A MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL IN DISTRIBUTED AND HETEROGENEOUS DATA SOURCES


University of Brasília – Electrical Engineering Department – Network Engineering Laboratory (UnB/ENE/LabRedes)

ABSTRACT

With the consolidation of the Internet and other computer networks, the need of sophisticated systems to retrieve information increased enormously. In these networks, information is distributed on several machines and sometimes in different formats. In this context, we propose an architecture that besides retrieving information can classify it according to some criteria. To implement the proposed architecture, we used some existing technologies such as mobile agents to transport information between data sources; the Naive Bayes model for classification; the Java language and the XML model for development and storage of the required information.

KEYWORDS

Mobile agents, distributed database, heterogeneous database, classification, aglets, XML.

1. INTRODUCTION

Because there are several technologies for data storage and a broad range of models for information distribution among hosts in a network, we frequently need to integrate information related to some domain, independently of its location and format. This necessity motivated a multiagent architecture to retrieve and classify information based on criteria related to its data source and domain. The proposed architecture is focused mainly on the communications and interactions involving an information management station and the nodes that store the data source. To achieve these functions we use mobile agents, which allow the architecture to be employed in a heterogeneous environment, including fixed and mobile devices such as wireless workstations and PDAs that often use fragile low rate connections. Mobile agents autonomy allow them to operate off-line in such devices, searching in local data sources, and communicating only after finish their tasks or when a connection is available, when they can return to the equipment that initiated the search. A second reason for using mobile agents is related precisely to their mobility, which allows computation to be moved to the data neighborhood instead of the data being moved to a computation facility. This way, only really interesting data will flow through the network, bringing more efficient network utilization. Of course, this requires a methodology for agent creation and destruction avoiding moving agents to generate more traffic than data transfer does. The proposed architecture was implemented in a Java environment and was analyzed using a study case focused on a medical area example.

This paper is organized as follows: section 2 presents the proposed architecture; section 3 presents its implementation; section 4 brings the study case and section 5 presents the conclusion and future works.

2. THE PROPOSED ARCHITECTURE

Figure 1 shows a typical environment where the proposed architecture can be applied. This environment includes a workstation and some machines (sources) that can store relational or object oriented database
management systems or even raw data files. A computer network interconnects these machines, though network topology and dimension are not analyzed in this proposal because these characteristics don’t influence the functionality of the architecture, but only application performance.

The proposed architecture (Figure 2) comprises a colony of agents that cooperate for retrieving information from different sources. Agents in data sources are responsible for keeping information from heterogeneous sources available in a homogeneous way for the navigator agents.

![Diagram](image1)

**Figure 1. Environment with distributed data sources.**

![Diagram](image2)

**Figure 2. The Proposed Architecture**

### 2.1 Processes in the Workstation

The workstation is responsible for the reception of search petitions through its interface, must forward solicitations to search agents, must organize information based on a classification model and must return the result back to the user. All the processes are illustrated in Figure 3.

**Application Interface**

This component receives user solicitations, through a web page or any other application, and forwards them to the controller. Also it sends back to the user answers from the controller.
Controller
The controller manages requisitions, being responsible to create and to control the searching agents, as well as being responsible for the discovery of data sources. The controller is also responsible for the communication with the classification model.

Classification Model
It receives and classifies information according to a specific algorithm within a certain application domain.

Place of Agents
This component is the architecture core, the place where the mobile agents are created, killed and where they cooperate in searching information from the data sources. This component is common to the workstation and data source. The workstation agents are classified in two groups: (i) discovery agents, which browses the network looking for hosts which have data sources to be searched; (ii) search agents, which are responsible for searching the information in the known data sources.

Auxiliary Databases
There are two auxiliary databases: (i) network data source, whose main goal is to reduce response time of information searching in a distributed environment, registering the known data sources, discovered by the discovery agent; (ii) cache: the knowledge database used in the classification model.

2.2 Processes in the Data Source

Data source hosts must receive the search and discovery agents, map requisitions to the data source and return searched information to the workstation, processes that are shown in Figure 4 and described below.
Place of agents
Hosts where the data sources are located must have a local agent, which waits the arrival and the requests of mobile agents and communicates with the middleware to forward these requests. The local agent also receives responses from the middleware and forwards them to the mobile agent.

Middleware
The middleware is responsible for the interface between the mobile agent and the data source, receiving the requests from the agents and forwarding them to the data source wrapper and forwarding the wrapper response back to the agents.

Wrapper
The wrapper makes available a view of a data source. For instance, a wrapper can create an object view for a relational data source or can create a XML view for an object-oriented database. Because it effectively searches information in data sources, the wrapper must know the data source format and how the information is stored in a specific data source. For each data source, the appropriate wrapper must be built or configured.

Data Source
For the purposes of the proposed architecture, data sources can either be a relational database management system, an object-relational database, an object-oriented database or a free-format file, because for each data source there will be always a wrapper to make data available in common predetermined format.

3. IMPLEMENTATION OF THE PROPOSED ARCHITECTURE

This section describes the implementation of the architecture components.

3.1 Processes of the WorkStation

The application interface for our study case was developed with JavaServer Pages (JSP) and servlets. The Controller component is a servlet, which was because it understands HTTP responses and requests. Also this servlet can exchange messages in the framework used for mobile agents implementation. The Classification Model is a Naive Bayes Model of Classification, which is simple and has good performance [16]. In this model, all text attributes (words) are independent, which means that the order of the words is irrelevant. There are two utilization modes for the Naive Bayes Model. In the first mode, the document is represented by a vector that informs if a word exists or not in the text, so that this model is adequate when the amount of attributes is fixed. The second mode uses as additional information the number of times that each word occurs in the document [16] and is used in the proposed architecture because we suppose texts that doesn’t have a fixed amount of words. The classification model stores its knowledge base in XML file format. For the Place of Agents component the IBM Aglet platform was used for the creation of the mobile agent environment, using this framework mobile agents server (Tahiti). Mobile agents were implemented using the Design Pattern Master-Slave [1], in which the master agent creates the slave agents. The master agents are fixed in the workstation, while the slave agents move between the workstation and the data sources. There are two sets of master/slave agents that collaborate for information retrieval and classification: (i) the MasterDiscoveryAgent receives HTTP requests to perform the active data sources Discovery and creates SlaveDiscoveryAgent agents, that effectively navigate in the net searching information; (ii) the MasterSearchAgent receives HTTP requests to retrieve information on distributed sources and creates SlaveSearchAgent agents for the searching.

3.2 Processes in the Data Source

The Agents Environment is the Tahiti Server, that is responsible for the reception of Discovery and Search slaves agents. The Middleware was implemented as a fixed and active agent (DataSourceAgent), which interacts with the SlaveSearchAgent. The Wrapper is a Java program that interacts with a relational database and obtains an object-oriented view from this database, using the hibernate framework [17]. This framework defines a query language (HQL – Hibernate Query Language) and the tools for object-oriented queries. Moreover, this framework features, such as lazy loading, optional use of a proxy and load of various related objects from a unique query, contribute to a high performance solution [17].
4. CASE STUDY

To verify the viability of the proposed architecture an application was developed to get and classify information from the medicine area. In this context, two hospitals named ‘X’ and ‘M’, located in different geographic areas, need to integrate data about patients attendance and classify this information to give support to doctors decisions. The information is stored in MySQL relational databases (Figures 5 and 6).

For our case study we physically connected two dedicated machines (‘sol’ and ‘lua’) using an ethernet network. In ‘lua’ we configured the Tomcat web server, the MySQL relational database and the Tahiti Aglet server. In ‘sol’, we configured the Thaiti Aglet Server and the MySQL relational database. The developed application had two functions: (i) verify the activity of data source, by which the MasterSearchAgent creates slave agents that will search all database sources and verify their activity; (ii) get and classify information, by which the user enters the words for information searching and classification. Then, the Discovery agents can look for texts in the active data sources of ‘sol’ and ‘lua’ machines.

To analyze results found in the searching and classification phases, fictional data was created in hospital ‘M’ and hospital ‘X’ databases. For the classification a knowledge database was created in XML format based on Naive Bayes Classification Model. Despite the usage of fictional data sources and knowledge database we obtained satisfactory results in our testbed.
5. CONCLUSION AND FUTURE WORKS

The presented architecture includes the retrieval and classification processes in a network with a variety of data source formats. But not all the implemented components have to be included on the solution. For example, the classification may not be needed in some situations.

The presented case study involves a set of small data sources, all in a network environment. In this context, a mobile agent model is an appropiated solution. An inconvenience of our implementation is that the Aglet Tahiti server must be in all machines. We haven’t analyzed the performance of the architecture in comparison to other proposals or configurations.

As future implementations we plan to apply our prototype in a larger set of data sources, in heterogeneous network architecture. In this situation security will be an issue to deal. The development of an autonomous discovery agent, capable to know where are the data sources, can also be a future work. Another development in this area is a research work to compare different distributed retrieval architectures, analyzing, for example, performance and network traffic.

REFERENCES