Towards full agent interoperability

Bruno Dillenseger, Huan Tran Viet — CNET France Télécom
{ bruno.dillenseger | huan.tranviet }@cnet.francetelecom.fr
BP 98, F-38243 Meylan CEDEX, France

abstract. This paper introduces a CORBA service dedicated to the transport of agent communication. It explains how this service completes the emerging standards in agent technology to enhance interoperability, while supporting peculiarities of mobile agents. It also shows how this service may be customized to support a variety of message-based agent communication models.

1 On agent interoperability and cooperation

1.1 Introduction

How can heterogeneous agents, in an open distributed environment, talk to each other, understand each other, and then negotiate, cooperate, etc.? As a matter of fact, these things that we call agents are expected to be autonomous and interact with their environment and other agents in order to exhibit smart behaviour and solve complex distributed problems.

On high levels, agents need a common knowledge core, featuring a minimal conceptual framework modelling their environment, and other agents. On lower levels, agents need an actual infrastructure to interact, a support for exchanging information. To summarize, agents must be able to *communicate*, which implies coping with several levels of heterogeneity (figure 1):

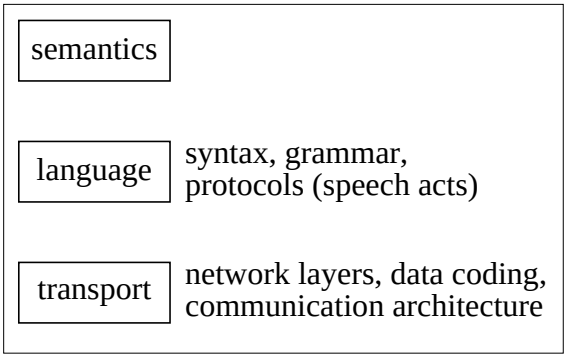

figure 1: the levels of heterogeneity for agent communication.

- on the upper level, a common semantics must be shared about the environment and things agents deal with;
- on the middle level, a common language must be specified, with its syntax, grammar and protocols (e.g. speech acts) so that negotiation and cooperation can be managed;
- on the lower level, a communication transport architecture must be specified in order to manage the network layers, computing resources and data coding.

As standardization is a key issue for full agent interoperability, we see that the current results mostly address the semantics and language levels, but that transport specification for agent communication is still in a very preliminary stage.

1.2 OMG and MASIF

The Object Management Group's first effort in the agent field resulted in the MASIF specifications adopted in 1998. Formerly known as MAF (Mobile Agent Facilities, 1997), the "Mobile Agent System Interoperability Facilities" specifications [6] finally focused on defining a conceptual framework, services and interfaces for CORBA-based interoperability between heterogeneous mobile agent platforms. The re-use of existing CORBA services (security, naming service, life cycle and externalization templates) is also analysed by MASIF.

MASIF's framework defines the concepts of *agent* (either *mobile* or *stationary*), hosted by *places*, run by *agent systems*, belonging to *regions*. Two interfaces are defined:

- the MAFFinder interface defines operations for place, agent and agent systems lookup;
- the MAFAgentSystem interface deals with the management of an agent system, its places and agents (creation, destruction, mobile agent migration).

There should be at least one server implementing the MAFFinder interface per region, and each agent system should implement the MAFAgentSystem interface.

MASIF suffers from many weaknesses: How can regions be interconnected? How can an agent system receive an incoming agent of a different agent system type? How may heterogeneous agents communicate? Moreover, it appears that mobility makes the reuse of today's CORBA services neither simple (e.g. naming), nor sufficient (e.g. security). Finally, although MASIF has been designed to take existing mobile agent platforms into account, the fact that only one implementation is available today makes it impossible to assess the actual support for interoperability. Nevertheless, MASIF has to be regarded as a first step, a basic infrastructure and conceptual framework for further specification and revision.

After the MASIF experiment of the OMG in the agent field, the Agent Working Group was created at the end of 1998 in order to provide OMG with a forum educating on agent technology, and to go further with the integration of agent technology to the OMG's specifications, taking into account agent standardization bodies such as FIPA.

1.3 FIPA

The Foundation for Intelligent Physical Agents was officially created and registered as a non-profit association in 1996, and gathers 48 members from 13 countries in 1998. As stated in

the foreword of [4], its purpose is *"to promote the success of emerging agent-based applications, services and equipment"*. Based on an international cooperation between academic and industrial teams active in the agent field, FIPA aims to produce *"specifications that maximise interoperability across agent-based applications, services and equipment"*.

As far as agent communication is concerned, FIPA's specifications are quite advanced on the semantics and language levels. The former level is addressed by an effort on ontologies, and the latter by the adoption of the speech-act based Agent Communication Language (ACL). On the transport level, a minimal architecture makes use of an "Agent Communication Channel" with a one-method interface for message routing, and a recommendation to support CORBA's Internet Inter-Operability Protocol for inter-platform communications.

1.4 Middleware-based experiments

[2] summarizes several experiments that we carried out during the past seven years, on the construction of agent platforms by adding middleware-based (mostly CORBA) communication features to existing programming languages. In these platforms, agents are designated by chosen unique names in a naming service, and send messages to each other either synchronously or asynchronously through personal mailboxes. Thanks to a group facility included in the communication middleware, messages may also be sent to groups in multicast or unicast modes. The PUMA platform [3] also achieves execution-safe agent mobility.

The most striking benefits of using a middleware such as CORBA is to reuse generic services for distributed objects, and to hide heterogeneity concerns (networks, operating systems, programming languages). But it appeared that directly integrating communication mailboxes into the agents had several drawbacks:

1. code for the mailbox has to be written for each programming language, and for each ORB;
2. since ORBs typically don't manage mobility, there is no straightforward and fully reliable way to achieve transparent communications for mobile agents.

3 Which communications for which agents?

3.1 Problems with basic RPC-style communications

RPC-like communications are widely used in today's agent platforms. The MASIF specifications claim that RPC-like communications (e.g. CORBA, DCOM, Java RMI) fulfil stationary agents' needs. But is this model really suitable for agents? In fact, many multi-agent systems, especially those featuring smart agents (DAI), assume the availability of a mailbox

service. This simple communication model preserves every agent's autonomy in reading and processing messages.

On the contrary, the direct use of RPC is a disturbance for the receiving agent. This is even worse for mobile agents, since on-going calls prevent it from moving (or sometimes do not, but result in inconsistencies in the agent state). We see that if agents directly make use of RPC-like communications, many extra features should be envisaged in order to control the incoming calls. On the contrary, it is easy to imagine a stand-alone mailbox infrastructure allowing the receiving agent to be autonomous about the way an incoming message has to be treated. Of course, the communication between agents and this infrastructure is likely to use RPC, but it may be hidden from the agent point of view, and, more than this, the infrastructure behaves like a buffer isolating the communicating agents from each other, avoiding activity interferences.

3.2 The "shared memory" approach

Several such communication infrastructure have already been designed. Typically, when agents don't communicate by mail, they use more or less structured shared memories. Distributed blackboards, or the Linda interactor model [5] are such infrastructures. The approach of the JavaSpace specifications is quite similar in that it defines an infrastructure of servers, to manage the deposit and retrieval of message objects. Thus we see great interest in a stand-alone (more or less distributed) communication infrastructure, isolating communicating agents. This is particularly relevant when the sender doesn't know the actual readers of a message/information. But isolating the senders and the receivers is also a good thing when agents are not permanently reachable, either because of network connectivity issues, or because of mobility. These shared-memory approaches are very powerful in that they often come with causality, persistence, transaction features. But these interesting features often result in heavy specifications, complex to write and implement (JavaSpace is still in a specification phase, and rely on other utilities that are just beginning to be specified), and not always easily understood by their users. Moreover, scalability is generally a problem.

4 A CORBA Mobile Agent Communication Transport Service

4.1 Why a MACTS?

The Mobile Agent Communication Transport Service aims at providing a standard communication infrastructure for mobile agents. In the context of the MIAMI project, the OMG MASIF standard is taken into consideration for mobile agent platforms management concerns.

But this standard defines nothing about communication between heterogeneous agents. The MACTS adds a CORBA-based communication transport facility to the Mobile Agent System Interoperability Facilities. Since FIPA specifications are also taken into account by the MIAMI project, our infrastructure also intends to be suitable for the transport of FIPA's ACL messages.

Providing a stand-alone transport service for agent communication must also be regarded as a continuation of our previous work (see section 1.4), with a step forward. This step consists of fully externalizing the communication system, which is used through CORBA interfaces. There are several advantages:

- it allows development effort to be concentrated on the common, shared infrastructure, while lessening the work on the various agent platforms;
- an agent is not necessarily a CORBA object (but just a CORBA client), which may be simpler for some agent platforms;
- it is not programming language dependent (unlike our C++ framework or JavaSpace);
- it isolates communicating agents from each other, making it easier to handle mobility and any non-permanent network connectivity problems (the MACTS server should run on a "well-connected" machine).

4.2 Overview

The MACTS is a message transport layer providing a mailbox-like service. To send and receive messages, an agent needs to create a *message port*, by invoking a *message port factory* (figure 2). Basically, these message ports are CORBA objects, that store incoming messages.

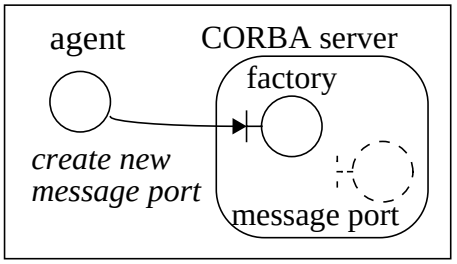

figure 2: message port factory

Then, the default behaviour implies that the recipient agent checks for new messages whenever it likes or can (figure 3). An agent may also create a *message port listener* and register

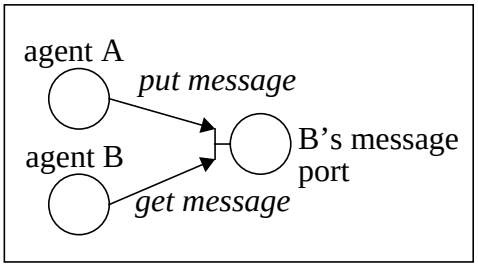

figure 3: message port in store mode

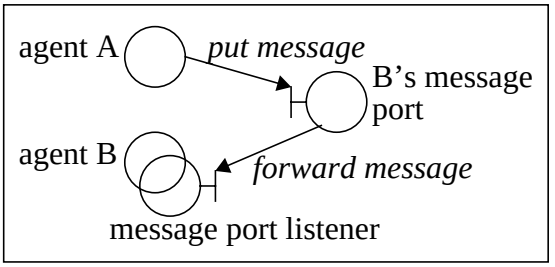

figure 4: message port in forward mode

it on its message port. The listener is a CORBA object that implements an interface dedicated to message reception. When a listener is registered, the message port switches from *store mode*

to *forward mode*. This interface is invoked by the message port for each new incoming message (figure 4). As soon as a communication failure occurs when calling the listener (typically, when the object reference of the listener is obsolete after it has moved along with the agent), the message port automatically switches to store mode without losing any message.

4.3 Implementation and usage

The MACTS relies on four IDL interfaces:

- **`MsgPortUser`** is the message port interface for adding a new message - this interface is used by the sender;
- **`MsgPortManager`** is the message port control interface, providing an operation for reading incoming messages, and operations for listener (de-)registration and message port destruction - this interface is used by the owner of the message port;
- **`MsgPortListener`** must be implemented by a message port listener in order to get incoming messages on the fly - this interface is used by the message port;
- **`MsgPortFactory`** is the message port factory for creating new message ports - this interface is used by agents.

These stationary message ports provide a multi-purpose and mobility-tolerant communication infrastructure for agents. An agent may have several messages ports in different places in order to improve either communication performance or reliability. According to its specific constraints, the agent may choose two communication models: either a pure asynchronous communication model, where the agent checks from time to time or periodically its message port content (store mode), or a more "reactive" model where the agent takes incoming messages into account as soon as possible (forward mode). In the latter case, the agent will have to register the new reference of the listener after each move in order to keep the "reactive" behaviour.

5 High-level customization through personalities

5.1 What is a MACTS personality?

A personality aims at hiding from the programmer CORBA-related "burdens" such as IDL compilation coming with the MACTS, and providing higher-level communication features. Moreover, a personality tries to show the genericity of the MACTS through various typical agent communication cases. Of course, hiding tasks such as IDL compilation implies that these personalities rely on a given programming language.

5.2 The “Mailbox personality”

The Mailbox personality is designed to use the MACTS infrastructure in a transparent manner and provide a high-level mailbox system, featuring:

- a mailbox address scheme consistent with the region concept of MASIF (agent_name@region_name), making it possible to communicate through regions thanks to the federation of CORBA naming services;
- mobility-tolerant store and forward modes;
- unicast and multicast features in regions.

From the programmer point of view, using the mailbox personality is as simple as creating an instance of Java class Mailbox, passing two strings as arguments (the agent name in the region, and the region name), plus a uniqueness flag. This flag specifies whether the requested name should be unique in the requested region. If the flag is true, the address is guaranteed to be unique by the Mailbox system, and the Mailbox instance is a personal mailbox, with a *unique address.* Otherwise, several Mailbox instances may have the same *multiple address*, and form a communication group in a given region with multicast (every member-mailbox receives the message) and unicast (an arbitrary member-mailbox receives the message) features.

For each Mailbox instance created, a MACTS message port is created in the factory of the requested region, and registered in this region’s naming service. A Mailbox instance provides methods to send asynchronous messages, and to get incoming messages on demand or on the fly, as well as miscellaneous management methods:

- **`discard()`** destroys the mailbox’s associated MACTS message port;
- **`empty()`** destroys pending incoming messages;
- **`getAddress()`** returns current mailbox’s address;
- **`noListener()`** switches the mailbox to store mode;
- **`receive()`** reads the next pending incoming messages;
- **`send2all()`** and **`send2one()`** send messages, respectively in multicast or unicast mode (only valid in case of "multiple" address recipients - any method may be used for a "unique" address recipient);
- **`setListener()`** switch the mailbox to forward mode.

Compared to the plain MACTS, the Mailbox personality hides IDL definitions and CORBA object references and provides high-level addresses with predictable names, and advanced features such as multicast/unicast message sending. Moreover, the Mailbox offers more transparency to mobility by automatically "reconnecting" mailbox listeners, high-level versions

of the message port listeners. This way, a mobile agent may move while continuing to receive messages, without blocking the senders during its migrations, without losing messages, and while keeping its consistency.

The Mailbox personality relies on a specific CORBA naming service usage federating the naming regions. This federation consists of running a naming service in each region (in the sense of MASIF), and binding a name context to each region name. If the region name represents the local region, it is bound to a local region context which contains the name bindings of the local mailboxes. If the region name represents a foreign region, then it is merely bound to the object reference of the actual naming context in the foreign name service. Using this mechanism allows transparent inter-region communication. Merging this approach with the MASIF specifications would result in region interconnection for management and mobility.

5.3 The FIPA personality

In the FIPA architecture [4], an *Agent Communication Channel* (ACC) routes messages between agents from an *Agent Platform* (AP) to agents resident on other APs by using a default protocol among ACCs. The exchange of messages among agents residing on the same AP is the duty of the *Internal Platform Message Transport* layer (IPMT), which is out of the scope of FIPA specifications. The support of mobility in the context of FIPA requires an agent-interoperable communication transport service, which allows a mobile agent to:

- interact locally with other agents (ACC included) on the destination platform;
- receive incoming messages when moving without modifying its contents, on the *Home Agent Platform* (HAP);
- receive and store incoming messages when it is provisionally disconnected from the network (e.g. on a mobile device).

The FIPA personality of the MACTS satisfies not only these requirements, but is also used as a means of communication supporting the IIOP protocol (recommended by FIPA) among ACCs. Because of its genericity, the MACTS does not directly meet the needs of FIPA agents. Therefore, the FIPA personality is a Java package aimed at providing a solution to the communication concepts of FIPA'98: an agent's *Globally Unique IDentifier* (GUID) and communication address, ACL message, *letter* object.

An agent must open a *letter port* to send and receive letters wrapping ACL messages. A letter object is composed of an *envelope* object and an ACL message. The letter port analyses only the content of the envelope for the message routing purpose, but is not required to read the content of the message (which may be encrypted for security reasons).

The letter port is an intermediate between an agent and its message port (figure 5). In the forward mode, the letter port receives a letter object from the agent, extracts the address of the destination message port, converts to a stringified form and then sends that string to the message port. On the destination side, the letter port builds a letter object from the forwarded string and passes it to the destination agent. When the agent moves or disconnects from the network, its letter port (and the associated message port) switches to the store mode. Incoming messages are immediately stored by the message port and then received by the agent on demand.

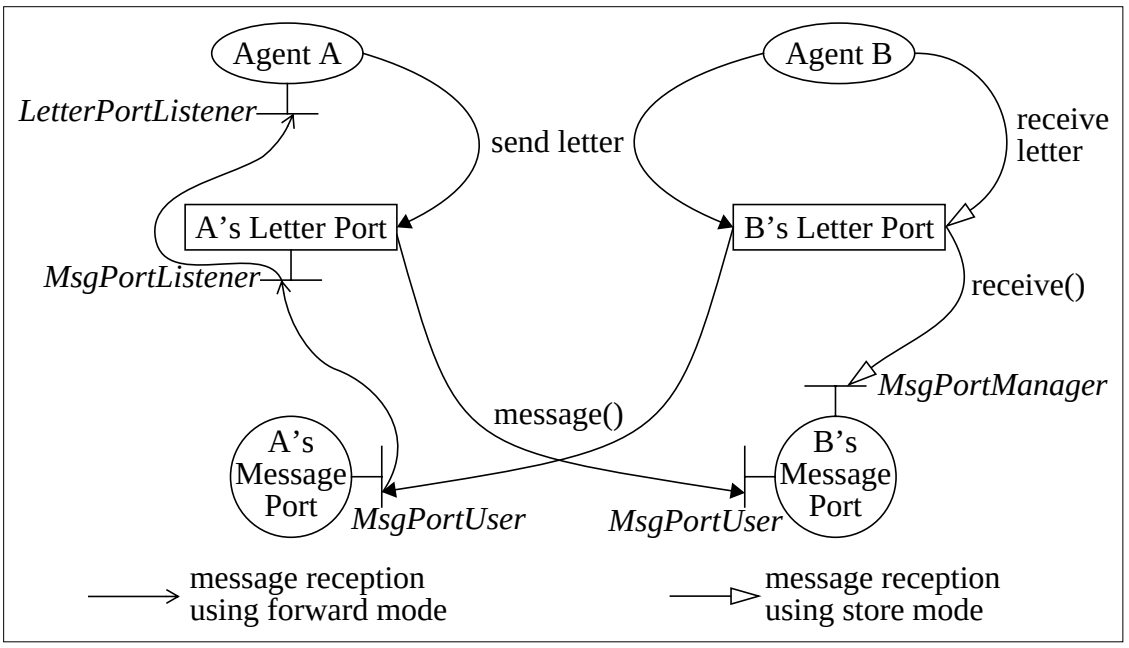

figure 5: the FIPA personality of the MACTS

Note that the letter port automatically handles serialization and deserialization in order to provide mobility transparency to mobile agent programmers.

6 Conclusion

In the context of the agent-technology emerging standards, this paper focused on the heterogeneous agent interoperability issue. It showed an interoperability level that is not covered in a satisfying manner today by standards such as MASIF and FIPA. Then, it presented a CORBA-based transport service for agent communication, compatible with the mobile agent paradigm by exhibiting a uniform, well specified and simple behaviour in message management. This service isolates the sender and the receiver of a message in order to protect their autonomy and to handle properly the connectivity issues related to mobility and network access. To illustrate the versatility of this service, a general-purpose mailbox system featuring advanced capabilities, as well as a FIPA'98 compliant communication architecture were

developed. These personalities show that several customized, easy-to-use, and agent-related communication systems can be built on top of the MACTS.

Some more work has to be done about the management of the MACTS infrastructure. For instance, some sort of garbage collection for unused message ports could be necessary. Moreover, the fact that the Mailbox personality has been much more complex to implement than the MACTS, makes us think about looking for extra generic features to enhance the MACTS itself.

References

[1] Agents Working Group. *Minutes of Meeting #1*. OMG Joint Internet SIG & Electronic Commerce Domain TF. Seattle, Washington, September 14-15, 1998.

[2] Bruno Dillenseger. *From Interoperability to Cooperation: Building Intelligent Agents on Middleware.* Lecture Notes in Artificial Intelligence 1437 (Proc. of IATA'98), Sahin Albayrak, Francisco J. Garijo Eds. Springer 1998, pp. 220-232.

[3] Bruno Dillenseger, François Bourdon. *Supporting intelligent agents in a distributed environment: a COOL-based approach.* Proc. of "Technology of Object-Oriented Languages and Systems", Versailles, France, 1995. Prentice Hall, pp. 235-246.

[4] *FIPA 98 Specification.* Foundation for Intelligent Physical Agents (Geneva, Switzerland), 1998.

[5] David Gelernter. *Generative Communication in Linda.* ACM Transactions on Programming Languages and Systems, Vol. 7, No. 1, pp. 80-112 (January 1995)

[6] *Mobile Agent System Interoperability Facilities.* Object Management Group TC Document orbos/97-10-05, November 10, 1997.

Acknowledgments

The work presented in this paper about the MACTS was carried out in the context of the MIAMI (Mobile Intelligent Agents for Managing the Information infrastructure) european project of the ACTS programme. Refer to http://www.fokus.gmd.de/research/cc/ima/miami/ for details.