# Adding New Communication Services to the FIPA Message Transport System

Javier Palanca, Miguel Escrivá, Gustavo Aranda,
Ana García-Fornes, Vicente Julian, and Vicent Botti

Dept. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
{jpalanca, mescriva, garanda, agarcia, vinglada, vbotti}@dsic.upv.es

**Abstract.** Agent communication is one of the most important aspects in the multi-agent system area. In recent years, several works have been developed that are related to the agent communication problem.

This paper presents a new method for agent and agent platform communication in accordance with FIPA proposals. It uses the Jabber protocol as a new message transport protocol (MTP). This protocol provides additional services that are not included in the current standard FIPA MTP. It provides facilities for "presence notification", "multi-user conference" and "security services". As a result of this work, a new plug-in for the JADE platform that incorporates this transport protocol has been developed.

## 1 Introduction

Some multi-agent platforms allow external communication by means of using their own protocol and schema of communications. Good performance can be achieved this way, but it is not possible to inter-operate with a different platform that does not understand this particular way of communication. There are platforms that are capable of communicating using standard protocols agreed upon by consortiums like the FIPA [1]. The most widely used standard protocol proposed by the FIPA is the Hypertext Transfer Protocol (HTTP) which is a protocol that was originally conceived to transfer web pages from a server to a web browser. Although this protocol is rather simple and allows some kind of bidirectional communication between client and server, it is not designed to sustain lengthy conversations over time as it is focused towards a 'query and result' routine. These are detected drawbacks in current Message Transport Protocols for agent platforms.

This paper proposes a new method of communicating agents and platforms using a more *human-oriented* way to develop conversations: Jabber [2,3], an Instant Messaging (IM) protocol designed to sustain lengthy bidirectional communications among entities on the Internet. This new Message Transport Protocol has been proposed to the FIPA consortium as a new preliminary specification. Besides, this paper introduces new services into agent communication that are only

K. Fischer et al. (Eds.): MATES 2006, LNAI 4196, pp. 1–11, 2006.

possible by integrating the Jabber technology in the FIPA Message Transport System.

A software add-on to the JADE agent platform that uses the new communication structure presented in this work has been developed and accepted by the JADE development team [4].

Next sections are structured as follows: Section 2 introduces a brief description about Instant Messaging and, in particular, the Jabber protocol; Section 3 introduces our proposal of using Instant Messaging to communicate agents. This section describes how an Instant Messaging protocol can be used to improve agent communication taking advantage of its features (like presence notification or multi-user conference); Section 4 describes our contributions to the Jabber protocol in order to support FIPA agent communication. This contribution have been submitted to the FIPA Consortium; Finally, conclusions are presented in Section 5.

## 2 Jabber Protocol

Jabber is an open protocol that is based on standard XML (eXtensible Markup Language) for the exchange of messages and presence information between two Internet points. This protocol was proposed by the Jabber Software Foundation [5].

The main use of Jabber technology is an extensible IM network that has similar features to other IM services like AIM, ICQ, MSN Messenger and Yahoo. Therefore, Jabber is an open, secure, and free alternative to consumer IM services. Under the hood, Jabber is a set of streaming XML protocols and technologies that enable any two entities on the Internet to exchange messages, presence, and other structured information in close to real-time.

Jabber is a communication protocol and can be applied to many scenarios. Its base is an XML-based messaging technology that can be used to communicate different entities. Its most common use is a decentralized instant messaging network, but this is not its sole application, as seen in the present work.

The term Jabber makes reference to a number of elements that, together, conform an infrastructure for the interchange of messages in real-time. The *technical* name of Jabber is XMPP, for eXtensible Messaging and Presence Protocol. It is a standard technology formalized by the Internet Engineering Task Force (IETF) based on the core protocols created by the open-source community [2]. The Jabber and XMPP terms can be used interchangeably.

XMPP technology has many advantages over existing proprietary closed ones. Being an open and free standard, which is also based in other open standards (XML, SSL, ...), it is not subject to sudden changes in its definition that may, for instance, render it to be incompatible with previous versions of the technology (as is the case with the MSN Messenger technology). It comes from a well-documented, open specification, which has eased the appearance and availability of many different types of clients, servers and *bridges* that enable the possibility

of interacting with other communication networks, such as email, SMS messages for mobile phones or other IM systems.

The Jabber network has a distributed architecture, which means that there is no central server that handles all the message delivery. Jabber servers cooperate together to bring messages from one user to another, like email servers do.

In the following list, we present the advantages of Jabber for the communication, specially over other similar proprietary networks: *Open, public, free, asynchronous, standard, tested, decentralized, secure, extensible and flexible.*

Bearing this in mind, it is clear that Jabber can be deployed to provide solutions far beyond the IM space as well as within it. Jabber applications beyond IM include network management, content syndication, collaboration tools, file sharing, gaming, remote systems monitoring and, now, agent communication.

## 3 Instant Messaging for Agents

In this section our contribution to a new agent communication framework is presented. This work presents the adaptation of the Instant Messaging technology to improve communication in multi-agent systems. We have studied instant messaging features that were interesting for agent communication and adapted them into a proposal for the FIPA standards.

### 3.1 Main Features

From a certain point of view, agents can be viewed as *'living'* intelligent entities, and so, all the advantages that benefit human users can also benefit agents. *'If IM is good for humans... Is it good for agents?'.*

Instant Messaging (IM) was initially developed to communicate humans. This technology has the ability of sending messages between two entities in a network. These messages are delivered in real-time (that's why it is called *'instant'* messaging) and are usually of text nature, although they can be any other kind of data (like binary or meta-information) if the protocol allows it.

Instant Messaging Networks also have more properties such as Presence Notification or Multi-User Conference. Presence Notification allows a user in the IM network to be notified when another user (who is a member of his/her contact list) changes his/her state.

Multi-User Conference is a technology that provides *chat rooms* where users can virtually *come together* and easily talk to more than one user. This feature is similar to the common Internet Relay Chats (IRC).

IM networks provide communication between human users in an unstructured way, using natural language. By changing the natural language used by humans for a common conversation with a standard structured language, like FIPA-ACL, and by inserting the necessary meta-information in a message (as shown in Section 4.1) it could be possible to re-use a common used Instant Messaging platform for agent communication. This is a more natural way to perform conversations between entities that involve interaction protocols.

Using an IM technology like Jabber for communicating agents seems like a natural step, since its innermost structure is quite analogous to what is commonly referred to as a multi-agent system (MAS) or platform. In IM, where there are users, in a MAS there are the agents; where there is a user directory, there is an agent directory; where there is a browseable components directory, there is a directory facilitator, and so on. From a certain point of view, one could say that agents behave as users that send or receive messages and make use of the resources and services of the platform.

When building a multi-agent platform, most of the design and implementation effort is put in the message transport mechanism. If an IM technology like Jabber is used, this *problem* is already solved. This model also simplifies the location issues of other users. The same user can connect from different locations without having to change addresses, the Instant Messaging server solves this.

Jabber is the ideal candidate to undergo this adaptation since it is standard, extremely adaptable, free, and has all the advantages and features presented in the previous section.

### 3.2 Presence Notification

Presence notification is one of the most useful features that Jabber provides. It is based on the Contacts List system. Contacts List is a mechanism through which a user can manage a list of *known friends* (called *roster*) to know at any time what the current status of any of the contacts is. Presence Notification allows a user to change its state in the network (e.g. *'Available', 'Busy', 'Don't disturb',* ...) to notify its contacts of its availability.

When a user changes its current status, all the other entities who share a bond with the user are automatically notified. An entity status usually means its availability and readiness to engage in a communication, but it is not closed to a short predefined list. Entities may define their own status and their meanings.

Two or more Jabber entities can be bonded (or *subscribed*) to each other. Entities who share a bond can make use of presence notification as described above. In order to form a bond, two entities must agree in a simple negotiation process that can be automated (–see Figure 1).

This simple and powerful system is useful to know when a contact is online and you can send messages to him/her. Otherwise, the contact is offline and a message sent to him/her will be stored in the server (so it will not be read in real-time).

Presence notification can be used for agent communication and Multi-Agent System platform communication. A *heartbeat pulse* (i.e., the ability for an entity to know when a bonded entity is online or not) can be built over this mechanism. This eliminates the need for a *'Ping'* service in the agents or in the platform, as all of them can be bonded to each other or to a hypothetical *central* AMS-like agent that can control the life-cycle of the rest of agents.

Bonds can also be used by agents to form *social circles*, that is, groups of agents that are aware of each other's presence and status. This eliminates the need for making several queries to a central *white pages* service or *yellow pages*

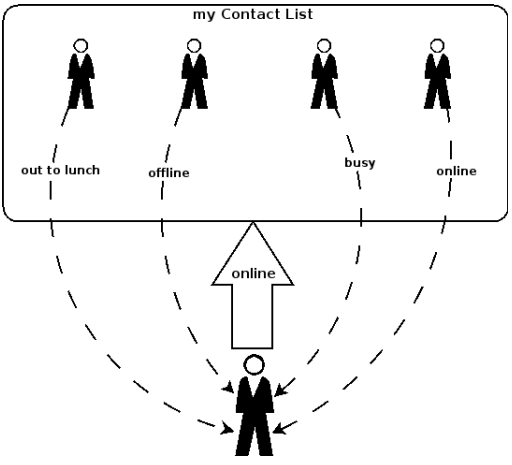


**Fig. 1.** Presence Notification

service to find out the availability of agents and services and, in the end, improves the general platform throughput.

### 3.3 Multi-User Conference

Another interesting Jabber feature is Multi-User Conference (MUC, in short). Jabber offers public conversation channels where users can join and make conversations with more than one user at the same time. These channels can be created by any entity (with an authorization in the server) and can be protected using a password, so only the entities that know the password can enter. A channel creator (or *administrator*) can also add or remove permissions to the entities that share a channel, such as the right to *speak* in it (–see Figure 2).

All these MUC features can be used to create forums dynamically for agents to connect to when performing some multi-agent activity. For instance, in a virtual auction scenario performed by agents, an agent playing the auctioneer role can create a password-protected channel and give the password only to previously certified agents. Then, in the channel, the auctioneer can give the right to speak only to agents who are going to play the bidder role, leaving the rest of agents in spectator mode only. Once the auction starts, every time a bidder wants to place a bid it just needs to *'speak'* in the channel, and all the other bidders and the auctioneer will know the bid at the same time.

MUC channels can also be used for more useful purposes. A simple method to send broadcasting announcements would be by sending a message to an appropriate chatroom. Every agent connected to that chatroom would receive the broadcast transparently. These chatrooms can be used to broadcast public platform information or to manage selected distribution lists. As an example imagine an scenario where a group of agents want to be notified when a particular

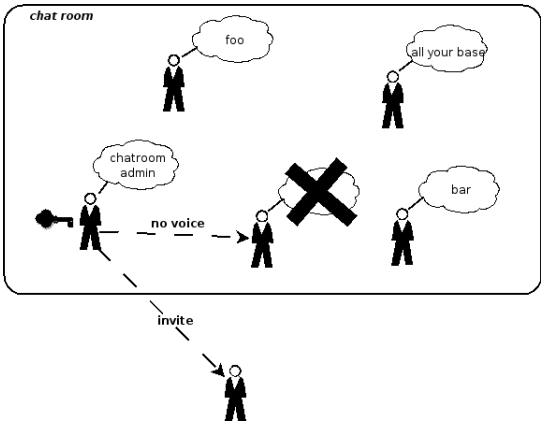


**Fig. 2.** Multi-User Conference

service is available. Thanks to Multi-User Conference mechanism, every agent that wants to be notified simply joins the chatroom associated with the service and when the service provider wants to announce the service availability sends a message to the chatroom. Note that the subscription list is automatically managed by the chatroom. Of course, this example does not replace the search service of the Directory Facilitator, it provides a new method similar to mailing distribution lists.

Using this powerful Jabber mechanism more MUC uses are quickly developed for agent communication improvement like federations, coalitions, hierarchies, etc. Also complex interaction protocols, like auctions, can be easily built with these facilities.

### 3.4 Distributed Network

As mentioned above, Jabber is decentralized by nature. Therefore, a Jabber user does not only choose a username, but also the server where the account will be created. What could at first seem to be a usability disadvantage, is one of the main advantages of Jabber as opposed to other IM systems.

For an agent, this means it can retain the same contact address even if it moves through different hosts, networks or even platforms. For a multi-agent platform, this feature provides several advantages. Let's see some aspects of this feature and their potential impact on an agent platform:

– Workload balance: Since there are servers distributed everywhere, divided by geography, by theme or by provider (university, city council, Internet provider...), the network workload generated by the users tends to distribute itself among all of them. In addition, some components of a Jabber server (users directory, network bridges, etc ...) can be executed in a different host than the one running the server itself, making the distribution better.

- Network reliability: When a Jabber server goes offline, the only affected users are those connected to that particular server. The rest of the Jabber network remains untouched. Therefore, only a relatively small number of users are affected by the mishap, instead of collapsing the entire communications network.
- Private services: The independence of the servers allows the set up of an IM network inside an intranet without needing any external resources from the Internet, as it is able to work on isolated networks.

### 3.5 Security

Using Jabber to communicate agents provides some built-in security mechanisms that help maintain the system's integrity.

- Logging to a Jabber server requires a **username and a password**. This mechanism prevents an unauthorized connection to the Jabber server if it has not been approved by the administrator. The administrator can set up the Jabber server to accept any registering attempt, or can set it up to deny any registering attempt, or only some privileged connections (i.e: some domain). This can be extended for agents in a way that only certain agents will be able to connect to a server (i.e join a platform).
- A connection to the Jabber server can be encoded with a symmetric cryptographic algorithm using SSL (Secure Socket Layer). SSL provides data cyphering, server **authentication**, message **integrity** and, optionally, client authentication for TCP-IP connections. This mechanism ensures that every message that flows through the Jabber network is strongly encoded and cannot be read by any malicious entity. This is necessary to ensure **confidentiality**.
- To ensure the authentication of a message, the Jabber server provides another mechanism to avoid identity theft. Jabber prevents identity spoofing by overriding a '*from*' field that does not match the sender's. In FIPA-ACL a conceptually similar message field exists: the '*sender*' field. This mechanism can be extended to watch and alter this '*sender*' field accordingly.

### 3.6 Performance and Scalability

Performance and scalability are important bottlenecks on current agent platforms [6, 7, 8, 9, 10]. Let's suppose a scenario where a Jabber server is integrated in a multi-agent platform as its message transporter to route XML messages. The server connected to the platform can be any Jabber server implementation, so that the most accurate version of the server can be selected depending on the platform constraints.

There is a reference implementation (called `jabberd`) that perfectly fulfills a standard use of a multi-agent platform (it supports around 10,000 concurrent connections). If more agent load is needed in the platform, other specialized servers can be used, like `WPJabber` (around 50,000 concurrent connections)

or `eJabberd`, a distributed (clustering) and fault-tolerant Jabber server that supports around $1,000,000$ concurrent connections (using the necessary *horsepower*).

### 3.7 Shared and Persistent Database

XML DataBase (XDB) is a Jabber server component that provides an interface to any kind of data source used to store and recover server-side data. A Jabber client can store any arbitrary XML on the server side.

Using this method, an agent can store private data on the server and retrieve it whenever necessary. Thanks to this mechanism agents can store the information needed to be persistent in their platform. An agent can recover this information even if it has been offline for a time. Part of the information stored in this XML Database can also be public. It is a simple method to advertise data to all the agents in the platform and make it accessible. The data stored might be anything, as long as it is valid XML. One typical usage for this namespace is the server-side storage of agent-specific preferences. Another usage is bookmark storage, scheduling and task information or any other interesting information.

## 4 Integrating Jabber in FIPA

The new Message Transport Protocol presented in this work is based on the data transfer representing the entire agent message, including the message envelope embedded in a XMPP message. Jabber uses three main building blocks in its communication: (the `<message/>` block, the `<presence/>` block and the `<iq/>` block), each of which has a different purpose. Jabber can perform all of its goals using only these blocks.

Next sections present the interface definition that has also been submitted to the FIPA consortium as a new preliminary specification.

### 4.1 Interface Definition

This new MTP has a proposed name called: `fipa.mts.mtp.xmpp.std`.

A FIPA Jabber message is represented by a `<message/>` Jabber element. Inside this element, there is the envelope of the FIPA message and the FIPA-ACL body of the message.

The message structure is detailed as follows:

- **Building Block**
  - The building block type is the `<message>` XMPP element.
  - The mandatory attribute `type` of the message element must have the value `normal` [2].
  - The `to` attribute of the message element is the physical XMPP address of the agent. (i.e. the Jabber ID of the agent or the Jabber ID of the platform)

```
<message id='2113' to='acc@foo.com' type='normal'>
        <body>
                (inform
                    :sender (agent-identifier
                        :name sender@bar.com
                        :addresses (sequence xmpp://acc@bar.com ))
                    :receiver
                       (agent-identifier
                        :name receiver@foo.com
                        :addresses (sequence xmpp://acc@foo.com )))
                    :language fipa-sl0
                    :ontology planning-ontology-1
                    :content   "((done task1))"
                )
        </body>
        <x xmlns='jabber:x:fipa' content-type='fipa.mts.env.rep.xml.std'>
                <envelope>
                    <params index="1">
                       <to>
                          <agent-identifier>
                            <name>receiver@foo.com</name>
                            <addresses> <url> xmpp://acc@foo.com </url> </addresses>
                          </agent-identifier>
                       </to>
                       <from>
                          <agent-identifier>
                             <name>sender@bar.com</name>
                             <addresses> <url> xmpp://acc@bar.com </url> </addresses>
                          </agent-identifier>
                       </from>
                       <acl-representation> fipa.acl.rep.string.std </acl-representation>
                       <payload-encoding> US-ASCII </payload-encoding>
                       <date> 20000508T042651481 </date>
                    </params>
                </envelope>
        </x>
</message>
```

**Fig. 3.** XMPP Example Message

– **Envelope Representation**
  - The envelope is placed in a `<x>` tag, which is used for ad-hoc extensions that add value, context, and information to any type of packet.
  - The XMPP namespace created for this purpose is called `jabber:x:fipa`.
  - The content type of the envelope is defined as an attribute of the `<x>` tag called `content-type`, where the value of the attribute is the component name given in each envelope specification.
  - The content of the `<x>` tag is the envelope body encoded in the defined representation.
– **Message Body**
  - The `<body>` tag of the message block contains the agent message payload. This payload is encoded with the representation described in the envelope.

Figure 3 shows an example of how a FIPA Jabber message is composed. This message is sent from the agent `sender@bar.com` to the agent `receiver@foo.com`,

which is resident on an Agent Platform that has an Agent Communication Channel with an external Jabber interface.

## 5 Conclusions

One of the main drawbacks detected on multi-agent platforms is the use of an inappropriate message transport protocol to communicate agents. To solve this problem, we have presented in this paper a new communication model using the Jabber Instant Messaging Protocol. We have also studied interesting Instant Messaging features and modified them in order to add new communication capabilities to multi-agent systems. This new communication model is FIPA compliant and supports all the required features, plus adding new ones that are made possible by using the Jabber technologies.

This protocol, which is based on Instant Messaging systems, uses a distributed network to route messages from one agent to another.

Nowadays, there are Instant Messaging Networks that are used by millions of people to communicate. These IM Networks can also be used by agents to communicate with each other.

Using the same IM Networks that humans do provides several advantages, such as a more comfortable and easy interaction between humans and agents. These networks have been extensively tested and can support a very high workload.

Other important matters are the new mechanisms presented in this paper (Presence Notification and Multi-User Conference) that can improve the agent communicative acts. These characteristics, which are not directly mentioned in the FIPA standard, perform new communication capabilities between agents which make them more versatile.

This work has been developed and tested with one of the most common and extended Multi-Agent Platforms: JADE. A plug-in [4] implementing this new Message Transport Protocol has been integrated in JADE and has been accepted by the JADE Team.

Thanks to this new MTP, a XMPP-capable platform (like the MAS platform developed by our research group: SPADE [11]) and a JADE platform can communicate with each other using the Jabber protocol. Actually, two or more JADE platforms can use this new XMPP plug-in to communicate their agents and take advantage of this new proposed Transport Protocol.

## Acknowledgements

This work is partially supported by the TIC2003-07369-C02-01 and TIN2005-03395 projects of the Spanish government.

## References

1. FIPA. Abstract architecture specification. Technical Report SC00001L, 2002.
2. Jabber Software Foundation. Extensible Messaging and Presence Protocol (XMPP): Core. Technical report, http://www.ietf.org/rfc/rfc3920.txt, October 2004.

3. Jabber Software Foundation. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. Technical report,http://www.ietf.org/rfc/rfc3921.txt , October 2004.
4. JADE XMPP-Plugin. http://jade.tilab.com/community-3rdpartysw.htm.
5. Jabber Software Foundation. http://www.jabber.org, 2005.
6. L. Mulet, J.M. Such, J. M. Alberola, V. Botti, A. Espinosa, A. Garcia, and A. Terrasa. Performance Evaluation of Open Source Multiagent Platforms. In *Autonomous Agents and Multi-Agent Systems Conference (AAMAS06)*, 2006.
7. E Cortese, F Quarta, and G Vitaglione. Scalability and performance of JADE message transport system. In *In Proc. of AAMAS Workshop on AgentCities, Bologna*, 16 June 2002.
8. David Camacho, Ricardo Aler, Csar Castro, and Jos M. Molina. Performance evaluation of Zeus, JADE, and skeletonagent frameworks. In *In Proc. of the 2002 IEEE Systems, Man, and Cybernetics Conference*, 2002.
9. K. Burbeck, D. Garpe, and S. Nadjm-Tehrani. Scale-up and Performance Studies of Three Agent Platforms. In *International Performance Computing and Communications Conference (IPCCC 2004)*, 2004.
10. L. C. Lee, D. T. Ndumu, and P. De Wilde. The stability, scalability and performance of multi-agent systems. *BT Technology Journal*, 1998.
11. M. Escrivá, Palanca J., G. Aranda, A. García-Fornes, V. Julian, and V. Botti. A Jabber-based Multi-Agent System Platform. In *Autonomous Agents and Multi-Agent Systems Conference (AAMAS06)*, 2006.