social networking architecture* that allows users to share and search for content in a fully decentralised way, while at the same time maintaining access control and ownership of their content. Our work builds upon research results from the peer-to-peer (P2P) paradigm, such as those utilising unstructured, small-world, and semantic overlay networks (SONs) [1–3]. Replacing the centralised authority with a distributed self-manageable community of nodes removes access control and ownership issues and ensures high-scalability and low maintenance costs. For this reason, recent efforts in the industry and the literature have also resorted to the P2P paradigm for building decentralised social networks and platforms (like Diaspora[3], KrawlerX[4], and OpenSocial[5]), and have developed architectures [4] and prototype systems [5] relying mainly on Distributed Hash Tables (DHTs). Contrary to DHTs that focus on providing accurate location mechanisms, $\mathcal{DS}^4$ emphasises on node autonomy, content-based grouping of nodes, and loose component architecture by relying on the SON paradigm. The designed system is scalable (requires no centralised component), privacy-aware (users maintain ownership and control over their content), automatic (requires no intervention by the user), general (works for any type of content), and adaptive (adjusts to changes of user content or interests).

In $\mathcal{DS}^4$, node organisation is achieved by executing *identification and grouping of semantic friends* (periodically) by each node. This protocol operates by establishing connections among *semantically similar nodes* (in addition to the social connections) and by discarding connections that are outdated or pointing to dissimilar nodes. The goal of this protocol is to create *groups/clusters of nodes* with similar interests. User queries can then be resolved by routing the query towards friends and nodes specialising to the query topic. In this way, content search is leveraged to another type of friendship often ignored in social networks: the *semantic friendship* emerging from common user interests or user profiles.

Contribution. In the light of the above, the contributions of this work are threefold:

- We define a distributed social networking architecture that offers fundamental social interactions, while emphasising on *content search*, *user autonomy*, and *data ownership*. To the best of our knowledge, this is the first approach to propose a *distributed social networking system* that introduces the notion of *semantic friendship* between users.
- We present the *protocols* and *services* that regulate node interactions. We provide details on the definition of the semantic friendship between users, present details on the distributed social and semantic search algorithms, and discuss *system implementation* issues.
- We show the importance of semantic friendship in such a distributed context by means of experimentation with real social networking data, both on a *simulated environment* and on a *prototype system*. Our experiments show that by resorting only to friends for content retrieval we achieve recall as

[3] https://joindiaspora.com/
[4] http://www.krawler.com/
[5] http://docs.opensocial.org/display/OS/Home

low as 10%, while introducing a basic notion of semantic similarity between nodes increases recall to 30%, but incurs high message traffic. Contrary, with the utilisation of semantic friendship we achieve a recall up to 80%, while message traffic is kept low.

Application Scenario. As an example of an application scenario let us consider Mary, a computer scientist whose main field of expertise is bioinformatics. Mary is interested in following the work of prominent researchers in the area and willing to share her own work and ideas with other researchers; this interaction would help her set-up her research agenda and generate innovative ideas. Currently, she would have to (i) resort to a number of digital libraries or scientific databanks, like DBLP or GenBank, to identify interesting bibliography, (ii) use one of the numerous centralised social networks, like LinkedIn or ResearchGate (aimed for professional or research use), to follow the work of other researchers, and (iii) use one of the numerous file hosting services, like Dropbox or Fileserve, to exchange her dataset with her colleagues. Clearly, Mary would benefit from accessing a Web 2.0-inspired solution that is able to provide socially-/semantically-aware search and data sharing with ownership and access control in an *integrated service*. This system would be a valuable tool, beyond anything supported in current centralised social networks, that would allow Mary to save both time and effort.

In our example scenario, consider a research community comprised of researchers working in different institutions and content providers of scientific material, like digital libraries, scientific publishers, or scientific databanks. In this context, each user will personalise and maintain its own node in the distributed social network that will act as an access point to the network services. A node will allow its user to socially connect with colleagues or collaborators and exchange ideas, manuscripts, or even datasets in a P2P fashion, thus maintaining full control of the data dissemination process. Additionally, a user might also be semantically connected in an automatic but user-centered fashion to other nodes with similar interests –maintained either by other researchers in the same field or by specialised content providers. Additionally, nodes may also be deployed by larger institutions, like research centers or content providers (e.g., CiteSeer, ACM, Springer), to provide access points for their content that may be freely disseminated or even priced (on the basis of pay-per-item or subscriptions). The $\mathcal{DS}^4$ architecture would be a promising solution to such a setting as it (i) emphasises the seamless integration of information sources, (ii) supports user-centered access control over the shared content, and (iii) enhances fault tolerance and requires no central administration authority.

The rest of the paper is organised as follows. Section 2 discusses related work. Section 3 introduces the proposed architecture, implemented services, and protocols that regulate node interactions, while Section 4 presents our experimental evaluation on a simulated environment and on the deployed prototype system. Finally, Section 5 concludes the paper and discusses future research directions.

2 Related Work

In this section, we discuss related work in the context of platforms for distributed social networks and social data management.

2.1 Distributed Social Platforms

All available social networks (e.g., Facebook, LinkedIn, Elgg) are currently based on centralised solutions both for storing and managing of content, which set scalability limitations on the system and reduce fault-tolerance. Industry has already detected these drawbacks and has lately turned into solutions that diverge from the centralised model of the existing systems by developing platforms, such as Diaspora, Krawler and OpenSocial, that provide APIs to support application hosting in remote application servers, owned and managed by the application providers. In a similar spirit, a strand of research work also moved towards hierarchical organisations for supporting distributed social networking. The distributed social systems SuperNova [6] and Scope [7] are based on a two-tier architecture, where nodes with higher computing capability become super-nodes and form an overlay to provide distributed data management of the P2P social network. Client nodes connect to super-nodes and rely on them for bootstrapping, sharing their content, and accessing the shared information. Although all of the platforms and system schemes propose the decentralisation of the social services, one of the main issues of the centralised architectures persists: the existence of a single point where user information is collected and may be exploited.

To alleviate the above disadvantage, distributed platforms for social online networks based on the P2P paradigm were proposed [8–10]. LifeSocial.KOM [10] is a plugin-based extendible social platform that provides secure communication and user-based data access control, and integrates a monitoring component that allows users and operators to observe the quality of the distributed system. Similar efforts aimed at spontaneous social networking; they include proposals for distributed social services in resource constrained devices (like tablets or smartphones) [11, 12] or in environments with no infrastructure guarantees (e.g., high-attendance events) [7]. All these approaches offer different types of distributed social platforms that allow users to create communities, share content, and send messages, but do not emphasise expressive content search mechanisms.

Finally, other approaches in distributed social networking emphasise on delivering innovative and competitive services; SCIMS [13] relies on an ontology-based model for managing social relationships and status, the work in [14] aims at personalising search results based on user context and friendship relations, while Gemstone [15] targets data availability in the absence of the data owner. To achieve this, a replica storage scheme based on social relationships, online patterns of nodes, and user experiences is utilised.

2.2 Distributed Social Data Management

Our work fits mainly into the area of data management in distributed social networks and is inspired by previous approaches on SONs [1–3] and on works that emphasise on distributed content location in social networks. Works like the eXO [4] and SoNet [16] systems are, similarly to $\mathcal{DS}^4$, inspired by the P2P paradigm to provide content location and management services on large-scale decentralised social networks. To do so, the authors rely on a structured overlay and exploit the accurate location mechanisms, but de-emphasise node autonomy. Contrary to these approaches, $\mathcal{DS}^4$ employs a loose component architecture and introduces a new type of social relations between nodes: the semantic closeness of

content. In this way, nodes that are similar in terms of content, create emergent groups likewise to the creation of social relations. Our work shares ideas with the SocialCDN system [17], where social caches (links among friends) are introduced as a way to alleviate the network traffic and optimise data dissemination (mainly by social updates). In [17], social cache selection is formulated as the neighbour-dominating set problem and a family of algorithms is proposed and evaluated. Contrary to $\mathcal{DS}^4$, where the emphasis is on efficiently supporting expressive content retrieval in the social paradigm, the emphasis on SocialCDN is on the reduction on network traffic to facilitate fundamental social interactions.

The loose component architecture and the emphasis on node autonomy of [18] resemble the architectural design of $\mathcal{DS}^4$, where an unstructured overlay network of nodes is utilised to support the distributed social infrastructure. However, the focus of [18] is on the design of gossip protocols for efficiently disseminating profile updates to all interested users and does not put any attention to the problem of content search and management.

Furthermore, the concept of creating and maintaining social connections in distributed infrastructures is affined with the problem of distributed data management in P2P networks. In SONs [3, 19], "social" connections between the peers (e.g., similarity of content, pattern, or distance in a physical level) are exploited to direct the search to nodes with relevant data (e.g., as in [20] that studies query routing strategies based on "social" relationships). Other works on SONs (e.g., [2]) focus more on the organisation of P2P networks as small-world networks, where peers self-organise in groups of similar interests to facilitate message-efficient query answering. Our work on $\mathcal{DS}^4$ borrows concepts and ideas from research on SONs and extends them for facilitating efficient and effective data management in a social network setting. We suggest that SONs offer the most promising architectural solution inspired from the P2P paradigm; it is a perfect fit for a distributed social networking scenario providing high decentralisation, high node autonomy, support for emergent semantic and social structures, and effective object location mechanisms. Contrary, DHT-based architectures [4, 5] ignore node autonomy (by enforcing deterministic key/content placement) and emergent structures (by enforcing network structure).

Finally, a large number of research in the domain of distributed social networks consists of studies on system security [5, 21], user privacy [5, 21–24], distributed access control [25, 26], and authentication mechanisms [25]. Clearly, security issues are also relevant in our design and the $\mathcal{DS}^4$ system could benefit by adopting approaches, like [25] or [23], that enforce user privacy and access control. However, the problem of security is orthogonal to our design and is not further analysed as it is not the emphasis of this work.

3 The $\mathcal{DS}^4$ Social Networking System

A distributed alternative for social networks should provide the same functionality as current solutions, while avoiding centralised storage and ownership of user data (that are anyway naturally distributed to users' personal computers). In our design, we aim at providing support for open social networks and innovative services, while preserving end-user privacy and information ownership. The benefits of distributed self-managing solutions extend over the architectural domain

to economic figures of social networks. With the proliferation of high-bandwidth internet connections and powerful off-the-shelf personal computers, a distributed solution based on P2P technology will realise a cost-effective way to develop and manage social networks, avoiding maintenance, storage, and administrative costs of centralised solutions.

3.1 Protocol Overview

To demonstrate the necessity and effect of semantic friendship in a distributed social networking system we have designed and implemented four different protocols that may be executed by each node. All the protocols outlined below, together with their implementation details with regard to each of the system services, are described in detail in Section 3.3. The performance of each protocol in terms of retrieval effectiveness and message traffic is evaluated in Section 4.

The FI Protocol. Under the FI protocol, each network node maintains a Friend Index (FI) routing table containing the contact details (i.e., IP and port) of the social neighbourhood of the node, comprised of explicitly declared friends that participate in the $\mathcal{DS}^4$ network. This is the most straightforward form of a distributed social network one could design and is used for providing the baseline comparison for our experiments. In the FI protocol neither interest identification nor semantic grouping is applied, since a node addresses a limited number of random friend node(s) at query time, thus performing a message-bounded flooding.

The FI+i Protocol. The FI+i protocol is similar to the FI protocol described above, but targets to demonstrate the usefulness of the semantic information in its simplest form. Under the FI+i protocol, when a node n connects to the $\mathcal{DS}^4$ network its interests are automatically derived by its local content as described in Section 3.2. Each node maintains a FI routing table containing the contact details mentioned above, together with the interest descriptions derived from the interest identification process executed locally by its social friends. In this way, at query time a node may address the friend(s) that are the most relevant to the query. The protocol is named after the combination of the FI with interests.

The SI+g Protocol. In the SI+g protocol, each node n maintains, in addition to the FI containing the contact details and interest descriptions of social friends, a Semantic Index (SI) routing table containing the contact details and interest descriptions of nodes sharing similar interests with node n (called semantic friends). The links in the SI form the semantic neighborhood of the node, and will be refined accordingly by using the semantic grouping service described in Section 3.3 by resorting *only to the* SI of other nodes. In this way, at query time the node(s) most relevant to the issued query are addressed. This protocol is designed to demonstrate the importance of semantic friendship in the content retrieval process, and is named after the combination of the SI with the grouping of semantic friends.

The FISI+g Protocol. The FISI+g protocol is similar to the SI+g protocol described above as the node maintains again two routing tables (SI and FI). However, under the FISI+g protocol the contents of SI will be modified during semantic grouping by resorting *both to the SI and FI* of other nodes. Similarly to the SI+g protocol, at query time the node(s) most relevant to the issued query are addressed. This protocol is designed to demonstrate the importance of the FI in the grouping of semantic friends, and show the potential of combining social and semantic friendship. The protocol is named after the combination of both FI and SI with the grouping of semantic friends.

3.2 Interest Identification

The interests of each node are automatically derived by its local content (meta-) data (e.g., user-defined tags, document titles, etc.) and naturally, a node may have more than one interests. The interests of a node are identified automatically, i.e., by applying a standard clustering algorithm like Hierarchical Agglomerative Clustering (HAC) [27] or K-means [27] on the node's local content repository. A node's interest is represented by the corresponding index terms (if an external reference system is used) or by the centroid vector (if content is categorised by clustering). Each node is then represented by the list of the centroid vectors of its interests.

3.3 $\mathcal{DS}^4$ Service Description

The main idea behind $\mathcal{DS}^4$ is to let nodes that are semantically, thematically, and socially close self-organise to facilitate the search mechanism. The services regulating node join, creation and maintenance of semantic friendship, query processing, and social interactions for all $\mathcal{DS}^4$ protocols mentioned above are discussed in the following sections.

Join Service. When a node connects to the $\mathcal{DS}^4$ network, it uses the join service which differs depending on the protocol followed. According to the FI protocol, a node joining the $\mathcal{DS}^4$ network needs only to locate its social friends and update its FI with their contact details.

Joining the network under the FI+i, SI+g, and FISI+g protocols involves a different procedure, since the node has to derive its interests as described in Section 3.2. In the FI+i protocol, after calculating its interests, the node contacts its social friends to notify them about its interests and asks in turn for their interests. This information is then used to update its FI. In addition to the FI, in the SI+g and FISI+g protocols, each node maintains also a semantic index SI_{ik} containing the contact details and interest descriptions of nodes sharing similar interests (called *short-range links*) and of a few nodes having different interests (called *long-range links*). This semantic index is maintained for each distinct interest I_{ik} of node n_i. A node may merge or split its semantic indices by merging or splitting its interests, depending on changes in its local content.

These (initially randomly selected) links form the semantic neighborhood of the node; the (short-/long-range) links contained in SI are refined accordingly by using the semantic grouping service (invoked either periodically or explicitly

by the user) described below. In the following, for simplicity of the presentation, we assume that each node n_i has only one interest I_i, and thus maintains only one SI_i. The same discussion applies for multiple interests, since nodes maintain one semantic index per interest and the semantic grouping service is applied independently for each one of them.

Semantic Grouping Service. The semantic grouping service is responsible (i) for updating social and semantic friends' interests both in FI and SI, and (ii) for reorganising the semantic neighborhood (SI) of the node by establishing new connections and discarding old ones, forming groups of nodes with higher semantic friendship (i.e., more similar interests).

For this reason, the FI protocol requires no semantic grouping and incurs no extra message cost at the operation of the system, as it uses only the FI without node interests. Additionally, semantic grouping for the FI+i protocol refers only to the regular update of the node interests stored at the FI, and incurs a cost proportional to the number of social links maintained in the FI.

The most interesting facet of the semantic grouping service is utilised in the SI+g and FISI+g protocols. In these protocols, the network nodes use the service to form groups of nodes based on their likelihood to have similar interests (i.e., their semantic friendship). As already mentioned in the previous section, the SI of each node is used to maintain two types of semantic friendship links: short-range links that are links to other nodes with similar interests, and long-range links that links to nodes having different interests. Short-range links are used to *create groups* (clusters) of nodes with high semantic friendship, while long-range links are used to *maintain the connectivity* of remote semantic groups (clusters) in the system. This procedure for grouping semantic friends is executed locally by each node and aims at clustering nodes with similar content, so as to allow forwarding of queries to social friends and clusters of semantic friends that are similar to the issued query.

Semantic grouping resembles *emerging group creation* behaviours in social networks, where users with similar interests utilise word-of-mouth to cluster around a fan page or another user. This process in $\mathcal{DS}^4$ may occur in two facets: manually, by explicitly selecting to become friends with other users, and automatically through the semantic grouping service. This procedure, may also function as a *recommendation mechanism* for creating new social friendships.

Each node n_i may (either periodically or when explicitly invoked) initiate the semantic grouping service. The node computes its average Semantic Neighborhood Similarity (SNS) to semantic friends as:

$$SNS_i = \frac{1}{|c_i|} \cdot \sum_{\forall n_j \in c_i} sim(I_i, I_j), \tag{1}$$

where $|c_i|$ is the number of n_i's connections (i.e., social friends and/or short-range links depending on the protocol). If the similarity computed is greater than a (user-defined) threshold θ then the node does not need to take any further action, since it is surrounded by nodes with similar interests. Otherwise, the node initiates a group refinement process by forwarding a message in the

network with a (user-defined) time-to-live (TTL) to collect other nodes' interests. In particular, under both SI+g and FISI+g protocols, a node n_i issues a FINDNODES $= (ip(n_i), I_i, P, t_g)$ message, where $ip(n_i)$ is the IP address of n_i, P is an (initially empty) list, and t_g is the TTL of the message. The issued message is forwarded with equal probability to either m randomly chosen nodes or the m most similar nodes to the message initiator. The rationale of applying either of the forwarding strategies is that the message initiator should be able to reach similar nodes both directly (through other similar nodes), but also indirectly (through propagation of the FINDNODES() message through non-similar nodes). The only difference between the SI+g and FISI+g protocols lies in the utilisation of the routing tables: the SI+g protocol uses only the semantic connections of a node (i.e., only the entries in the SI routing table), while the FISI+g protocol uses both the semantic and social connections (i.e., entries in both the SI and FI routing tables) to compute neighbourhood similarity and propagate the message. In this way, we are able to study the effect of social friendship in the grouping of semantic friends.

Each node that receives the FINDNODES() message adds its contact details and its most similar interest in list P of the message, reduces TTL by one, and forwards the message in the same manner. When the TTL reaches zero, the message (containing the contact information and interests of all nodes that received it) is sent back to the initiator node, which uses the collected information to refine the (short-/long-range) links contained in the *SI*. Additionally, to speed up semantic grouping, every intermediate node receiving the FINDNODES() message may utilise the message information to refine its semantic connections. The pseudocode providing a high-level description of semantic grouping in protocols SI+g and FISI+g is given in Figure 1(a).

Query Processing Service. Queries are issued as free text or keywords and are formulated as term vectors. Subsequently, the node issues a query message in the network with a (user-defined) TTL t_q using its social (FI) and/or semantic (SI) connections depending on the protocol implemented. All the nodes receiving the query message reduce t_q by one and apply the same forwarding technique; the query message is not forwarded further in the network when $t_q = 0$. Additionally to the forwarding of the query message, each node executes the query locally, identifies matching content and returns appropriate pointers and metadata to the query initiator. Finally, the query initiator collects all responses, produces a list with the candidate answers ordered by similarity to the issued query, and presents the list to the user. Figure 1(b) summarises the steps of the query processing algorithm.

Query forwarding depends on the implemented protocol, as not all protocols have the same routing indices. In the FI protocol a node forwards the query to m social friends found in the *FI*, without resorting to any form of semantic relatedness between the issued query and the contacted nodes. This is a baseline scenario used to demonstrate a distributed social network without explicitly designed content retrieval capabilities.

Contrary to the FI protocol, where the query q is forwarded in an uninformed way, in the FI+i protocol q is forwarded in an informed way: a node forwards a

```
1: compute SNS_i
2: if SNS_i < θ then
3:    P ← { }
4:    initiate message FindNodes = (ip(n_i), I_i, P, t_g)
5:    issue FindNodes to neighbours n_j of n_i
          ▷entries from SI in the case of SI+g
          ▷entries from SI and FI in the case of FISI+g
      the issuing nodes are with equal probability
      m random or
      the m most similar to n_i
6:    P ← P ∪ {(ip(n_j), I_j)}
7: reduce message TTL t_g by 1
8: do the same for the neighbours of n_j
9: repeat until message TTL t_g = 0
10: return list P to n_i
```

(a)

```
1: if sim(q, I_i) > θ then   ▷against n_i's interest
2:    R_i ← { }
3:    if sim(q, d) > θ then   ▷against n_i's local content
4:       R_i ← R_i ∪ (d, sim(q, d))

5:    initiate message Query = (ip(n_i), q, t_q)
6:    issue Query to neighbours n_j of n_i
      the issuing neighbours are
      all short-range links of n_i
      and the friends of n_i similar to q
7: else
8:    issue Query to
      the m connections of n_i most similar to q
9: reduce query TTL t_q by 1
10: do the same for each visited node n_j
11: repeat until query TTL t_q = 0
12: return answer sets R_j to n_i
13: rank results R ← ∪R_j by similarity to q
```

(b)

Fig. 1: Pseudocode for (a) grouping of semantic friends and (b) query processing

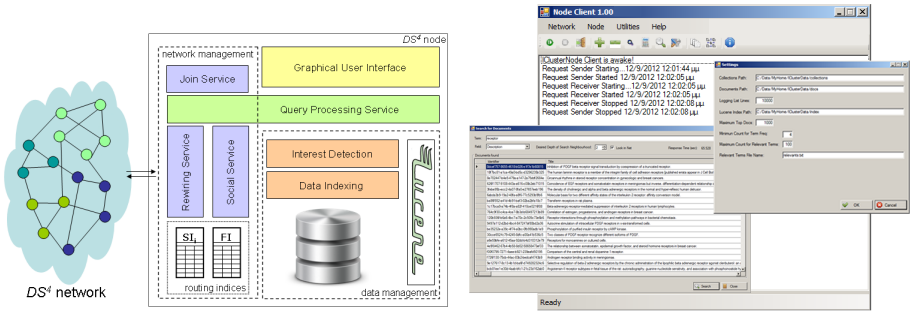
query message to its m social friends that havethe highest similarity to q, i.e., to those nodes in its FI that have interests closer the issued query.

Finally, query forwarding in the SI+g and FISI+g protocols is the same and involves forwarding the query to (i) a number of semantic friends that are expected to have interests similar to the issued query and are contained in the SI of the node (semantic facet of the search), and (ii) a number of social friends contained in the FI of each node that are similar to the issued query (social facet of the search). Specifically, the message initiator compares the query against its interests and, if similar, the query is forwarded to all of its short-range links and similar friends, i.e., the message is *broadcasted* to the node's neighborhood (*query explosion*). Otherwise, the query is forwarded to the nodes (found in either the SI or the FI) that have the highest similarity to the query (*fixed forwarding*). The query routing strategy induced by both protocols is referred to in the literature as the *fireworks* technique [1, 2] and combines low message traffic with high efficiency in clustered distributed environments.

Social Networking Services. All social networking functionality is implemented by resorting on unicast or multicast messages to the explicitly declared friends in the FI. Each node may add (or remove) a friend from the FI, send a personal message to one of his friends, post an announcement to a subset of his friends, or follow a friend (i.e., subscribe to posts or content changes). To add or remove a friend requires only local changes in the FI. Additionally, to communicate with another friend a node n_i issues a FriendComm $= (ip(n_i), msg, f)$ message containing the IP address of n_i, a free text bulletin, and a flag indicating the type of message (post or follow). The node receiving a FriendComm() message makes the message content available to the user.

3.4 Prototype Implementation

The $\mathcal{DS}^4$ prototype system is build upon Microsoft .NET Framework v4.0 using C# and the Lucene v2.9.1.2 library for performing the content clustering



(a) (b)

Fig. 2: (a) $\mathcal{DS}^4$ node architecture and (b) GUI with results and settings screen

and interest identification. Figure 2(a) shows a high-level view of a $\mathcal{DS}^4$ node and the different types of services implemented. A user in $\mathcal{DS}^4$ may utilise the node join service to connect to the social network and invoke interest creation to automatically cluster the content to be shared and identify one of more user interests. Additionally, the user may also manage his own collection in the local index store and add, remove, or modify the content or the metadata (e.g., tags). Interest creation may be invoked by the user when a significant amount of content in its local store has changed, or when the user wants to add/remove an interest. Apart from sharing the content with the rest of the community, the user may use the query processing service to issue Boolean, multi-keyword, and wildcard queries on the shared (meta-)data and discover new content. The content discovery process is automatic and returns (i) relevant results from the users' local store and (ii) content created by friends (social search) or nodes specialising on the query topic (semantic search). Finally, a user may refine/refresh its connections manually by invoking the semantic grouping service at any time. All actions are facilitated through a graphical user interface (Figure 2(b)).

4 Experimental Evaluation

The experiments in this section are designed to demonstrate the necessity and importance of semantic friendship in a distributed social network, and showcase the feasibility and qualitative benefits of our design. In the next sections, we show that content searching restricted only in social friends (protocol FI) is not effective in distributed social networks (returns only 10% of the relevant content), while small improvements are observed when we maintain the interests of social friends and use them at query time (protocol FI+i). We also show that by utilising the notion of semantic friendship and creating groups of nodes with similar interests (protocols SI+g and FISI+g) recall is significantly improved (we receive up to 80% of the relevant content), while keeping message traffic low. Finally, we demonstrate that our prototype implementation –apart from supporting fundamental social interactions– may be also used as a full-fledged content search engine in a distributed social network setting.

4.1 Simulation Measurements

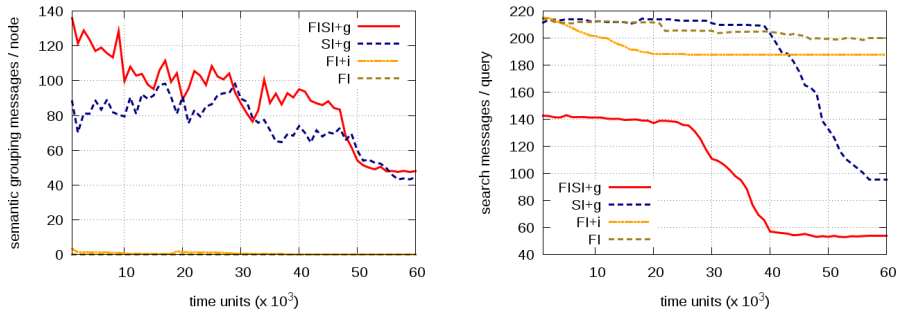
Set-Up and Measures. The search efficiency and effectiveness of $\mathcal{DS}^4$ and the effect of semantic friendship have been tested on a data set derived from a home crawl of a real-life social network. For our simulations we used a crawl of the Delicious social bookmarking site performed in November 2011, comprised of about 700K users, 10M tags, and 21M bookmarks belonging to 170K categories. The social graph followed power-law distribution in social friendships, with the average number of social friends being 11. Notice that social friendships in Delicious are not symmetric, i.e. user A following (being a friend of) user B does not imply that B is also a friend of A. The average overlap between two social friends was 15% in friends, 17% in tags, and around 5% in data (bookmarks).

In each graph of this section, we randomly selected a part of 200K users from the user graph of our data set as nodes in the distributed social network, and averaged our results over 10 different network topologies and 100 queries per network. The queries employed in the evaluation are tags that were strong representatives of bookmark categories. In our simulation each node periodically tries to improve its semantic friends by initiating a semantic grouping procedure. The base unit for time used is the period t; the beginning of the semantic grouping procedure for each node is chosen uniformly at random from the time interval $[0, 4Kt]$, and its periodicity is selected in the same way from the time interval $[0, 2Kt]$ (and differs for each node). The simulation was run for all four protocols described in Section 3.1.

The performance of $\mathcal{DS}^4$ is evaluated in terms of the effectiveness of the retrieval process and the communication load incurred (both for the semantic grouping and the query processing). The accuracy of retrieval is evaluated using *recall* (i.e., percentage of qualifying answers retrieved with respect to the total number of qualifying answers in the network). Notice that, in our setting, precision is always 100% since only relevant content is retrieved. Additionally, the *network load* is measured by the total number of messages (requests and answers to requests) exchanged by the nodes during semantic grouping or querying.

The simulator was implemented in C/C++ and all experiments were run on Linux. The baseline parameter values used are $|SI| = 20,\ \theta = 0.5,\ t_g = 3,\ t_q = 5,\ m = 3$; a parameter setup is better than another if it achieves better recall for less communication load. In the following, we present how the most important of the parameters affect both the effectiveness and efficiency of the retrieval; the rest of the parameter setup experiments are briefly discussed at the end of this section due to space reasons.

Communication Load. Figure 3(a) presents the number of semantic grouping messages per node over time. The plots presented in the figure correspond to the four protocols discussed earlier in the paper. As expected, the FISI+g and the SI+g protocols impose a need for node organisation, which in turn leads to message traffic at node grouping time. Compared to the protocols that do not reorganise node connections (FI+i and FI), FISI+g and SI+g load the network with messages to make nodes discover semantically similar friends and get organised in semantic groups. Because of this, the network initially presents a high message overhead indicating that nodes are still randomly connected and seek



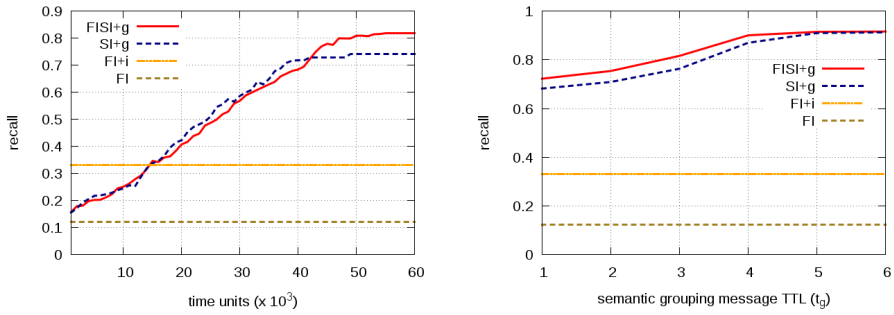
(a) (b)

Fig. 3: (a) Semantic grouping messages/node and (b) search messages/query over time

the network for semantic friends. Algorithm FISI+g imposes more traffic load at the network compared to SI+g, since nodes use both semantic and social connections to decide the initiation of semantic grouping. However in both cases, as nodes self-organise into semantic groups, message traffic is greatly reduced (over 55%). Notice also, that the FI+i protocol imposes the network with some minor message overhead (subject to the number of social friends of nodes) in order to update the nodes' FI with the interests of social friends.

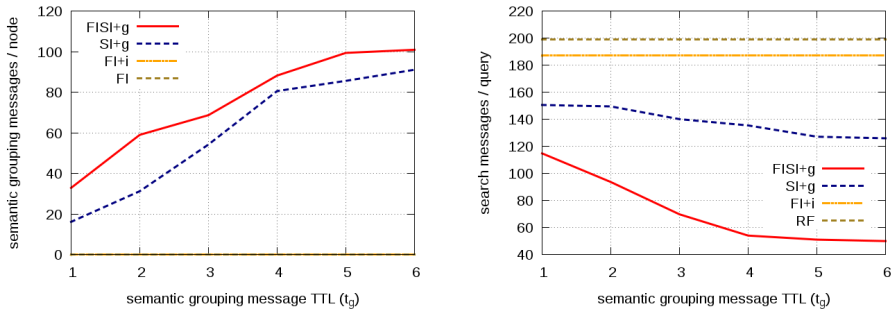
Figure 3(b) shows the number of messages per query over time for the four different protocols. Initially, a high number of search messages is needed to retrieve the available data relevant to a query in all cases. However, this message overhead is decreased when one of the FISI+g or SI+g protocols is used (more than 55%) as nodes continuously get organised into groups with similar interests, which proves helpful at query time. The FISI+g protocol imposes the network with less communication load compared to the SI+g protocol, indicating that using both social and semantic connections helps (i) at grouping time towards discovering semantically similar nodes and create cohesive semantic neighborhoods and (ii) at query time towards discovering the available data by exploring less node connections. Notice also, that the FI+i protocol shows better network performance compared to the FI protocol, indicating that using semantic information to forward the query to the social friends helps the query discover the available data.

Retrieval Effectiveness. Figure 4(a) illustrates the retrieval effectiveness of the network as a function of time for the different protocols. At the beginning, the values for recall are low (around 15%) for protocols FISI+g, SI+g, and FI, as nodes utilise only social friends (as no semantic grouping is yet initiated) to find the available data. After some time, when nodes start to organise into semantic groups with the FISI+g and SI+g protocols, we can observe the effect of the different strategies to retrieval effectiveness. The FI and FI+i protocols do not improve recall since network connections do not change over time, as they utilise only social friends. However, FI+i achieves 3 times better recall compared to FI, indicating that semantic information proves useful; even if search resorts only to social connections, using friends' descriptions to forward the query may improve performance. The FISI+g and SI+g protocols demonstrate improved retrieval



(a) (b)

Fig. 4: Recall (a) over time and (b) versus semantic grouping TTL (t_g)

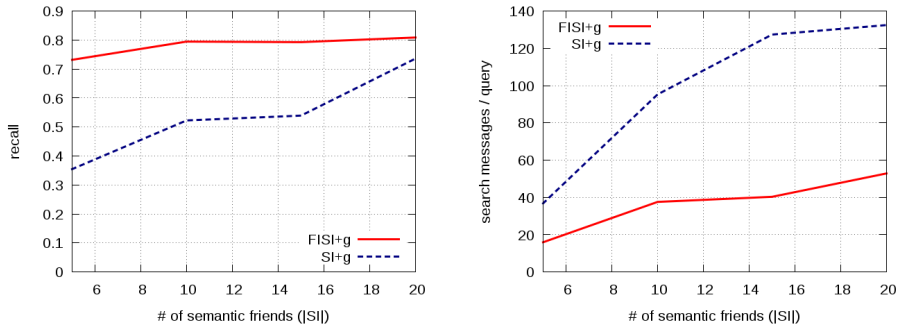


(a) (b)

Fig. 5: (a) Grouping messages/node and (b) search messages/query when varying the semantic grouping TTL (t_g)

performance overall, achieving up to 2.5 (resp. 8) times better recall than FI+i (resp. FI). Besides, the FISI+g protocol outperforms SI+g achieving up to 10% better recall, indicating that social friends provide added value in the semantic grouping process.

Varying t_g. Figure 4(b) investigates the retrieval effectiveness when social friends have been discovered (at time unit $50Kt$) for various values of t_g (i.e., the system parameter denoting how far the message will be sent in the network). As expected, when either of the FI+i and FI protocol is used, recall is not affected by the semantic grouping message TTL ($t_g = 0$ for both protocols), since these protocols do not reorganise nodes' semantic connections. On the other hand, the FISI+g and SI+g protocols improve their retrieval performance (FISI+g by 26% and SI+g by 30%) when greater values for t_g are used, since nodes explore a larger part of the network, thus increasing the probability to discover nodes with similar interests and populate SI with more relevant semantic friends. Then, at query time, the issued queries have higher probability to find the available data by exploring the nodes' semantic connections. Because of this, the retrieval performance of the FISI+g and SI+g protocols converges for higher values of t_g, indicating that organising the network in cohesive semantic neighborhoods may improve recall without resorting to social connections.



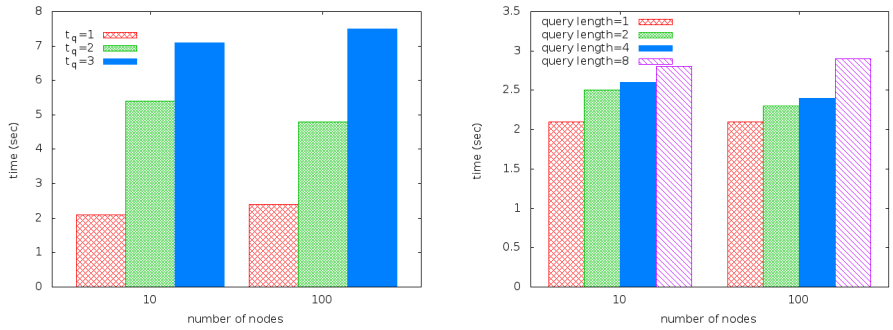
(a) (b)

Fig. 6: (a) Retrieval effectiveness and (b) search messages/query when varying the number of semantic friends ($|SI|$)

Figures 5(a) and 5(b) show the message overhead of the network (in terms of semantic grouping and search messages) when semantic friends have been discovered (at time unit $50Kt$) while varying the semantic grouping message TTL. As expected, communication load, when either the FI+i or the FI protocol is used, is not affected by t_g as nodes do not reorganise their semantic connections. In terms of grouping messages, the FISI+g and SI+g protocols cause more communication load in the network as t_g increases, since the message is sent further in the network to discover semantically similar friends. On the other hand, this further exploration of the network causes more cohesive semantic connections, which in turn improve (FISI+g by 2.3 times and SI+g by 1.2 times) communication load in terms of search messages. Recall that the FISI+g protocol uses more grouping messages as it uses both the semantic and the social connections to discover semantic friends, and needs less search messages to find the available data as more cohesive semantic groups have been eventually formed.

Varying $|SI|$. Figure 6(a) shows the retrieval effectiveness of the FISI+g and SI+g protocols when semantic friends have been discovered (at time unit $50Kt$) and how it is affected by the size of SI. Retrieval effectiveness is not demonstrated for the FI+i and FI protocols since they do not use semantic friends to find relevant data. The FISI+g protocol remains almost unaffected to changes in the number of semantic connections. This was expected since FISI+g uses both social and semantic connections to calculate SNS and forward the semantic grouping messages, explore a large part of the network, and achieve (an already) high recall. On the other hand, the SI+g protocol improves the retrieval performance (by 200%) when nodes store more semantic friends, since this protocol relies heavily on SI.

The number of search messages per query for the same setting is presented in Figure 6(b). Communication load at query time is affected by the number of semantic friends at each node for both the FISI+g and SI+g protocols; the network load increases when more semantic friends are stored, since queries are eventually broadcasted to the semantic neighbourhoods. Finally, the FISI+g protocol imposes the network with overall less communication load compared to the SI+g protocol as it manages to achieve a more cohesive semantic organisation.



(a) (b)

Fig. 7: Response time in the $\mathcal{DS}^4$ prototype when varying (a) query TTL and (b) query length

Other Parameters. The way other system parameters affect the performance of $\mathcal{DS}^4$ is independent of the protocol used. For greater values of θ (similarity threshold) (i) semantic grouping messages are increased as the semantic grouping process is initiated more often, (ii) search messages are decreased as less nodes are considered similar to a query, and (iii) recall in increased as less data matches a query. Additionally, recall and search messages are linear to the increase in t_q in all cases, while semantic grouping messages remain unaffected to t_q variations. Finally, recall and communication load are highly affected by increases in m (message fanout).

4.2 Prototype Measurements

Set-up and Measures. Apart from the simulation results aiming at showing the effectiveness and efficiency of the $\mathcal{DS}^4$ protocols, we also conducted a series of experiments on the prototype implementation. We implemented $\mathcal{DS}^4$ upon Microsoft .NET Framework v4.0 using C# and used the Lucene v2.9.1.2 library for performing the content clustering and interest identification of the nodes. Our experiments were executed on a local-area network that consists of 10 workstations connected by a Gigabit ethernet connection running up to 10 nodes per machine, that is 100 nodes in total.

To obtain a content-rich set-up for the experimentation on the prototype, the $\mathcal{DS}^4$ nodes were populated with a subset of the OHSUMED TREC[6] collection containing 30,000 medical articles belonging to 10 different categories. Each node was randomly assigned documents from different categories so as to support *multiple interests* per node, and the queries were issued from random peers in the network. The network was allowed a small bootstrapping period to perform node grouping and was subsequently used for social networking and content retrieval. The response time shown in the graphs is wall-clock time between the time a user issues a query and the time he receives the final result ranking.

Response Times. Figure 7(a) shows the response time for different values of the t_q parameter and for two network setups (consisting of 10 and 100 nodes).

[6] http://trec.nist.gov/data/t9_filtering.html

As it was expected, the response times between networks of different size do not present a significant variation, as the dominant factor for introducing a delay is the increase of the query TTL. Increasing the query TTL results in reaching nodes several hops away from the query originator and also, in receiving more relevant results that have to be collected, ranked, and presented to the user.

In Figure 7(b) we observe the response time when varying the query length for the two network setups. Queries posed in this setting are comprised of multi-word terms under the VSM model that were matched against both the local document store of the node and the results received from other network nodes. Query length refers to the number of terms contained in an issued query. We observe that $\mathcal{DS}^4$ is not very sensitive to changes in the query length.

The extensive experimental evaluation of the prototype system also considered the message costs, achieved recall, and grouping quality and the findings were in trace with the simulation results. Our recall values were as high as 78%, while the search cost in terms of bandwidth was kept low (at all times less than 2MB/query for all the search messages issued for a specific query). Our network analysis showed that the cost of local search is 40-50% of the response time, the cost of the network communication was 30-40% of the time, while 10-20% of the time was the collection, ranking, and display of results. The full scale evaluation of the prototype is omitted due to space reasons.

Storage and Memory Requirements. In terms of storage and memory requirements, $\mathcal{DS}^4$ is kept lightweight. Its memory trace when idle is around 8.5MB (8756KB) while its storage costs are largely dominated from the size of the shared data. Only one instance of the Lucene library (with a memory trace of 556KB on average) is loaded in main memory and serves all nodes hosted in this machine. In our evaluations, we managed to run as many as 30 nodes in a single workstation; however the experiments were executed with up to 10 nodes per machine for avoiding bottlenecks.

5 Summary of Results and Outlook

We have presented a social networking system that introduces semantic friendship between users to facilitate content search on top of the default social interactions. Our simulation experiments show that semantic friendship improves the observed recall of the system significantly, while search costs are reduced. Moreover, measurements from the implementation of the proposed system show that it achieves low response times, while supporting an expressive query language.

We plan to deploy $\mathcal{DS}^4$ in large-scale environments like PlanetLab and extend the proposed protocols to cover also subscriptions over content/tags with aggregation (e.g., notify me when a newly published document matches my continuous query and k of my friends have tagged it as interesting).

References

1. Ng, C.H., Sia, K.C., Chang, C.H.: Advanced Peer Clustering and Firework Query Model in the Peer-to-Peer Network. In: WWW. (2002)
2. Raftopoulou, P., Petrakis, E.: iCluster: a Self-Organising Overlay Network for P2P Information Retrieval. In: ECIR. (2008)

3. Voulgaris, S., van Steen, M., Iwanicki, K.: Proactive Gossip-based Management of Semantic Overlay Networks. CCPE **19**(17) (2007)
4. Loupasakis, A., Ntarmos, N., Triantafillou, P.: eXO: Decentralized Autonomous Scalable Social Networking. In: CIDR. (2011)
5. Narendula, R., Papaioannou, T., Aberer, K.: My3: A Highly-Available P2P-based Online Social Network. In: P2P. (2011)
6. Sharma, R., Datta, A.: SuperNova: Super-Peers based Architecture for Decentralized Online Social Networks. In: COMSNETS. (2012)
7. Mani, M., Nguyen, A.M., Crespi, N.: SCOPE: A Prototype for Spontaneous P2P Social Networking. In: PerCom Workshops. (2010)
8. Mitchell-Wong, J., Goh, S., Chhetri, M., Kowalczyk, R., Vo, Q.: A Framework for Open, Distributed and Self-Managed Social Platforms. In: VECN. (2008)
9. Abbas, S., Pouwelse, J., Epema, D., Sips, H.: A Gossip-Based Distributed Social Networking System. In: WETICE. (2009)
10. Graffi, K., Gross, C., Mukherjee, P., Kovacevic, A., Steinmetz, R.: LifeSocial.KOM: A P2P-Based Platform for Secure Online Social Networks. In: P2P. (2010)
11. Stuedi, P., Mohomed, I., Balakrishnan, M., Mao, Z., Ramasubramanian, V., Terry, D., Wobber, T.: Contrail: Enabling Decentralized Social Networks on Smartphones. In: Middleware. (2011)
12. de Spindler, A., Grossniklaus, M., Norrie, M.C.: Development Framework for Mobile Social Applications. In: CAiSE. (2009)
13. Kabir, M.A., Han, J., Yu, J., Colman, A.: SCIMS: A Social Context Information Management System for Socially-Aware Applications. In: CAiSE. (2012)
14. Upadhyaya, B., Choi, E.: Social Overlay: P2P Infrastructure for Social Networks. In: NCM. (2009)
15. Tegeler, F., Koll, D., Fu, X.: Gemstone: Empowering Decentralized Social Networking with High Data Availability. In: GLOBECOM. (2011)
16. Liu, G., Shen, H., Ward, L.: An Efficient and Trustworthy P2P and Social Network Integrated File Sharing System. In: P2P. (2012)
17. Han, L., Punceva, M., Nath, B., Muthukrishnan, S., Iftode, L.: SocialCDN: Caching Techniques for Distributed Social Networks. In: P2P. (2012)
18. Mega, G., Montresor, A., Picco, G.: Efficient Dissemination in Decentralized Social Networks. In: P2P. (2011)
19. Crespo, A., Garcia-Molina, H.: Routing Indices for Peer-to-Peer Systems. In: ICDCS. (2002)
20. Raftopoulou, P., Petrakis, E., Tryfonopoulos, C.: Rewiring Strategies for Semantic Overlay Networks. In: DPD. (2009)
21. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: A Privacy-Preserving Online Social Network Leveraging on Real-Life Trust. IEEE Communications Magazine (2009)
22. Gunther, F., Manulis, M., Strufe, T.: Key Management in Distributed Online Social Networks. In: WoWMoM. (2011)
23. Greschbach, B., Kreitz, G., Buchegger, S.: The Devil is in the Metadata - New Privacy Challenges in Decentralised Online Social Networks. In: PerCom Workshops. (2012)
24. Xue, M., Carminati, B., Ferrari, E.: P3D - Privacy-Preserving Path Discovery in Decentralized Online Social Networks. In: COMPSAC. (2011)
25. Backes, M., Maffei, M., Pecina, K.: A Security API for Distributed Social Networks. In: NDSS. (2011)
26. Buchegger, S., Schioberg, D., Vu, L.H., Datta, A.: PeerSoN: P2P Social Networking - Early Experiences and Insights. In: SNS. (2009)
27. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)