MAS: Activity 5 – The JADE platform Intro

Alexandru Sorici
25.03.2018

The **Java Agent DEvelopment Framework (Environment) – or JADE**[1] – is a popular agent developement framework for the development of **software agents** that feature **strong mobility** – are able to move from one machine to another – and communicate through **FIPA-ACL**[2] messages (agents are FIPA-compliant).

Jade agents are based on various types of **behaviors**. Popular behaviors are **one-shot behaviors**, **ticker and waker behaviors** and **cyclic behaviors**. Use the Resources to learn more about Jade programming (see also the resources on `cs.curs`[3]page).

This introductory lab to the JADE platform is meant to accustom you to the following programming elements:

- start a JADE platform (containers and agents) programmatically
- send arguments to agents at setup
- add and remove various types of simple behaviors: one-shot, ticker, waker or cyclic behaviors
- create and send ACLMessages; use appropriate meta-properties (e.g. reply-with, reply-by, in-reply-to)
- create filter templates for received ACLMessages

Scenario setup
To exercise these elements, the proposed scenario is the following. A set of agents is configured in a tree-like hierarchy of father-child relationships. Each agent is initially given only the ID of their parent agent. When started, each parent agent has a limited period in which it accepts registration requests from child agents. Similarly, the first order of business for each child agent is to register with their parent. These two start behaviors are already implemented and exemplify the usage of WakerBehaviors and TickerBehaviors.

At setup, the agents also receive an integer value. After the initial registration phase, the objective is that each of the agents becomes aware of the maximum value in the entire group. To do this they engage in a 3-step process.

1. (Procedure start): The parent agents request the value from all their children.
2. (Upward pass): The parent agents await message from all their registered children. They compute the maximum among the received values and forward the maximum to their parent. The root agent in the tree is where the upward pass ends
3. (Downward pass): After the upward pass, each agent awaits a message from its parent, in response to upward pass to receive the final maximum value. Upon

[1] http://en.wikipedia.org/wiki/Java_Agent_Development_Framework
[2] http://en.wikipedia.org/wiki/FIPA-ACL
[3] http://cs.curs.pub.ro/2015/mod/folder/view.php?id=5498

reception, the agent will forward the maximum value to its children (if it has any). After receiving the final maximum value, each agent prints it out and the terminates.

The parent-child relationships and the value for each agent are configured in a `config.csv` file under the `data` folder.

Roadmap
To enable the 3-step process described above, you can use the following recommendations:

- (Procedure start)
  - Use a one-shot behavior to trigger the procedure (i.e. ask the child agents to send their values upward). For the message you are sending set protocol, conversation id, as well as reply-with meta-properties to be able to identify the type of conversation (protocol), which child your talking to (conversation i) and where in the conversation you currently are (reply-with). Use ACLMessage.REQUEST as the message performative.
  - In the child agents, use a cyclic behavior to listen for incoming requests that trigger the procedure for determining the maximum value. Define a MessageTemplate to listen for such requests and condition it on the sender (the parent), the performative, the protocol and the conversation id, which should tell you what kind of request it is (i.e. the one for determining the max value).
- (Upward pass)
  - In leaf child agents, define a one-shot behavior to send the value. Add the behavior as a consequence of receiving a REQUEST from a parent agent. In the message set the conversation-id and in-reply-to meta-properties to match the values received in the initial request. Set the reply-with meta-property value that the parent agent must use when announcing the final maximum value in the downward pass stage.
  - In the parent and intermediary agents, extend a simple behavior to await for all the upward-pass value sending messages. Define a MessageTemplate for the messages from the child agents. Again, use a MessageTemplate to await for messages filtered by, protocol, conversation id and in-reply-to meta properties.
- (Downward pass). Work similarly to the upward pass. Intermediary and leaf child agents should extend a simple behavior once they have sent their computed max value, to listen for the final response from their parents. Define a MessageTemplate to filter for the same conversation id, but using the in-reply-to value set during the upward pass. If the agent is a parent, use a one-shot behavior to forward the final maximum value.
  After the agent has forwarded the value, it will call doDelete() to finish its activity. In the `takeDown()` method, the agent should print the maximum value.

Must read (from resources:)

- Jade Programmer's Guide pages 47–48 (section 3.8) to see how to start the platform programmatically.
- Section Agent Tasks in Jade-Basic to learn about behaviors.
- Section Agent Communication in Jade-Basic to learn about messaging.

Resources:
Root Jade site: `http://jade.tilab.com/`

Main documentation + papers page: http://jade.tilab.com/papers-index.htm
Documentation at: http://jade.tilab.com/doc/index.html
Tutorial: http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf
Programmer's Guide: http://jade.tilab.com/doc/programmersguide.pdf

Cum să raportați activitatea:

- **la sfârșitul laboratorului:** trimiteți arhiva conform cu instrucțiunile de mai jos.
- **la terminarea taskurilor** aferente laboratorului (înainte de următorul laborator, altfel cu depunctare): trimiteți din nou arhiva, conform cu aceleași instrucțiuni, eventual adăugând ceva la nume.

Conținutul arhivei: numai directorul `src`, arhivat într-o arhivă cu numele `PrenumeNume_MAS-N.zip`, unde `N` este numărul laboratorului pe care l-ați rezolvat.

Cum trimiteți: trimiteți arhiva în atașament la un mesaj către adresa alex.sorici+mas@gmail.com. Dacă adresa este corectă și există atașament, veți primi un mesaj automat de confirmare.

Notă: Folosiți adresa de mai sus numai pentru a trimite activitatea de laborator. Pentru alte probleme folosiți modalitățile de contact indicate la curs.