An Application of Mobile Agents as Personal Assistents in Electronic Commerce

Volker Roth[1], Mehrdad Jalali[1], Roger Hartman[1], and Christophe Roland[2]

[1] Fraunhofer Institut für Graphische Datenverarbeitung
Rundeturmstraße 6, 64283 Darmstadt, Germany
{vroth|jalali}@igd.fhg.de
[2] Thomson-CSF Communications
66, Rue du Fossé Blanc
BP82, 92231 Gennevilliers Cedex, France
Christophe.Roland@enst.fr

Abstract. In this paper we describe the architecture of a Web-integrated personal commerce assistant application based on mobile agents. The assistant's task is to do some high-level shopping on behalf of a user. In our case the assistant organises the catering of a birthday party.

Keywords: mobile agents, electronic commerce, delegation, World Wide Web, Java

1 Introduction

Mobile agents [7] push the flexibility of distributed systems to their limits since not only computations are dynamically distributed but also the process and code that performs them. Information gathering and electronic commerce are application areas in which mobile agent technology may offer substantial benefits [5]. In the course of the European ESPRIT Project AIMedia (Targeted Advertising on Interactive Media) – a two-year project now running for one and a half years – we developed a proof of concept application based on mobile agent technology by which we intend to demonstrate some of the benefits that can be expected from using mobile agent technology. The basic idea is to delegate a high-level shopping task to a mobile agent (the personal commerce assistent, PCA). In our case, the agent's goal is to organise the catering for a birthday party. We give details of the application in Section 4. The graphical front-end to the application is managed by a dedicated agent that may be accessed through the World Wide Web using commonplace browsers. The integration of our mobile agent framework into the World Wide Web and its application to the PCA application is the subject of Section 3.

A number of mobile agent systems are in existence at present; basic information on about 60 such systems (including ours) was collected in the run-up

to the ASA/MA'99 Conference[1]. Each platform implements a particular flavour of mobile agents and puts emphasis on different aspects of agent mobility. The focus of our developments is on providing applicable and flexible security mechanisms. This is where our platform differs from most comparable ones. In Section 2 we outline the general architecture of our platform. A thorough discussion of the security architecture is beyond the scope of this article; this is covered by two further articles issued for publication.

2 SeMoA Overview

In this section we describe the basic concepts and architecture of a Mobile Agent Server called SeMoA that is under active development at the Fraunhofer Institute for Computer Graphics. Like many others we chose Java as the implementation language and agent programming language. First, Java was gaining widespread attention so we could expect that a sea of third party support would become available, and second, Java already provided a framework for security and class loading that suited our needs very well. Obviously, Java had considerable influence on our design. Apart from the leverage of the Java 2 Sandbox and AccessController model, we decided that SeMoA:

- should have a minimal kernel and simple lightweight interfaces
- be modular, flexible and easily extendable
- build on JCE/JCA to render it independent of particular cryptographic mechanisms
- enforce strict separation of agents
- provide practical and enforcable security mechanisms

SeMoA should not require particular agent transport mechanism or dictate a particular agent communication mechanism. Instead, we wanted it to leverage existing protocols and infrastructures (such as SMTP, POP, IMAP, HTTP, FTP). Both, transport and language support should be done through extensions, that might be provided through agents or special types of plugins (services).

The general architecture of the SeMoA Server is based on the concepts of agents and services (see also Figure 1). Agents may register services through a central *Registry* which makes the services visible to other agents. Special types of services (called plugin services) can be automatically registered and configured by the server on start up. The primary difference between services and agents is that agents are active and services are passive. Agents group a number of resources and permissions such as threads and filesystem storage; services

[1] `http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/mal.html`

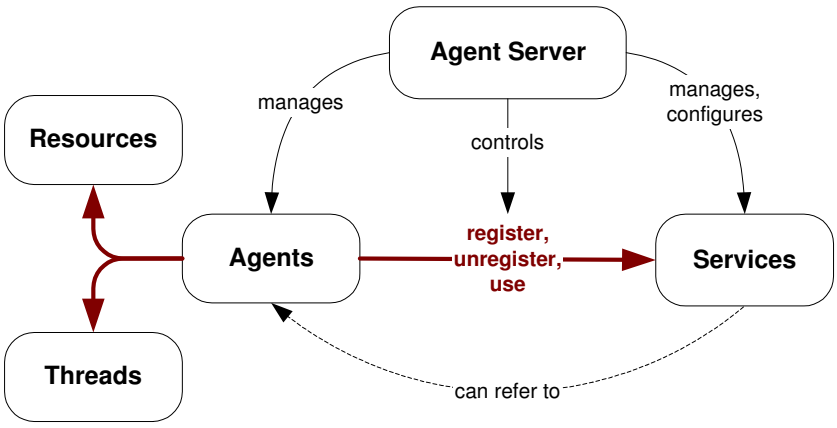

Fig. 1. This figure illustrates the relationships between concepts of the platform.

run in the caller's thread and thus do not require the type of management required for agents. However, services may pose as a front for agents (comparable to proxy objects) that restrict access to agents with regard to a particular purpose or aspect of the agent (see also Figure 2). Services may be registered on a number of service levels (depending on the permissions of the registering agent) that share a common access policy. For each service level the following rights can be granted:

- Register a service.
- Unregister a service.
- Request (use) a service.
- Receive events distributed on the given level.
- Send event on the given level.

Services are identified by the name of their intended level and a well-known service name, and are associated with a well-known interface or class (or subclasses thereof). The Registry supports dictionary operations on the available services that are subject to permission checks. Lookups are based on a given level and service name. The Registry may also list the service levels (public) and the names of services registered on a given level (requires appropriate permission). Since agents may simply instantiate local service classes and use this to circumvent the security checks, the root class of all services captures the Access Control Context [4] on creation and provides convenience methods for the execution of Privileged Actions restricted to the captured context. This ensures that services cannot leak access rights when instantiated directly.

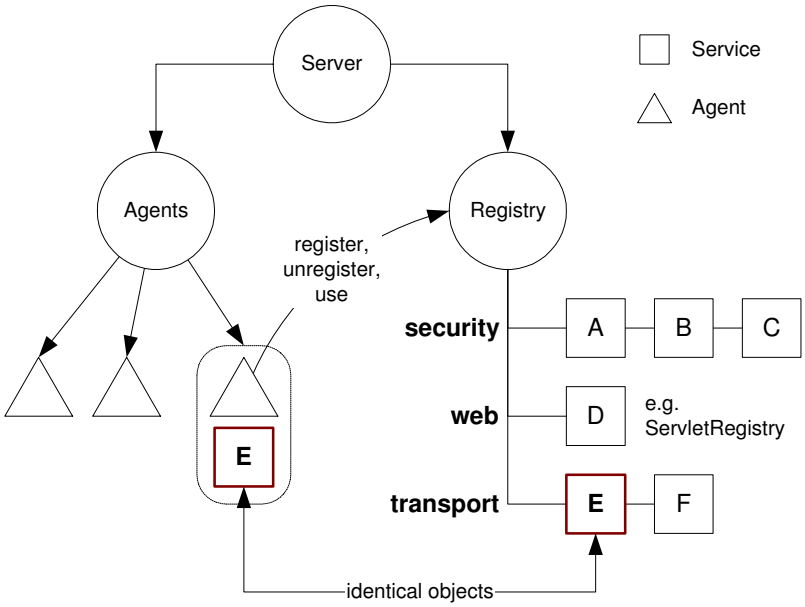

Fig. 2. Agents may register services that provide controlled access to particular aspects of an agent.

Transport mechanisms for transporting agents to remote servers are provided by transport services registered at the *transport* level. A central gateway object (the *outgate*) scans the transport level for registered transport services and matches the advertised transport protocols against the destination specified by agents through a *ticket*. The ticket basically contains a set of alternative URL that define the destination and desired transport protocol of the agent. Incoming transport is handled by agents that for instance listen on network ports and dispatch incoming agents to a central gateway object reserved for incoming agents (the *ingate*). Both ingate and outgate feed agents through a pipeline of filter services registered at the security level. These filters take care of various cryptographic operations required to handle certain aspects of agent and server security. In the spirit of Chess et al. [3] the server is the *Agent Meeting Point* (AMP), the *Registry* corresponds to the *Shallow Router* (albeit on a very low level compared e. g. to a subset of KQML), and the *ingate* corresponds to the *Concierge*.

3 Web Integration

It is hardly conceivable that mobile agent technology could achieve noteworthy market penetration without offering some sort of integration with the Web. For this reason we developed a *Web agent*. This agent speaks HTTP and implements a Servlet engine according to Version 2.1 of Sun's Servlet specification. Servlets are meant to be server-side extension for Web servers such as the popular Apache Server.

Our Web agent registers a service (the *Servlet registry*) with the service registry of its hosting agent server which allows other agents to register Servlets they bring with them (see also Figure 3). Any pre-existing Servlet may be used as long as it conforms to the standardised API. This allows agents to interact with humans through HTML pages. Going even further, mobile agents may carry ordinary Applets that handle – in a standard browser – client side presentation of results or status information generated by its serving mobile agent. Probably the most straightforward application of this approach is the configuration of agents before they are sent away to roam the Internet. We chose this approach for the configuration of the PCA described in Section 4.

Although our Web agent is static at present there is no reason why it shouldn't be able to migrate itself if needed. Hence our Web agent in principle supports the easy and transparent migration of a complete Web server by means of a command. This might become interesting in combination with comparisons of ISP prices for Web hosting. Private home page owners might also appreciate such a feature if Web space providers offer reduced prices. In that case said owners may simply instruct their "home page agent" to move somewhere else.

The Servlet registry and the Web agent do not restrict access to a particular agent's Servlets yet. Anyone who knows the exact URL chosen by the agent upon registering its Servlets may send requests to it. We plan to add a non-public service whereby agents also pass for instance a set of public-key certificates of those who are allowed to contact the Servlet. Contacting non-public Servlets then will require the use of transport layer security mechanisms for authentication and encryption.

4 Personal Commerce Assistent

Based on our mobile agent platform and the Web agent described in Section 3 we developed the personal commerce assistant (PCA) – an ensemble of services and agents that handle a specific high-level shopping task on behalf of a user. By "high-level" we mean a shopping task that is formulated at an abstraction layer above the bare product level (although the outcome probably is a detailed shopping list consisting of products and quantities).

In our case this task is to organise the catering for a birthday party. The agent's task is to select a number of recipes for food and beverages, and to suggest a present. Hypothetical users of the PCA are Admittedly those who give the party – who usually do not buy their own presents. Yet this serves as an example that the shopping task may include more than just food. In principle, decoration can be ordered as well to give the party a certain style. This may well include finding suitable music (for instance in the popular MP3 format) ready to be played at the party.

The PCA scenario consists of the user interface agent, the search agent, a recipes server, a Yellow Pages server, and the servers of retailers. The user either runs a lightweight agent server himself or attaches his browser to the PCA run by some service provider (the browser is used in both cases as the user interface).

The user interface agent registers a Servlet with the Servlet registry of the Web agent on its local platform. The user interface agent provides pages via the Servlet enabling the user to configure the PCA with information on the person giving the party and the expected number of guests. So far, this information includes the sex, age group, category of hobbies, preferred beer brand, and "theme" of the food (either Chinese, American, European, or none in particular). The beer brand is ignored if alcohol is not allowed for the given age group. On submitting the information the user interface agent generates and returns an URL that can be used to retrieve the results once they become available. At the same time it selects two meals (by name) and beverages based on the entered information. If the age group is above 18, five cocktails are also selected (by name). The user interface agent then creates and configures a search agent (this is illustrated in Figure 3).

The search agent then migrates to a recipes server on which it resolves the names of the selected recipes against a small database of recipes. The recipes are normalised to one serving. The quantities are multiplied by the expected number of guests by the search agent. The recipes database consists of a PostgreSQL database that is connected to the agent server by means of a service that issues queries over JDBC. The search agent also looks for a picture database service and retrieves pictures of the selected meals and cocktails if available. Recipes consist of a list of general product entries without specific details such as brand names. The search agent then proceeds to the Yellow Pages server, locates the retailer information service and queries for the sites of retailers that may offer the required products. This information service is based on a flat file database with entries of the known retailers and the product categories offered by them. The search agent assembles its subsequent itinerary and hops from one retailer to the next until all products are found or the itinerary is finished.

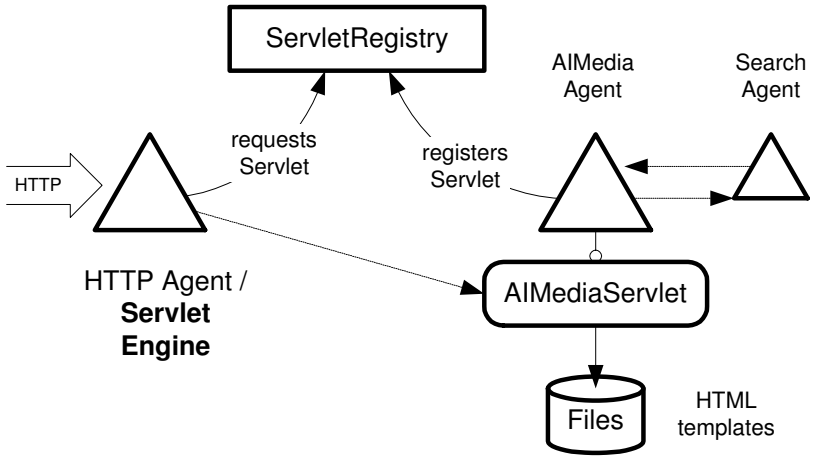

Fig. 3. The integration of Web servlets carried by mobile agents in the server architecture through a service offered by the Web agent.

On each retailer server the search agent looks for a product information service, which is used to retrieve detail information on the required products. This information service connects via JDBC to further PostgreSQL databases with product records kindly provided for testing purposes by our project partners J. Sainsbury's, United Kingdom, and OTTO Versand Hamburg.

The search agent then returns to the originating server, passes the results to the user interface agent, which formats them into a nicely looking HTML page with pictures of the food and beverages. This page is then made available through the URL initially passed as the response to the submission of the user information. It contains a complete shopping list with product details, numbers of required units and capacities per unit.

For demonstration, we usually use three agent servers. The first agent server runs the Web agent and user interface agent. The second server runs the recipes and yellow pages service. The third and final server runs the product information services (see Figure 4).

5 Extensions

Within the AIMedia project additional software components were developed which are targeted at enriching existing Web sites with personalised services by using proxy servers. These extensions are integrated with the mobile agent architecture by means of services registered in the server. These services link to

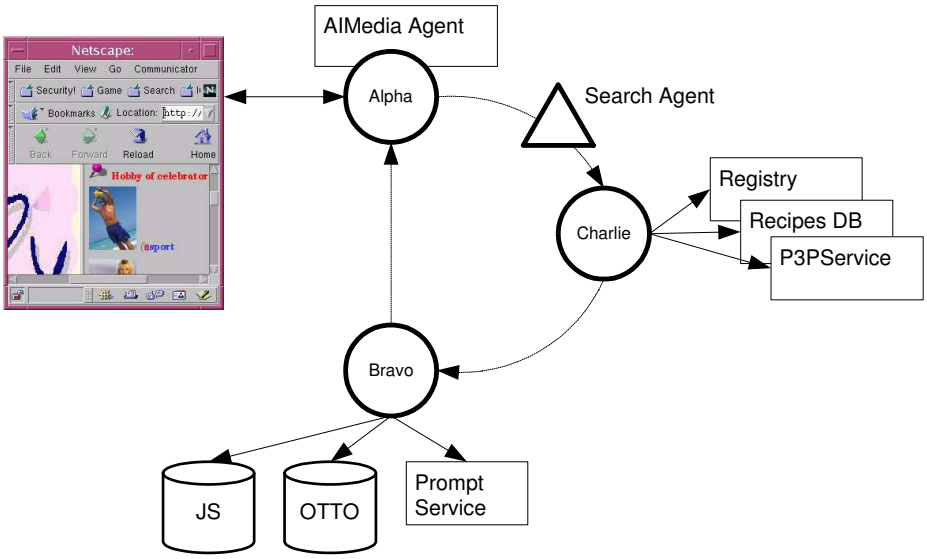

Fig. 4. The architecture of the AIMedia demonstrator.

the components also used in a pure Web-based approach and make their functionality available to mobile agents.

One such component supports the exchange of personal profiles as set forth by the World Wide Web Consortium's Platform for Privacy Preferences Project[2] (P3P). The development of P3P occurred within a consensus process involving representatives from more than a dozen W3C member organizations, as well as invited experts from around the world.

P3P is designed to help users reach agreements with services (Web sites and applications that declare privacy practices and make data requests). As the first step towards reaching an agreement a service sends a machine-readable proposal in which the organization responsible for the service declares its identity and privacy practices. A proposal applies to a specific realm, identified by a URI or set of URIs. The set of statements that may be made in a proposal is defined by the harmonized vocabulary, which is a core set of information practice disclosures. These disclosures are designed to describe what a service does rather than whether it is compliant with a specific law.

P3P does not attempt to guarantee or enforce privacy in itself, but relies on a complementary set of measures to earn the trust of users. Hooks for technical

[2] http://www.w3.org/P3P/

protection mechanisms, trusted third party assistence, and auditing are provided. Prosecution of fraudulent privacy assertions is part of the concept.

We chose P3P in order to allow users to specify personal information in a clearly specified and somewhat controlled way; information provided by users may be used by retailers to provide personalised offers through the AIMedia Shopping Assistant component. The integration of the P3P components is illustrated in Figure 5.

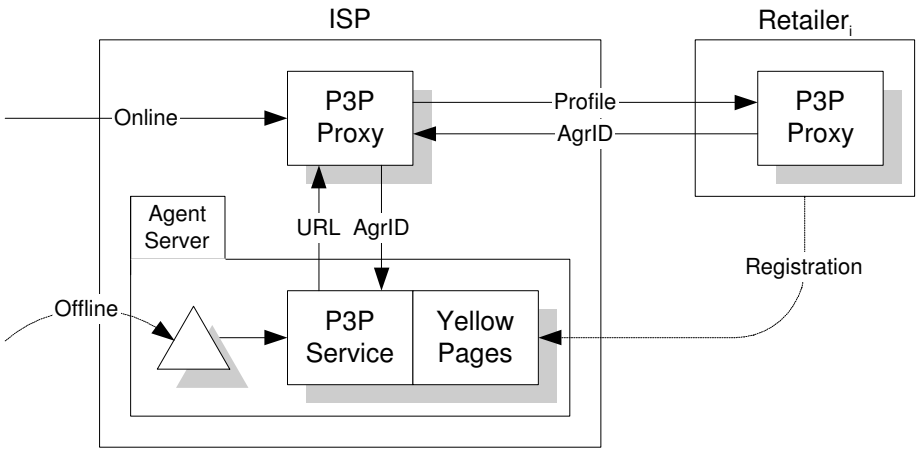

Fig. 5. The P3P negotiation services are integrated into the agent server by means of special services that forge the link.

We assume that the user registers profile information with a trusted service provider, for instance an Internet Service Provider (ISP). PCAs of the user migrate to the ISP's agent server and announce to a P3P service available in the agent server where they intend to go by passing the destination URL. In exchange they receive an *agreement ID* that was established by the P3P proxy of the ISP in the course of the profile negotiation with the retailer's P3P proxy. Retailers must register an URL with the yellow pages that will be used to contact them (see Figure 5) prior to the first negotiation.

The user's PCA then migrates to the desired retailer agent servers and looks for product information. The agent's browsing may be regarded as a session of the agent's owner who browses a Web site of the retailer using an standard browser. The PCA may collect personalised prompts by passing selected products and the agreement ID it received while querying the P3P service on the ISP. The agreement ID allows the retailers to take into account the personal informa-

tion that was provided by the agent's owner through the trusted intermediary. The prompt format is the same in the PCA scenario and the online scenario in AIMedia which is based on ordinary Web browsers. The PCA presents these prompts upon return to the user.

6 Conclusions

The amount of information that is available on the Internet is growing steadily, leading to decreased transparency of the offers. More and more time needs to be invested in order to sift through the amounts of data searching for valuable and relevant information. Agent technology offers compelling advantages with regard to the problems in Internet-based electronic commerce.

Agents may focus on a specific domain and apply domain-specific knowledge in order to optimize their strategies and the quality of results. In electronic marketplaces agents often act as middlemen of either the buyer or seller. For instance Jango [2], and BargainFinder[3] [1] are comparison-shopping agent systems that represent the buyer. However, they often work like intelligent search engines that request and digest information from the original site and store the resulting data centrally. Queries by the user are resolved against this database although the final purchase must be done through the original Web site.

Since original sites are designed to communicate offers to humans the search engine needs to parse the Web pages and extract the relevant information from it. This limits the information (in particular meta-information that is digestible by automated processes) that can be extracted about products. From a technical vantage, agent to agent based systems may relieve the retailer's burden of providing elaborate user interfaces. This may be handled by agents instead. Retailers may concentrate on an efficient and functional representation of offers [8]. The data exchanged between agents can be represented in a way more suitable to automated processes.

Mobile agents may seek for relevant products directly at the source of information which obsoletes the need to transfer massive numbers of Web pages in order to build the database from which summaries are extracted. Since the search criteria and filter algorithms may be distributed with mobile agents, software updates and the introduction of new mechanisms and rules is simple.

The greatest hindrance that we experienced is the lack of abstraction from the actual data in the retailer's databases. In our view appropriate models, product ontologies and agent communication models are prerequisites for a flexible and widespread application of agents in electronic commerce with regard to applications such as the one described in this article. Mobility adds to the benefits

[3] BargainFinder is not available on the Web anymore

of agent systems since a large number of specialised and highly focused services can be distributed easily. Users may run searches and continuously monitor sources of information in the background even while being detached from the network.

Our model and architecture facilitates the deployment of such services and integrates nicely with existing infrastructure of the World Wide Web. The mechanisms used to create Web-enabled agents correspond to mechanisms already in use to extend Web servers and thus feature a flat learning curve.

In the remaining period of the project, the PCA application prototype will be subject to a number of user test in order to measure their responses to this approach and how they feel towards delegating task such as the implemented one to an agent.

7 Acknowledgments

Parts of this work were sponsored through the ESPRIT project *AIMedia: Targeted Advertising on Interactive Media*, project number 26983. We would like to thank our project partners, in particular Sabine Geissel from OTTO Versand Hamburg, and Ian Hawkins and Mark Venables from Sainsbury's for their kind support and for providing the testbed databases.

References

1. Cstar. Internet resource at URL `http://bf.cstar.ac.com/`, 1999.
2. Exite shopping. Internet resource at URL `http://www.jango.com/`, Version current 6th Dec. 1999.
3. David Chess, Benjamin Grosof, Colin Harrison, David Levine, Colin Parris, and Gene Tsudik. Itinerant agents for mobile computing. *IEEE Personal Communications*, pages 34–49, October 1995.
4. Li Gong. *Java™ Security Architecture (JDK 1.2)*. Sun Microsystems, Inc. in [6], relative URL: `file:/docs/guide/security/spec/security-spec.doc.html`.
5. Danny B. Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, March 1999.
6. Sun Microsystems, Inc. *JDK 1.2 Documentation*, 1998. Available at URL: `http://java.sun.com`.
7. James E. White. Mobile agents. In J. Bradshaw, editor, *Software Agents*, chapter 18, pages 437–472. AAAI/MIT Press, Menlo Park, CA, 1997.
8. Rüdiger Zarnekow and Walter Brenner. Diensteebenen und kommunikationsstrukturen agentenbasierter elektronischer märkte. *Informatik Spektrum*, 22(5):344–350, October 1999.