# Simulation Environment for Mobile Agents using Java for Wireless sensor Network

Sonia[1],
[1]Research Scholar,
Department of Computer Science & Engineering,
Mewar University, Rajasthan, India

Prof. S Niranjan[2],
[2]Professor,
Department of Computer Science & Engineering,
Mewar University,
Rajasthan, India

Dr. Yashpal Singh[3]
[3]Associate Professor,
Computer Science & Engineering Department.
GITAM, Kablana,
Jhajjar, Haryana, India

**Mobile agents are a distributed computing paradigm based on code mobility that has already demonstrated high effectiveness and efficiency in IP-based highly dynamic distributed environments. mobile agents may provide more benefits in the context of WSNs than in conventional distributed environments. In this paper we present the design, implementation and experimentation of Mobile Agent Platform an innovative Java-based framework for wireless sensor networks based Mobile agent technology which enables agent-oriented programming of WSN applications. This paper gives an overview of what the mobile agents are, what they should do and how they can be implemented in Java. It seems to be the best available language for making mobile agents roaming through the Internet for the time being. We describe our Java-based mobile agent platform to create mobile agent called JADE (Java Agent Development Environment) and present its programming interface. It also presents the JADE software describing its intended uses, hardware /software requirement as well as being a walkthrough of JADE internal architecture.**

***Index Terms—JADE (Java Agent Development Framework), WSN (wireless sensor network)***

## I. MOBILE AGENTS IN WIRELESS SENSOR NETWORK

The wireless sensor network is a technology which employs a large finite number of unattended, intelligent sensor nodes these are battery powered sensor nodes constrained in energy supply. As the internet constantly expands, the amount of available on-line information expands as well [3]. The issue of how to efficiently find, gather, and retrieve this information . An agent is a computer system that is situated in some environment, which autonomously sense the environment and respond accordingly. There is a need of a selection of criteria in an environment to establish an agent-based system. Distributed computing models can bring benefits to big data. And there are two categories of distributed computing: client/server model and mobile code model. Mobile agent model consists of remote computation, code on demand, and mobile agent model

Authors may prepare their papers for review using any word processor, one or two columns, single or double spaced. Please follow the writing style specified in this document.

### *A. Mobile Agent*

Mobile agents have been developed as an extension to and replacement of the client-server model. A mobile agent system provides the execution environment for mobile agents and also provide a framework in which mobile agent applications can be developed and managed. [2] The mobile agent technologies can be classified as:

- **Mobile Software Agent**: A mobile software agent in a sensor network is the executable process with its state and code, and it can migrate from a sensor to another one in the network.
- **Mobile Hardware Agent :** These devices are powerful hardware units of processing, memory, communication, and mobility capabilities. These agents traverse the network to collect information from ordinary sensor nodes.

There are three issues of Mobile Agent.

1) Its distributed nature:
   a) Focus on system architectures and protocols for managing executions of mobile agent objects.
   b) Security, fault tolerance.
2) How one could program this technology:
   a) Code mobility, safety, programming constructs
   b) Agent communication languages
3) The element of intelligence is absorb in it by the possibilities in Artificial Intelligence Research with:
   a) Focus on intelligence, learning, and cooperation

### *B. Life Cycle of a Mobile Agent*

The life of a mobile agent is modeled with the stages it goes through called lifecycle mode. The stages of the model are: Creation, Starting Deactivation, Disposal and cloning.

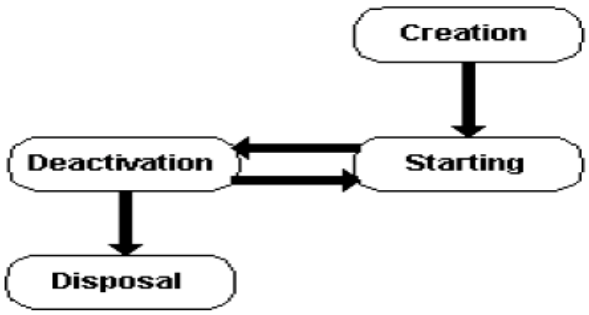


Fig 1: Life cycle of Agent

- Creation: Creation of the agent is done only once when new agent is created. Every agent gets its unique id, initial state and then it is prepared for further instructions.
- Starting : The server initializes the agent and gives it a thread of execution after which the agent resumes its execution. All the agents are executed in parallel on the host.
- Deactivation: The agent stops all its calculations and stores its state and intermediate results to a disk. It can be used for making checkpoints before performing some unsecured operations or moving to unknown host.
- Disposal: means the agent terminates all its activity and frees all resources it's using. After that, its state is lost forever.

There are various mobile agent system like Agent TCL, ARA, JADE, WADE, Telescript and Odyssey, Concordia, Mole, Tacoma, Sumatra, Voyager and java to Go. Though these systems differ in their goals, motivations, and implementations, they all (more or less) provide common functionalities that support: the migration of agents, the communication between agents, various programming/interpreted languages, and various forms of security.

## II. JADE (JAVA AGENT DEVELOPMENT ENVIRONMENT)

JADE conceptualizes an agent as an independent and autonomous process that has an identity, possibly persistent, and that requires communication with other agents in order to fulfill its tasks. This communication is implemented through asynchronous message passing and by using an Agent Communication Language (ACL) with a well-defined and commonly agreed semantics [3]. JADE (Java Agent Development Framework) is a software framework to develop distributed agent-based applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. It also includes [6]:

- A runtime environment where JADE agents can "live" and that must be active on a given host before one or more agents can be executed on that host.
- A library of classes that programmers have to/can use (directly or by specializing them) to develop their agents.
- A suite of graphical tools that allows administrating and monitoring the activity of running agents.

### *A. Programming Environment for JADE*

The most important issues about the mobile agents is the selection of implementation language, System requirements. In earlier days, the mobile agent programming which resulted in languages like TCL, Oblique, and Rosette; even C and C++ languages were used for this purpose [4].

### *B. The system requirement to run JADE*

Jade has been completely implemented in Java. Its capabilities can only be fully exploited by using the Java programming language. The minimal system requirement is the JDK 1.2 Runtime or later. Java Developing Kit 1.1 (JDK 1.1) and JDK 1.2 with their possibilities, like Remote Method Invocation (RMI) that allows object methods to be called over the network.

### *C. Features of JADE*

JADE offers the following list of features to the agent programmer:

- FIPA-compliant Agent Platform, which includes the AMS (Agent Management System), the DF (Directory Facilitator), and the ACC (Agent Communication Channel), all automatically activated at the agent platform startup;
- Distributed agent platform. Agents are implemented as one Java thread and Java events are used for effective and lightweight communication between agents on the same host. Parallel tasks can be still executed by one agent, and JADE schedules these tasks in an efficient way;
- Directory Facilitator (DF): This console allows to register/deregister/modify/search for agents and services, it allows also to federate the DF with other DF's and to propagate searches to the federated DFs.
- FIPA-compliant MTPs (Message Transport Protocol) to connect different agent platforms. It provide transport mechanism and interface to send/receive messages to/from other agents.
- Dummy Agent, a tool to compose custom messages to send and display the received messages. Messages can also be saved to a file and loaded from a file. The tool allows to simulate by hand the communicative behavior of an agent.
    - Automatic registration of agents with the AMS;
    - Graphical user interface to remotely manage the life-cycle of agents and agent containers.

## III. ARCHITECTURE OF THE JADE AGENT PLATFORM

The JADE Agent Platform complies with FIPA97 specifications and includes all those mandatory agents that manage the platform that is the AMS, and the DF. The AMS (Agent Management System) that provides the naming service (i.e. ensures that each agent in the platform has a unique name) .The DF (Directory Facilitator) that provides a service by means of which an agent can find other agents providing the services he requires in order to achieve his goals

### A. *Containers and Platforms*

Each running instance of the JADE runtime environment is called a **Container** as it can contain several agents. The set of active containers is called a **Platform**. A single special **Main container** must always be active in a platform and all other containers register with it as soon as they start. If another main container is started somewhere in the network it constitutes a different platform to which new normal containers can possibly register.

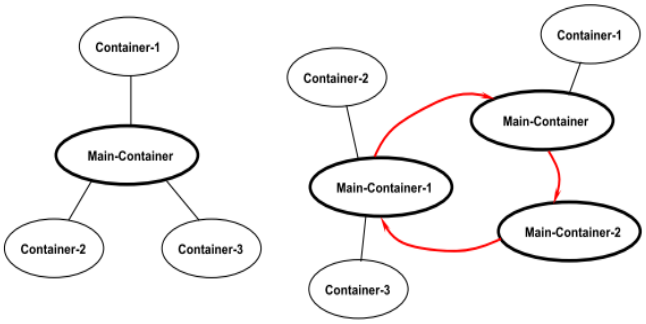


Fig 2: Containers and Main-Container

Main Container supported following features are:

- Managing the Container Table (i.e. the set of all the nodes that compose the distributed platform).
- Managing the Global Agent Descriptor Table (i.e. the set of all the agents hosted by the distributed platform, together with their current location).
- Managing the MTP table (i.e. the set of all deployed MTP endpoints, together with their deployment location).
- Hosting the platform AMS agent.
- Hosting the platform Default DF agent

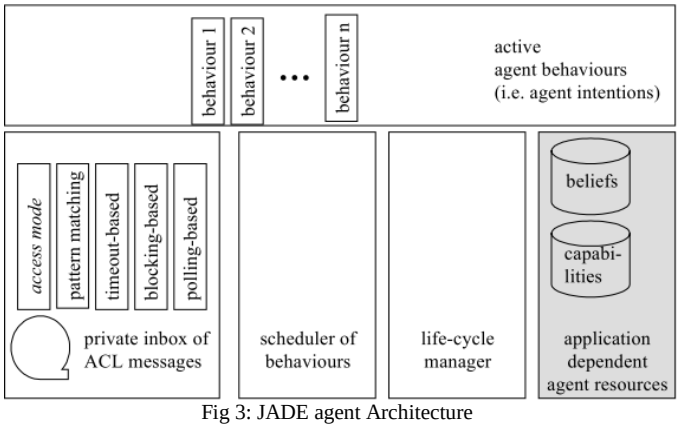


Fig 3: JADE agent Architecture

JADE distributed architecture relies on a special node, named Main Container, to coordinate all other nodes and keep together the whole platform. Ordinary containers will then be able to connect to the platform through any of the active Main Container nodes; the different copies will evolve together using cross-notification.

### B. *Agent Task– The Behavior Class*

A behavior represents a task that an agent can carry out and is implemented as an object of a class that extends jade.core. behaviours. Behavior. Each class extending Behaviour must implement the action() method, that actually defines the operations to be performed when the behavior is in execution and the done() method, that specifies whether or not a behavior has completed and have to be removed from the pool of behaviors an agent is carrying out. This means that when a behavior is scheduled for execution its action() method is called and runs until it returns. Therefore it is the programmer who defines when an agent switches from the execution of a behavior to the execution of the next one.

In JADE there is a single Java thread per agent. Since JADE agents are written in Java, however, programmers may start new Java threads at any time if they need. If you do that, remember to pay attention since the advantages mentioned in this section are no longer valid. The path of execution of the agent thread is depicted in Figure below:

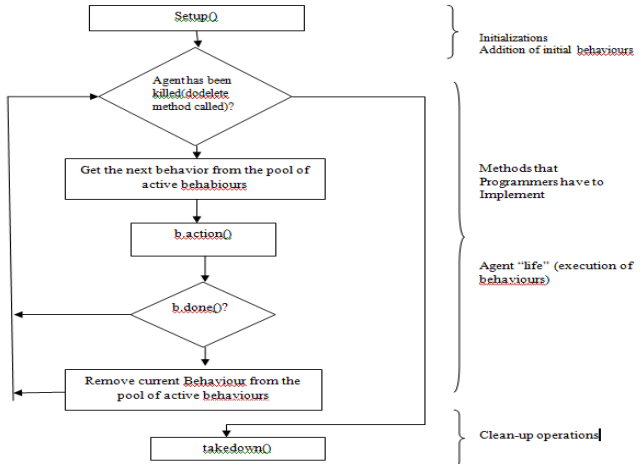


Fig 4: Agent Thread path of execution

## IV. CREATING AND EXECUTING JADE AGENT IN ECLIPSE ENVIRONMENT

Before creating and JADE agent we have to download JADE software from the JADE web site http://jade.tilab.com/. We get five compressed files: JADE, JADE-examples, JADE-doc, JADE-bin, JADE-all

We installed JADE-bin, Eclipse and jdk-1.4 archive in our system. Having uncompressed the archive file, a directory tree is generated whose root is jade and with a lib subdirectory. This subdirectory contains some the jade.jar jar file that has to be added to the CLASSPATH environment variable. To work with command prompt we have to set the CLASSPATH but here we use eclipse to implement agent code. Before creating agent, create a java project in eclipse as follows:

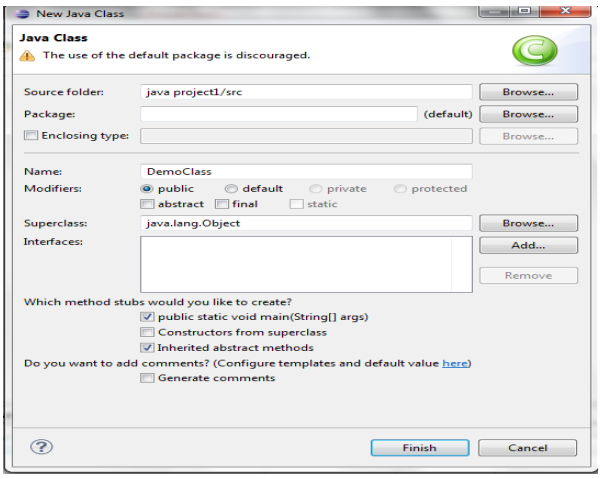


Fig 5: Java Project in eclipse

### A. *Eclipse Configuration*

- Importing External files: Create a new project in eclipse and add all jars of JADE in the library. For importing packages in jade library Go to project->properties->java build path-> Libraries tab Add external Jar file to work with JADE Packages.

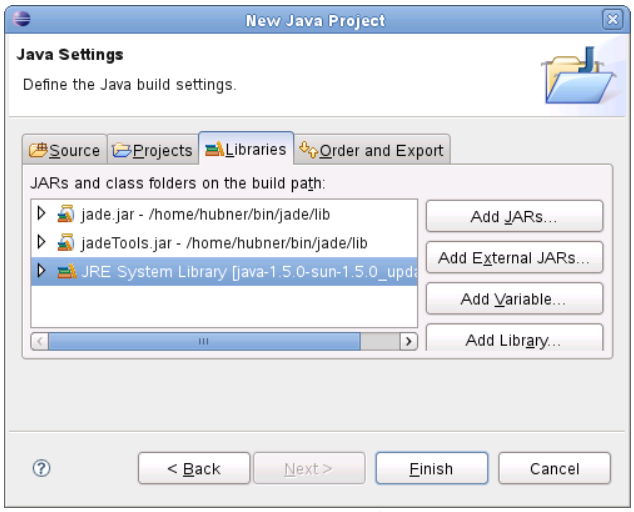


Fig 6: Adding External files in project

- Creating Project and Agent Class: Now we can use JADE classes directly in our project. Add a new class named HelloAg as a public class to the project, specified jade.core.Agent as the parent class. Override the setup method in the new class to say hello to the console. Create a simple hello agent using the eclipse new class wizard and fill the fields.

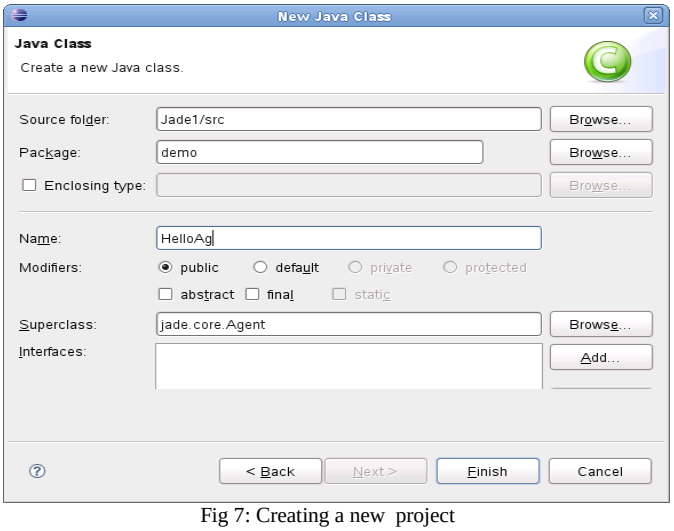


Fig 7: Creating a new project

The finished agent class looked as follows:

```
import jade.core.Agent;

public class HelloAg extends Agent
{
    protected void setup()  {
        System.out.println("Hellod,
I'm" +getLocalName());
    }
}
```

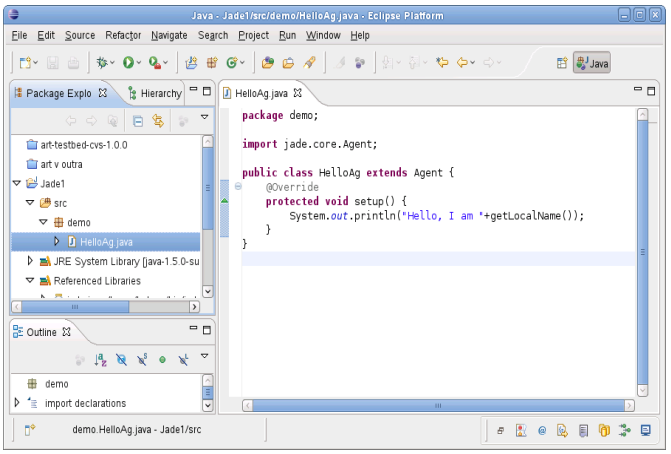


Fig 8: Adding HelloAg Agent in project

- Run Configuration: This agent can not run as a program, we need to start a JADE container and add the agent inside the container. To start a container to run this agent do the following run configuration:

  For that we will do:

  Run->Run Configure-> Java Application-> right click-> New This window will open and in this window write in Main Class text box as jade.Boot.

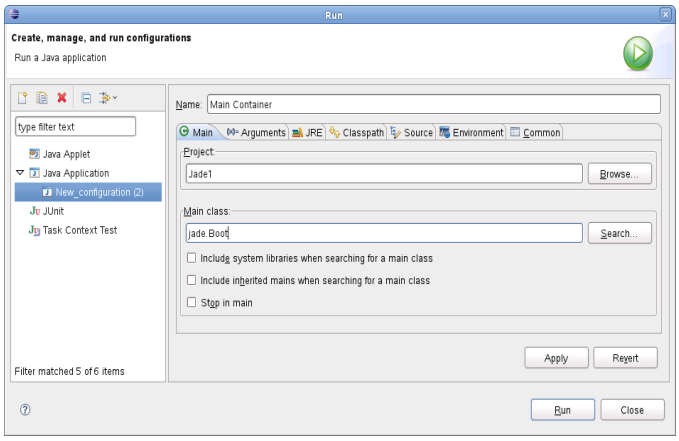


Fig 9: Run configuration

- Passing arguments to an agent: These arguments can be retrieved, as an array of Object, by means of the getArguments() method of the Agent class. Arguments on the command line are specified included in parenthesis and separated by spaces. C:\ jade > java jade. Boot Agent: DemoAgent. But in eclipse we can pass argument as: Now in click on Argument tab and write as

  -gui bob: demo.HelloAg

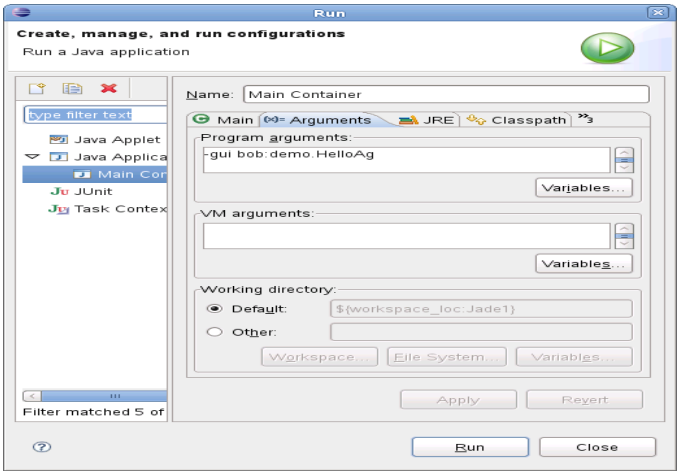


Fig 10: Passing argument to agent

- Run: Finally, execute the application by clicking on Apply and run.

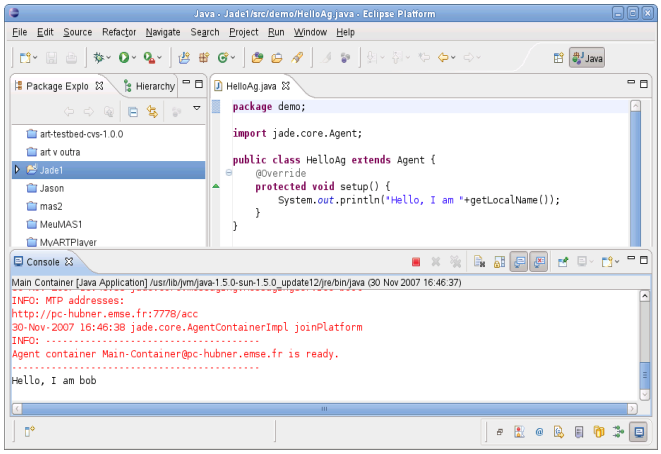


Fig 11: Execution of agent

## V. REMOTE MONITORING AGENT

The above output is the JADE disclaimer that is printed out each time the JADE runtime is started. The Remote Monitoring Agent (RMA) allows controlling the life cycle of the agent platform and of all the registered agents. The distributed architecture of JADE allows also remote controlling, where the GUI is used to control the execution of agents and their life cycle from a remote host.

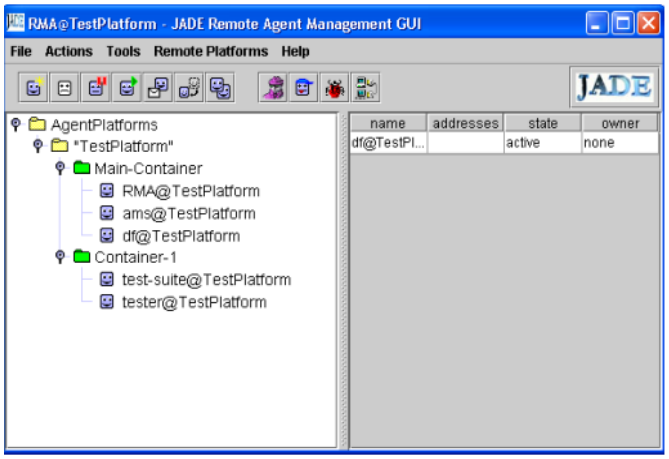


Fig 12: Remote Monitoring Agent

### A. *Agent Termination*

Agent can be suspended, Resumed, killed, migrated, cloned, Freezed as well as can send messages to another agent uging ACL language.

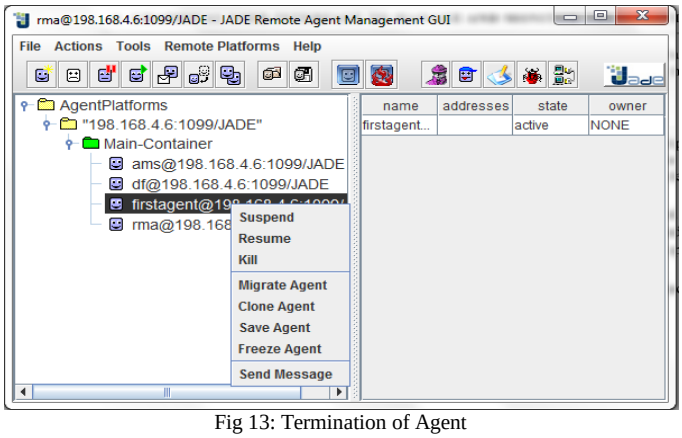


Fig 13: Termination of Agent

### B. *Communication– The ACL Message Classes*

The communication paradigm adopted is the asynchronous message passing. Each agent has a sort of mailbox (the agent message queue) where the JADE runtime posts messages sent by other agents. Whenever a message is posted in the message queue the receiving agent is notified. A message in JADE is implemented as an object of the **jade.lang.acl.ACLMessage** class that provides get and set methods for handling all fields of a message.

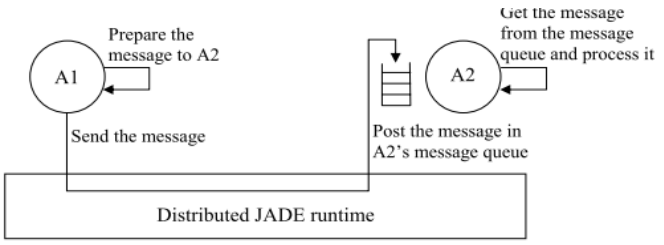


Fig 14: The JADE asynchronous message passing paradigm

*a.* JADE messaging Architecture:

A JADE platform is a distributed environment composed of several run-time containers launched over one or more hosts across a network. In the platform, an important role is played by the Main Container where the FIPA service agents live. The ACL language Messages exchanged by JADE agents have a format specified by the ACL language defined by the FIPA international standard for agent interoperability. There is two type of agent communication , intra-platform and inter-platform agents communication.

**Intra-platform:** It involves agents living in the same platform and JADE uses its internal message transport protocols (IMTPs) for implementing delivery services. JADE delivers messages using event passing when both the sender and the receiver agents live in the same container.

**Inter-platform:** In the inter-platform scenario, interaction among agents is achieved by the Agent Communication Channel (ACC), which is physically distributed across all the containers of the platform. each container can be launched with one or more message transport protocols (MTPs) and the entire platform is able to internally route the messages and select the best MTP for each situation JADE provides a Java interface both for implementing new ad-hoc IMTP and MTP. IMTP has been implemented for JADE integrated with Leap to provide inter-container communication in wireless environment.

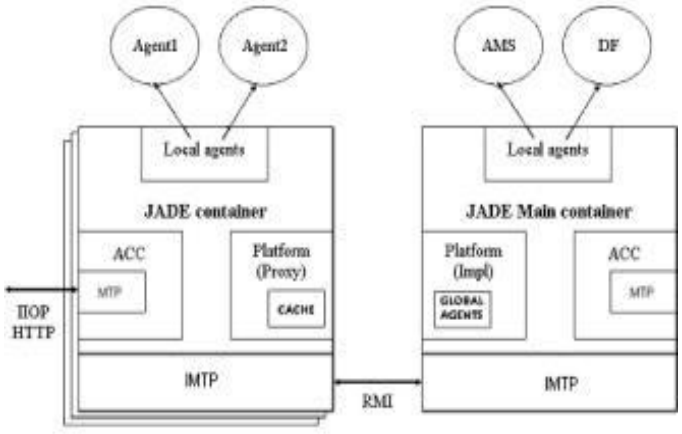


Fig 15: Components of Jade Messaging Architecture

*b.* Sending and Receiving Messages:

This format comprises a number of fields and in particular:

- The sender of the message by calling send() method.
- The list of receivers by mean of specifying receive () method.
- The communicative intention indicating what the sender intends to Assuming you have an Agent ID (AID) for the recipient, sending a message is easy[7]

```
ACLMessage msg = new
ACLMessage(ACLMessage.INFORM);
msg.setContent("Message #" + n );
msg.addReceiver( new AID( name, AID.ISLOCALNAME )
);
                    send(msg);
```

We can add several receivers to the message by addReceiver() method and the one send broadcasts it to all of them. There are 2 basic ways for the receiver to get its messages. By using blockingReceive() and with receive() methods.

Using BlockingReceive() method, the receiving agent suspends all its activities until a message arrives:

```
ACLMessage msg = blockingReceive();
```

The second method, with receive(), examines the message queue, returning a message if there is one or null otherwise. This is the normal technique used when an agent is involved in parallel activities and has multiple active Behaviours.

```
ACLMessage msg = receive();
  if (msg != null)
                    <.... handle message...>
   else
<... do something else like block() ...>
```

Each Agent Container is an RMI server object that locally manages a set of agents. It controls the life cycle of agents by creating, suspending, resuming and killing them. Besides it deals with all the communication aspects by dispatching incoming ACL messages, routing them according to the destination field (:receiver) and putting them into private agent message queues; for outgoing messages, instead, the Agent Container maintains enough information to look up receiver agent location and choose a suitable transport to forward the ACL message. Following figure shows the Agent registered with the DF.

## VI. CONCLUSION

JADE design tries to put together abstraction and efficiency, giving programmers easy access to the main FIPA standard assets while incurring into runtime costs for a feature

only when that specific feature is used. This "pay as you go" approach drives all the main JADE architectural decisions: from the messaging subsystems that transparently chooses the best transport available, to the address management module, that uses optimistic caching and direct connection between containers.

Since JADE is a middleware for developing distributed applications, it must be evaluated with respect to scalability and fault tolerance, which are two very important issues for distributed robust software infrastructures.

## REFERENCES

[1] A. Fuggetta, G. Picco, G. Vigna, "Understanding Code Mobility", IEEE Transactions on Software Engineering, Vol. 24, No. 5, pp. 352-361, May 1998.
[2] David M. Chess, Colin G. Harrison, and Aaron Kershenbaum. "Mobile Agents: Are they a good idea?", IBM Research Report.
[3] http://jade.tilab.com/community-faq.htm
[4] Damir Horvat 1, 3 , Dragana Cvetkoviü "Mobile Agents and Java Mobile Agents Toolkits" Proceedings of the HICSS – 2000, Maui, Hawai'i, USA, January 2000
[5] Peine, H., Stolpmann, T., "The Architecture of the Ara Platform for Mobile Agents, " Department of Computer Science, University of Kaiserslautern, Germany, 1998. http://www.unikl.de/AGNehmer/Projekte/Ara/Doc/
[6] http://jade.tilab.com/doc/JADEProgramming-Tutorial-for-beginners.pdf
[7] http://www.iro.umontreal.ca/~vaucher/Agents/Jade/primer4.html