Chat Platform with JADE

Sai Krishna Kalyan
Universite Lumiere Lyon 2

Abstract

This document is one of the deliverables of a case study project. The source code and the execution script can be found on the github repository. The main purpose of this project is to implement a chat service using Jade. JadeAndroid libraries were used to implement chat client in Android. Swing GUI and Jade was used to implement chat client. This service consists of three modules namely Android chat client, Standard chat client and a chat server.

1 Introduction

JADE is a distributed agents platform, which has a container for each host where you are running the agents. Additionally the platform has various debugging tools, mobility of code and content agents, the possibility of parallel execution of the behaviour of agents, as well as support for the definition of languages and ontologies. Each platform must have a parent container that has two special agents called AMS and DF. AMS (Agent Management System) controls the platform. Is the only one who can create and destroy other agents, destroy containers and stop the platform. DF (Directory Facilitator) provides a directory which announces which agents are available on the platform.
This Chat platform will consists of a chat server that will have a Main Container that manages the interaction with components. There will also be a standard chat client that will allow PC users to chat using the standard JVM and agents. Android client will allow participating to the chat with an android device.

2 Jade Platform

JADE Architecture JADE is a middleware which facilitates the development of multi-agent systems under the standard FIPA for which purpose it creates multiple containers for agents, each of them can run on one or more systems. Jade architecture is a container-based platform. In-side a container several agents can be run acting with certain behaviours. There can be multiple containers being executed on different hosts thus achieving distributed platform.
An application based on JADE is made of a set of components called Agents each one having a unique name. Agents execute tasks and interact by exchanging messages.
Communication is based on an asynchronous message passing paradigm. Message format is defined by the ACL language defined by FIPA. (Agent Communication Language)
FIPA is an IEEE Computer Society standards organisation that promotes agent-based technology and the interoperability of its standards with other technologies.
There are three libraries used to implement this platform namely jade.jar, chatOntology.jar, chatStandard.jar. The chatOntology.jar contains precompiled classes implementing concepts, predicates and actions that must be known by agents in the system. These classes are packaged in a separate jar file since thy must be available in all modules. The chatStandard.jar file contains precompiled classes (but ontology ones) implementing the Standard Chat Client and Chat Manager Agent modules.

Android Platform

Applications intended to run in an Android environment are fully written in Java. JadeAndroid is used to develop this chat application which can be downloaded from the JADE website. It is an add-on that provides support for using Jade Leap on Android Platform. Android is the software stack for mobile devices including the operating system released by Google. According to the Android philosophy all operations like processing and communication are asynchronous. Android SDK and Jade Libraries were used to develop this application. We need to import jadeAndroid.jar to the workspace so that we can implement the two service class jade.android.RuntimeService and jade.android.MicroRuntimeService.

2.1 Project Setup

This paper is based on development and testing in the following setup.

- Eclipse Development IDE
- Jade and JadeAndroid Libraries
- Ant Build Tool
- Android SDK
- Android AVD

Jade Android Implementation

The android manifest need to be given permission to access the internet to allow communication with the chat server. All the user interface related artefacts are under the layout folder. The application code is organised into two classes chat.client.gui and chat.client.agent.
The chat.client.gui package contains Android related classes such as the Application class and Activity classes. The chat.client.agent package contains the class of the Chat Client Agent and the interface that it provides to the GUI components.
Consistent with the Android architecture, the JADE runtime is wrapped by an Android service. More specifically, the jadeAndroid.jar library includes two service classes jade.android.RuntimeService and jade.android.MicroRuntimeService that wrap a full container and a split container, respectively.

Binding to the service
The first operation to activate the JADE runtime from an Android Activity is to bind to the MicroRuntimeService. As a result a jade.android.MicroRuntimeServiceBinder object is retrieved such that it will be possible to perform all JADE management operations.

```
serviceConnection = new ServiceConnection() {
public void onServiceConnected(ComponentName
className, IBinder service) {
// Bind successful
microRuntimeServiceBinder =
(MicroRuntimeServiceBinder)
service;
...
};

};
```

Starting Agent Container
Having retrieved the MicroRuntimeServiceBinder object it is now possible to start a JADE Split Container as shown in the code snipped below

```
microRuntimeServiceBinder.
startAgentContainer(pp,
new RuntimeCallback<Void>() {
@Override
public void onSuccess(Void thisIsNull) {
...
}
```

Starting an Agent
The ApplicationContext is passed to the agent as argument. This will allow the agent to access Android API when needed.

```
microRuntimeServiceBinder.startAgent(nickname,
ChatClientAgent.class.getName(),
new Object[] { getApplicationContext() },
new RuntimeCallback<Void>() {
@Override
public void onSuccess(Void thisIsNull) {
// Agent successfully started
...
} );
```

Agent interactions
The handleSpoken() method is used in the ChatActivity to make the agent forward a chatted message to all chat participants.
The getParticipantNames() is used in the ParticipantsActivity to show the list of people currently participating to the chat.

```
public interface ChatClientInterface {
public void handleSpoken(String s);
public String[] getParticipantNames();
}
```

3 Starting the Chat Platform

To start the chat platform we need to start the server and the clients as mentioned in the steps below.

- From a shell/DOS-prompt move to the folder standard/bin
- Run startPlatform.sh to start the chat server
- Run startChatParticipant.sh to start chat client
- Enter a Nick Name and click ok

Android Client

- Install the chatClient.apk application on your phone
- Launch Jade Chat application
- Change the Container Parameters if required. (host/port)
- Enter a Nick Name and you can start chatting.

4 Conclusion and Future Work

The current application is based on Ant builds. This could be upgraded to Gradle builds for android and Maven builds for Java. Also improvements with the layout and the UI to improve the look and feel of the application as the current UI is quite outdated.

References

[1] Github.com/krishnakalyan3/JadeChatPlatform

[2] Jade Android Programming Tutorial http://jade.tilab.com/doc/tutorials/JadeAndroid-Programming-Tutorial.pdf

[3] Jade Programming Tutorial http://jade.tilab.com/documentation/tutorials-guides/