MULTIAGENT SYSTEM FOR SEARCHING IN FOAF NETWORKS

Zdeněk MAZAL, Doctoral Degree Programme (1)
Dept. of Intelligent Systems, FIT, BUT
E-mail: mazal@fit.vutbr.cz

Supervised by: Doc. František Zbořil

ABSTRACT

FOAF – the Friend of a Friend project is currently one of the most spread applications of W3C's Semantic Web. FOAF is about creating machine–readable homepages describing people, the links between them and the things they create and do. Most of the software written for FOAF is concerned with visualisation of FOAF files, especially the social networks they include. Exploring such networks can be fun, however searching them for some specific information is much more useful. The problem with searching the whole network is that it requires large storage capacity, connectivity etc. We are addressing this problem by proposing a multiagent system based on peer–to–peer communication for searching in FOAF networks.

1 INTRODUCTION

As most of us have already experienced, searching the World Wide Web (WWW) is not an easy task. The reason for this is that the WWW consists of a huge number of pages, which are from the searching point of view more or less an unstructured text with very little meta information. This has lead the World Wide Web Consortium (W3C) to starting of a new project – the Semantic Web (SW) [9]. The goal of SW is to create a framework for sharing well structured data between applications on the Internet. SW is based mainly on Resource Description Framework (RDF) [6] – RDF defines a data model for describing resources using triples of form (subject–predicate–object). For integration of this concept in current WWW, existing technologies are used – namely Uniform Resource Identifier (URI) and XML serialisation. For defining relations between resources and properties, languages for creating vocabularies are used; most common are RDF Schema and Web Ontology Language (OWL). These standards (technically W3C's recommendations) do not describe vocabularies for any particular domain, just a common framework for creating them.

The applications of the SW are not so wide spread yet, however RSS (RDF Site Summary, Rich Site Summary or Really Simple Syndication – all of these names are used in literature) and Friend of a Friend (FOAF) [2] have already found their way to the public audience. We will deal with FOAF in the rest of the article.

FOAF is technically an RDF vocabulary. It defines classes and properties for describing people, their interests, projects, chat accounts, weblogs and what is most important, provides ways to create links to FOAF files of people they know. The main idea of the project is that if people publish their personal information in FOAF files, together with links to their friends' FOAF files, machines will be able to crawl these files and present interesting facts to the user. Most of the programs written for FOAF like [10] or [4] however provide only a simple visualisation of the contents of file at some URL presented by user, allowing the user to manually navigate in the graph, the possibilities of searching are usually limited – one example of a bot answering questions about IRC users is available at [7]. The reason for this is that crawling and storing the content of the files requires significant connectivity and disk space. There are two possibilities how to address this problem. One can either create a big Google–like server crawling the web or a peer–to–peer network of hosts (agents) where every agent has a limited storage capacity and connectivity, but is able to communicate with other hosts. The results of user queries are then achieved mainly by communication and cooperation. We have chosen the second approach and we will describe the architecture of the multi agent system in the rest of the article.

The multi agent paradigm was already described elsewhere (a good introduction can be found at [8]), we expect the reader to be familiar at least with the basic concepts.

2 SYSTEM ARCHITECTURE

The overall system architecture can be seen in figure 1. Each agent consists of three main layers; each layer has a specific purpose.

The top layer is used for accessing the actual content of the SW (i.e. the FOAF files), crawling the pages and (optionally) storing the results in a local database. We are using JENA [5] SW framework for implementation of major part of this layer's functions. More details about JENA are presented further in the article.

The middle layer has two main tasks – it provides user interface and carries out the reasoning of the agent, i.e. controls and assigns tasks to the top and bottom layer. We were considering two possibilities for implementation of this layer: (1) use some available Belief Desire Intention (BDI) system or (2) create a reactive agent. We are planning to implement both, so we can compare the approaches; in the first prototype we are using the reactive rules. These two top layers can run as a standalone application omitting the agent communication.

Finally, the bottom layer is responsible for communication with other agents. This layer is being implemented using the JADE [1] middleware, which provides convenient methods for agent communication and management. Some interesting aspects of each individual layers together with more detailed description of JENA and JADE are presented in the next section.

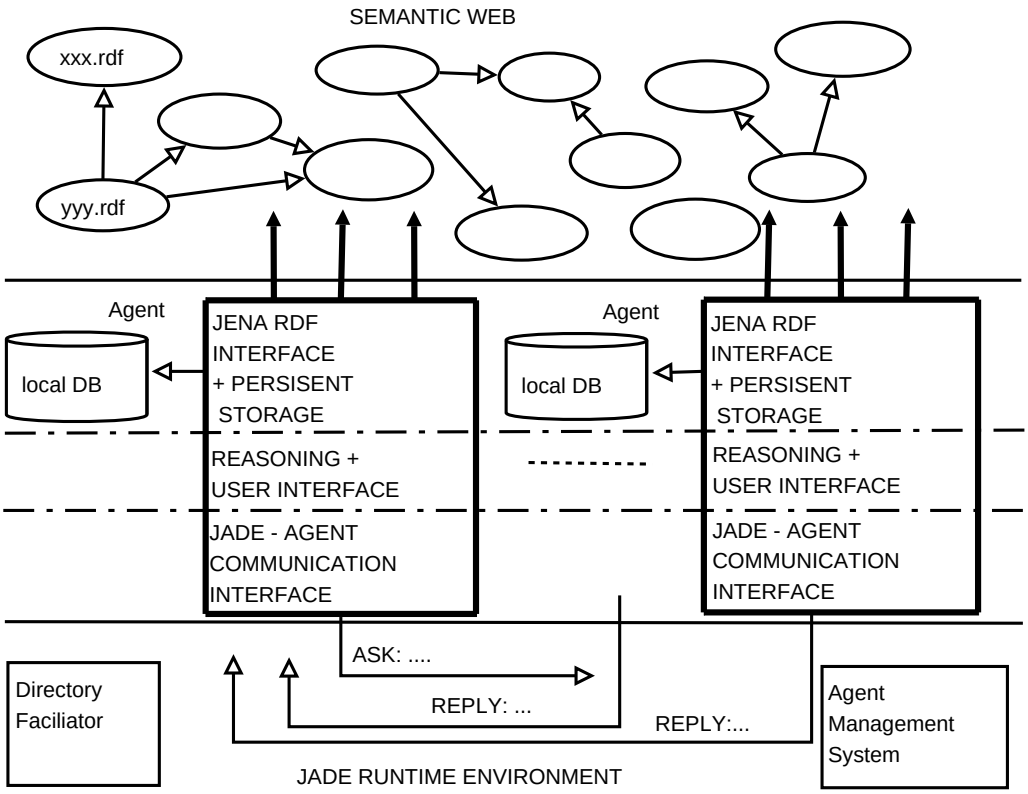

Figure 1: Overall system architecture of the multiagent system for searching in FOAF files

3 AGENT LAYERS AND FURTHER DETAILS

3.1 TOP LAYER – JENA

As we already mentioned, the main purpose of the top layer is crawling the FOAF files on the WWW and storing the triples in a local database. Most of the required operations are already implemented in JENA [5], an open source framework for creating SW applications in Java, which is developed in HP laboratories. JENA provides a programmatic environment for handling RDF, RDFS and OWL, a rule–based inference engine, implementation of a SPARQL query language and means for storing the triples in a database.

JENA doesn't have any special support for FOAF, so our main focus at the top layer was on creating algorithms for extraction and searching of desired properties from the graphs, because several different constructions for expressing the same fact are allowed in FOAF (for example a person can be described as a member of class Person or a more general class Agent). What's more, FOAF can be mixed with other RDF dictionaries, such as Dublin Core (DC), RSS and others. Extracted links to other FOAF files are passed to the middle layer, which decides what files will be crawled next.

3.2 MIDDLE LAYER – USER INTERFACE AND REASONING

The user presents its request to the agent via a search form. The form includes fields for all properties that can be used for specifying searching constraints – currently only a basic subset of properties defined by FOAF (URI, name, nickname, e-mail, e-mail SHA1 hashcode etc.). The user can also choose whether he wants to find some specific information about the person (like e-mail address) or all available data should be retrieved.

After the user submits the form a query for local data is prepared and executed. Results of this query (if there are any) are presented to the user. At the same time the request to other agents is also sent – the protocol of this communication is discussed further in the article. The replies of the agents don't include directly the triples replying to the query, only links to FOAF files, which should be relevant. These files are then processed by the local agent (user has a complete control over this process and can stop processing of selected files).

While being idle, the agent crawls the links extracted from FOAF files that were either entered by user or processed when executing some query. This process can again be controlled by the user – globally via preferences (by general rules on how much crawling should be done etc.) and also a list of the files that are scheduled to be crawled next can be updated.

In the prototype implementation we use the reactive agent paradigm, i.e. the agent responds with an action according to some predefined rule whenever an event occurs. However our plan is to use a BDI reasoning in the next version. Regarding to the software used, JADEX – a BDI extension to JADE seems to be natural choice.

3.3 BOTTOM LAYER – JADE

The bottom layer is responsible for communication among agents. We implement this layer using JADE [1] (Java Agent DEvelopment Framework). JADE is an open source middleware for creating multi agent systems communicating on peer–to–peer basis. It is written in Java and is compatible with relevant standards of the Foundation for Intelligent Physical Agents (FIPA), which ensures very good interoperatibility. JADE provides convenient classes for wrapping the higher layers into a reactive agent body, a distributed runtime environment where the agents "live", and some useful tools for debugging of the system. The JADE environment provides also facilities for dynamic creation of agent communities using a system of Directory Facilitor agents (yellow pages services).

The style of ontology support implementation in JADE (using Java classes and object) is not suitable for use with RDF and therefore the bottom layer treats the content of the messages as plain text and only passes it to the middle layer where it is processed.

JADE provides a framework for common agent communication scenarios defined by FIPA. The scenario currently used in our system is FIPA–Request, we also plan to include FIPA–Subscribe for implementation of long term queries. As a content language for the messages, the RDF is used – FIPA defines such use of RDF in the specification [3], which however still has the experimental status. The request messages include the specification of the person queried by user; it doesn't include the actual query (from various reasons, one of them is security). The reply includes a list of FOAF files, which should contain the queried information wrapped in RDF.

4 CONCLUSION

We described the architecture of a multi agent system for searching in FOAF networks. The prototype of the system is currently being implemented. We have done some testing of a standalone agent (the top two layers), however the overall performance of the system very much depends on the number of agents in the system and the communication among them. This will be the main subject of our future work and we hope to be able to use the results and experiences gained in a more general SW application in the future.

ACKNOWLEDGMENTS

This article is a part of research supported by Czech Science Foundation grant GACR 102/05/H050.

REFERENCES

[1] Bellifemine F., Poggi A., Rimassa G.: JADE – a FIPA–compliant agent framework. CSELT internal technical report. Part of this report has been also published in Proceedings of PAAM'99, London, April 1999, pp.97–108.

[2] Brickley, D.: The foaf project [online]. 2000 [cit. 2006-03-01]. Available from WWW: <http://www.foaf-project.org>.

[3] Foundation for Intelligent Physical Agents: FIPA RDF Content Language Specification [online]. 2001 [cit. 2006-03-01]. Available from WWW: <http://www.fipa.org/specs/fipa00011/XC00011B.html>.

[4] Frederiksen, M.: FoaF Explorer [online]. 2002 [cit. 2006-03-01]. Available from WWW: <http://xml.mfd-consult.dk/foaf/explorer/>.

[5] Hewlett-Packard Development Company LP: Jena – A Semantic Web Framework for Java [online]. [c2003–2005] [cit. 2006-03-01]. Available from WWW: <http://jena.sourceforge.net/>.

[6] Manola, F., Miller, E.: RDF Primer [online]. 2004 [cit. 2006-03-01]. Available from WWW: <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.

[7] Useful Information Company: FOAFBot: IRC Community Support Agent [online]. c2000–2005 [cit. 2006-03-01]. Available from WWW: <http://usefulinc.com/foaf/foafbot>.

[8] Wolldridge, M.: An Introduction to MultiAgent Systems. Wiley, Chichester 2002.

[9] World Wide Web Consortium: Semantic Web [online]. 2001, [cit. 2006-03-01]. Available from WWW: <http://www.w3.org/2001/sw/>.

[10] Foafnaut [online]. [2003] [cit. 2006-03-01]. Available from WWW: <http://www.foafnaut.org/>.