A Generic Infrastructure for Object Streaming And Mobile Agents on ProActive

DAVID PINTO, ADRIANA BERISTAIN & MARGARITA MARQUEZ
Faculty of Computer Science
Benemérita Universidad Autónoma de Puebla
14 Sur & Blvd.Valsequillo CU
PUEBLA, PUE., MEXICO
{dpinto, beristain, mtmargarita}@cs.buap.mx, http://proactive.cs.buap.mx

Abstract: - We propose a parallel and distributed component framework for building applications that use mobile agents and object streaming. This generic infrastructure is based on ProActive, a middleware (programming model and environment) for object-oriented parallel, mobile, and distributed computing. We have extended ProActive by implementing a hierarchical and dynamic component model for both cases. Future applications will be able to take advantage of this generic infrastructure in order to develop complex applications based on ProActive, and that use mobile agents and/or object streaming.

Key Words: - Parallel Computing, Distributed Computing, Mobile Agents, Object Streaming.

1 Introduction

The objective of this research is to propose new architecture for the automation, identification and integration of reusable software components, and for mobile agents and object streaming in particular.

Designing generic infrastructures is a very widely studied research topic in the computer science community. For instance, Concordia [1], a new framework for developing and executing mobile agents, offers a full-feature middleware infrastructure for the development and management of network-efficient mobile agent application for accessing information anytime, anywhere, and on both wire-based and wireless devices. Concordia has been implemented in Java to ensure unimpeded interoperability and platform independence among developed agent applications. Another approach for the infrastructure of mobile agents is given in [2], an HTTP-based infrastructure for mobile agents that has very important features. This approach allows agents to move among hosts, in order to communicate with other agents. It also supports agents written using diverse languages, and furthermore permits the agent's programmers to implement a variety of interaction schemes based on a general mechanism for agent communication. The agent infrastructure uses the *Hypertext Transfer Protocol* (HTTP) for agent transfer and communication; by taking advantage of this widely accepted, platform-independent mechanism, the agent infrastructure makes it easy both for providers to offer agent-based services and for users to access them. The Aglet from IBM [3] is another approach to mobile agent infrastructures; in this case, the aglet represents the next step forward in the evolution of executable Internet content, (according to IBM). In this approach, a program code is introduced that can be carried along with its state information. Aglets are Java objects than can move from one Internet host to another. An aglet executed on one host can suddenly halt its execution, dispatch itself to a remote host, and resume its execution there. When the aglet moves, it takes along its program code as well as its data.

On the other hand, the development of generic infrastructures for object streaming has also been proposed. Although basically intended for multimedia streaming similar to that proposed in [4], another approach has been developed by HP [5], whose main objective is to investigate Internet object streaming systems. In some cases, like in [6], the use of object streaming is motivated by some particular applications, as in the case of augmented reality.

The importance of developing generic infrastructures is evident in the above-mentioned approaches. It is necessary for researchers in parallel and distributed computing areas to develop generic infrastructures, in order to facilitate the development of future distributed and parallel applications. The idea behind these approaches is to therefore identify, design and create reusable software components that can fit within a set of criteria, in this case for mobile agents and object streaming.
The generic infrastructure proposed in this paper is

based on ProActive, a middleware (programming model and environment) for object oriented programming and parallel, concurrent and distributed computing. We have extended ProActive by implementing a hierarchical and dynamic component model for mobile agents and object streaming. The following is a description of ProActive, as well as some of its special features.

ProActive is based on a well-studied programming model, and is a Java library for parallel, distributed, and concurrent computing that also features mobility and security in a uniform framework [7]. With a reduced set of simple primitives, ProActive provides a comprehensive API which allows for simplifying the programming of applications distributed on local area networks (LAN), on workstation clusters, or on Internet grids.

The library is based on an active object pattern that contains the following characteristics:

- A remotely accessible object
- A thread as an asynchronous activity
- An actor with its own script
- A server of incoming requests
- A mobile and potentially secure agent

ProActive is only made of standard Java classes, and requires neither changes to the Java Virtual Machine, nor preprocessing or compiler modification; programmers simply write standard Java code [8]. Based on a simple meta-object protocol, the library is itself extendable, permitting system adaptations and optimizations. ProActive currently uses the RMI Java standard library as a portable transport layer.

ProActive features the following:

- Asynchronous calls: Typed Messages (Request and Reply)
- Automatic future-based synchronizations: wait-by-necessity
- Migration, mobile agents (compatible with Swing and AWT)
- Remote creation of remote objects
- Reuse polymorphism between standard objects and remote objects
- Group communications with dynamic group management
- Libraries for sophisticated synchronizations, collaborative applications
- Transparent, dynamic code-loading (up and down)
- Seamless management of the RMI Registry and Jini.

Although ProActive is now a powerful library for parallel, distributed and concurrent computing, it lacks a generic infrastructure for mobile agents and object streaming. Identifying, designing and creating such components is the focus of this research. In addition, this proposal will extend the library, thereby giving it more versatility and application. It will furthermore facilitate the development of distributed and parallel applications, especially for those which require the use of mobile agents and/or object streaming.

2 Generic Infrastructure

A generic infrastructure is basically a minimum set of primitives necessary to develop applications in any field. In order to program mobile agents and/or object streaming applications, it is mandatory to have a set of instructions that allow programmers to focus on the problem at hand rather than on implementation details. The following two sections explain the set of primitives needed for mobile agent and object streaming infrastructure. In both cases, we identify the components of the architecture, the functionality of each component and discuss the interaction among the components.

3 Mobile Agents

3.1 Introduction

Mobile agents constitute a new approach to the architecture and implementation of distributed systems. A mobile agent is a program with a persistent identity that can both move around a network and communicate with its environment and other agents [9]. Possible applications for mobile agents include network management, information retrieval, distributed simulation, remote device control, active documents and mobile computing.

Mobile agents have the following characteristics:

- Autonomous or semi-autonomous.
- Guided to execute tasks.
- Are sent like objects.
- Asynchronous.
- Able to communicate.
- Can operate without connection.
- Can suspend their execution.
- Able to be duplicated.
- React to changes in their environment

In order to relate the previous features to our ProActive model, we must first consider the features of the active object [5]: transparent localization,

controlling activity and handling synchronization. The combination of both concepts facilitates the creation of a mobile agent by means of an active object [10].

In order to be useful, an agent needs to interact with its host and other agents. The information contained in an agent should be offered and/or negotiated with other agents during the exchange of services. Agents should also be able to move inside heterogeneous computer networks. This is possible only if a common work characteristic such as a standardized agent infrastructure exists for agent operations throughout the entire network.

Given the above-mentioned, we propose an interactive mobile agent infrastructure that manages the creation, manipulation and monitoring of generic mobile agents. The objective is to allow the programmer to choose the agent's behavior, and to then have the infrastructure serve as a support in that same agent's creation, manipulation and monitoring.

3.2 Generic Infrastructure for Mobile Agents

In this subsection we identify the components needed for our infrastructure of mobile agents. Furthermore, we describe the functionality of each component, as well as the way they interact with each other.

First, we have identified a set of components for a generic mobile agent infrastructure. These components have been modeled by a UML diagram in Figure 2, and are as follows:

Two main entities were identified:
Agent: This entity is a pattern for programming an agent with a particular activity (see Figure 1).
Agents' server: This entity is the one in charge of the agent's handling, from the creation until the end, obtaining the agent's result.

During the design process, the following components for the agents' server were identified:

1. GiaController.
2. GiaLaunchAgent.
3. GiaInforAgent.
4. GiaResultAgent.

The description of these components is carried out as follows:
Initially the **GiaController** is activated, whose process carries out the following activities:

- Initiate an agent.
- Obtaining the agent's information (state, position).
- Obtaining the results obtained by the agent.
- Suspending and renewing the execution of an agent.
- Viewing the list of activated agents.

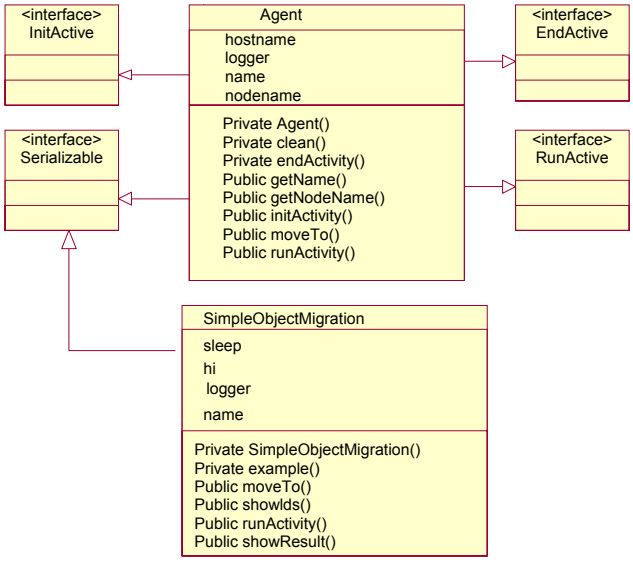

Figure 1. Agent Class.

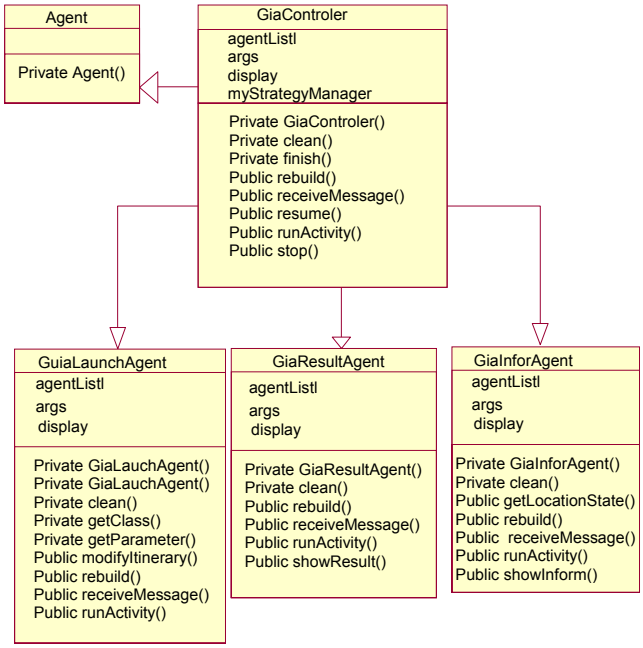

Figure 2. Generic Mobile Agent Infrastructure.

- **GiaLaunchAgent** is the component responsible for obtaining the necessary data for initiating an agent with a given itinerary (established within the class, in order to determine a new itinerary and the necessary parameters according to the agent's type).
- **GiaInforAgent** is the component that provides information about the state and the generated agent's position to the GiaControler.

- **GiaResultAgent** is the component responsible for providing the results that the agent has obtained after the execution to the GiaControler.
- The **Agent** interacts with the above-mentioned components before providing information of their state, position and the results obtained.

The modeling of the components is presented in UML, by means of a class diagram shown in Figures 1 and 2.

The characteristics involved in the current development for the generic ProActive agent are the following:

- Operation without connection.
- Suspension of execution.
- Ability to be duplicated.
- Reaction to changes in their environment.

4 Object Streaming

4.1 Introduction

A stream of objects is an orderly sequence of objects that has a source (input stream) and a destination (output stream). The stream provides a uniform interface of objects in such a way that the programmer is unaware of the type of data contained in the stream. The stream is merely treated as an object, which thereby gives it the qualities of an object.

4.2 Generic Infrastructure for Object Streaming

The transmission-reception of a stream is a technique that is becoming more and more important in the area of networks, and particularly in multimedia. HTTP is an example of a protocol with stream, which allows the viewing of text before the images are totally transmitted. Up until now, the streams that java provides, as well as the shipment of a stream, are technically at a relatively low level of abstraction, mainly at the net protocol level (for example, RTP [11], or Real Time Transport Protocol, and RTSP [12], or Real Time Stream Protocol).

Nowadays, a deep necessity exists for defining models and techniques for object streaming with a high level of programming abstraction with object-oriented languages, particularly in Java. The main idea is to maintain the object streaming at a high level of abstraction, thus avoiding having programmers worry about implementation details related to this stream. A generic infrastructure should be generated in such a way that would allow the appropriate methods to be easily summoned, and that would obtain a transparent stream of objects in a parallel and distributed environment.

In this paper we propose a generic infrastructure with ProActive that has as its main objective the transmission of objects in a stream, without loss of objects and control of the stream by the user. The goal of this approach is to improve the effectiveness of objects-distributed programming, and to permit the user to have total control of the objects inside the same stream.

In ProActive, the active objects are fundamental, so it is important to dedicate a part of this document to discuss them. In this way, it is possible to design a set of basic primitives for object streaming. Programmers will be able to make use of these elements, which in fact are basic units of activity and distribution, to build parallel, distributed, and concurrent applications using ProActive. The modeling of this stream is made by the use of active objects. An active object [5] contains its own thread, and a thread will execute only methods invoked in the same one by other active and passive objects from the subsystem to which the active object belongs. With ProActive the programmer doesn't need to manipulate the object threads explicitly, contrary to the standard Java. The active objects can be created in any of the hosts involved in the computation. Once an active object is created, its activity (the fact that it has its own thread) and its localization (local or remote) are perfectly transparent. In fact, any active object can be manipulated as if it were a passive instance of the same class. The previously indicated generic infrastructure is designed with the help of active objects. It uses a model client-server, of which three active objects are used for the client's side:

- **ClientAO**: In charge of sending the transfer petitions to the server.
- **ViewControlAO**: Interacts with the user via a graphic interface.
- **TransferAO**: Receives the transfer of the object made by the server.

The server also has three active objects:

- **ServerAO**: In charge of receiving the client's petitions and assisting each request.
- **TransferAO_Server:** In charge of carrying out the connection between hosts in order to execute the stream. This object is very important since it has the task of controlling the stream. The user can decide at any given moment whether to carry out transfer changes, change the IP or Port destination, change transfer priorities, or to carry

out actions such as canceling or definitely finishing the transfer. For many of these actions, it will be necessary to close the stream, the connection or maybe both, since TransferAO_Server is the object in charge of executing them.

- **ViewControlAO_Server:** Responsible for verifying that the actions mentioned above will be shown to the user.

The architecture of this proposal can be seen in Figure 3. The software components designs are in UML and are shown in Figure 4.

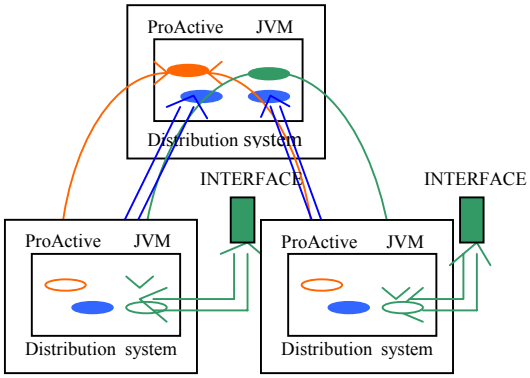

Figure 3. Infrastructure Architecture.

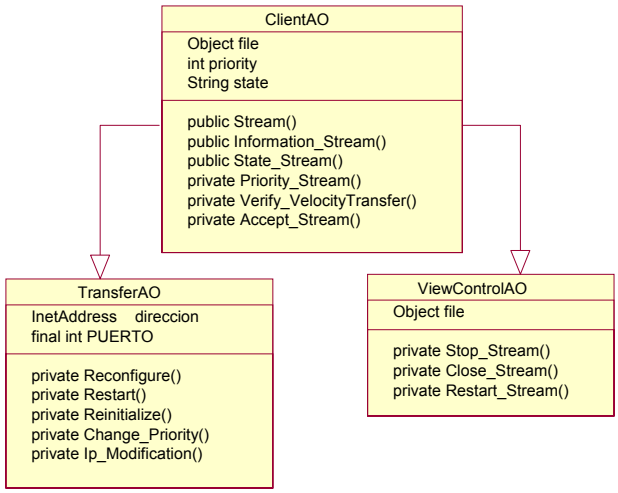

Figure 4. Classes Diagram.

The advantages of designing this infrastructure are the following:

- Interleaving & LOD's (Link Out of Delay) synchronization.
- Remote control of transfers.
- Parallel transfers with groups.

5 Conclusion

A generic infrastructure has been presented for mobile agents and object streaming on ProActive. We defined the components of each infrastructure and their functionality, as well as the way that they interact together. The developed infrastructure has extended the library of parallel, distributed, concurrent computation developed in INRIA (ProActive). The use of active objects has been recaptured for the implementation of both platforms; taking this development as a basis, programmers of parallel, distributed and concurrent applications which make use of mobile agents and/or object streams, will be able to focus their efforts on the problem at hand rather than on the application implementation. It is necessary to analyze the way an agent will obtain all necessary features, especially in sending the order to stop the execution of an agent and to resume its execution at another host, while maintaining all its data.

References:

[1] D.Wong, N. Paciorek, T.Walsh, *Concordia: An Infrastructure for collaborating mobile agents*.

[2] A.Lingnau, O. Drobnik., *An Infrastructure for mobile agents: requirements and architecture*

[3] IBM Tokyo Researh Lab, *Aglets: Mobile Java Agents*, URL=http:/www.ibm.co.jp/trl/projects/aglets.

[4] Duc A. Tran, Kien A. Hua and Tai Do, *ZigZag: An efficient Peer-to-Peer scheme for Media Streaming*, , Infocom 2003.

[5] HP Labs : Research : *MMSL : Projects : Streaming Media Systems*, URL= http:// www.hpl.hp.com/research/mmsl/ projects/streaming.html.

[6] Tokunaga, E., Zee A., *Object-Oriented Middleware Infrastructure for Distributed Augmented Reality*, ISORC 2003.

[7] D. Caromel, *Towards a Method of Object-Oriented Concurrent Programming*, Communications of the ACM, Volume 36, Number 9, pp 90-102, September 1993.

[8] INRIA Sophia Antipolis, *Manual de ProActive.*

[9] W. Cockayne, M. Zyda, *Mobile Agents*, Mamming, 1998.

[10]F. Baude, D. Caromel ,F.Huet,J.Vayssiere Proceedings of HPCN Europe 2000.

[11]Henning Sinul Zrinne, Stephem L. Casner, Ron Frederick, Van Jacobson, A Transport Protocol for Real Time Applications, Internet Engineering Task Force, 2001

[12]H. Shulzrinne, A.Rao, R. Lanphier, *Real Streaming Protocol (RTSP),* Columbia U./Netscape/Real Networks,1998.