Development of flexible peer-to-peer information systems using adaptable mobile agents

Conference Paper · January 2004

DOI: 10.1109/DEXA.2004.1333532 · Source: IEEE Xplore

3 authors:

Jean-Paul Arcangeli
Paul Sabatier University - Toulouse III
37 PUBLICATIONS **168** CITATIONS
SEE PROFILE

Sébastien Leriche
Ecole Nationale de l'Aviation Civile
26 PUBLICATIONS **91** CITATIONS
SEE PROFILE

Marc Pantel
Ecole Nationale Supérieure d'Electrotechnique…
105 PUBLICATIONS **531** CITATIONS
SEE PROFILE

Development of Flexible Peer-To-Peer Information Systems using Adaptable Mobile Agents

Jean-Paul Arcangeli, Sebastien Leriche
IRIT-UPS
118, route de Narbonne, 31062 Toulouse France
{arcangel, leriche}@irit.fr

Marc Pantel
IRIT-ENSEEIHT,
2, rue Camichel, 31071 Toulouse France
pantel@enseeiht.fr

Abstract

Wide-area networks provide an easy access to many different distributed and heterogeneous data sources. The development of automated operating tools is still complex, particularly because of evolution and adaptation requirements (to the data sources structures and to the network Quality of Service). The purpose of this paper is to evaluate the advantages of adaptable mobile agents in order to simplify the development and the deployment. As an experiment, we develop a prototype of peer-to-peer information system, and we show how agent mobility and adaptation abilities help in implementing various forms of adaptation: adaptation to the execution context, access to new servers with initially unknown communication protocols, dynamic modification of search algorithms based on results provided by the servers. Then, we show how these techniques can be easily extended to other problems such as search and upgrade of software components.

1. Introduction

Nowadays, large amounts of data, services, hardware and software resources are connected by wide-area networks. Data and computing grids and peer-to-peer computing have emerged in such a distributed and heterogeneous context. Peer-to-peer information systems are based on symmetric relations between nodes (clients and servers) which both require and provide (possibly temporarily) data or services.

Distributed information systems must take care of software and hardware heterogeneity (interaction mechanisms with servers, data representation...). Besides, their execution contexts are highly unstable (availability and state of servers, network, services and data) and lack central organization (deployement over multiple administrative domains; no possible assumption on the availability and on the quality of services). Consequently, they must deal with dynamic variations of the execution context, unanticipated and uncontrolled upgrade of information servers, and client specific requirements. Hence, both static and dynamic adaptation abilities are useful.

Taking into account adaptation requirements increases complexity and costs of software engineering. Thus, appropriate software techniques and tools are necessary in order to facilitate development, maintenance, and deployment. In this work, we are interested in the mobile agent paradigm enhanced with individual adaptation properties.

A software agent is an autonomous computing entity with private knowledge and behavior. A mobile agent is a software agent able to move at runtime with its code, data, and computational state [7]. Basically, agents move in order to replace remote interactions by local ones (in order to limit network traffic) and to customize remote services. An adaptable agent can change some of its operating and functional mechanisms at runtime. Mobility and adaptation are controlled by the agent itself. In order to fit wide-area networks, agents communicate in asynchronous mode.

This paper relates the experimental development of a flexible peer-to-peer prototype system which allows to share and retrieve distributed documents. It relies on adaptable mobile agents provided by a Java library. Our goal is to evaluate their advantages and drawbacks for constructing and deploying efficient distributed and heterogeneous information systems (efficiency concerns software engineering costs, response times to queries, and quality and relevance of retrieved data).

Section 2 briefly presents related work on the use of mobile agents for information retrieval. Section 3 presents a primary version of the prototype based on agent mobility. Section 4 details how adaptation mechanisms allows to refine and specialize the prototype. Finally, we conclude on our experience and draft some future works.

Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA'04)
1529-4188/04 $ 20.00 IEEE

2. Related work

The early works on mobile agents suggested information retrieval as a potential *killer application*, especially in case of multiple distributed data sources, large volumes of data, numerous interactions between clients and sources, and client specific search procedures [6]. In [4], authors propose a simple application of textual document retrieval in a local network with centralized organization: mobile agents move queries to servers depending on their complexity and on the state of the network; mobile agents interact with network-sensing agents and with stationary agents (one per site) which serve as interfaces with the local servers. The DBMS-Aglet system [11] implements a Java mobile agent-based solution for querying distributed heterogeneous databases *via* the *Web*: a mobile agent moves on the remote server site, gets the JDBC pilot on-the-fly, performs the search locally, and goes back to the client site with the results. Compared to an applet-based solution, it turns out that the agent-based solution is more efficient regarding response times, robustness, scalability and flexibility.

M3 -MultiMedia database Mobile agents- [10] is a system for content-based retrieval and indexing of multimedia data on top of Java and CORBA. A mobile agent roams from site to site, searches information locally using client-specific code, memorizes results, uses and updates them along the trip. As far as we know, M3 is one of the only systems which truly exploits the main advantages related to agent mobility : customization and evolution. Besides, several works propose mobile agents for the *Web* crawling process by search engines as an alternative to the centralized strategy *download first, process later* [8, 12].

However, even if significant general works on mobile agents have been conducted, they have seldom been used in practice. Limitations come often from security constraints (protection of hosts, agents, and communications) [13]. *Ad hoc* solutions have been proposed but they generally rely on strong assumptions and restrictions. The problem of security is still widely open but quite simple and realistic solutions seems attainable in the context of distributed information retrieval (see [10] and [12]) based on encryption, authentication, confidence assumptions, sand-boxing...

3. Development of a prototype system for information sharing and retrieval

3.1. Specification of the prototype

The prototype, called JAVANE, allows to share, search and exchange documents (music, picture, text, program...) distributed across a large scale heterogeneous network, in accordance with a peer-to-peer model and algorithms inspired by eDonkey[1]. On each node, JAVANE allows users to publish, manage and share some of their files (server role) and to search and download files from the network (client role). Each site has informations about other existing servers on which searches can be performed. These links of knowledge between sites evolve with time.

During this study, several protocols have been defined and implemented with mobile agents, all inspired by existing P2P applications like eDonkey or Kazaa. Search agents crawl the network using mobility or by creating other agents, dynamically discovering on each new site some information about other existing sites on which they will be able to jump. Download is done exploiting the parallelism resulting from the distributed replication of data, with adaptation to the network bandwith.

While crawling and exploring the network, clients also discover meta-informations in particular knowledge about peers. In our protocol, the client gets knowledge *via* a simple message sent by the mobile agent. This knowledge is used to update client's database about known servers. With this mechanism, the client can access to new servers, improve server selection and increase his ability to find pertinent information.

3.2. Implementation

3.2.1. JAVACT : a middleware for adaptable mobile agent-based programming. The implementation is based on JAVACT [3], a free software under LGPL license developed in our group. JAVACT[2] is a standard Java middleware which provides a mobile agent-based programming model with fine individual adaptation mechanisms. It is not a platform dedicated to the development of multi-agent systems but a tool for large scale distributed programming.

JAVACT applications run on networked places (Java virtual machines) connected by a lower-level middleware (e.g. Java RMI, CORBA, SOAP, or even TCP/IP sockets) whose use is hidden to the programmer. JAVACT agents are actors [1] i.e. software autonomous entities that communicate asynchronously, process serially the messages they receive, and change their behavior at runtime. JAVACT actors can move from site to site and resume their activity after they move. JAVACT library provides about ten methods for programming behaviors of agents. The ones we use to implement the search protocol are: $go(place)$, $myPlace()$, $send(message, targetAgent)$ and $suicide()$.

JAVANE core is a JAVACT place extended (by inheritance) with a local information server. As any Java-based software, JAVANE is portable. Consequently, it can easily run in an heterogeneous network.

1 http://www.edonkey2000.com

2 http://www.irit.fr/recherches/ISPR/IAM/JavAct.html

```
public class SearchBehavior extends JavAneBehavior
                            implements Searcher, javact.util.StandAlone {

    public SearchBehavior(client, clientInfo, energy, queryStamp, Params) {
        ...
    }

    public void run() {
        if (energy <= 0) {
            suicide();
            return;
        }
        // Has the current place been already visited ?
        if (! markedPlace(myPlace(), queryStamp, client)) energy -= E_VISITED;
        else {
            // Database interrogation and sending of results to the client
            localResults = LocalDB.fileSelect(Params);
            if (localResults != null && localResults.length > 0) {
                send(new JAMprocessResults(place, localResults), client);
                energy -= E_FOUND;
            } else energy -= E_NOT_FOUND;
        }
        // Random moving (if it remains energy)
        if (energy > 0) {
            String[] places = getPlaces(myPlace());
            String nextPlace = places[random.nextInt(places.length)];
            go(nextPlace);
        } else suicide();
        // Collection of meta-information
        String[] placesInfos = getPlacesInfos(myPlace());
        send(new JAMaddPlacesInfos(placesInfos), clientInfo);
    }
}
```

Figure 1. Behavior of searcher agents

3.2.2. The search protocol. Let us see how the search protocol is implemented. Two Java classes implement the behaviors of the client agent (the first one to deploy searcher agents, the second one to recover file descriptors). One class implements the functional behavior of searcher agents (see Figure 1 in which declarations and exception processing have been abstracted). The $run()$ method is automatically executed when an agent in installed on a place (declared in the $StandAlone$ interface).

3.3. Evaluation

The experiment confirms advantages of mobile agents in a wide-area distributed and heterogeneous context in which no centralization is possible and in presence of instability and volatility of resources and data. Agent's autonomy and dynamic adaptation ability support efficient and robust protocols, which are implemented very simply.

In our system, deployment and dynamic adaptation of the process (search and information discovery) are automatic without demanding interactions with the client. They depend on the execution context (availability of network bandwidth and of servers, links between places, and server contents) and on the results found. Protocols are defined and chosen by the client and shipped on server sites. Running remotely client-specific search processes moves computing load from the client site to the server site and limits network traffic to the useful data (and not all data as it should be with the standard client-server approach). Besides, searcher agents require limited rights only; therefore, they can *a priori* be considered as non dangerous for servers.

JAVANE has been implemented with several search protocols, tested and validated in various network and system configurations. Four master students have been charged of the development after they have trained a few hours on mobile agents and actors. The programming model proved to be simple and easy-to-handle, and JAVACT abstractions on distribution, communication, and synchronization simplified the development. Concise code has been written naturally, thus contributing to increase software reliability: for example, there are approximately 120 lines of code for the entire search protocol presented above.

4. Specialization of the retrieval process

Adaptation consists in modifying and specializing (statically or dynamically) some operational or functional features. This section shows how the primary prototype can be easily extended and enriched. Agents' adaptation abilities allow to address some issues raised by peer-to-peer computing (communication problems, server heterogeneity and uncontrolled upgrade) and support sophisticated protocols for information discovery in the network.

4.1. Operational adaptation

Operational adaptation concerns internal mechanisms of agents: message sending, message reception, behavior changing, life cycle... In order to allow operational adaptation, JAVACT proposes an architecture of agent based on replaceable and customizable micro-components, each of them implementing an operational mechanism.

Several specialized components have been developed: encrypted communications, message sending tolerant to disconnections (for wireless communications with temporary loss of signal), etc. These components have been transparently plugged in the prototype without affecting the functional code.

4.2. Functional adaptation

Functional adaptation concerns agents' knowledge and know-how. JAVACT functional adaptation mainly relies on actor's abilities to change its behavior (i.e. private knowledge, methods for message processing, and state) and to create other actors dynamically.

Let us consider local and global aspects of the search process (Figure 2). An agent who moves on a server site must adapt to the local interaction protocol and to the form of data (dialog client-provider, A_1). It must be able to run locally a personal indexing and retrieval code, or even a fine querying plan (local search, A_2). At least, it must be able to process locally the resulting data and decide what to do with them (operating on results, A_3). Lastly, network crawling must be adaptable depending on the state of the execution system and on the results found (global search, A_4).

The collect of informations on the network could be personalized in the same way.

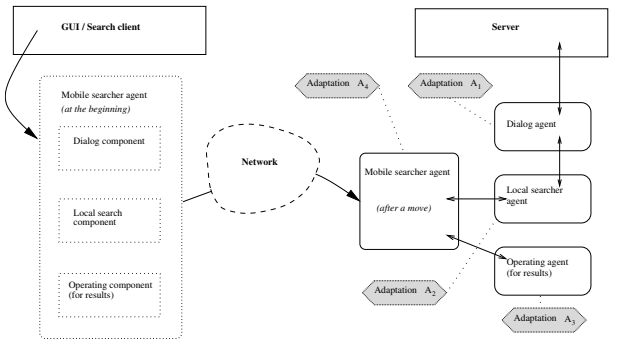

Figure 2. Points of functional adaptation

4.2.1. Agent adaptation to the heterogeneity of servers. We assume that if a client knows a peer (an information server), it also knows how to interact with it. In order to do that, the client has a dialog component for each known server. This component is embedded within the mobile agent behavior to serve as an interface with the server on the remote place. Consequently, when a new server is discovered (from a third peer), the client must acquire both its reference and the convenient dialog component (servers have to provide an additional functionality to supply it). The searcher agent must dynamically acquire it too (by means of behavior change) before visiting the new server.

However, this protocol must be extended in order to support unanticipated and uncontrolled upgrades of servers. Because of network's decentralization and size, it cannot be assumed that clients are informed of servers upgrades. Thus, agents may reach a server while carrying an obsolete dialog component. They must then require dialog components from the server site (for security reasons, authentication of the provider could be necessary).

In practice, the dialog component is an actor behavior possibly instantiated by client specific data. On server site, the searcher agent creates a dialog agent from the dialog component, and activates it with a message containing the query and its parameters.

Finally, using mobile searcher agents allows dynamic adaptation and customization of the search process and information discovery without demanding remote interaction with the client.

4.2.2. Local search adaptation. Client-specific procedures advantages have been presented in section 3.3. A local searcher agent is created from the corresponding embedded dialog component. It can run a sophisticated querying plan from the complete data collection without network bandwith constraints. It can also evolve by changing its behavior.

4.2.3. Operating on results. From the corresponding embedded component, the mobile agent creates an agent dedicated to the post-processing of results on the server site. Post-processing means sending files or descriptors to the remote client, possibly after encryption or compression, or sending them locally to the searcher mobile agent. The component implements client's policy which can depend on the obtained results and on the state of the network.

4.2.4. Global search adaptation. Agent mobility is proactive : the searcher agent decides by its own to move according to a client-specific policy which can be based on an evaluation of the quality of the results found, on the state of the network, on knowledge on potential servers, etc. It can stop or continue the search, go back to the client site, or possibly submit another query *via* the local searcher agent. Thus, the agent's itinerary is dynamically and autonomously built and decision are made from and close to relevant data.

4.3. Examples of functional adaptation

Originally, JAVANE allows to retrieve and download documents from a collection distributed over a peer-to-peer network. This section illustrates the enhancing of JAVANE with new functionalities by doing adaptations at the levels A_3 and A_4.

4.3.1. Remote procedure calls. The problem is to find a remote software service, to transmit parameters to the service, to execute it and send back the results. Adaptation is required at the level A_3 : post-processing consists in running the service found on the server site. The component is built on the client site and initialized with the desired parameters, then moved and executed on the server site by the corresponding agent.

4.3.2. Search of software components. The problem is to search a software component for an update purpose. Installing the new component (on the client site) may demand search and installation of other components. Adaptations are required at the levels A_3 and A_4. At the level A_3, the found software component must be sent back to the client with the corresponding documentation. At the level A_4, the global search must handle the search of other components if required.

4.3.3. Update of software components. The problem is to find a remote software component which provides a service replicated on different servers, and to replace every

copy of the component by another (in this case, the update is performed on the server site by the component provider client). Adaptations are required at the levels A_3 and A_4. At the level A_3, operating on result consists in replacing the component found by the one supplied by the client at the beginning of the update process. At the level A_4, the search agent must go all over the network in order to update servers where it is necessary.

5. Conclusion and future works

In order to evaluate the advantages of adaptable mobile agents to support information systems distributed over wide-area networks and grids, we have experimented the development of a flexible prototype which allows to share and retrieve documents in peer-to-peer mode. In such a distributed, heterogeneous and decentralized context:
- absence of hardware failures, and permanence and constant quality of services cannot be assumed,
- users are not aware of the true state of the execution context when querying the system.
Therefore, the search process must take in account instability and adapt to unavailability and non-anticipated evolutions of servers, services and data.

On one hand, due to their autonomy, adaptable mobile agents allow dynamic operational and functional adaptation of the search process to the execution context: communication encryption, compression of data, client-specific algorithms, dynamic routing of agents and exploration of the network, discovery of information and servers, etc. Customizing the search process and moving it on server sites should decrease the volumes of data moved over the network and consequently the response times (and should possibly improve the relevance of results).

On the other hand, adaptable mobile agents help in developing flexible and extensible peer-to-peer information system (our one is quite simple but also realistic and powerful). Advantages mainly set in behavior changing ability and in the separation between operational and functional aspects. By separating client-specific procedures from server basic services, maintenance of servers is simplified and limited to a local operation, while agent's adaptation features increases robustness and tolerance to servers unavailability and upgrades. Additionally, deployment and redeployment on the network are automatic without demanding remote control from clients or servers. Actually, adaptable mobile agents help in deploying an adaptive component-based search process.

This work should be continued in the following directions:
- Development of techniques and tools to secure composition of micro-components and preserve consistency at the agent level.
- Improvement of expressiveness of distribution and security by enhancing JAVACT with the concept of domain [9] derived from works on formalization of security and organization problems [5].
- Technology closely related to the one presented in this paper should be used in the context of the GRID-TLSE[3] project in which the third author is involved [2]. The objective is to build an expertise web site for sparse matrices in the context of open grids. Distributed services for sparse linear algebra must be located, then executed and results sent back to the client. Services can result from composition of other ones which must be located and executed recursively.

References

[1] G. Agha. *Actors: a model of concurrent computation in distributed systems*. M.I.T. Press, Cambridge, Ma., 1986.

[2] P. Amestoy and M. Pantel. Grid-tlse: A web expertise site for sparse linear algebra. In *Sparse days and Grid computing at St Girons workshop*, 2003.

[3] J.-P. Arcangeli, C. Maurel, and F. Migeon. An API for high-level software engineering of distributed and mobile applications. In *8th IEEE Workshop on FTDCS*, pages 155–161. IEEE-CS Press, 2001.

[4] B. Brewington, R. Gray, K. Moizumi, D. Kotz, G. Cybenko, and D. Rus. Mobile agents in distributed information retrieval. In *Intelligent Information Agents*. Springer, 1999.

[5] L. Cardelli. Abstractions for mobile computation. In *Secure Internet Programming*, pages 51–94, 1999.

[6] D. Chess, C. Harrison, and A. Kershenbaum. Mobile agents: Are they a good idea ? Technical report, IBM, 1994.

[7] A. Fuggetta, G. Picco, and G. Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, 1998.

[8] J. Hammer and J. Fiedler. Using mobile crawlers to search the web efficiently. *Int. Journal of Computer and Information Science*, 1(1):36–58, 2000.

[9] A. Hurault, V. Hennebert, and M. Pantel. Repartition et mobilite en JAVACT : une approche derivee d'un modele formel. *LMO'2004, RSTI-L'Objet*, 10(2-3):47–60, 2004.

[10] H. Kosch, M. Doller, and L. Boszormenyi. Content-based indexing and retrieval supported by mobile agent technology. In *Second International Workshop on Multimedia Databases and Image Communication*, pages 152–166, 2001.

[11] S. Papastavrou, G. Samaras, and E. Pitoura. Mobile Agents for World Wide Web Distributed Database Access. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):802–820, 2000.

[12] P. Thati, P.-H. Chang, and G. Agha. Crawlets: Agents for high performance web search engines. In *Mobile Agents 2001, LNCS 2240*, pages 119–134. Springer-Verlag, 2001.

[13] C. Tschudin. Mobile agent security. In *Intelligent Information Agents*. Springer-Verlag, 1999.

3 http://www.enseeiht.fr/lima/tlse/