Mobile agent technology for the management of distributed systems – a case study

Claudia Raibulet *, Claudio Demartini

Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy

Abstract

Mobile agent technology has become an alternative approach for the design of distributed systems to the traditional client–server and message-based architectures. The main difference between the two technologies is the key solution they provide for the distribution problems: local interaction and mobile logic versus location transparency. An important example of their application, where the advantages of mobile agent technology are exploited best, is the management of distributed systems. This paper presents an implementation example of the mobile agents used to access and to manage information stored in the distributed database repository (DDR), as specified within the ESPRIT Project “Network Oriented Application Harmonization” (NOAH). © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Mobile agent; Distributed system; Information repository; System technical management

1. Introduction

The increasing development of distributed systems requires more and more efficient technologies to ensure the access and the management of the network resources. Features such as efficiency, flexibility, velocity and security should be related to all the operations performed within distributed systems.

In this context, mobile agent technology is considered an enhancement of distributed technologies as it provides powerful and efficient mechanisms to develop applications for distributed and heterogeneous systems. To deal with the increasing amount of data/resources available nowadays, and the large number of tasks which have to be performed to manipulate the data/resources, mobile agent technology offers the possibility of executing these tasks in an automated way with minimal human intervention. This allows the user to concentrate attention on other activities. The solution proposed by mobile agent technology for the distribution problem includes basic functions such as creation, migration and execution of the mobile agents, as well as specialized functions that involve agent security and management.

Mobile agent technology has become popular primarily because of the efficient way it provides for access and manipulation of remote information. It is based on a different approach for the design of distributed systems [10,12]. The traditional client–server and the message-based architectures treat local and remote resources in the same way, in that they use location transparency. From the user’s point of view all information is stored locally. In contrast to the traditional
technologies, the mobile agent architecture considers that local interaction is more efficient than the remote one. The solution proposed by mobile agents is for software to migrate to remote hosts, where information is stored, so that the user's request can be executed [3]. It means that agents can take decisions or fulfill previously designed operations in an autonomous way on the remote site, even if the home host becomes temporarily unavailable. This feature, called mobile logic, gives flexibility to the design of distributed systems.

Local interaction and mobile logic also lead to the achievement of load balancing in the whole distributed system. The agents move from host to host to execute the user's requests and return the result to the host where they were created. They act autonomously according to the mobile logic they incorporate.

The applications which are best suited to mobile agent technology are electronic commerce [7], telecommunication [1], information retrieval, and management of distributed resources [2,4]. Usually, such applications are characterized by asynchronous transactions, low bandwidth, high latency, remote information retrieval, multi-processing, or distributed task processing features.

This paper describes an example of the application of mobile agents used to access and manage information related to an industrial system. The DDR model [5] represents a distributed information repository designed to ensure a common view of the system for any application. The immediate advantage of the DDR model is a unified perspective for several activities that concern the distributed resources of the industrial system (e.g., installation, maintenance and technical management). The specification of the DDR model has been one of the goals of the ESPRIT Project named “Network Oriented Application Harmonization” (NOAH).

The DDR has been implemented as an object-oriented database in the Java environment. As support for the mobile agents' implementation, the Voyager platform [8,9], developed by ObjectSpace, Inc. has been chosen.

The rest of this paper is organized as follows. Section 2 contains general considerations about mobile agent technology. Section 3 contains a description of the DDR architecture. Section 4 discusses the reasons that led to the usage of mobile agent technology to access and manage DDR distributed information. Section 5 presents aspects related to the implementation approach and conclusions and the further work is considered in Section 6.

2. Mobile agent technology – general considerations

An agent is defined as a self-contained software element that acts autonomously on behalf of a user (e.g., person or organization) [6,12]. Each agent has its own thread of execution, which means that it can perform tasks on its own initiative. A mobile agent has, in addition, the unique ability to migrate from one host in a network to another [6,12].

The definition of a mobile agent contains at least two key words that raise important issues, which must be considered when developing mobile agent systems.

The first key word refers to the autonomy of a mobile agent. This is a feature that allows the agent to act on its own by using the data and the mobile logic which it incorporates, and does not need human intervention or guidance. The user can perform other activities in the meantime. Further, the time needed to fulfill the tasks is reduced because interaction with the user is avoided. The agent is designed to deal with any situation that may occur during execution. There are however cases when the lack of human supervision may lead to unexpected and undesired actions, such as the violation of the private information associated to the agent’s owner [6]. The relationship between the agent and the owner should be based on trust because the agent has access to information stored in the user’s profile; information that may be private. Due to the fact that an agent can collect, manipulate, and distribute data, including user's personal data, it is possible that there is a violation of privacy when the agent communicates and/or exchanges data with its environment and/or agents belonging to different systems. In order to defend the user's privacy, privacy-enhancing technologies (PETS) (e.g.,
Identity Protector) [11] should be added or integrated in mobile agent systems.

The second key word in the definition refers to the mobility of an agent. The mobility feature enables the agent to travel to the host where the data is physically stored. In this way the transfer of a large amount of data in the network is prevented. The agent accesses the data locally and so provides only the results to the user. There are at least two advantages to this approach: low traffic in the network and a better use of network resources. The migration of an agent within the network should respect rules concerning the agent transport, agent interaction and security, otherwise it is not possible to ensure interoperability among mobile agents which belong to different vendors’ systems.

Security has always been considered to be the weak point of mobile agent technology. The security problem regards the security of the host that may receive a potentially malicious agent, as well as the security of the agent that should be protected from potentially malicious hosts. Closely related to this aspect, is legal responsibility which concerns authentication of mobile agents, secrecy, privacy, data quality, and many other similar aspects. Questions such as: “Is the agent really what it pretends to be?” or “Has it the right to access this kind of data?” should have a clear response because otherwise the host will not know how to react to the agents’ requests.

3. The DDR model

3.1. Scope

The main goal of the DDR is to provide a general information model that can describe any distributed system during the different stages of its life cycle. The DDR specification unifies the actual representation of the devices within complex systems, enabling heterogeneous devices and their links to be described according to a common approach, regardless of their specific implementation. The advantage of this approach is that the model provides a common view for the installation, the maintenance, and the management of distributed resources in the system.

STEP/ISO 10303 AP 212 [14,15] has been chosen for the specification of the DDR model because it includes all the concepts necessary to describe a plant (e.g., devices, interfaces, networks, functions, variables, etc.). The objective of the AP 212 is to specify the tasks of the operational activities of each plant and the relationships between them. It analyses the data used by individual activities and/or shared by several activities. Based on the previous affirmations, attention is directed to the abstraction of generic activities and data structures that provide the basis on which efficient activity models and data models can be built up. The AP 212 is used not only to create data models to represent and to exchange data, but also to build distributed data repositories.

3.2. The DDR architecture

Within the DDR model architecture, four distinct components are identified:
- DDR System Manager;
- DDR Database;
- DDR Plant Manager;
- Management Information Base (MIB).

As shown in Fig. 1, the distributed system is made of one or more plants that are connected to the DDR System Manager. All the plants in the network are also directly connected to each other.

The DDR System Manager contains general information related to the whole distributed system, such as a complete directory of the system’s resources or the network wide topology information. The detailed information associated to each plant is stored locally in the DDR Databases and the MIBs. This repartition of the information has at least two advantages. The first is that the System Manager does not have to store a huge amount of information that can over time get proportionally bigger together with the development of the distributed system. The second refers to the velocity of information updates or changes that may be carried out over time for the plants.

The DDR Database contains information related only to the local plant, such as a complete directory of the devices connected to the plant or general data that provides a basic description and the basic features of the devices. Every device in the
distributed system has a local MIB associated with it. The MIB contains all the detailed information that concerns the respective device (e.g., effective values of the device parameters, device status, etc.).

The DDR System Manager receives the user's requests from the System Management Application. Depending on the scope of the request, the System Manager creates an agent to execute the query. The agent will be sent to one or more DDR Plant Managers. The Plant Manager first interrogates the DDR Database and only if the information stored here is not enough to perform the query will it also interrogate the MIBs. The result is then sent to the System Manager.

4. Mobile agents for the management of the DDR information

Two main categories of management operations have been identified among the user requirements: the read management operations and the set management operations. The first category includes device identification, checks of the parameter values, and monitoring of the device status, while the second refers to the set or modification of device parameters, and the set of functional configuration for the devices.

4.1. Three types of queries for the management of the DDR

This section discusses three examples of queries used for the management of DDR information:

1. List all the type T devices of the industrial plant that have parameter X set on the Y value.
   This type of query implies that every plant in the system that has at least one device of type T connected to its local network will be interrogated. A query has to be sent to each plant in order to read parameter X for the type T devices, and to check its value.

   In a client–server approach, the DDR System Manager creates a number of threads equal to the
number of plants that contain at least one device of type \( T \), each thread having the task of performing the query for one particular plant. After receiving all the partial results from each plant, the System Manager puts them together and sends the final result to the System Management Application. An alternative client–server-based solution may create only one thread in the System Manager that will sequentially interrogate all the plants and then return the result to the user. The first solution performs the query in a minimum time, but introduces a computing overhead in the System Manager application. The System Manager has to handle all the requests and answers sent and received to/from the plants, and this means a high load for its resources and a decrease in its performances. The second solution generates only one thread within the System Manager application, but it will need significantly more time to perform the query. Both solutions generate the same number of request-answer messages to execute the query.

A solution based on mobile agent technology implies the creation of an agent designed to migrate from plant to plant, and to read and check the parameter \( X \) value for the type \( T \) devices. The System Manager is responsible for creating the mobile agent and for receiving the result of the query provided by the agent. Finally, it forwards the result to the user. The mobile agent incorporates the logic necessary to execute the query, as well as the logic to migrate from host to host. It has an autonomous behavior that allows it to act asynchronously to the System Manager.

The mobile agent based solution achieves load balancing in the whole system by the agent migration. It improves the time taken to perform the query by using local interaction. The traffic in the network is due only to the migration of the agent from host to host (compared to the \( 2 \times \text{number_of_plants} \) messages exchanged in a client–server architecture).

(2) Monitor the value of parameter \( X \) belonging to the \( Y \) device every \( N \) s. Do operation \( Z \) and notify the DDR System Manager if the value is out of a certain range. Such a query implies reading the parameter value in a local MIB that is associated with the above mentioned device, and then validating this value.

A client–server solution creates, within the System Manager application, a separate thread that every \( N \) s sends an interrogation message to the local MIB. The System Manager verifies the returned parameter value and decides if it is valid or not. In the case of a fault, it sends a message to the local MIB to execute operation \( Z \), and a notification message to the user.

A mobile agent solution sends an agent designed to perform this query to the plant where the device is connected. The agent reads and validates the parameter value every \( N \) s, using local interaction. In the case of a fault, the agent executes operation \( Z \) and then sends a notification message to the System Manager that will forward it to the user.

The client–server solution uses remote interaction, which, in this case, has two important disadvantages: it increases the traffic in the network and introduces an overhead in the execution of the query. The mobile agent approach eliminates these two disadvantages by using local interaction. Further, in the case of a fault, the mobile agent immediately executes operation \( Z \) at the device side. Time may play an crucial role in a high-traffic network within real-time industrial systems.

(3) A new resource \( X \) has been added to the industrial plant. Update the related information in the DDR Database and the DDR System Manager. As mentioned before, the DDR System Manager stores information related to the entire industrial plant, such as the network topology related data. When adding a new resource in the network, the DDR System Manager and the DDR Database data should be updated.

The DDR implementation requires the newly added device to signal its presence in the system. A client–server approach solves this problem by using a pre-defined protocol established between the newly added device and the DDR System Manager or the DDR Database. There are cases in which this protocol must be changed or personalized for a specific device due to the new or changed features of the device. In a client–server architecture, such a protocol would need complex modifications that involve upgrading of the software in the whole system (e.g., the DDR System Manager application, as well as DDR Databases application) and this may be costly or inefficient.
A mobile agent based solution sends an agent to the DDR Database and to the DDR System Manager to add the information related to its presence in the system. The mobile logic of the agent can be changed or personalized for each new added resource, without modifying any of the other applications in the system. Mobile logic gives an important degree of flexibility to the design of distributed systems. Furthermore, the scalability of the system is implicitly ensured.

This type of query can also be used in a situation when a device becomes unavailable due to a temporary or a permanent fault. The direct advantage in such cases is that any fault can be immediately identified and fixed; an aspect that is very important in industrial systems.

4.2. Four reasons to use mobile agents for the management of distributed information

From the discussion presented above it follows that, there is nothing in the management queries that cannot be solved by the client–server and the message-based architectures. Although, it suggests that in the case of the management of distributed resources, mobile agent technology provides a more efficient solution then the client–server based approach. This is due to the way in which distribution problems are treated. Traditional technologies achieve communication between distributed resources by using location transparency, while mobile agents provide local interaction for communication and mobile logic facilities. The mobile agents migrate to remote hosts and perform different operations according to the mobile logic they incorporate. They use local interaction to access the distributed information because they move to where the data is stored.

Four important reasons that sustain the use of mobile agents for the management of distributed resources can be identified:

1. mobile logic and local interaction: in order to execute the user’s requests, the mobile agents move to the remote hosts where data is stored, and access and manipulate the data using local interaction; the mobile agents contain the logic necessary to perform the user’s requests autonomously from the host where they were created;

2. load balancing: mobile agent technology ensures a better and more balanced use of the network’s resources through the migration of the agents; in the DDR model, the computational tasks are performed not only by the DDR System Manager, but also by the DDR Plant Managers;

3. low traffic in the network: the management of distributed systems also assumes the monitoring of different resources; this kind of operation may increase the traffic in the network; the mobile logic feature offers the possibility to monitor distributed resources by local interaction, ensuring minimum traffic in the network;

4. flexibility: mobile agent technology leads to the design of flexible applications. The modification of a mobile agent can be carried out without any other changes being made to the software within the system; a direct advantage of the use of mobile agents is that they improve the scalability of the distributed applications: for example, a new kind of resource may be added in the network by designing a mobile agent that will append the necessary information in the System Manager or Databases, without any other modification of the software in the system being made.

5. Implementation of the mobile agents using the Voyager platform

The implementation of the mobile agents must be supported by a platform that provides the mechanisms and the functionality necessary for their proper creation and execution in distributed environments. In November 1997, the object management group (OMG) [18] specified within its first mobile agent standard named Mobile Agent System Interoperability Facilities (MASIF) [6], the requirements that have to be supplied by any mobile agent platform. These requirements are related to the support for execution, management, mobility, security, communication, transaction and unique identification of agents. Platforms for the implementation of mobile agents include Voyager (ObjectSpace) [8], Aglets Workbench (IBM) [16], Grasshopper (IKV+) [17], and Telescript (General Magic) [13].
The Voyager platform has been chosen for the implementation of the DDR. Based on the most common available industry standards, its scope is to unify and to simplify the development of distributed applications. Voyager is a pure Java distributed platform that provides a complex and competitive state-of-the-art object request broker (ORB) [9] in order to support the implementation of distributed applications. The list of the ORB features includes dynamic class loading, remote construction, distributed garbage collection, dynamic aggregation, mobility, autonomous mobile agents, and many others.

5.1. Mobile agents in the DDR model

In the DDR model, all the computers connected in the system must have the Java Development Kit and the Voyager platform installed in order to support the creation, migration, execution, and destruction of the mobile agents. Usually, the agents are created by the DDR System Manager on receiving a user’s requests. The agents are then sent to perform the queries and to provide the results to the System Manager that will then forward them to the user. Mobile agents are also created within the local plants when a new resource is added. This kind of agent is designed to add the information related to the new resource in the local DDR Database and in the DDR System Manager. The mobile logic of an agent contains the information necessary for its migration, execution, and destruction.

5.2. Creating a mobile agent

Within the Voyager platform, a mobile agent is defined as a common, simple class that implements the Serializable interface. An object of this class becomes a mobile agent at runtime when a secondary object, called an agent facet, is attached to it. The agent facet enables the primary object to use the methods defined within the IAgent interface and to exploit the mobility facility provided by the Voyager platform. The aggregate object is then treated as a single unit. The facility of adding new code and data to an object at runtime is called dynamic aggregation and it is considered to be an important enhancement for the object modeling that complements the inheritance and polymorphism mechanisms.

5.3. Moving a mobile agent

The IAgent interface provides a moveTo method that implements the support for the migration of the mobile agents. This method has three parameters: the location to which the agent will move, the name of a method implemented by the agent that will be executed when the agent reaches the new location, and the parameters for the agent method, if any. If the agent has to migrate to another location after finishing its mission on a host, the last instruction executed on the present location will contain the moveTo method.

5.4. Executing a mobile agent

On reaching a new host, the mobile agent starts its execution with the method specified in the parameters of the moveTo instruction. The agent continues its execution in order to perform the query for which it has been designed, and to obtain the requested results.

5.5. Destroying a mobile agent

By default, the mobile agents created within the Voyager platform are autonomous. This means that a mobile agent cannot be garbage collected even if there are no more local or remote references to it. The IAgent interface provides a method called setAutonomous that sets the autonomy of an agent to true or false. Usually, after achieving its goal, the mobile agent sets its autonomy to false in order to specify that it has finished its job and it can be garbage collected.

6. Conclusions and further work

This paper has presented a case study of mobile agents’ usage for the management of the DDR distributed resources. The DDR model validation has been demonstrated with an example of an actual system: a Temperature Loop in the IAM-Pilot laboratory at ENEL in Milan [5].
Mobile agent technology should be seen as an alternative approach to the client–server traditional architectures, not as a universal solution for distributed systems. In the case of the management of distributed resources, a comparison between a client–server solution and a mobile agent based approach shows that mobile agent technology offers important advantages, such as flexibility and scalability of the system, load balancing, on-demand services, low traffic in the network, and many others. These advantages are due to the way in which mobile agents treat distribution problems by using local interaction and mobile logic.

The mobile agents require a proper and secure environment for their implementation and execution. For the implementation of the mobile agents, the Java environment and the Voyager platform have been chosen. Java is considered the de facto standard for mobile agents, while Voyager provides a complex platform that supports the development of distributed applications and the implementation of the mobile agents is implicit.

Further work will be related to the security of the mobile agents and of the hosts that receive them in public networks. The mobile agents should be protected against potentially malicious hosts. The hosts should also be protected against malicious actions that may be performed by the mobile code they receive and execute.

References