# Data Agents System: A Query Service based on Mobile Agent Technology

Rosane Maria Martins, Luci Pirmez, Luiz Fernando Rust da Costa Carmo
NCE/UFRJ – Núcleo de Computação Eletrônica/Universidade Federal do Rio de Janeiro
Tel.: +55 21 598-3160 – Caixa Postal: 2324 – Rio de Janeiro - RJ
Brasil

## ABSTRACT

The rapid growth of the Internet and the World Wide Web (Web) provides access to vast amounts of valuable information information. However, the problem of information overload is an obstacle to the practical use of potentially useful information on the Web. The use of mobile agents in this kind of applications represents a novel approach and potentially solves most of the problems that exist in centralizes client;server solutions, because they are programs with a persistent identity which moves around a network and can communicate with this environment and other agents. We present a possible solution for this problem: the Data Agent system - a mobile agent application for the retrieval of distributed structured information in a scenario of several on-line bookstores. This system was developed for Web-based distributed access to database systems based on Java-based mobile agents. This paper describes the project architecture and its implementation that is based on IBM´s Aglets Workbench. It also emphasizes the obtained results with the several experiments realized, concluding that the implementation of the system shows that its performance is comparable to, and in some case outperforms the current approach.

**KEY WORDS:** Mobile Agents, Aglets, Multi-Agents System

## 1. INTRODUCTION

Nowadays the number of available information sources is growing rapidly. Finding and combining the relevant information is becoming a critical task. There is a need for facilities that perform these integrating tasks and thus overcome problems such as distribution and heterogeneity. These facilities are often referred to as integrated systems. In a integrated system, the user is not exactly aware of which and how many information sources that are used, neither does he knows how they are used. The user is provided with the vision that only one information source exists.

In this paper, the mobile agent-oriented approach to this problem is discussed. Software agents can be used to structure and integrate information. To illustrate it, we present the Data Agents system [2] - a mobile agent-based prototype for the retrieval of distributed structured information in a scenario of several on line bookstores. The proposed system is based on a group of agents that try to find simultaneously the users' interest products in the several virtual places known by them, presenting the results in an homogeneous way.

In the remainder of this paper, we provide background in the area of agents (sections 2 and 3), introduce the architecture and implementation project and discuss the performance evaluation of the Data Agents system (section 4). Finally, we conclude with a discussion of future directions of this work (section 5).

## 2. RELATED WORK

Software agents have become very popular in the last six or so years. They have been used successfully to filter information, match people with similar interests and automate repetitive behavior. More recently, the capabilities of agents have been applied to electronic commerce, promising a revolution in the way we conduct transactions. One of these examples is the Andersen Consulting's Bargain Finder. This is a sophisticated broker designed to aid in online shopping applications by gathering information, from 9 websites, and delivering the price and shipping terms of a certain good requested by the user [11].

Like Bargain Finder and others systems based on collaborative filtering technology [16], Firefly [12] helps consumers find products. However, instead of filtering products based on features, Firefly recommends products via an automated "word of mouth" recommendation mechanism called collaborative filtering. The system first compares a shopper's product ratings with those of other shoppers. After identifying the shopper's "nearest neighbors" (i.e., users with similar taste), the system recommends products that neighbors had rated highly but which the shopper may not yet have rated, potentially resulting in serendipitous finds. Essentially, Firefly uses the opinions of like-minded people to offer recommendations. The system is used to recommend commodity products such as music and books, as well as harder to characterize products such as web pages restaurants.

Others aspects of value added by agents in electronic commerce is their usefulness in the dynamic process of transacting a deal. There are several agent systems that assist the customer in negotiating terms of a transaction : AuctionBot [13], FishMarket [14], Kasbah [15], and Tete-a-Tete [16].

AuctionBot is a general purpose Internet auction server at the University of Michigan. AuctionBot users create new auctions to sell products by choosing from a selection of auction types and then specifying its parameters (e.g., clearing times, method for resolving bidding ties, the number of sellers permitted, etc.). Buyers and sellers can then bid according to the multilateral distributive negotiation protocols of the created auction. In a typical scenario, a seller would bid a reservation price after creating the auction and let AuctionBot manage and enforce buyer bidding according to the auction protocols and parameters. What makes AuctionBot different from most other auction sites, however, is that it provides an application programmable interface (API) for users to create their own software agents to autonomously compete in the AuctionBot marketplace. However, as with the Fishmarket Project, it is left to the users to encode their own bidding strategies. Fishmarket is not currently being used as a real-world system, but it has hosted tournaments to compare opponents' hand-crafted bidding strategies along the lines of Axelrod's prisoner's dilemma tournaments.

MIT Media Lab's Kasbah is an on-line, multi-agent consumer-to-consumer transactions system. A user wanting to buy or sell a good creates an agent, gives it some strategic direction, and sends it off into a centralized agent marketplace. Kasbah agents pro-actively seek out potential buyers or sellers and negotiate with them on behalf of their owners. Each agent's goal is to complete an acceptable deal on behalf of its user subject to a set of user-specified constraints, such as a initial asking (or bidding) price, a lowest (or highest) acceptable price, a date by which to complete the transaction and restrictions on what parties to negotiate with and how to change the price over time. Kasbah's agents automate much of the Merchant Brokering and Negotiation stages for both buyers and sellers.

Tete-a-Tete provides a unique negotiation approach to retail sales. Unlike most other online negotiation systems which competitively negotiate over price, Tete-a-Tete's consumer-owned shopping agents and merchant-owned sales agents cooperatively negotiate across multiple terms of a transaction including warranties, delivery times, service contracts, return policies, loan options, gift services, and other merchant value-added services. Tete-a-Tete considers product features and merchant features equally throughout negotiations to help the shopper simultaneously determine what to buy and who to buy from. This integration of Product and Merchant Brokering through integrative negotiations has the unique benefit that constraints on product features can affect the decision of who to buy from. For example, only a certain merchant may be able to support a particular product configuration. Likewise, constraints on merchant features can affect the decision of what to buy. For example, if no merchant can accommodate the overnight delivery of a specific product, an alternate product which can be delivered overnight may be determined to have a better overall value.

As we've seen, there are several agent based e-commerce applications, however, we've not found any one which applies the advantages mobility property. Because of this, the paper purpose is to improve the performance of an e-commerce prototype using mobile agent technology.

## 3. BACKGROUN MATERIAL

### 3.1 Mobile Agents

With the development of network technology, the whole computing environment has changed profoundly and become highly distributed, heterogeneous and dynamic. Traditional client/server (C/S) model can no longer meet the needs of complicated distributed computing because of its inflexibility.

Mobile Software Agent (MSA) is a new distributed computing model that can meet the needs of current computing environment. In C/S model, computing entities are static and passive, but in MSA model, they can migrate and finish computing and are implemented through agent migration and interaction. [9]

A Mobile Agent has the unique ability to transport itself from one system in a network to another. This ability allows a mobile agent to move to a system that contains an object with which the agent wants to interact and then to take advantage of being in the same host or network as the object. After its submission, each mobile agent proceeds autonomously and independently of the sending client. When the agent reaches a server, it is delivered to an agent execution environment. Then, if the agent possesses necessary authentication credentials, its executable parts are started. To accomplish its task, the mobile agent can transport itself to another server, spawn new agents, and interact with other agents. Upon completion, the mobile agent delivers the results to the sending client or to another server.[3,4]

In order for these agents to exist within a system or to themselves form a system they require a framework for implementation and execution. This is known as the agent environment.

### 3.2 IBM Aglets: Java Mobile Agent Technology

The Aglets Software Developer Kit (ASDK) [10] was developed at IBM Research Laboratory in Japan. It is a framework for programming mobile network agents in Java. From a technical point of view, the IBM's mobile agent called "aglet" (agile applet), is a lightweight Java object that can move automosuly from one computer host to another for execution, carrying along its program code and state as well as the so far obtained data.

Unlike an applet's short and boring period of execution, an aglet can exist and execute tasks forever. One of the main differences between an aglet and the simple mobile code of Java applets, is the itinerary that is carried along with the aglet. By having a travel plan, aglets are capable of roaming the Internet collecting information from many places. The itinerary can change dynamically giving the aglet the sense of self-governing and the look of an intelligent agent (that of course is in the hands of the prrogrammer).

An aglet can be dispatched to any remote host that supports the Java Virtual Machine. This requires from the remote host to have preinstalled Tahiti, a tiny aglet server program implemented in Java and providedby the Aglet Framework. A running Tahiti server listens to the host's ports for incoming aglets, captures them, and provides them with an aglet context (i.e., an agent execution environment) in which they can run their code from the state that it was halted before they were dispatched. Within its context, an aglet can communicate with other aglets, collect local information and when convenient halt its execution and be dispatched to another host. An aglet can also be cloned or disposed.

## 4. DATA AGENS SYSTEM

### 4.1 Project Architecture

E-commerce is becoming an attractive means of conducting business. At present, numerous web sites offer products for purchase over the Internet. However, none of these sites is truly automated, as human intervention is required for browsing, selecting and ordering products. Moreover, such sites are essentially passive catalogs of products and prices, with mechanisms for receiving orders from buyers.

The agents for electronic commerce considered in this context are those that somehow help the users to shop over the Internet. This type of agent, called shopping agent, may carry out several tasks, such as: to help the user decide what product should be purchased; to make suggestions based on its knowledge of its owner; to find out new things, discounts and special prices; to find stores that sell the desired product or service, among other things.

In order to show the feasibility of the search process for structured and distributed information through the mobile agents technology, this paper proposes the development of a multiagent and mobile system called "*Data Agents*" in the context of several "on line" bookstores. The goal is to accelerate the retrieval of distributed structured information. This is achieved by improving the phase of the process of data selection, in which the agents run parallel among the servers related to them and at the end returning with all the information requested by the user, without the need to make a call to each one of the servers separately. The information obtained is then presented in a uniform and organized way. Using the information thus presented by the system, it is much easier for the user to choose a product with the most satisfactory characteristics.

Among the possible functions described above, the Data Agents Agent is intended to help find the stores that sell the desired product and to list the prices of the products found.

The operation of the prototype to achieve this objective is the following:

- user selects the specific product and the desired characteristics of that product (these characteristics will be the restrictions for the search);
- the purchase agent searches for products with the desired characteristics among products of that type;
- as a result of the search, Data Agents sends an e-mail or shows a screen to the user with a list of products, their respective prices and where they can be found.

The following architecture is proposed to enable the Data Agents system to have the functionality mentioned above and, in the future, to be applied to many products and stores.

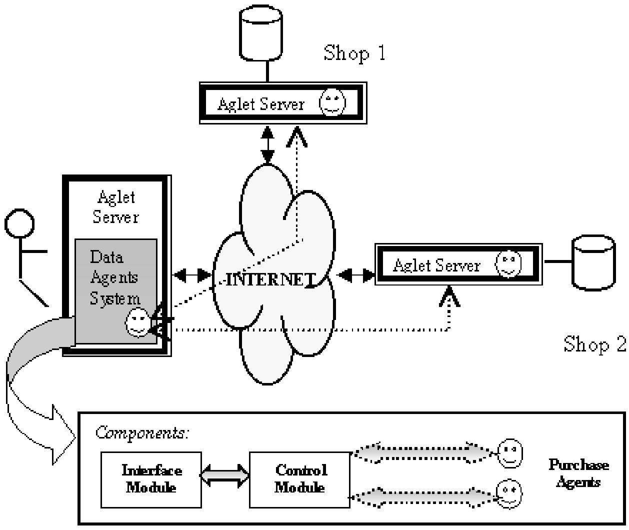

Figure 1 – Project Architecture

The system presented comprises the following components: Interface Module, Control Module and Purchase Agents. What follows is a detailed list of the system components:

**Interface Module**: this is the component through which the user contacts the system and places his order. This module is also responsible for presenting the result obtained by the group of agents to the user.

"Title", "author" ,"price range" and "type of itinerary" are the information that the user must provide to the Interface Module so that it may request the Control Module to create and dispatch the purchase agents according to the restrictions imposed by the user.

There are three possibilities of choices for itineraries:

a. <u>*one agent for each server:*</u> According to the quantity of servers registered in the system, one agent is created for each server and dispatched to do its task. When each agent arrives at its destiny, it does its search, send the result as a message to Control Module and "dies".

b. *only one agent that visits all the servers:* It is created only one agent that has in its travel plan the addresses of all servers. It will go to all servers, one by one, do the search, send the result as message to Control Module and "dies" at last visited server.

c*. one agent that goes through the servers until to find the first occurrence*: It is created only one agent that contains in its travel plan the addresses of all servers. But it will travel to next server only if doesn't find any book at former server, that is, the agent travels until to find the first occurrence that satisfies the order user.

As soon as the result manager (a component of the control module) compiles all answers received, it sends these answers to the interface module so that they are delivered to the user: on the screen or via e-mail.

**Control Module**: This module is responsible for the creation and release of purchase agents to begin the search requested by the buyer. This module also aggregates the results found by the different agents. There is a control module for each type of product available in the system, e.g. a control module for books and a different one for CDs.

After receiving the user’s requirements from the interface module, the Control Module creates the agents according to such requirements and sends them to the addresses available at a Storage Structure.

Storage Structure is a hash structure that contains the addresses of the various stores associated with the system. There is a storage structure for each type of product researched by the system.

When the Control Module receives a request to send an agent, the latter is created on the “aglet” layer according to user's requirements and travels through the runtime layer, which converts the agent into an array of bytes and such array, on its turn, passes on to the ATP layer – Agent Transfer Protocol, to be sent to its destination. This protocol, then, builds a “bit stream” that contains both general information, such as the system name, and its identification, such as the “byte array” resulting from the runtime layer.

Upon returning to the server with the information from its search, each purchase agent sends its contents to the Result Manager (Control Module), so that the Results Manager may aggregate all answers obtained and send them to the interface module.

**Purchase Agents :** Make contact with the stores by accessing their databases, place the order and interpret the answers generated, converting them into a format that is understood by the control module. Before proceeding to their destination, the agents are coded in bit stream: the first segments are general information, such as the agent’s identification, and the last segment is the byte array, the agent per se: code and state. The goal of the agents is to check the information found at their destination address, selecting only the information considered relevant and recommended according to the pre-determined rules. Such information shall represent the basis of rules to be used by the agent to make appropriate decisions in the process of evaluation of the items found.

With this architecture, the extension of this system to deal with new products and new stores is simple, although it is necessary to build a control module for each new product.

So, several resources about the same type of information could be organized in different groups and could be answered by a specific member of the protptype (bookshops, CD shops, newspapers). This fact would allow a great and better vision to the user in order to him can compare the prices of the several products of the web.

## 4.2 Project Implementation

This project was based on an experiment done to investigated and test the suitability of using mobile agents in a distributed and multiplataform environment to produce a solution to a purchase order in a global market. The scenario is based on a process very conventional and common nowadays: the search of books in virtual bookshops. The Data Agents system uses a parallel query architecture in order to query pricing and availability of user specified books. The system then combines the filtered results as a summary to the user, finding the best price and providing a unified interface for different vendors, thus negating the need for the user to navigate to different stores and deal with separate user interfaces.

The model architecture is characterized by the presence and interaction of three types of agents:

**Purchase agent –** is the only mobile component of the model; it travels through the net until arrive at the hosts where does the wanted searches. Each purchase agent when it is created, it receives all the necessary data to do the search. After receiving it, this agent starts its trip towards the first host of its address list. When it arrives at each host defined in its itinerary, the purchase agent executes the lista() method, that does the connection with database locally and does the query. After receiving the result, the purchase afent send a message to control module that has as a argument a vector with all the data from done query converted into a string.

**Control agent –** it is created by the interface agent. It is responsible for the control and creation of the quantity of necessary purchase agents in agreement with the requeriments received from the interface agent. After its creation, the control agent creates a data structure for each purchase agent within the drivers (that will be used), the host location (atp address), the users and the passwords. Next, it is created a wait interface that indicates the control module was created and it is waiting for the return of purchase agents.

**Interface Agent** – it is the first agent that is created. It presents a graphic interface where the user specifies the data for the search and creates the control agent providing to it the search parameters through the messages.

The interaction among the agents works through a series of precoded messages that can or cannot have any argument for the agent. For example: the message “caminho” has as argument the vector that indicates the places that the purchase agent has to visit (its itinerary), while the message “iniciar” indicates that the purchase agent has to start its trip to do the wanted searches.

So, with the agents organized this way, only the purchase agent travels over the net – it goes to the host where it does the search locally and afterwards it sends the results obtained through the net. The whole process is started with the creation of interface module that creates a graphic interface where the user specifies his preferences to the search. The Control Module when receives the user requirements, it creates one or more agents, converts them in an array of bytes. This array is passed to the ATP layer (Agent Transfer Protocol) in order to it can be sent to its destiny. This protocol builds, so, a bit stream that has general information, such as system and identification name and the array of bytes origined from runtime layer.

Each purchase agent when returns to source host, sends all its contents to the control module component, Result manager in order to him can aggregate all answers obtained and send them to the interface module. The interface module, per se, sends the result to the user by e-mail or by screen.

### 4.3 Performance Evaluation

A simulation environment was developed in order to implement and test the Data Agent system and establish a basic structure of programming in a distributed system.The operation system used it was Windows NT 4.0 and all the code lines were written in Java and ASDK library. The tests were done at  computers: Pentium 233, 64 Mb RAM, with the JDK 1.1.7A.

The proposed system acts in a simulated environment of searching of books over the Internet. The main goal is to enable a hogh degree of automation of Internet market. At any time, a user can delegate tasks to a handler agent. Such task is to look for good offers matching a certain interest. Even if the user is off-line, he can be notofied about the ordered query through e-mail.

The entity-relationship model used by this system and available in each host is a simple model with four tables: AUTORES (“authors”), GÊNEROS (“genders”), LIVRO_AUTOR (“book_author”) and LIVROS(“books”). The query is done, in fact, in the tables: autores, livros and livro_autor. To test the operation of JDBC calls, it was used the ACCESS and MICROSOFT SQL SERVER 7.0 softwares.

The performance evaluation is based on execution time of the purchases agents according to three types of itineraries available in this system. It was evaluated the time that these agents take to arrive the data sources, do the search and send the results to the control module. The results presented at the table 1 were represented in milliseconds.

**Table 1. medium execution time of the purchases agents**

| Quantity of Bookshops | Itinerary 1 | Itinerary 2 | Itinerary 3 |
|---|---|---|---|
| 5 | 988 | 1800 | 110 |
| 10 | 1044 | 1998 | 232 |
| 25 | 3558 | 5282 | 492 |

For a better comprehension of the table , we consider:

- itinerary 1 - one agent for each server;
- itinerary 2 - only one agent that visits all the servers;
- itinerary 3 - one agent that goes through the servers until to find the first occurrence.

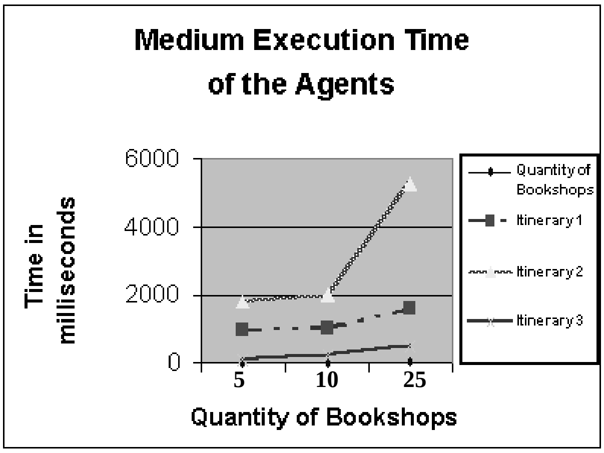

Figure 2 – Medium Execution Time of the Agents

The figure shows that the agent behavior change considerably with the change of the type of itinerary. But the difference of performance (considering the speed of the agents) is more meaning between the itineraries 2 and 3. But considering the results quality of the search realized, we have to accept that the results obtained by the agents with the itineraries 1 and 2 are more complete. Of course that the processing time is different depending on the itinerary used. But, we can conclude that if we have until tem server for the agents visit, the system works well.

We also have to emphasize that with the itinerary 2, the answer time grows quickly when the number of bookshops is great. It happens because, in this case,  the mobile agents have their size added that becomes difficult the migration. Concerning to itinerary 3, it is obvious that this type of itinerary has the minor processing time, because, in this case, the agent returns to the client machine when it finds the first occurrence that satisfies the user.

Finally, analyzing  the executing time and the information received quality, we can notice that the itinerary 1 gives a better answer because we have several purchase agents working together to attend the user interests.

## 5. CONCLUSION

The figure shows that the agent behavior change considerably with the change of the type of itinerary. But the difference of performance (considering the speed of the agents) is more meaning between the itineraries 2 and 3. But considering the results quality of the search realized, we have to accept that the results obtained by the agents with the itineraries 1 and 2 are more complete. Of course that the processing time is different depending on the itinerary used. But, we can conclude that if we have until tem server for the agents visit, the system works well.

We also have to emphasize that with the itinerary 2, the answer time grows quickly when the number of bookshops is great. It happens because, in this case, the mobile agents have their size added that becomes difficult the migration. Concerning to itinerary 3, it is obvious that this type of itinerary has the minor processing time, because, in this case, the agent returns to the client machine when it finds the first occurrence that satisfies the user.

Finally, analyzing the executing time and the information received quality, we can notice that the itinerary 1 gives a better answer because we have several purchase agents working together to attend the user interests.

## REFERENCES

[1] Papaioannou, T. *On the Structuring of Distributed Systems: The Argument for Mobility.* Loughborough University: Doctoral Thesis, February 2000.

[2] Martins, R.M.; Pirmez, L., Carmo, L.F.C.. Applications of Aglet Technology in *Proc. of ICSC'99*, Hong Kong, 1999), 399-408.

[3] Chess, D.; et al. Itinerant Agents for Mobile Computing. *Journal IEEE Personal Communications, Vol.2, Nº 5*, October, 1993.

[4] General Magic Inc. *Mobile Agents*. http://www.genmagic.com

[5] Lange, Danny B.; Oshima, Mitsuru. *Programming and Deploying Java Mobile Agents with Aglets.* (Massachusetts: Addison-Wesley, 1998).

[6] Bredin, J. *Market-based mobile agent planning: a thesis proposal.* http://www.cs.darmouth.edu/~jonathan/agents/proposal/index.html

[7] Klotkin, G.;Rosenberg, J.S. Negotiation and Task Sharing among autonomous agents in cooperative domains. *Proc. of IJCAI´89.* Detroit Michigan, 1989, 912-917.

[8] Klotkin, G.;Rosenberg, J.S. Mechanism for Automated Negotiation in Stated Oriented Domains. *Journal of Artificial Intelligent Research*, 1996, 163-238.

[9] Chong, Chen; Jiwen, Huo; Kai, Bi; Zhongfan, Mai. Mobile Software Agent Model and the Architecture of JMSAS System. *Proc. of MATA'99*. Ottawa, Canada, 1999, 37-52.

[10] Aglets Workbench, by IBM Japan Research Group. http://aglets.trl.ibm.co.jp

[11] Andersen Consulting. *BargainFinder.* http://www.url.http://bf.cstar.ac.com/bf

[12] Orlanta, Do; March, Eric; Rich, Jennifer; Wolff, Tara. *Intelligent Agents & The Internet: Effects On Electronic Commerce and Marketing* http://asterix.ist.utl.upt/massdist/agcompras/referencias.htm

[13] Wurman, P. R., Wellman, M. P., Walsh, W.E. *The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents.* http://auction.eecs.umich.edu/papers.html.

[14] Rodríguez-Aguilar, J. A, Noriega, P., Sierra, C. e Padget, J. *A Java-based Electronic Auction House.* http://www.iiia.csic.es/~sierra/Publications.html.

[15] Chavez, A. , Maes, N. Kasbah: An Agent Marketplace for Buying and Selling Goods, *Proc. of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, Londres, UK, 1996, 75-90.

[16] Moukas, A, Guttman, R., Maes, P. *Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective*.http://ecommerce.media.mit.edu/papers/ker98.pdf

.