Supporting Social Networks with Agents

Federico Bergenti
Dipartimento di Matematica e Informatica
Universitá degli Studi di Parma
43124 Parma, Italy
e-mail: federico.bergenti@unipr.it

Agostino Poggi, Michele Tomaiuolo
Dipartimento di Ingegneria dell’Informazione
Universitá degli Studi di Parma
43124 Parma, Italy
e-mail: {agostino.poggi, michele.tomaiuolo}@unipr.it

Abstract—Multi-agent systems are among the most interesting areas in software research and they have been significantly contributing to the development of both theory and practice of complex distributed systems. Social networks are probably the systems that exhibit the highest degree of similarity with multi-agent systems because members of a social network interact as agents do in a multi-agent system. Multi-agent system may be the best solution to enrich and facilitate the interaction and the collaboration among the members of a social network. After an introduction on the relationships between multi-agent systems and social networks, the paper describes how multi-agent systems can support the interaction and the collaboration among the members of a social network through a set of services.

Keywords—Multi-agent-systems, social networks, agent-based services, information sharing.

I. INTRODUCTION

Multi-agent systems are among the most interesting areas in software research and they have been significantly contributing to the development of the theory and the practice of complex distributed systems (see, e.g., [1][2][3]). One of the main challenge of multi-agent systems was to become the primary means to support legacy systems interoperability and to facilitate the design and implementation of scalable distributed systems (see, e.g., [4][5]). However, in the last decades, service-oriented technologies have experienced an impressive progress and they seem to have good chances to compete with multi-agent systems as primary means for the development of scalable and interoperable distributed systems. However, service-oriented technologies cannot provide the autonomy that agents provide and they cannot offer the social and proactive capabilities of agents. As a result, the design and implementation of flexible adaptive distributed systems may be difficult if we go with service-oriented technologies only.

An integration of multi-agent systems with service-oriented technologies seems to be the most suitable solution for the implementation of scalable and interoperable distributed systems (see, e.g., [6][7][8]). However, in some application areas, multi-agent systems can be considered a suitable means for directly providing services. Social networks are one of such areas. Indeed, social networks and multi-agent systems exhibit deep similarities and the members of a social network often interact as agents do in a multi-agent system [9]. Hence, it is possible to envisage in the long term networks of humans and agents, where agents provide services aimed at improving the exchange of information and supporting the collaboration among members.

This paper has the goal of showing how multi-agent systems can be a suitable means for enhancing social networks with intelligent services. Next section presents the relationships between multi-agent systems and social networks. Section 3 shows a prototype of a peer-to-peer social network that take advantage of agents for providing services to its members. Section 4 discusses related work. Finally, Section 5 concludes with a discussion about the main advantages of using multi-agent systems in social network platforms and introduces the directions of our future work.

II. SOCIAL NETWORKS AND MULTI-AGENT SYSTEMS

Social networks are typically described as finite sets of actors and relations defined on them [10]. In this context, an actor is essentially any social entity, such as an individual, a corporate, or a collective social unit; and a relationship can be any kind of social tie that establishes a link between a pair of actors.

Nowadays, the most widely known social networks are Web platforms, often called online social networks, where users not only put or read content, but they are also linked with relationships. The diffusion of online social networks is opening new scenarios for envisaging novel kinds of applications, either to support new social networking activities, or to exploit established relationships among users and use them to offer higher-level services. One promising direction is the use of a social network as a sort of grid platform for accessing and composing services, to build rich virtual computing environment [11]. However, such scenarios would require highly adaptive services in order to react to potentially differentiated and dynamic environments.

The situation becomes even more interesting and promising when location-based communities and services are taken into consideration. In all such cases, software agents are a natural fit for mediating access to local software- or hardware-based services, including access to data, sensors, monitors, printers and various kinds of actuators. Given their ability to negotiate and plan in a dynamic social context, software agents are also good for composing locally available services and resources, following existing trust relationships with other persons and agents located in the user’s proximity area. New trust relationships can also be created, on the basis of reputation and
mutual acknowledgement, through the incremental and controlled exchange of profile data.

Especially in the case of completely distributed or federated social networking platforms, multi-agent systems can play an important role. Indeed, one of the very specific features of multi-agent systems is the sociality of agents, i.e., their ability to communicate in a semantic way and to develop trust relationships among them. Moreover, agents can (i) express their communication acts by means of acknowledged standards for interoperability among diverse systems, like FIPA; (ii) and exchange messages directly, in a peer-to-peer way. Therefore, it is not surprising that these two technologies are often applied together for developing advanced social platforms. In particular, multi-agent systems have been used as (i) an underlying layer, or middleware, for developing social networking platforms; and (ii) a technology to increase the autonomous and intelligent behavior of existing systems.

For the first type of solutions, many of the distinguishing features of multi-agent systems can be fully exploited. Multi-agent systems provide semantic communication among agents, which is handy for expressing all the different actions that users can perform in a social platform. The different types of messages can be understood according to their meaning and applied according to existing trust relations among the users and their respective agents. In addition, complex negotiation protocols help creating acknowledgements and trust among users, in an automatic or assisted way, without exposing sensitive data. Mobility can also be useful for moving the computation closer to data, if massive analysis is needed, but it can also be handy for adding functionality to a node of a distributed social platform or to a user’s client application.

In the second case, agents are mainly used because of their proactive and reactive behaviors that can provide recommendations of both users and content, and that can enable the personalization of results. Reactive abilities are particularly important in a social networking environment where events happen continuously and users can be easily distracted by the huge information flow, which is associated with highly connected social networks. Sensing the environment and executing automatic tasks can reduce this overload significantly. Goal-oriented behaviors, on the other hand, can support users in prosecuting their long term objectives about friend and content discovery, i.e., to discover known persons registered in the network, to make new acquaintances with users with common interests, to find interesting content hidden in less relevant data or from new sources.

III. A PROTOTYPE OF A MULTI-AGENT SOCIAL NETWORK

As an example of the use of the described services we present a distributed social networking system, called Blogracy, whose goal is to provide adaptive and composite services on top of its core features. At the lower level, Blogracy uses widespread and stable peer-to-peer technologies, such as distributed hash tables and the BitTorrent protocol, for coping with the intrinsic defects of centralized architectures and to become the basis of solid distributed social networking platforms. At the higher level, it takes advantage of multi-agent systems for simplifying the implementation of social network services in a decentralized setting.

The architecture of the application is modular and composed of two basic components: (i) an underlying module for basic file sharing and DHT operations, built as an extension of existing implementations, and (ii) an OpenSocial container, i.e., a module providing the services of the social platform to the local user through a Web interface. Additionally, the system supports autonomous agents for providing (i) recommendations of both users and content, (ii) personalization of results, and (iii) trust negotiation mechanisms.

The Blogracy system itself relies only on users’ nodes for its operation and users need to perform background tasks on their own, in a distributed way. A layer of autonomous agents takes charge of assisting the user in finding new interesting content and connections, and in pushing the local user’s activities to followers.

The multi-agent architecture of Blogracy is sketched in Figure 1 where a Personal Assistant (PA) monitors the local user’s actions in the platform and it learns the user’s profile, beyond information provided explicitly. The PA receives the user’s queries, forwards them to the available Information Finders (IF) and presents the results to the user. Moreover, a PA provides the local user with recommendations about possibly interesting content and connections available in the network. Another task performed by the PA is the personalization of results. Indeed, as a social network becomes larger and more richly interconnected, users unavoidably face some form of information overflow. A PA, using a user’s profile, can arrange presented data in a way that highlight the most interesting pieces of information.

An IF is an agent that searches information on the repository contained in the node where it lives, through an automatic TF-IDF indexing algorithm and explicit hashtags associated with local posts. It provides such information both to its user and to other trusted users. An IF receives users’ queries, finds appropriate results and filters them by using its user’s access policies. An Information Pusher (IP) is an agent that monitors the changes in the local repository and that pushes new information to the PA of interested subscribers that are currently connected. An IP forwards content produced by the local user and by her/his remote acquaintances to other contacts, according to privacy preserving policies and to recent queries made by other users.

Over the OpenSocial container, Blogracy can also provide functionalities for pervasive online social networking, specifically for creating locality and proximity groups. In this case, the system has to rely on highly adaptive services both to sustain the basic operations of the location-based social networking and to provide advanced functionalities. For this purpose, each node of the social network has to host multiple agents, with different levels of agency. Some of the more important agents are (i) the Neighborhood Manager (NM) agent, which cooperates with lower level agents to discover the users in its neighborhood; (ii) the Trust Negotiator (TN) agent, which is involved in the decisions regarding privacy and access rules; and (iii) the OpenSocial agent, which provides a bridge towards the underlying Blogracy modules.
A user may own multiple nodes (e.g., an instance on the smartphone and an instance on his home computer) and, since the actual location of the user is important for our application, the nodes in the different devices negotiate which one should be considered active (i.e., which one determines the user location). In fact, the nodes can either determine the device that registered an explicit user action or they can ask the user to select the device he/she is currently using.

Apart from the personal circles defined by each user, we also have two additional kinds of groups: proximity groups and location groups. Proximity groups are centered on each member of the social networking system and represent physical closeness to such a member. Proximity groups are extremely fluid in the sense that users can physically move and consequently the set of users belonging to a proximity group varies over time. Each user configures the hysteresis, or stickyness, of his proximity group, i.e., how long other users are considered part of it after they are no longer physically close to him/her. Although a proximity group may be entirely public, for privacy reasons it is safer to consider only proximity groups that are subset of other groups (or of the set union of all groups, i.e., only “friends” are part of a proximity group). The NM agent informs the OpenSocial agent when users enter and leave the proximity group and the latter notifies the OpenSocial container about it.

On the other hand, a location group is associated with the users in the proximity of a given location (e.g., a classroom or a museum room) and it has a host (i.e., a node) that both identifies and supports the group. Moreover, a location group is associated with a location profile maintained either on the central server or on its host. In fact, a location, although logically different from a regular user, works in the same way and a location group is essentially a proximity group for the location.

The availability of a generic TN agent is also important since users joining a proximity group or a location group are not necessarily connected a priori in the social network, and they may need to acknowledge their profile attributes before practical social interaction. Such a negotiation requires the controlled exchange of credentials and of policies, without disclosing unnecessary sensible information, yet establishing trust if possible [12]. In [13], a generic library supporting zero-knowledge proof for attribute verification is presented. The same mechanisms can also facilitate the creation of trust in social networks.

Agents present different degrees of autonomy and intelligence. For example, lower level agents are mostly reactive; e.g., they inform the NM agent when a new node is discovered. The NM agent itself has some degrees of autonomy and intelligence, and it has the following duties:

- It aggregates information from the agents that discover new peers;
- It informs the OpenSocial of the state of neighborhood;
- It tries to present a consistent view, merging data from the different sources; and
- It configures the discovering agents according to high-level criteria, such as battery consumption and hardware availability.

The OpenSocial agent is basically a gateway to the OpenSocial container and it translates the other agents’ requests for the OpenSocial container. A TN agent is a true agent that performs potentially complex negotiations on its user’s behalf and, depending on the configuration, it may work in full autonomy.

IV. RELATED WORK

In order to understand the relationship between multi-agent systems and social networks it is important to understand the intrinsic computational properties of social networks themselves. Indeed, without such properties, social networks would not be an effective structure for multi-agent systems and multi-agent systems could not be used to build software systems supporting social networking. The first insights on such properties came from Milgram’s experiment that led to the investigation of the so-called small world phenomenon [14]. In Milgram’s experiment, a group of randomly chosen people received the name and address of another randomly chosen person living in a distant city. Then, people were asked to route a mail message toward the target person chosen only among their friends or close acquaintances. The experiment pointed out that: (i) people are connected through very short chains of acquaintances, with a 5-6 links length, in average; and (ii) people is able to route the messages to the target person using local information and performing local actions.

A result of the Milgram’s experiment is that the behavior of people was similar to that of rational autonomous agents. In fact, every person chooses his/her successor in his/her list of acquaintances considering elements like geographical proximity or profession similarity, which is essentially using only local and elementary information to pursue a global complex goal, with no need to use their humanity. From our point of view, this is a particularly relevant conclusion, since it points to the emergence of a global behavior from local strategies, a feature that is one of the key properties of multi-agent systems.

More recently, the studies on the small world problem led to two computationally-based approaches to search for people within social networks (a comprehensive review of different
algorithms and their performance is presented in [15]). The Milgram’s original experiment led to a machine-based approach consisting in the problem of looking for a remote agent given its unique identifier. A successive approach deals with finding a specific agent who matches some criterion, such as having a given capability or expertise. This is quite similar to the problem of navigating one’s social network in search for someone with a given expertise or for an answer to a specific question. In an enterprise setting, this is the problem of looking inside the organization for someone able to solve a specific problem or able to answer to a specific question. When solved with agent-based techniques, this problem resembles the “collaborative filtering” one and is usually termed as “expert finding”, and authors use such definitions interchangeably.

The expert finding problem is similar to Milgram’s original problem in that the social network of each node is the search space in which the request is processed. It should be emphasized that both problems strongly rely on the local search ability and the occurrence of the small world phenomenon, i.e., on the fact that two random individuals are preferably mostly connected by short chains of acquaintanceships. If social networks were not searchable it would be impossible to efficiently find a person matching some criteria unless personally known and, then, the Milgram’s experiment would have failed. On the other hand, if the chains were very long, the search would be not feasible.

A pioneering research on this subject was done in [16][17]. These papers describe ReferralWeb, an agent based interactive system for reconstructing, visualizing, and searching social networks on the Web whose main focus is selecting an expert of a given field in one’s (extended) social network.

In ReferralWeb a social network is modelled by a graph where the nodes represent individuals and an edge between nodes indicates that a direct relationship between the individuals has been discovered. For ReferralWeb a direct relationship is implied when the names are in close proximity in any document publicly available on the Web, e.g., home pages, co-authorship in published papers or organization charts in institutional websites. ReferralWeb does not require its users to fill a user profile describing their skills.

The constructed network is then used to guide the search for people or documents in response to user queries. A person can: (i) ask to find the chain between himself/herself and a named individual; (ii) search for an expert in a given topic providing a maximum social radius (the number of links in the chain connecting the person performing the query with the expert); and (iii) request a list of documents written by people close to a given expert.

The key idea of ReferralWeb is to use the social network to make more focused and effective searches. It is not meant to be a tool to create social networks, i.e., to help people socializing. ReferralWeb also emphasizes the importance of the referral chains themselves as means to build trust on the selected experts.

Yenta is a matchmaking system that helps people with similar interests to get in touch [18]. Yenta agents do not query the Web; instead, they scan user’s e-mails, Usenet posts and (possibly) documents in order to discover their users’ interests and hobbies. The idea is that many potentially interesting people do not write publicly and so they become invisible to tools relying on public data. Collected data are then used to introduce users’ to each other. Considering that in the nineties Web communities were built around the idea of common interests rather than on personal acquaintance, the system was a truly distributed social networking system, at least for the time.

Community Organizer is a system where agents help the users by gathering and exchanging information, visualizing contexts, and recommending or assisting their users in making choices [19]. Each user has a personal agent and a set of additional community agents have the function of providing shared information, knowledge, or contexts within the community and act as mediators for informal communications between people. In particular, each personal agent acquires the user profile and visualizes potential communities around the user. The community agent collects the user profiles and maintains information on potential communities. Upon a request from a personal agent, the community agent first computes potential communities around the owner of the personal agent, and then sends the necessary data (users in the potential communities and their relevance) to the personal agent.

Shine (SHared INternet Environment) is a fully peer-to-peer framework for network community support. The system has been implemented and a presented in [20]. The framework provides design guidelines and enables different applications to share program components and to cooperate, and it features a peer-to-peer architecture through which personal agents can flexibly form communities where users can exchange information with peer agents. Essentially, Shine is a middleware for collaborative workspaces especially tailored to implement various collaborative workspaces.

Shine provides a personal agent to each and every single user and three core modules compose each agent: the Person database, the plan execution module and the communication module. In addition, one or more applications are installed in each agent. Such applications provide their services to the user by means of functionalities of the core modules via a dedicated API.

The Person database of Shine holds data on people and on personal agents. The data include information on the agent and on the user whom the agent is associated with, as well as other agents and people known to the agent. An agent holds the data required to form a community that is suitable to the user in the Person database and it exchanges data among other agents when necessary. In the Shine architecture, the user and his/her personal agent correspond in a one-to-one manner. Therefore, in the Person database, data on both a user and his/her personal agent are stored without distinguishing between them.

As socialware is designed to support communities, Shine’s authors added the concept of person set. Each community is represented in the Person database as a person set and the framework provides operations for dealing with such sets, e.g., functions to broadcast messages to the members of a community. In this way Shine agents can flexibly determine
the range of broadcasting by regarding a person set as the destination list.

In Shine a peer-to-peer network is formed directly connecting the communication modules of groups of agents. The function of such modules is simply to exchange messages with each other. Given the fact that the agents reside in a ubiquitous computation environment, the module is layered so that only the lower layer depends on the environmental details.

Agents in Shine are goal-driven through plans: a plan is description of agent action rules. Multiple plans are executed concurrently in the plan execution module of each agent. Some plans are prepared to perform services of applications while other plans are provided by Shine to do fundamental or common tasks. A plan acts in response to external events, e.g., receiving a message from another agent, a user input or a modifications in the Person database.

MARS is a multi-agent referral system that finds experts on the basis of personal agents able to learn the user’s preferences and interests, and able to build an expertise model of the other users on the basis of their responses [21]. Each user is assigned an agent who: (i) learns the user’s preferences and interests and (ii) maintains a view of its user’s acquaintances, that are used to prioritize incoming queries, possibly issuing referrals when other users might be more suitable to answer a given query. Each agent first rates, according to the user’s feedback, those agents that provided an answer and those agents that referred to them, and then, it modifies its neighbors accordingly. Consequently, the referral system evolves to reflect the changes in the social network.

A response to a query specifying what information is being sought, if given, may consist of an answer or a referral, depending on the query and on the expertise of the answering agent. If an agent is reasonably confident that its expertise matches the query, it directly answers; otherwise, it yields referrals to other supposed expert agents.

Each agent maintains models of its acquaintances. An agent sends its query initially only to some of its neighbors, that are the individuals with the closest acquaintances. The agent who receives a referral may pursue it even if the referred party is not already an acquaintance; good acquaintances are going to be promoted to neighbors on an intuitive basis. When new neighbors are considered (included) some of previous ones will be discarded, since the number of neighbors is bounded. The authors of MARS decided that reputation should increase slowly, but fall quickly and that rewards and penalties are greater for agents nearer to the answering agent. This implies that a bad decision results in bad reputation, but if agents just started a chain of referrals leading to a bad agent, then the penalty is modest.

The expertise model is captured through a classical vector space model [22]. Term vectors are used to express both the profile of the user and the acquaintance model for each of its acquaintances. Since a term vector also models the required expertise, the cosine of the angle between the user vectors with the subject vector yields the competence of a user in a given subject. Intuitively, when there are two agents with expertise in the same direction, the one with the greater expertise is more desirable.

Each agent learns its user’s profile and its acquaintance models based on an evaluation of the received answers as well as on the referrals that led to them. A referral graph, which is local to each agent, encodes how the computation spreads, as a query originates from an agent and referrals or answers are sent back to this agent.

MAgNet is a middleware based on software agent technology that enables social networking services for users in the mobile network domain [23]. The first experimentation showed that its services can enable mobile users to define and customize their social relations with other users, as well as use the created relations to plan and manage group events. MAgNet is built taking advantage of both JADE [24] and FOAF [25].

SNIS is a multi-agent system where agents utilize the connections of a user in the social network to facilitate the search for items of interest [26]. In particular, each agent is associated with a user and it observes the user’s activities and, in particular, the ratings and comments provided by the user to items retrieved from the social network. SNIS has been experimented in the Flickr domain [27]: the system scans photos posted by all of the user’s contacts and gathers statistics about their categories and user comments (which represent user interest) and such information is used to facilitate the search for items of interest.

V. CONCLUSIONS

Multi-agent systems have the potential to become one of the most important means to develop intelligent services for social networks that can automate the execution of tasks that their members currently perform manually. The coordination, knowledge management, and learning capabilities provided by multi-agent systems can already be used to help the members of a social network by easing the execution of complex tasks and by acting on their behalf to perform tasks in collaboration with other members of the social network.

In particular, the provision of intelligent services will be simplified by the integration of multi-agent systems with semantic Web technologies, as well as with other related technologies for providing semantic services. Given that the composition of services is considered one of the most promising areas of application of agents, the improvement on the composition techniques will greatly depend on the quality of the semantic description of services available to the agents.

Therefore, besides extending and improving the prototype presented in this paper, and experiment it with different types of users, our work will be dedicated (i) to enhance the definition of the content exchanged by agents with semantic Web technologies [28], and (ii) to support the semi-automatic annotation of the information published by the members of the social network [29]. Moreover, our work will be also dedicated to introduce new social network services. In particular, we plan to develop a set of services for the collaborative work [30][31].
REFERENCES


