A DHT-based Multi-Agent System for Semantic Information Sharing

Agostino Poggi and Michele Tomaiuolo

Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma
Viale U. P. Usberti 181/A, 43100 Parma, Italy

{agostino.poggi, michele.tomaiuolo}@unipr.it

Abstract. This paper presents AOIS, a multi-agent system that supports the sharing of information among a dynamic community of users connected through the Internet thanks to the use of a well-known DHT-based peer-to-peer platform: BitTorrent. In respect to Web search engines, this system enhances the search through domain ontologies, avoids the burden of publishing the information on the Web and guaranties a controlled and dynamic access to the information. The use of agent technologies has made the realization of three of the main features of the system straightforward: i) filtering of information coming from different users, on the basis of the previous experience of the local user, ii) pushing of some new information that can be of interest for a user, and iii) delegation of access capabilities, on the basis of a reputation network, built by the agents of the system on the community of its users. The use of BitTorrent will allow us to offer the AOIS systems to the hundreds of millions of users that already share documents though the BitTorrent platform.

1 Introduction

Nowadays, the Web is the most powerful means for getting information about any kind of topic. However, the Web assigns a passive role to the large part of its users. Therefore, when Internet must be used to allow the active sharing of information among the members of a community, the use of a peer-to-peer solution may provide several advantages [22].

This paper presents a system, called AOIS (Agents and Ontology based Information Sharing), trying to couple the features of peer-to-peer and multi-agent systems. The next section introduces related work on multi-agent systems for information retrieval. Section three describes the main features and the behaviour of the AOIS system. Section four describes how this system has been designed and implemented by using some well-known technologies and software tools. Section five briefly discusses the testing of the system. Finally, section six reports some concluding remarks, gives a short introduction about the first experimentation of the system and presents our future research directions.

2 Related Work

Multi-agent systems have always been considered one of the most important ingredients for the development of distributed information management systems and for proving the different services needed in such systems [13]. In particular, several interesting works demonstrate: how multi-agent systems are a suitable means for the management of information in a community of users, how they can take advantage of a peer-to-peer network for performing a distributed search of information and how the use of ontologies and user profile allows an improvement of the quality of their work.

DIAMS is a multi-agent system that provides services for users to access, manage, share and learn information collaboratively on the Web [5]. DIAMS can be considered one of the most complete multi-agent infrastructures for the management and retrieval of information in a community of users. In fact, it supports the searching and retrieval of the information from local and/or remote repositories and it encourages the collaboration among its users by supporting the sharing and exchange of information among them.

ACP2P (Agent Community based Peer-to-Peer) is an information retrieval system that uses agent communities to manage and search information of interest to users [18]. In the ACP2P system, an agent works as a delegate of its user and searches for information that the user wants by coupling the typical propagation of the query on the peer-to-peer infrastructure. It supports the community with the identification of the agents that may have such information through the use of the experience gained in its previous interactions. The experimental results of the use of the ACP2P system demonstrated that the use of the agent experience provides a higher accuracy in retrieving information.

CinemaScreen is a recommender system, which combines collaborative filtering and content-based filtering [26]. The first method requires matching a user with other users with similar behaviours and interests. The second method requires matching the items on the basis of their characteristics (CinemaScreen, in particular, deals with genres, actors, directors etc.). While both mechanisms exhibit weaknesses in particular situations, their combination allows better performances since the very beginning of the system activity. The system is built in the form of an intelligent agent, but apparently it is modelled as an essentially centralized system.

On the other hand, pSearch is a decentralized information retrieval system [30]. In this system, which is P2P but non-flooding, document indices are distributed through the network according to a classification of document content. The document semantics is generated and managed through a technique called Latent Semantic Indexing [34]. The resulting system is proven to be efficient in the number of nodes to contact to perform a search.

In [11] a social resource sharing system is presented. In this case, it uses a form of lightweight knowledge representation, called folksonomy, is used. In fact, the conceptual structures of ‘taxonomy’ are created bottom-up by ‘folks’, thus creating an emergent semantics, instead of using the more rigid approach of the traditional Semantic Web

Sanchez and his colleague proposed an integrated agent-based ontology-driven multi-agent system that automatically retrieves Web pages that contain data relevant to the main concepts of a specific domain [27]. The multi-agent system is based on

the use a Web-based ontology learning method able to automatically build ontologies for any domain [20], and then on a set of agents that use such ontologies for the retrieval, filtering and classification of information.

3 AOIS

AOIS (Agents and Ontology based Information Sharing) is a multi-agent system composed of different agent platforms connected through the internet that supports the sharing of information among a community of users. Each agent platform acts as a "peer" of the system and is based on five agents: a personal assistant, a repository manager, an information finder, an information pusher, and a directory facilitator.

A personal assistant (PA) is an agent that allows the interaction between the AOIS system and the user. This agent receives the user's queries, forwards them to the available information finders and presents the results to the user. Moreover, a PA allows the user to be informed about the new information that other users made available and that may be of her/his interest. Finally, a PA maintains the information that a user may share allowing her/him to add and remove information in a repository where information is partitioned on the basis of the topics of interest of the user.

A repository manager (RM) is an agent that builds and maintains both the indexes for searching information and the ontologies describing the topics of interest of its user. Each time the user adds or removes some information, the OM updates the corresponding index and ontology.

An information finder (IF) is an agent that searches information on the repository contained into the computer where it lives and provides this information both to its user and to other users of the AOIS system. An IF receives users' queries, finds appropriate results, on the basis of both the queries and the topic ontology, and filters them on the basis of its user's policies (e.g., the results from non-public folders are not sent to other users).

An information pusher (IP) is an agent that monitors the changes in the local repository and pushes the new information to the PA of the users whose previous queries match such information.

Finally, the directory facilitator (DF) is responsible to register the agent platform in the AOIS network. The DF is also responsible to inform the agents of its platform about the address of the agents that live in the other platforms available on the AOIS network (e.g., a PA can ask about the address of the active IF agents).

The exchange of information among the users of an AOIS system is driven by the creation of both a search index and an ontology for each topic. The search index allows the ranking of information on the basis of the terms contained in a query. The ontology allows to identify additional information on the basis of the terms contained in the ontology that have some semantic relationships (i.e., synonyms, hyponyms, hypernyms, meronyms and holonyms) with the terms contained in the query. Both the search index and the ontology are automatically built by the RM on the basis of the information stored in the topic repository.

The following subsections describe the behaviour of the AOIS system through six practical scenarios and introduce a detailed description of how members can be added

to an AOIS community and how security and privacy are managed to show how AOIS copes with the problems of working in a real open community.

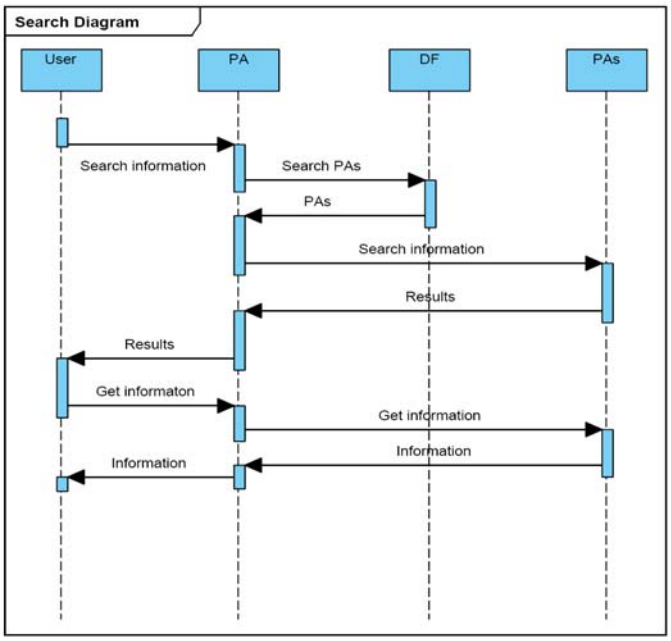

Fig. 1. Searching scenario UML sequence diagram.

3.1 Information Searching Scenario

The first scenario describes how a user can take advantage of the agents of the AOIS system for searching information. This scenario can be divided in the following five phases (see also figure 1):

1) a user requests her/his PA to search information on the basis of a topic, a set of keywords. The PA asks the DF for the addresses of available IF agents and sends the topic and the keywords to such agents (information search request phase);
2) each IF checks if the querying user has the access to at least a part of the information about the topic stored in the corresponding topic repository, and, if it happens, searches the information on the basis of both the received query and a set of additional queries obtained by replacing each keyword of the received query with the possible substitutes contained in the topic ontology. Moreover, the IF sends the received query to the local IP and RM agents: the IP adds the query to the profile of the corresponding remote user and the RM add the query keywords to the list of the keywords for updating the repository ontology (information search execution phase),
3) each IF filters the searching results on the basis of the querying user access permissions of the querying user sends the filtered list of results to the querying PA (information filtering and sending phase);

4) the querying PA orders the various results as soon as it receives them, omitting duplicate results and presents them to its user (information presentation phase);
5) after the examination of the results list, the user can ask her/his PA for retrieving the information corresponding to an element of the list. Therefore, the PA forwards the request to the appropriate IF, waits for its answer and presents the information to the user (information retrieval phase).

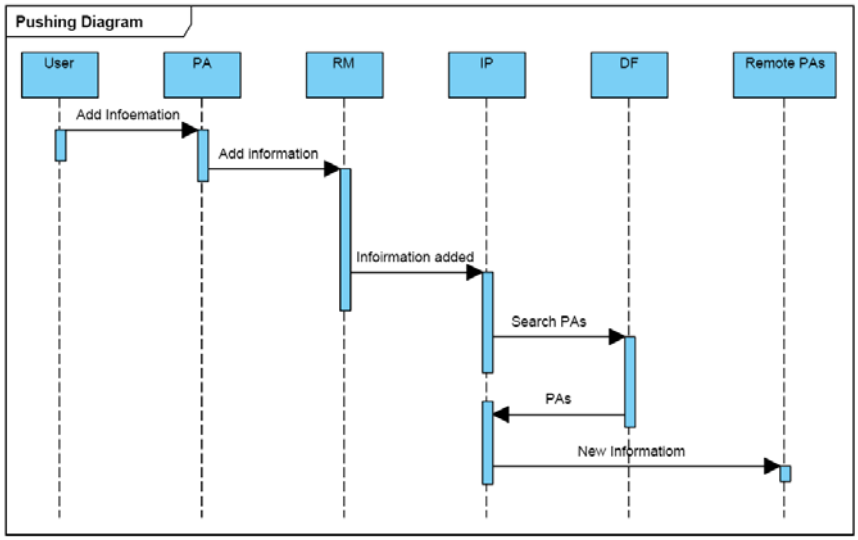

Fig. 2. Pushing scenario UML sequence diagram.

3.2 Information Pushing Scenario

The second scenario illustrates how a user can take advantage of the AOIS system to be aware about the availability of new information of her/his interest. This scenario can be divided in the following five phases (see also figure 2):

1) a user requests her/his PA to add some information in a specific topic repository and the PA propagates the request to the RM (information addition request phase);
2) the RM adds the information in the repository, updates the indexes for the searching of information and then informs the IP about the new information (information addition phase);
3) the IP checks if such new information satisfy some queries maintained in the profiles of the remote users and, when happens, then the IP either sends such information to the PA of the remote user (if the corresponding AOIS platforms are alive), or maintains such an information until such a platform becomes alive again. (information pushing phase);
4) Of course, when a PA receives a list of pushing results, it presents them to its user (information presentation phase);
5) after the examination of the results list, the user can ask her/his PA for retrieving the information corresponding to an element of the list. Therefore, the PA

forwards the request to the appropriate IF, waits for its answer and presents the information to the user (information retrieval phase).

3.3 Repository Creation Scenario

The third scenario illustrates how a user can take advantage of the AOIS system for the creation of a repository for maintaining the information about a specific topic. This scenario can be divided in the following four phases:

1) a user requests her/his PA to create a repository for a specific topic indicating the set of terms (named ontology top terms) that describe such a topic and listing a set of information to store in the repository. The PA propagates the request to the RM (repository creation request phase);
2) the RM creates the repository, builds the topic ontology finding the semantic relationships (i.e., synonyms, hyponyms, hypernyms, meronyms and holonyms) among the top terms, adds the set of information, builds the search indexes and then informs the PA about the creation (repository creation phase);
3) the PA asks its user if she/he wants to populate the ontology with terms extracted from the information stored in the repository and the maximum permitted semantic distance between a new and a top term (ontology population request phase);
4) If the user enables the operation, the PA asks the RM to analyse the repository search indexes for finding the terms that are in direct or indirect relations with the top terms of the ontology. Of course, in the case of indirect relationship, each new term is only added if it satisfies the maximum semantic distance constraint (ontology population phase).

3.4 Repository Updating Scenarios

The forth and fifth scenarios illustrate how a user can take advantage of the AOIS system for updating both the search indexes of a repository and the related ontology.

The forth scenario is driven by the user that wants to add some information to a repository. This scenario can be divided in the following four phases:

1) the user requests her/his PA to add some information to a repository and the PA propagates the request to the RM (information addition request phase);
2) the RM adds the information in the repository, updates the indexes for the searching of information and then informs the PA about the new information (information addition phase);
3) the PA asks its user if she/he wants to populate the ontology with terms extracted from the new information added in the repository and the maximum permitted semantic distance between a new and a top term (ontology updating request phase);
4) If the user enables the operation, the PA asks the RM to analyse the repository search indexes for finding the terms that are in direct or indirect relations with the top terms of the ontology. Of course, in the case of indirect relationship, each

new term is only added if it satisfies the maximum semantic distance constraint (ontology population phase).

The fifth scenario starts when the user logs to the system and her/his RM has some new keywords coming from last remote users queries. This scenario can be divided in the following two phases:

1) when the user logs the system, the PA gets the new keywords from the RM and asks its user if she/he likes to add some of them as top terms of the ontology (keywords selection request phase);
2) if the user selected some of the keywords to populate the ontology, the PA asks the RM to update the ontology finding the semantic relationships (i.e., synonyms, hyponyms, hypernyms, meronyms and holonyms) among the new and the old top terms (ontology updating phase);
3) Then the PA asks her/his user if she/he wants to populate the ontology with terms extracted from the information stored in the repository and the maximum permitted semantic distance between a new and a top term (ontology population request phase);
4) If the user enables the operation, the PA asks the RM to analyse the repository search index for finding the terms that are in direct or indirect relations with such new top terms of the ontology. Of course, in the case of indirect relationship, each term is only added if it satisfies the maximum semantic distance constraint (ontology population phase).

3.5 Community Management Scenario

The fifth scenario illustrates how an AOIS user can connect to an existing community and how a community can deal with new join requests.

This scenario can be divided in the following four phases:

1) the user has to be introduced into the community by a member who plays the role of introducer;
2) the new member is acknowledged by the introducer by receiving a proper token, which testifies the acceptance into the community;
3) once the new member has been acknowledged by the introducer, the latter also sends a list of all other members of the community, with their basic profile and contact information, to the new member;
4) the new member registers all members into the local list of contacts, and updates the information of the DF of her/his AOIS platform on the basis of the profiles received by the other members of the community;
5) the new member then sends a join request to all other members, together with all relevant credentials, including the token received from the introducer;
6) the other members of the community add the new user's profile to the local list of contacts and adds his services into the local DF.

3.6 Security and Privacy Management

The information stored into the different repositories of a AOIS network is not accessible to all the users of the system in the same way. In fact, it's important to avoid the access to private documents and personal files, but also to files reserved to a restricted group of users (e.g.: the participants of a project). The AOIS system takes care of users' privacy allowing the access to the information on the basis of the identity, the roles and the attributes of the querying user, as defined into a local knowledge base of trusted users. In this case, it is the user that defines who and in which way can access to her/his information, but. Moreover, the user can also allow grant the access to unknown users by enabling a certificate based delegation, built on a network of the users registered into the AOIS community. In this sense, the system completely adheres to the principles of trust management. For instance, if the user U_i enables the delegation and grants to the user U_j the access to its repository with capabilities C_0 to the user U_j, and U_j in turn grants to the user U_k the access to its the repository with the same capabilities C_0 to the user U_k, then U_k can access U_i's repository with the same capabilities of U_j.

The definition of roles and attributes is made in a local namespace, and the whole system is, in this regard, completely distributed. Local names are distinguished by prefixing them with the principal defining them, i.e. a hash of the public key associated with the local runtime. Links among different local namespace, again, can be explicitly defined by issuing appropriate certificates. In this sense, local names are the distributed counterpart of roles in role based access control frameworks [14]. This model is centred on a set of roles. Each role can be granted a set of permissions, and each user can be assigned to one or more roles. A many to many relationship binds principals and the roles they're assigned to. In the same way, a many to many relationship binds permissions and the roles they're granted to, thus creating a level of indirection between a principal and his access rights. Like roles, local names can be used as a level of indirection between principals and permissions. Both local names and roles represent at the same time a set of principals and a set of permissions granted to those principals. But, while roles are usually defined in a centralized fashion by a system administrator, local names, instead, are fully decentralized. This way, they better scale to Internet-wide, peer-to-peer applications, without loosening in any way the principles of trust management.

In AOIS, the user can not only provide the permission to access his own files, but can also assign the permission to upload a new version of one or more existing files. In this case the PA informs his/her user about the updated files the first time he/she logs in. This functionality, together with the trust delegation, can be useful for the members of a workgroup involved in common projects or activities

4 Implementation

The AOIS system has been designed and implemented taking advantage of agent, peer-to-peer, information retrieval and security management technologies and, in

particular, of five main components: JADE [3], BitTorrent DHT [6], Nutch [1], WordNet [17] and JAWS [28].

AOIS agent platforms have been realized by using JADE [3,4,31]. JADE is probably the most known agent development environment enabling the integration of agents and both knowledge and Internet-oriented technologies. Currently, JADE is considered the reference implementation of the FIPA (Foundation for Intelligent Physical Agents) specifications [8]. In fact, it is available under an LPGL open source license, it has a large user group, involving more than two thousands active members, it has been used to realize real systems in different application sectors, and its development is guided by a governing board involving some important industrial companies.

The JADE development environment does not provide any support for the realization of real peer-to-peer systems because it only provides the possibility of federating different platforms through a hierarchical organization of the platform directory facilitators on the basis of a priori knowledge of the agent platforms addresses. Therefore, we extended the JADE directory facilitator to realize real peer-to-peer agent platforms networks thanks to DHT indexing mechanisms and popular file-sharing protocols.

In the first prototypes [23], we used JXTA protocols to augment JADE with the desired peer-to-peer features [9]. In fact, FIPA had acknowledged the importance of the JXTA protocols, and it had released some draft specifications for the interoperability of FIPA platforms connected to peer-to-peer networks. In particular, in the “FIPA JXTA Discovery Middleware Specification” a Generic Discovery Service (GDS) is described, to discover agents and services deployed on FIPA platforms working together in a peer-to-peer network. However, no advancement has then been made for these specifications and JXTA itself has not gained the expected popularity and maturity.

For these reasons, we turned to BitTorrent as one of the most widespread and solid file-sharing platform, which can be configured and extended to work in a completely decentralized fashion [6]. Actually, BitTorrent names both a file-sharing protocol and a particular application, implementing the protocol itself. Other applications, available for many existing hardware/software platforms, implement the protocol.

Basically, BitTorrent requires a tracker server to host so-called torrent files. A torrent file contains the updated list of seeds from which a particular resource can be obtained. For our purposes, i.e. to build a decentralized collaborative network, we preferred avoiding this basic approach. Moreover, in the recent past it has been proven vulnerable to both technical disruptions and legal actions. In fact, today some alternatives allow the realization of trackerless systems.

Azureus was the first application to introduce a Distributed Hash Table to supplement the centralized index. Today, this indexing mechanism is supported through a standard plug-in and is called Distributed Database (DDB). Vuze is an evolution of Azureus that uses such an approach [32]. The Vuze DDB is based on the Kademlia algorithms by Maymounkov and Mazières, which are essentially used to associate the hashes of files and chunks with their current locations (seeds), in a fully distributed fashion [16]. A widespread standard to share a reference to a file is a magnet-uri, which contains the hash of the file. Both nodes and shared files have globally unique, 160 bit long, identifiers. Each node maintains a small routing table

with contact information for a small number of other nodes; the routing table is more detailed for closer nodes. The distance is measured according to the XOR metric defined by Maymounkov and Mazières. The information regarding the peers sharing a given file is stored on nodes with ID close to the hash of the file. When a node wants to download a file for which it knows the hash, it asks further information to the nodes it knows with ID closer to the file hash. These nodes answer with the list of peers downloading the file if they have such an information, otherwise they return a list of nodes with IDs even closer to the file hash, which should be queried afterwards.

Apart from basic file sharing protocols, however, a generic service advertisement system needs some mechanisms to discover a, possibly semantically enriched, service description, starting from some requested features and desired quality of service. In principle, keywords and tags can be associated with any file, and in particular with a service description, quite easily even over DHTs. In fact, some decentralized file sharing platforms use their DHT to implement two different indexes: one for associating seeds to file IDs, the other for associating file IDs with some keywords. However, the keyword index is hardly verifiable in an automated way and in the real world it proved to be particularly weak with respect to pollution and index poisoning attacks. Montassier et al. provide a measure of the credibility of the keyword index of the popular KAD network and show that around 2/3 of the contents are polluted [19]. For this reason, in current implementation of our service discovery system, keyword indexing is based on a DHT, but the keyword-service association is only trusted if provided by a trusted node, participating in the same collaborative network, and possibly other sources suggested by those trusted nodes. Specifically, each node can associate some attributes with the descriptions of the services it provides. This information is then published in the DHT under a unique key, which is obtained by combining the attribute and the node's identity in the system. Other than the attributes associated with a service in the DHT, a node can then analyze the obtained descriptions, in detail, to choose a particular service among those with the basic set of attributes.

Currently, also other applications support some form of DHT indexing. In particular, the BitTorrent application introduced a mechanism named Mainline DHT, which is also based on Kademlia. The queries available under the Mainline DHT allow a robust exchange of information and well support the BitTorrent file-sharing protocol. However, they are unsuitable for our purposes. In order for the DHT to map an attribute to some service descriptions, it is necessary to use arbitrary keys (the node identifiers combined with the attribute) on the DHT. Essentially, a couple of put/get queries would be needed, which would simply associate a given key to a given value on the DHT. Those queries, instead, are readily available under the Vuze DDB, and thus make it a preferable choice when implementing a generic service advertisement and discovery system. As a consequence, we decided to use Vuze DDB for both our logical DHTs, although in principle the DHT mapping hashes to the files that generated them could have been the Mainline DHT.

Regarding the Vuze platform, it has a modular architecture, where functionality can be added with plug-ins. The main application exposes to the plug-ins only a restricted programming interface, which is nonetheless sufficient for our purposes. Consequently, we decided to implement the service discovery system as a Vuze plug-in. As Vuze is implemented as a modular system, it is possible to run it without any

graphical user interface, and thus to use its backend features inside other applications, too. In our case, the Vuze backend is used for realizing the needed discovery, location and sharing services inside a full agent-based system based on JADE. The multi-agent system acts as a Vuze plugin, while Vuze APIs are used as low level primitives for implementing the needed services inside the multi-agent system.

Even if there are some specific tools and software libraries for searching information in a local repository (see, for example, Beagle [2] and Google Desktop Search [10]), we adapted Nutch [1], an open source web-search software, for searching the local repository. It has been done because it is very easy to develop Nutch plugins for extending its capabilities (we used this feature for using its term extraction module for building the topic ontologies) and because is available a Nutch plugin, that extends keywords based search through the use of OWL ontologies [33]. Figure 3 shows a graphical description of the work done by the Nutch core software and by its two plugins for indexing and building the topic ontologies and for using them for searching information.

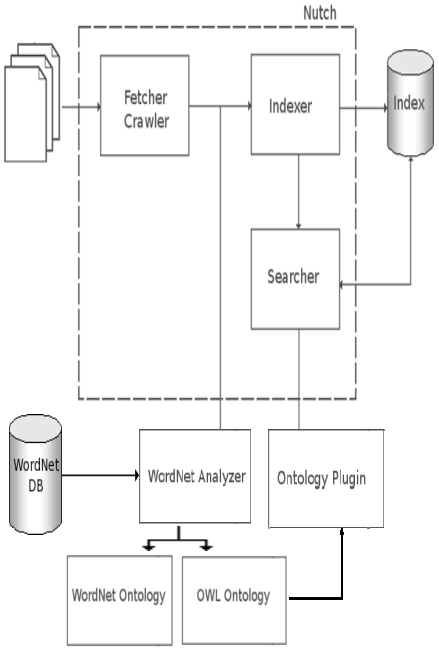

Fig. 3. Indexing and ontology management subsystem.

As introduced above, topic ontologies are built by a Nutch plugin. This plugin receives the terms extracted from the information to be indexed by the Nutch software. Then, accessing the WordNet lexical database [17,24] though the use of the JAWS Java software library [28], for each term it identifies the top terms of the ontology and the other terms extracted from the information that have some semantic relationships (i.e., synonyms, hyponyms, hypernyms, meronyms and holonyms). At the end of this process, all the terms that have a semantic distance greater than the one fixed by the user are removed and then the WordNet ontology is saved as an OWL file.

As introduces before, authentication and authorization are performed on the basis of the local knowledge base of trusted users, though they can be delegated to external entities through an explicit, certificate based delegation. In this sense, the system completely adheres to the principles of trust management. The definition of roles and attributes is also made in a local namespace, and the whole system is, in this regard, completely distributed. Local names are distinguished by prefixing them with the principal defining them, i.e., a hash of the public key associated with the local agent platform. Links among different local namespace, again, can be explicitly defined by issuing appropriate certificates. The theory of AOIS delegation certificates is founded on SPKI/SDSI specifications [7], though the certificate encoding is different. As in SPKI, principals are identified by their public keys, or by a cryptographic hash of their public keys. Instead of s-expressions, AOIS uses XML signed documents, in the form of SAML assertions [21], to convey identity, role and property assignments. As in SPKI, delegation is possible if the delegating principal issues a certificate whose subject is a name defined by another, trusted, principal. The latter can successively issue other certificates to assign other principals (public keys) to its local name. In this sense, local names act as distributed roles [14].

Finally, the extraction of a digest for each search result is required to avoid the presentation of duplicate results to the user. This feature is provided by a Java implementation of a hash function [24].

5 Testing

Practical tests on the first prototype of the AOIS system were done installing the system in different labs and offices of our department asking some students and colleagues to use it for sharing and exchanging information. Moreover, we tested the system setting some computers of a Lab with different access policies and distributing information on their repositories providing, in some cases, different copies of the same information on different computers. The tests covered with success all the system features and the searching and pushing of information satisfied our expectations.

Moreover, a part of the experimentation was oriented to compare the results of the searching and pushing operations based on the use of topic ontologies with the ones based only on the use of keywords and what happened is that: i) the use of topic ontologies increases the number of results, but very few were of no interest for the users if, in particular, the users chose a good set of top terms.

Up to now, we do not perform a numeric analysis of the results, but only a qualitative analysis derived from a discussion with the people that used the system. The main result is that usually the quality of search and pushing operations mainly depends on an appropriate set of top keywords. Therefore, the goodness of an ontology usually does not depends on the keywords extracted from the information of the repository, but mainly depends on an appropriate initial set of top keywords and then by the introduction of the other appropriate keywords coming from the queries of remote users.

6 Conclusions

In this paper, we presented a peer-to-peer multi-agent system, called AOIS (Agents and Ontology based Information Sharing), that supports the sharing of information among a community of users connected through the Internet. AOIS is an evolution of a previous system [15], called RAIS (Remote Assistant for Information Sharing), that performed a similar task, but was implemented by using a different search technology (i.e., Google Desktop Search) and did not take advantage of topic ontologies for the search of information. The first prototypes of the AOIS system used the JXTA protocols to provide the peer-to-peer features useful to support the interaction among remote users. However, JXTA is not used in the most known and used peer-to-peer applications and so it is suitable to test the features of prototypes, but it is unsuitable for developing real application. Therefore, the last AOIS implementation is based on BitTorrent, one of the most widespread and solid file-sharing platform, which can be configured and extended to work in a completely decentralized fashion.

AOIS derives a large part of its features from the systems for information sharing described in the related work section. However, it offers a new feature that seems to improve the quality of search and pushing operations: the creation of a topic ontology through the use of a set of initial terms (i.e., the top terms), its automatic extension through the information maintained by the user, the possibility of controlling the semantic distance from the top terms and the terms automatically added, and, in particular, the possibility of using the terms contained in the queries of the other users for refining the ontology, allow the construction of high quality ontologies. Then, a topic ontology can be customized by each user, but taking into account of the implicit suggestions of the other users of the community, Moreover, its implementation based on some well-known software tools guarantees good performance and reliability.

The first prototypes of the AOIS system was experimented in some “artificial” communities involving researchers and students of our University and the results of the experimentation encouraged us in the further development. The introduction of BitTorrent in its last prototype will allow us to plan a real and large experimentation thinking to the hundreds of millions of users that already share documents though the BitTorrent platform and may be interested in using such a system.

The current implementation of the system maintains in the remote user profiles all the queries she/he did. Often the information retrieved through some old queries might be not yet of interest for the remote user. Therefore, we are working on a more sophisticated technique for managing remote user profiles: all the queries are stored together with the time they were executed; every day the IP checks the remote user profiles and for all the queries that are older than a fixed duration (e.g., a week), it asks the PA about the interest of its user in maintaining such queries and refreshes the execution time for all the queries for which it receives a positive acknowledge.

The creation of topic ontologies may be a difficult activity because it requires the identification of an appropriate set of top terms and its completion through the use of an appropriate set of information. Therefore, the possibility of using the topic ontologies built by other users may be an important feature of the system. In fact, we are working to the possibility that PA agents can require some topic ontologies to other PA agents and then either directly use them for driving the search or build new topic ontologies by merging them with the local topic ontologies.

Beyond the definition of the top terms and of the maximum semantic distance between terms, users have not the possibility of managing the topic ontologies. But this would be a very important feature in the future, when the system will allow the use of topic ontologies defined by other users and the merging among different topic ontologies. In the current version of the system the topic ontologies are also saved as OWL files because the search ontology Nutch plugin requires an OWL file for proving ontology based search. Therefore, users may manipulate topic ontologies by using one of the available tools for manipulating OWL ontologies (e.g., Protégé [24]). However, in the OWL view of the topic ontologies there is not information about the top terms. Therefore, we are developing a very simple graphical tools (based on the use of the Jung software library [12]) that: i) shows the graph defining an ontology, ii) distinguishes top terms from the other terms, iii) distinguishes the different kinds of semantic relationships among terms, iv) allows the introduction of new terms and the deletion of existing terms, v) allows the introduction and the deletion of the "top" attribute to any term, and vi) allow the modification of the maximum semantic distance (when such a distance is reduced, the tool removes all the terms that do not satisfy the new constraint).

Acknowledgments

This work is partially supported by the Italian Ministry MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca).

References

1. Apache Foundation (2011) Nutch software. Available from http://nutch.apache.org.
2. Beagle Team (2011) Beagle software. Available from http://beagle-project.org.
3. Bellifemine F, Poggi A, Rimassa G (2001) Developing multi agent systems with a FIPA-compliant agent framework. Software Practice & Experience 31:103-128.
4. Bellifemine F, Caire G, Poggi A, Rimassa G (2008) JADE: a Software Framework for Developing Multi-Agent Applications. Lessons Learned. Information and Software Technology Journal 50:10-21.
5. Chen, JR, Wolf SR, Wragg SD (2000) A Distributed Multi-Agent System for Collaborative Information Management and Sharing. In: Proc. of the 9th ACM International Conference on Information and Knowledge Management, pp. 382-388.
6. Cohen B (2003) Incentives build robustness in BitTorrent. In Proceedings of the First Workshop on Economics of Peer-to-Peer Systems. Berkeley, CA.
7. Ellison, C, Frantz, B, Lampson, B, Rivest, R, Thomas B, Ylonen T (1999) SPKI Certificate Theory. RFC 2693.
8. FIPA Consortium (2011) FIPA Specifications. Available from http://www.fipa.org.
9. Gong L (2001) JXTA: A network programming environment. IEEE Internet Computing, 5(3):88-95.
10. Google (2011) About Google Desktop Search software. Available from http://desktop.google.com.

11. Hotho, A, Jäschke R, Schmitz C, Stumme G (2006) Information Retrieval in Folksonomies: Search and Ranking. In: The Semantic Web: Research and Applications, Lecture Notes in Computer Science, vol. 4011, pp. 411-426. Springer, Berlin, Germany.
12. Jung Team (2011) Jung software. Available from http://jung.sourceforge.net.
13. Klusch M (2001) Information Agent Technology for the Internet: A survey. Data & Knowledge Engineering 36(3):337-372.
14. Li N, Mitchell JM (2003) RT. A Role-based Trust-management Framework In: Proc. of the Third DARPA Information Survivability Conference and Exposition (DISCEX III), pp. 201-212. Washington, DC.
15. Mari M, Poggi A, Tomaiuolo M, Turci P (2008) Enhancing Information Sharing Through Agents. In: Agent-Oriented Information Systems IV, Lecture Notes in Computer Science, vol. 4898, pp. 202-211. Springer, Berlin, Germany.
16. Maymounkov P, Mazières D (2002) Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In: Peer-to-Peer Systems, Lecture Notes in Computer Science, vol. 2429, pp. 53-65. Springer, Berlin, Germany.
17. Miller GA (1995) WordNet: A Lexical Database for English. Communications of the ACM 38(11):39-41.
18. Mine T, Matsuno D, Kogo A, Amamiya M (2004) Design and implementation of agent community based peer-to-peer information retrieval method. In: Cooperative Information Agents VIII, Lecture Notes in Computer Science, vol. 3191, pp. 31-46. Springer, Berlin, Germany.
19. Montassier G, Cholez T, Doyen G, Khatoun R, Chrisment I, Festor O (2011) Content pollution quantification in large P2P networks: A measurement study on KAD. In: Proc. of the IEEE International Conference on Peer-to-Peer Computing, pp. 30-33, Kyoto, Japan.
20. Moreno A, Riano D, Isern D, Bocio J, Sanchez D, Jimenez L (2004) Knowledge Exploitation from the Web. In: Practical Aspects of Knowledge Management, Lecture Notes in Computer Science, vol. 3336, pp. 175–185, Springer, Berlin, Germany.
21. OASIS (2011) SAML specifications. Available from http://saml.xml.org.
22. Parameswaran M, Susarla A, Whinston AB (2001) P2P Networking: An Information-Sharing Alternative. Computer 34(7):31-38.
23. Poggi A, Tomaiuolo M (2011) A Multi-Agent System for Information Semantic Sharing. In; Proc. of the 5th International Workshop on New Challenges in Distributed Information Filtering and Retrieval, Palermo, Italy.
24. Princeton Universty (2011) Wordnet. Available from http://wordnet.princeton.edu.
25. Rivest RL (1992) The MD5 Message Digest Algorithm. Internet RFC 1321.
26. Salter J, Antonopoulos N (2006) CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering IEEE Intelligent Systems 21(1):35-41.
27. Sanchez D, Isern D, Moreno A (2006) Integrated Agent-Based Approach for Ontology-Driven Web Filtering. In: Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science, vol. 4253, pp. 758-765. Springer, Berlin, Germany.
28. Southern Methodist University (2011) JAWS software. Available from http://lyle.smu.edu/~tspell/jaws.
29. Stanford University (2011) Protégé software. Available from http://protege.stanford.edu.
30. Tang, C, Xu Z, Dwarkadas S (2003) Peer-to-peer information retrieval using self-organizing semantic overlay networks. In: Proc. of ACM SIGCOMM, pp. 175–186.
31. Telecom Italia (2012) JADE software. Available from http://jade.tilab.com.
32. Vuze BitTorrent Client (2012) Available from http://vuze .sourceforge.net.
33. W3C Consortium (2009) OWL 2 Web Ontology Language Overview. Available from http://www.w3.org/TR/owl2-overview.
34. Wiemer-Hastings PM (1999) How Latent is Latent Semantic Analysis? In: Proc. of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99), pp. 932-941.