JADE API for Mobile Agents

Integrated Master's in Informatics Engineering

Intelligent Agents

2017/2018

Synthetic Intelligence Lab

Filipe Gonçalves

Paulo Novais

ISLab

Mobile Agents

- A Mobile Agent is an executing program that can **migrate**, at **times** of its own choosing, from **machine to machine** in a **heterogeneous network**
- Mobile Agents are an effective paradigm for **distributed applications**, and are particularly attractive for partially **connected computing**
- Using JADE application developers can **build** mobile agents, which are able to **migrate** or **copy** themselves across **multiple network hosts**

Agent Life Cycle

- **INITIATED:** Agent object is built, but hasn't registered itself yet with the AMS
- **ACTIVE :** Agent object is registered with the AMS, has a regular name and address and can access all the various JADE features
- **SUSPENDED :** the Agent object is currently stopped since Its internal thread is suspended
- **WAITING :** the Agent object is blocked, waiting for an event
- **TRANSIT:** the mobile Agent enters this state while it is migrating to the new location. The system continues to buffer messages that will then be sent to its new location

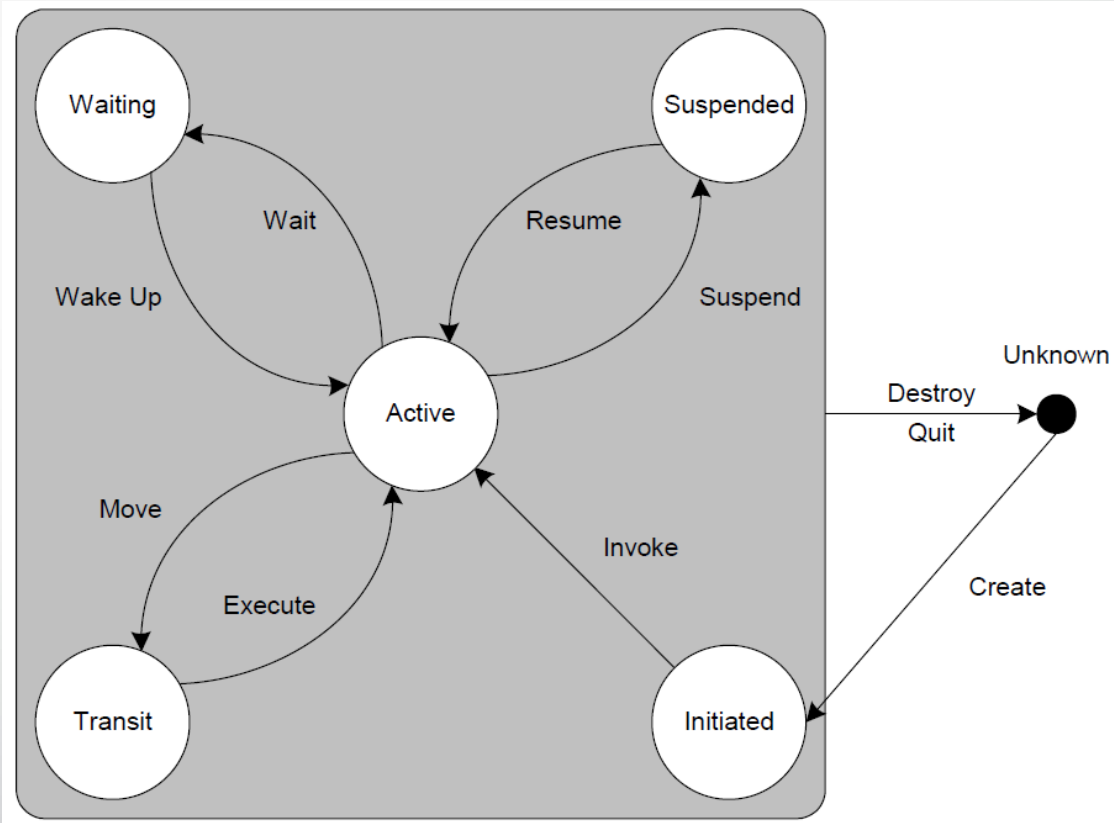

Figure 3 - Agent life-cycle as defined by FIPA.

Mobile Agents

- **Moving** or **cloning** is considered a **state transition** in the life cycle of the agent, which can be initiated either by the agent itself or by the Agent Management Service (AMS)
- **Moving** or **cloning** an agent involves sending its code, resources and state through a network channel
- Mobile agents need to be **location aware** in order to decide **when** and **where** to move
- ***JADE*** makes available a couple of matching methods in the Agent class for resource management and provides a proprietary ontology (***jade-mobility-ontology***) holding the necessary concepts and actions, contained within the ***jade.domain.mobility*** package

JADE API for Agent Mobility

- **Java Methods**:
  - ***doMove():*** allows a JADE Agent class to migrate elsewhere
    - **INPUT**: a jade.core.Location single parameter representing the intended destination
  - ***doClone():*** allows a JADE Agent class to spawn a remote copy of itself under a different name
    - **INPUT**: a jade.core.Location parameter (intended destination) and a String containing the name of the new cloned Agent

- **NOTE**: ***jade.core.Location*** is an abstract interface. Applications agents must ask the AMS for the list of available locations and choose one or request another agents location.

JADE API for Agent Mobility

- **doMove() sub-methods:**
  - **beforeMove()**: called at the starting location normally to release any local resources used by the original instance
  - **afterMove()**: called at the destination location and executed when the agent transition is completed
- **doClone() sub-methods [similar to beforeMove() and afterMove()] :**
  - **beforeClone()**
  - **afterClone()**

JADE Mobility Ontology

- JADE provides the class ***jade.domain.mobility.MobilityOntology*** which contains all the **concepts** and **actions** needed to support **agent mobility**
- The ontology allows access to a single, shared instance of the JADE mobility ontology, through the ***getInstance()*** method, for **accessing** the **AMS**
- The ontology contains **five concepts** and **two actions**:
  - **Concepts**:
    - ***mobile-agent-description***
    - ***mobile-agent-profile***
    - ***mobile-agent-system***
    - ***mobile-agent-language***
    - ***mobile-agent-os***
  - **Actions**:
    - ***move-agent***
    - ***clone-agent***

JADE Mobility Ontology

- ***mobile-agent-description***: describes a mobile agent going somewhere. It is represented by the ***MobileAgentDescription***

Slot Name	Slot Type	Mandatory/Optional
Name	AID	Mandatory
destination	Location	Mandatory
agent-profile	mobile-agent-profile	Optional
agent-version	String	Optional
signature	String	Optional

JADE Mobility Ontology

- ***mobile-agent-profile:*** describes the computing environment needed by the mobile agent. It is represented by the ***MobileAgentProfile*** class.

Slot Name	Slot Type	Mandatory/Optional
system	mobile-agent-system	Optional
language	mobile-agent-language	Optional
os	Mobile-agent-os	**Mandatory**

JADE Mobility Ontology

- ***mobile-agent-system:*** describes the runtime system used by the mobile agent. It is represented by the ***MobileAgentSystem*** class
- ***mobile-agent-language:*** describes the programming language used by the mobile agent. It is represented by the ***MobileAgentLanguage*** class
- ***mobile-agent-os:*** describes the operating system needed by the mobile agent. It is represented by the ***MobileAgentOS*** class

Slot Name	Slot Type	Mandatory/Optional
name	String	Mandatory
major-version	Integer	Mandatory
minor-version	Integer	Optional
dependencies	String	Optional

JADE Mobility Actions

- **Every mobility related action** can be **requested** to the **AMS** through a FIPA-request protocol, with ***jade-mobility-ontology*** as ontology value and ***FIPA-SLo*** as language value
- ***move-agent:*** the action of moving an agent from a location to another. This action moves the agent identified by the ***name*** and ***address*** slots of the ***mobile-agentdescription*** to the location present in the ***destination*** slot. It is represented by the ***MoveAction*** class.
  - **INPUT**: single, unnamed slot of type ***mobile-agent-description***
- ***clone-agent:*** the action performing a copy of an agent, possibly running on another location. It is represented by the ***CloneAction*** class.
  - **INPUT**: two unnamed slots:
    - ***mobile-agent-description*** type
    - String agent_name

JADE Mobility Actions

- The agent has to create a new ***MoveAction*** object, fill its argument with a suitable ***MobileAgentDescription*** object, filled in turn with the ***name*** and ***address*** of the **agent to move** and with the ***Location*** object for the destination.
- Then, a single call to the ***Agent.getContentManager().fillContent(..,..)*** method can turn the ***MoveAction*** Java object into a **String** and write it into the content slot of a suitable request ACL message
- **Example:**

```
 private ACLMessage request;

public GetAvailableLocationsBehaviour(MobileAgent a) {
  // call the constructor of FipaRequestInitiatorBehaviour
  super(a, new ACLMessage(ACLMessage.REQUEST));
  request = (ACLMessage)getDataStore().get(REQUEST_KEY);
  // fills all parameters of the request ACLMessage
  request.clearAllReceiver();
  request.addReceiver(a.getAMS());
  request.setLanguage(FIPANames.ContentLanguage.FIPA_SL0);
  request.setOntology(MobilityOntology.NAME);
  request.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
  // creates the content of the ACLMessage
  try {
    Action action = new Action();
    action.setActor(a.getAMS());
    action.setAction(new QueryPlatformLocationsAction());
    a.getContentManager().fillContent(request, action);
  }
  catch(Exception fe) {
      fe.printStackTrace();
  }
  // creates the Message Template
  // template = MessageTemplate.and(MessageTemplate.MatchOntology(MobilityOntology.NAME),template);
  // reset the fiparequestinitiatorbheaviour in order to put new values
  // for the request aclmessage and the template
  reset(request);
}
```

AMS Request Example

- To **move** agent ***Peter*** to the location called ***Front-End***, it must send to the AMS the following ***ACL request message***:

```
(REQUEST
    :sender (agent-identifier :name RMA@Zadig:1099/JADE)
    :receiver (set (agent-identifier :name ams@Zadig:1099/JADE))
    :content (
        (action (agent-identifier :name ams@Zadig:1099/JADE)
            (move-agent (mobile-agent-description
                :name (agent-identifier :name Johnny@Zadig:1099/JADE)
                :destination (location
                :name Main-Container
                :protocol JADE-IPMT
                :address Zadig:1099/JADE.Main-Container )
                )
            )
        )
    )
    :reply-with Req976983289310
    :language FIPA-SL0
    :ontology jade-mobility-ontology
    :protocol fipa-request
    :conversation-id Req976983289310
)
```

- The ***clone-agent action*** works in the same way, but has an additional String argument to hold the name of the new agent resulting from the cloning process

AMS Request Example

- ***where-is-agent*** action has a single AID argument, holding the identifier of the agent to locate. This action has a result, namely the location for the agent, that is put into the content slot of the inform ACL message that successfully closes the protocol
- **Example**: request message to ask for the location where the agent Peter resides

```
(REQUEST
    :sender (agent-identifier :name da1@Zadig:1099/JADE)
    :receiver (set (agent-identifier :name ams@Zadig:1099/JADE))
    :content (
        (action
        (agent-identifier :name ams@Zadig:1099/JADE)
        (where-is-agent (agent-identifier :name Peter@Zadig:1099/JADE))
        )
    )
    :language FIPA-SL0
    :ontology JADE-Agent-Management :protocol fipa-request
)
```

```
(INFORM
    :sender (agent-identifier :name ams@Zadig:1099/JADE)
    :receiver (set (agent-identifier :name da1@Zadig:1099/JADE))
    :content ((result
        (action
        (agent-identifier :name ams@Zadig:1099/JADE)
        (where-is-agent (agent-identifier :name Peter@Zadig:1099/JADE))
        )
        (set (location
            :name Container-1
            :protocol JADE-IPMT
            :address Zadig:1099/JADE.Container-1
            )
        )
    ))
    :reply-with da1@Zadig:1099/JADE976984777740
    :language FIPA-SL0
    :ontology JADE-Agent-Management
    :protocol fipa-request
)
```

AMS Request Example

- ***query-platform-locations*** action takes no arguments, but its result is a set of all the Location objects available in the current JADE platform.
- **Example**: request message to ask for the location where the agent Peter resides

```
( REQUEST
    :sender (agent-identifier :name Johnny)
    :receiver (set (Agent-Identifier :name AMS))
    :content (( action (agent-identifier :name AMS)
                ( query-platform-locations ) ))
    :language FIPA-SL0
    :ontology JADE-Agent-Management
    :protocol fipa-request
)
```

```
( INFORM
    :sender (Agent-Identifier :name AMS)
    :receiver (set (Agent-Identifier :name Johnny))
    :content (( Result ( action (agent-identifier :name AMS)
                        ( query-platform-locations ) )
                    (set (Location
                        :name Container-1
                        :transport-protocol JADE-IPMT
                        :transport-address IOR:000….Container-1 )
                        (Location
                        :name Container-2
                        :protocol JADE-IPMT
                        :address IOR:000….Container-2 )
                        (Location
                        :name Container-3
                        :protocol JADE-IPMT
                        :address IOR:000….Container-3 )
                    )))
    :language FIPA-SL0
    :ontology JADE-Agent-Management
    :protocol fipa-request
)
```

JADE Mobile Agent Patterns

- The ***Location*** class implements ***jade.core.Location*** interface, so that it can be passed to ***Agent.doMove()*** and ***Agent.doClone()*** methods

- A typical behaviour pattern for a JADE mobile agent is **to ask the AMS for locations** (either the complete list or through one or more where-is-agent actions); then the **agent** will be able to **decide if, where and when to migrate.**

Running Mobile Agents

- A singleton instance of the JADE Runtime can be obtained via the static method ***jade.core.Runtime.instance()***, *which provides two methods:*
  - ***JADE main-container***
  - ***JADE remote container*** (i.e. a container that joins to an existing main-container, forming a distributed agent platfom)
- **INPUT**: ***jade.core.Profile*** object parameter that keeps the configuration options (e.g. hostname, port number)
- **Notice that,** having created the agent, it still needs to be started via the method start()

Remote Container Code Example

```
import jade.core.Runtime;
import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.wrapper.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class JavaApplication2 {

    public static void main(String[] args) {
        // Get a hold on JADE runtime
        Runtime rt = Runtime.instance();
        // Create a default profile
        Profile p = new ProfileImpl();
        // Create a new non-main container, connecting to the default;
        // main contianer (i.e. on this host, port 1099)
        ContainerController cc = rt.createAgentContainer(p);
        // Create a new agent, a DummyAgent
        // and pass it a reference to an Object
        Object reference = new Object();
        Object argsl[] = new Object[1];
        argsl[0] = reference;
        AgentController dummy;
        // Fire up the agent
        try {
            dummy = cc.createNewAgent("inProcess","jade.tools.DummyAgent.DummyAgent", argsl);
            dummy.start();
        } catch (StaleProxyException ex) {
            Logger.getLogger(JavaApplication2.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

Mobile Agents Exercise

1. Install and configure Apache Ant
   Tip: Search "install Apache Ant"
2. Download jade-bin and jade-src and unzip them in same location
3. Move to the jade directory and type command: "ant examples"
4. Go to jade/lib directory and execute in console:
   - java -cp "jade.jar" jade.Boot -gui [open main container]
   - java -cp "jade.jar" jade.Boot -container [create new containers]
   - java -cp "jade.jar;jadeExamples.jar" jade.Boot -gui -container mobile:examples.mobile.MobileAgent [start MobilAgent]
5. Move and clone mobile agent between the different containers using the MobileGui

   Source Code: jade\src\examples\mobile\MobileAgent.java

   .\GetAvailableLocationsBehaviour .\ServeIncomingMessagesBehaviour

Bibliography

- Bellifemine, F., Poggi, A., & Rimassa, G. (1999, April). JADE–A FIPA-compliant agent framework. In *Proceedings of PAAM* (Vol. 99, No. 97-108, p. 33).

- Bellifemine, F., Caire, G., Trucco, T., & Rimassa, G. (2002). Jade programmer's guide. *Jade version*, *3*.

- Bellifemine, F., Poggi, A., & Rimassa, G. (2001, May). JADE: a FIPA2000 compliant agent development environment. In *Proceedings of the fifth international conference on Autonomous agents* (pp. 216-217). ACM.

JADE API for Mobile Agents

Integrated Master's in Informatics Engineering

Intelligent Agents

2017/2018

Synthetic Intelligence Lab

Filipe Gonçalves

Paulo Novais

ISLab