

Agent Migration over FIPA ACL Messages

Joan Ametller, Sergi Robles, and Joan Borrell

Computer Science Dept. Universitat Autònoma de Barcelona
08193 Bellaterra, Spain Joan.Ametller@uab.es

Abstract. In this paper, we present the design and implementation of an inter-agency mobility (migration) mechanism for agents. This migration is based on FIPA ACL messages. We also evaluate the performance of this implementation. Agent mobility is an essential requirement for some electronic commerce applications, such as those in the Sea-of-Data (SOD) family. The majority of agent systems at present do not support mobility or do not respect standards. This situation does not encourage inter-operability. Our implementation provides a Mobile Agent System that is compatible with the FIPA standard, and SOD applications can be implemented using it. Our solution has been confirmed as feasible by performance evaluation, even in situations of extreme agent load. Already existing applications may make use of our idea to increase their functionality and flexibility, as our results have been integrated in the well-known JADE agent platform.

1 Introduction

Sea-of-Data (SOD) applications are those in which the user is interested in the results of processing large amounts of data in several distributed locations. These data may not leave their location for a number of reasons, such as legal requirements, bandwidth limitation, or restrictions due to the acquisition process (the data may be acquired on demand). The user's connection may be intermittent, subject to sudden cuts or irregular, either because data processing requires a lot of time or because the user is connected via an unstable network, or from a simple device in a pervasive computing environment.

These applications are extremely useful in areas such as intrusion detection systems, medical image processing or satellite image analysis. They also provide new perspectives for electronic commerce, as this data processing may be sold to clients as an indirect service.

Because of the characteristics and restrictions of SOD applications, it is difficult, or almost impossible, to implement them using traditional technologies. Mobile agent systems appear to be the most feasible solution for their implementation. In these systems, autonomous software entities (agents) may move across a network of execution platforms (agencies). The applications code can thereby be executed wherever the data are located, without the need for centralizing all the processing. Furthermore, the user, once the mobile agent has been sent, can remain disconnected. The use of agents also enables the execution of the application to be parallelized.

There are two basic requirements for developing electronic commerce applications in the Sea-of-Data field. The first essential requirement is an agent system that is interoperable with other agent systems, or the equivalent, an agent system that meets FIPA specifications.

FIPA (the Foundation for Intelligent Physical Agents) [4] has so far been the main body that has promoted interoperability between agents. It provides the specifications for a standard language for agents (*Agent Communication Language - ACL*), ontologies structuring the semantic contents of these messages, and an abstract agency model.

Of the existing agent systems (for a complete review, see for example the doctoral thesis [10]), one of the few that meets FIPA specifications is JADE (Java Agent DEvelopment Framework) [8]. It is also easily adaptable to the needs of SOD applications because it is open source. JADE is currently the agent system used as a basic platform by most groups carrying out research and development in the field of agents [9].

The second essential requirement, obviously, is to provide the chosen agent system with mobility, in order to have an interoperable mobile agent system with open source. This second requirement is not an immediate task.

Firstly, the FIPA specifications have not gone into sufficient depth in the field of mobile agents. The only initiative has been to propose a specification for agent mobility of an optional nature and with little precision [4]. This specification has become obsolete because it has not been implemented by any agent system. Secondly, previous experiences, such as the development of the MARISM-A extensions [3] for the JADE platform, show that there are various possible ways to provide agents with mobility (based on sockets, Remote Method Invocation, etc.), but few of them meet the basic FIPA interoperability specifications.

In the case of JADE, this platform includes a mechanism for transporting agents internally within a single agency by means of introducing the concept of distributed agent containers. This approach, as analyzed in this article, is not valid for constructing a mobile agent system between different agencies. It has several disadvantages in terms of fault tolerance and system scalability.

In this article, we present a proposal for providing inter-agency mobility (migration) to the JADE platform based on the ACL language, the basic mechanism for communication between FIPA agents. This proposal has been let known to the managers at JADE and the FIPA technical committee. This proposal will enable us to have an open system for mobile agents, in which a number of applications can be implemented, including Sea-of-Data and electronic commerce related.

The second and third sections concentrate on presenting the main characteristics of FIPA and JADE, respectively. In the fourth section we describe our proposal for agent migration based on ACL. The fifth section includes the evaluation of our proposal's performance. Finally, the sixth section is given over to conclusions and suggestions for continuing our work.

2 FIPA

FIPA is a non-profit making organization that was established in 1996 with the aim of promoting specifications for facilitating interoperability between agent systems. The model proposed by FIPA consists of concrete specifications enabling interoperability, based on a high-level abstract architecture.

FIPA's abstract architecture [6] defines all components that should be found in an agent platform as high-level entities. Some of these components are optional, and others

are compulsory. The FIPA specifications are a concrete production of this architecture and are basically focused on two points.

Firstly, the basic elements which the platform should consist of are defined, as well as some aspects of internal high-level functioning such as the states that form the life cycle of the agents. Three components are defined - AMS, DF and ACC. AMS (Agent Management System) is an agent which any new agent in the platform must locate in order to send it a registration request when its execution starts. This agent is mandatory in all FIPA platforms. This agent thereby maintains a register of descriptions of all agents in the platform that may be consulted by any agent, thus providing a directory service. The DF (Directory Facilitator) is an optional component which may or may not be represented by an agent, and provide agents with a Yellow Pages service. The ACC (Agent Communication Channel) is a high-level interface, through which messages are sent from one agency to another, using an MTP (Message Transport Protocol). Figure 1 shows the location of these components within FIPA agencies.

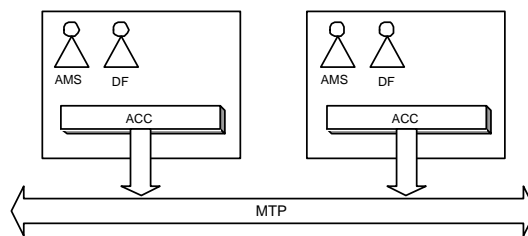


Fig. 1. FIPA Agency Model

Secondly, the specification gives minute details of the entire process and characteristics involved in the sending and structure of messages between agents (ACL). This part of the specification describes the common language used between the agents and the way concepts are represented in the messages. It also specifies the functioning of the transport system that agencies must use to send messages to other agencies. This is the equivalent of saying that an agent is capable of sending a message to another agent belonging to a different platform and that agencies are capable not only of establishing communication, but also of continuing the protocol started according to the content of the message. The level of understanding between the two agents will be greater or lesser depending on the knowledge that they have of the concepts sent in the message, but there is always a minimum level of comprehension that means that the conversation is not blocked by an unusual message.

Platforms meeting FIPA specifications are known as FIPA-compliant. There is a European project called *AgentCities* [1] which is an initiative for creating a network of FIPA-compliant platforms. Every platform offers services available to everybody thanks to the interoperability provided by FIPA.

Agent migration capacity has not been dealt with in detail by FIPA. This is because while on the one hand it is feasible to standardize a message transport layer, on the

other it is not so feasible to standardize a migration between heterogeneous systems. In this last case, the system should enable agents to migrate and continue with their execution within an environment for which they have not been designed. If we also consider that the platforms may be written in different programming languages, and the security mechanism problems posed by executing remote code in a machine, the complexity of a specification of this type grows exponentially. However, some time ago a specification for migration appeared that gave details of how the migration process should be carried out, sending the code of agents within ACL messages. This specification was not completed and became obsolete because it was never implemented.

3 The JADE platform

The platform chosen for implementing migration was JADE, because it is a widely adopted platform within the software agent development and research communities. It is open source and complies with FIPA specifications. This platform facilitates its agents' mobility, but as shown below, does not meet the requirements for a real migration.

The platform is divided into a large number of functional modules, which can be placed into three categories in general terms:

Core. The core of the platform is formed by all components providing the necessary execution environment for agents' functioning.

Ontologies and Content Languages administration. This consists of the agency's mechanisms for carrying out information processing in ACL messages, and the internal structures that the agency and agents will use to represent this content.

Message transport mechanisms. Mechanisms and protocols used to send and receive messages at both intra-agency and inter-agency level.

At the core of the JADE platform is the concept of the container, which is the minimum execution environment necessary for an agent to operate. Each container in JADE is executed in a different Java virtual machine, but they are all interconnected by RMI (Remote Method Invocation).

Containers do not only enable groups of agents to be separated into different execution groups, but agencies may also be distributed in various machines so that each has one or several of them. One of the different existing containers is the principal, which represents the agent itself and which gives orders to all the others. JADE also provides mobility between containers. For this reason, if the agency is distributed in various machines, agents can move between them. However, accepting this type of mobility as migration could be considered a mistake. "Satellite" containers are highly dependent on the principal and many operations carried out by the agents within them end up passing through the central node. Furthermore, the connections between them (carried out by RMI) must be permanent, as if not, many errors due to the loss of link may be generated. As we can see, using this type of mobility as a typical migration ends up making mobile agent systems' scalability disappear because a certain type of operations is centralized in a single node. However, it may be very useful to use the diagram of containers to distribute the processing of agencies that have to bear a heavy load or to isolate some agency types within a single agency for security reasons.

These details lead to the necessity for inter-agency migration, which is carried out through a non-permanent channel and makes a system of mobile agents available that is much more scalable, and in which agencies are totally independent units. This independence is not only desirable from the point of view of fault tolerance, but also because of privacy.

4 Our proposal for migration using ACL

The idea of creating a migration using ACL messages came from FIPA's specification regarding mobility, where this type of migration is proposed. However, as mentioned above, this specification only gives a general outline of the ontologies, the protocol, and the life cycle of a mobile agent. It has not been updated due to the lack of implementations and has become obsolete within the FIPA specifications. For these reasons, we have found that there is a need to propose extensions to the specification to cover situations that it does not deal with. These extensions have been approved by the managers at JADE and the FIPA technical committee.

The design for a migration using ACL means that transmission of mobile agents between two agencies will be carried out using the message system between agents. In other words, the agent (both the code forming it as well as the state that it is in) will travel as the content of a message between two agents. Specifically, the agent will travel in the message between the AMS agents of each of the agencies involved in a migration.

Because the agencies have mechanisms for sending and receiving messages, using a parallel transmission protocol is not necessary. This is an advantage in interoperability terms and enables agents to be transmitted using the various message transport protocols (HTTP, IIOP, SMTP, etc). Furthermore, this is achieved in a totally transparent way.

The first logical step in this process is to design the ontology and the protocol that will be used in the exchange of messages between the two agencies. This protocol has the movement of the agent as its final purpose. Defining an ontology basically consists of defining the elements that will form the content of an ACL message to give a common interface between the two parties when extracting the information of the message.

The two possible migration models that are proposed in the initial FIPA specification deal with one migration directed by the agent and another directed by the agency. In our implementation, we have decided to adopt the migration directed by the agency, which is more robust, as it enables the agency to decide which migration requests are accepted and which are not.

The ontology initially specified by FIPA is made up of seven elements: five concepts ("mobile-agent-description", "mobile-agent-profile", "mobile-agent-system", "mobile-agent-language", and "mobile-agent-os") and two actions ("Move" and "Transfer").

Of these concepts, we only use the first one, "mobile-agent-description". This is because it is very difficult to develop systems that enable agents with different architectures to migrate with total interoperability, at the current level of agent technological maturity. These agents could have been written in different languages or executed in different operating systems. For this reason, these concepts are never used, assuming that mobile agents which migrate move between the same agencies. Obviously, if the agency has been developed in a language like Java, a migration between agencies lodged

in different operating systems is possible. However, this is a characteristic of this language which is transparent to the agency, and therefore does not involve the need to use the concept “mobile-agent-os”, for example. In any case, although we do not use it, we maintain these concepts to ensure compatibility in case it is possible to make agents migrate between agencies implemented in a different way or with different languages.

The concept “mobile-agent-description”, on the other hand, is highly useful to us. Within it are several fields that define the characteristics of the mobile agent in question. Among others, these include the characterization of the code and its data.

Of the two actions specified, we have only implemented “Move”, for migrations directed by the agency. We do not take the “Transfer” action into consideration, which can be used for migrations directed by the agent, although it is supervised by the agency.

Once the content of the ACL messages was described, we moved on to enumerating the protocol by which the migration process is carried out. A diagram of the messages exchanged during the migration process can be seen in figure 2.

- Firstly, the agent wishing to migrate starts a conversation with the AMS agent of the local platform to make a request for migration. This request is the first step in the standard FIPA-Request protocol and consists of a Request-type message with a Move action and a *MobileAgentDescription* (*MobileAgentDescription* is the name of the class that implements the concept of “mobile-agent-description”), in which the code and data fields are empty.
- When the AMS agent receives the request for migration from a mobile agent, the first thing it does is to decide whether to accept it or not according to a given criterion. If the migration is accepted, the AMS agent sends an *Agree* message to the agent, or if not, a *Refuse* message.
- If the migration is accepted, the first thing that the AMS agent does is to obtain the code and data (serialized instance) of the agent that made the request and fills in the code and data fields of the *MobileAgentDescription*.
- The next step is to make contact with the AMS belonging to the agency to which the mobile agent wishes to migrate. To this end, the local AMS must start a parallel conversation to that between the agent and the remote AMS.
- When the remote AMS receives the request, the agent’s code and data travel within the *MobileAgentDescription* that the local AMS has prepared.
- Following its own criteria, the remote platform decides whether to accept the incoming agent. If so, it responds with a *Agree* message, and if the agent does not meet the requirements specified by the agency to execute it, it responds with a *Refuse* message.
- The remote AMS loads the agent’s class, deserializes its instance and restores its execution. Once this entire process has been successfully completed, the standard FIPA-Request protocol is completed by sending an *Inform* message to the local AMS.
- The final step in the protocol consists of informing the agent that started the process. If the process has been successfully completed, the original agent is destroyed.

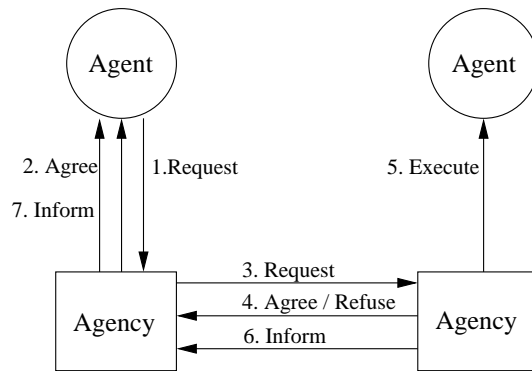


Fig. 2. Exchange of ACL messages in the migration protocol

4.1 Design components

After describing the ontology and the protocol used, the components necessary to carry out the entire process are listed below.

Migration Manager The Migration Manager component is responsible for carrying out all operations enabling the execution of an agent to be suspended in the local agency. Also, it enables its state and code to be processed. These operations allow to insert the agent in an ACL message. In the remote agency, it is responsible for undoing the messages that contain agents, decoding them and acting together with the agency to restore the agent's execution.

Class Loader The class loader is the component that constructs the representation of the class in the memory so that it can be used in the code. The bytecode of the class is extracted from the ACL message and loaded during the deserialization time of an incoming agent. At the same time, each classloader provides a space of different names from the others, so that two agents created with one class with the same name do not conflict within a single platform if they are loaded by different classloaders.

Mobile Agent We have created the *MobileAgent* class, inheriting the properties of a basic agent and adding the functionality of being able to migrate to it. This functionality provides it with the *doMigrate(dest)* method, which starts the migration protocol when invoked.

Conversation Modules These modules were implemented using the behaviours of the JADE model [2]. Behaviours represent agent tasks and are also a useful tool for designing the protocols that govern conversations using ACL. There are basically two components of this type developed to provide mobility. The first is the agent's behavior. This component is launched when the function *doMigrate(dest)* is invoked and is responsible for supervising the migration from the agent's point of view. The second was implemented within the AMS agent to help it administer its part of the migration protocol. This behavior has a double functionality as it was designed for playing the roles of the local AMS and the remote AMS. The most complex part

of the implementation of this component is its functioning as a local AMS: parallel conversations (AMS-Agent and AMS-AMS) which depend on each other must be taken into consideration.

5 Performance evaluation

The objective of performance evaluation is to confirm how the migration System implemented influences the normal functioning of an agency. The migration process involves many processes such as the serialization of agents, their encoding to Base64, the sending of large ACL messages, and dynamic class loading. The aim is to determine whether the implementation of the migration is feasible or whether it involves an acceptable decline in the performance of the agency.

We have seen how our proposal for migration sent agents within ACL messages. The JADE platform is able to support the massive sending and receiving of messages. The main difference between a migration of this type and sending inter-agency messages lies in two areas. Firstly, the size of an ACL message containing an agent's code and data is in general significantly greater than the average size of a normal ACL message. Secondly, the pre-process involved in extracting the agents from the ACL message, as well as that of creating the message based on the agent, makes the typical treatment of a message somewhat complicated.

The trials carried out took place on machines based on Pentium IV at 2GHz with 256 Mb of RAM, a Java 2 virtual machine, and a VLAN within an Ethernet network switched to 100 Mbps, in stable and low load situations. The version of the JADE platform installed in each was 2.61.

The aim of the trials was to check the processing behavior of overloaded agencies faced with a significant number of simultaneous migrations. An agent with the test process using almost the entire computing capacity was used, to obtain the maximum decline in the agency's performance. Once the time taken for this process in a totally unloaded agency was calculated (see Row 1 of Table 1), this time was calculated again, but submitting the agency firstly to hundreds of simultaneous migrations from incoming agents, and then to hundreds of migration requests from outgoing agents. The aim of these two trials was to confirm the performance in the two facets of migration - the sending and reception of agents. Finally, the process was repeated but loading the same amount of static agents. This enabled us to compare the load of the agents with migration. The trials were carried out five times for each experiment. Table 1 shows the results for 300 agents, sized from 3,4 to 15,2 Kbytes. The processing of the line of messages in JADE, and therefore the requests for migration, was carried out using a limited number of execution threads, meaning that the processing of the sending and reception of messages was treated in an almost sequential way. The 300 migrations evaluated represent an overload situation in the transport system during a significant time period with regard to the duration of the test process.

The results in table 1 show how the migrations carried out decrease the efficiency of the test process by an average of 15 seconds, but it can also be seen how the agents' temporary load weight (see Row 4 of Table 1) is similar to that of the migration (see Rows 2 and 3 of Table 1).

| Test/time (seconds) | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 |
|------------------------------|--------|--------|--------|--------|--------|
| Test Process (TP) | 211 s | 209 s | 212 s | 212 s | 209 s |
| TP + 300 incoming migrations | 227 s | 228 s | 226 s | 228 s | 226 s |
| TP + 300 outgoing migrations | 225 s | 227 s | 227 s | 227 s | 226 s |
| TP + 300 loads | 225 s | 227 s | 226 s | 225 s | 225 s |

Table 1. Performance evaluation

The conclusions to which the results obtained lead us are as follows. Firstly, the decline in the system’s capacities during periods in which there is a massive and constant migration flow is not unacceptable, as this flow does not paralyze the system or markedly affect the processes carried out in the agency. Secondly, if the duration times for the test process during the migrations and during agent loading are compared, it can be deduced that most of the resources used during the migration process are due to the classloading of the agents. Classloading is not the typical migration process as although it is used in it, it also takes place when new agents are input into the agency. From this we can deduce that the procedures that are exclusive to migration (negotiation, serialization, etc.) are a load that is hardly significant.

6 Conclusions

In this paper, we have presented a proposal for the mobility of agents between various agencies, based on the agent communication language (ACL) proposed by FIPA. This has been the key to fulfilling the double objective of maintaining consistency with the generally accepted agency model and also permitting interoperability between platforms of a different type.

The implementation has been carried out as an extension of JADE, the most commonly used agent platform at present. Those responsible for the development of this platform are aware of our implementation and it could be considered to be included in future versions. During the design of the mechanism, a subsequent extension in order to guarantee the security of information has been taken into consideration. The flexibility it provides enables it to be rapidly adapted for using encoding and authentication schemes, as anticipated by the FIPA technical committee for security.

After carrying out various experiments, we have confirmed that the system’s performance does not decline even when it has to administer hundreds of migrations while other agents are executed. Our migration proposal is therefore feasible for both large and small systems. Optimizing the mechanism and the natural extension of the agency model also mean it can be used in small mobile devices, such as the PDA.

There are basically three areas in which this work may be continued. Firstly, the scheme could be expanded, by adding privacy and authenticity protection mechanisms, considering the security aspects that will arise in new ACL specifications. Secondly, to improve overall performance, the Java threads planner could be modified, to include a reactive architecture [7]. This would make the system suitable for processing thousands and tens of thousands of agents. Finally, another of these lines could be the development of applications using our scheme.

7 Acknowledgements

This work has been partially funded by the Spanish Gov. Commission CICYT (TIC2000-0232-P4), and Catalan Gov. Department DURSI (2001SGR 00219).

References

- [1] AgentCities.NET. European Commission funded 5th Framework IST project. November 2001. <http://www.agentcities.net>
- [2] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa. JADE Programmers Guide. July 2002.
- [3] CCD Research Group: MARISM-A, An Architecture for Mobile Agents with Recursive Itinerary and Secure Migration. <http://www.marism-a.org> (2003)
- [4] FIPA Agent Management Support for Mobility Specification. Foundation for Intelligent Physical Agents. Geneva - Switzerland 2000.
- [5] FIPA. Foundation for Intelligent Physical Agents. <http://www.fipa.org>
- [6] FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents. Geneva - Switzerland 2002. <http://www.fipa.org>
- [7] L. Hazard, J.F. Susini, F. Boussinot. "Junior Reactive Kernel". Inria Research Report 3732, July 1999.
- [8] JADE, Java Agent DEvelopment Framework. <http://jade.cse.lt.it>.
- [9] S. Poslad, S. Willmott. Agent Engineering for Open Systems. 5th European Agent Systems Spring School. Universidad Aut3noma de Barcelona - February 2003
- [10] S. Robles. Mobile Agent Systems and Trust, a Combined View Toward Secure Sea-Of-Data Applications. Phdthesis, Universitat Aut3noma de Barcelona, 2002.