Escola Tècnica Superior d'Enginyeria
Universitat Rovira i Virgili

[DΣIM]

Multi-Agent Systems

Jordi Pascual – jordi.pascual@urv.cat

JADE messaging and special agents

MESIIA – Master's Degree in Computer Security Engineering and Artificial Intelligence
MAI - Master's Degree in Artificial Intelligence

Outline

1. Messaging
2. Message templates
3. Sniffer and Dummy agents
4. More resources

1. Messaging

- Agents communicate by exchanging messages
- The **ACLMessage** class represents a single message
- Each agent has a **message queue shared** by all its behaviours
- Agents exchange **ACLMessages** through:
  - **Agent.send()**: sends a message to the receiver
  - **Agent.receive()**: gets a message if there is any available in the message queue (**non-blocking receive**)
  - **Agent.blockReceive()**: suspends the agent unit it receives a message (**blocking receive**)

2. Message templates

- An agent should be capable of handling **simultaneous conversations**
- As the message queue is shared, the **MessageTemplate** class provides a way to build message patterns and filter messages
- Available attributes to filter:
  - FIPA Performative: INFORM, PROPOSE, CONFIRM, …
  - Sender
  - Conversation ID
  - Language
  - Content
  - Others
- Logical conditions can be built with: **AND**, **OR** and **NOT**

2. Message templates

The receiver is only interested in **Request** performative messages **AND English** language

```
public class FilteringBehaviour extends SimpleBehaviour {
        private boolean finished = false;
        MessageTemplate messageTemplate = null;
        Agent agent;

        public FilteringBehaviour(Agent agent) {
                MessageTemplate performativeFilter = MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
                MessageTemplate languageFilter = MessageTemplate.MatchLanguage("English");
                messageTemplate = MessageTemplate.and(performativeFilter, languageFilter);
                this.agent = agent;
        }

        @Override
        public void action() {
                ACLMessage aclMessage = agent.blockingReceive(messageTemplate);
                if (aclMessage != null) {
                        myLogger.log(Logger.INFO, "Message matching template received: " + aclMessage.getContent());
                        finished = true;
                }
        }

        @Override
        public boolean done() {
                return finished;
        }
}
```

3. Sniffer and Dummy agents: Exercise

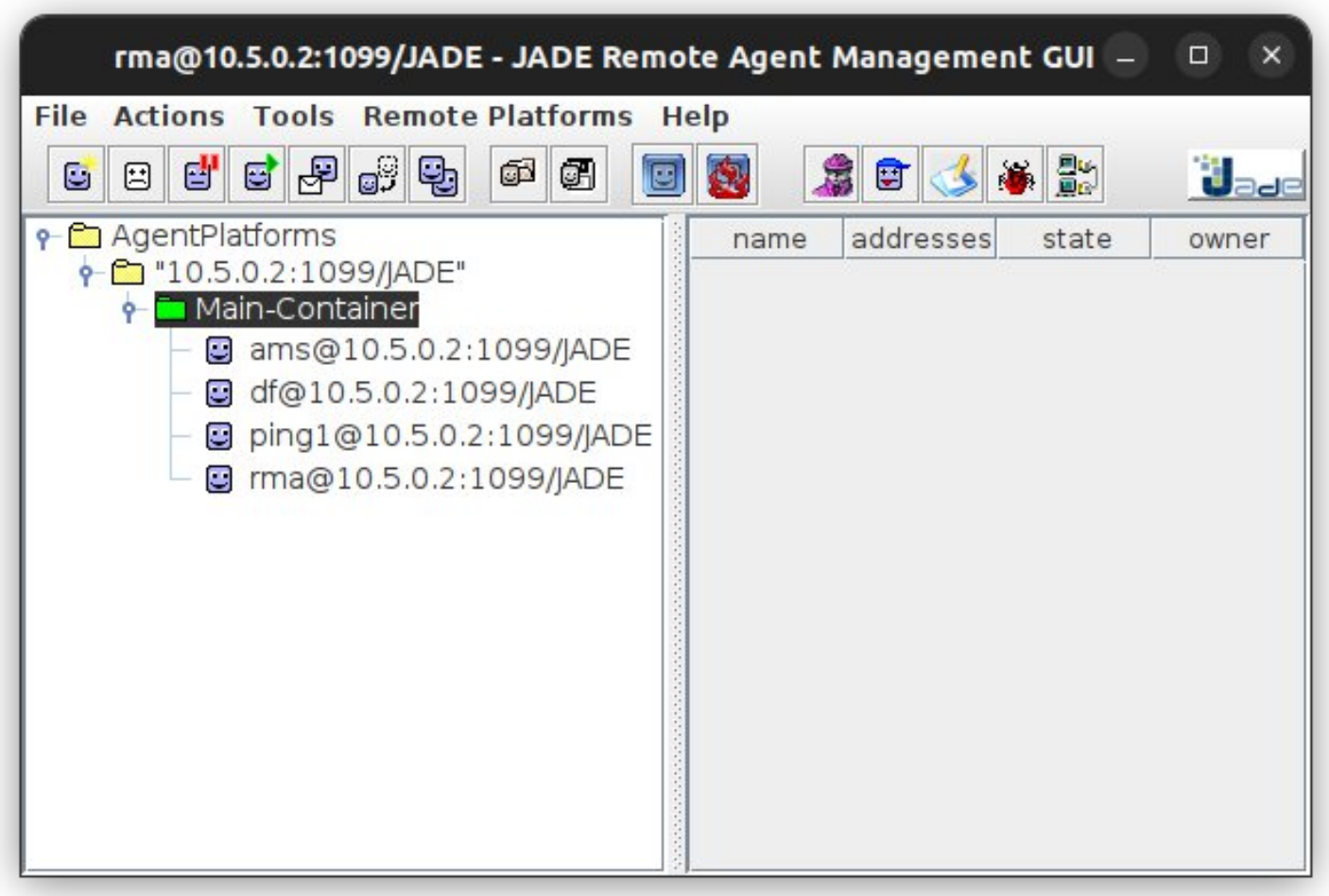
- In this exercise we will see the Sniffer and Dummy agents
- We will use the PingAgent from the JadeExample project
- This agent:
  - Registers to the DF
  - Replies **pong** to **REQUEST** messages which have **ping** as the content
  - Otherwise replies with an error message

3. Sniffer and Dummy agents: Exercise

This is a guided exercise. Follow the next steps:

1. Create a new Maven profile named **ping-agent** with the following arguments: **-gui**, **-agents**, **ping1:urv.imas.PingAgent**
2. Build and execute the profile: **mvn install -P ping-agent exec:java**
3. Check the following agents are on the Main-Container:
    1. **AMS**: agent that runs the Agent Management System
    2. **DF**: agent that runs the Directory Facilitator
    3. **RMA**: agent that runs the Remote Agent Management
    4. **Ping1**: agent of class PingAgent

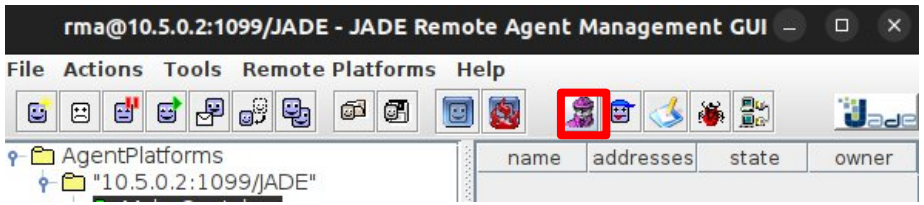
3. Sniffer and Dummy agents: Exercise



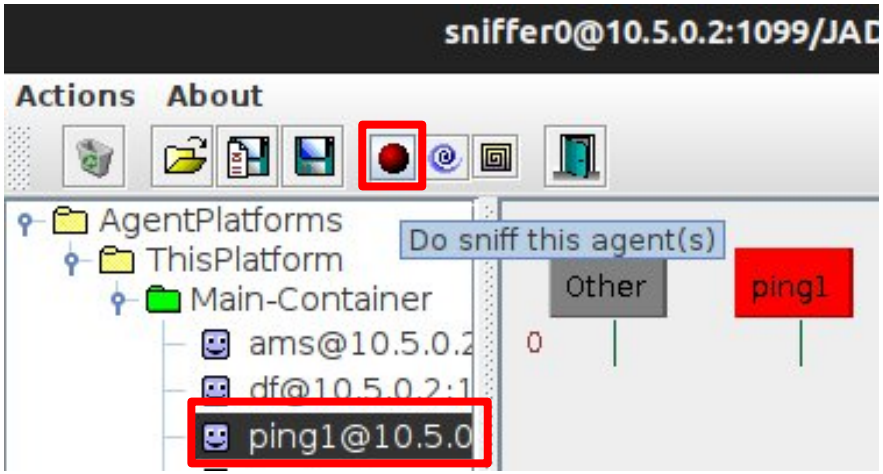
3. Sniffer and Dummy agents: Exercise

Now we will create a Sniffer Agent. This agent allows to see the messages being sent and received by the agents

4. Select the Main-Container and start the sniffer agent. A sniffer GUI will open



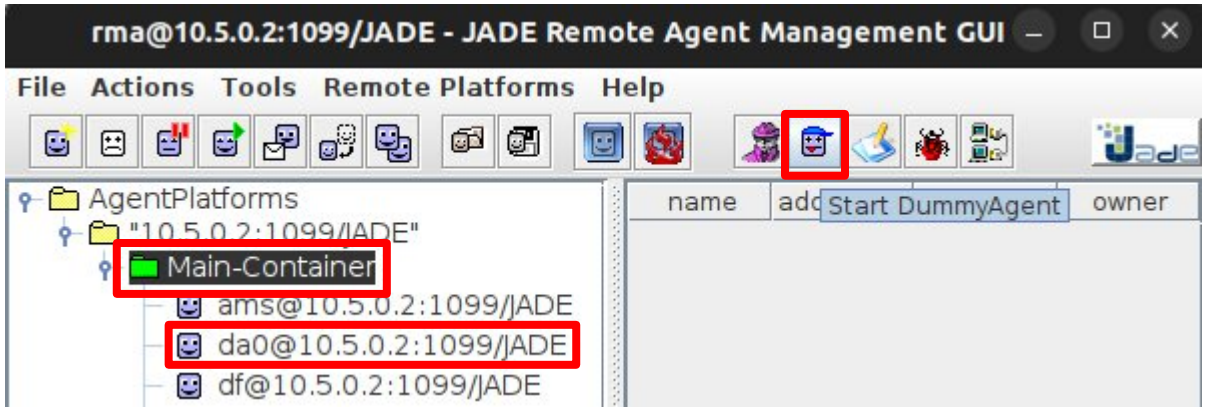
5. Select the **ping1** agent, and press the **Do sniff this agent(s)** button



3. Sniffer and Dummy agents: Exercise

Now we will create the Dummy agent. It is a ready-made agent in the RMA for manually sending messages

6. Select the Main-Container and start the Dummy Agent. A new agent named *da0* should be created and its GUI should open

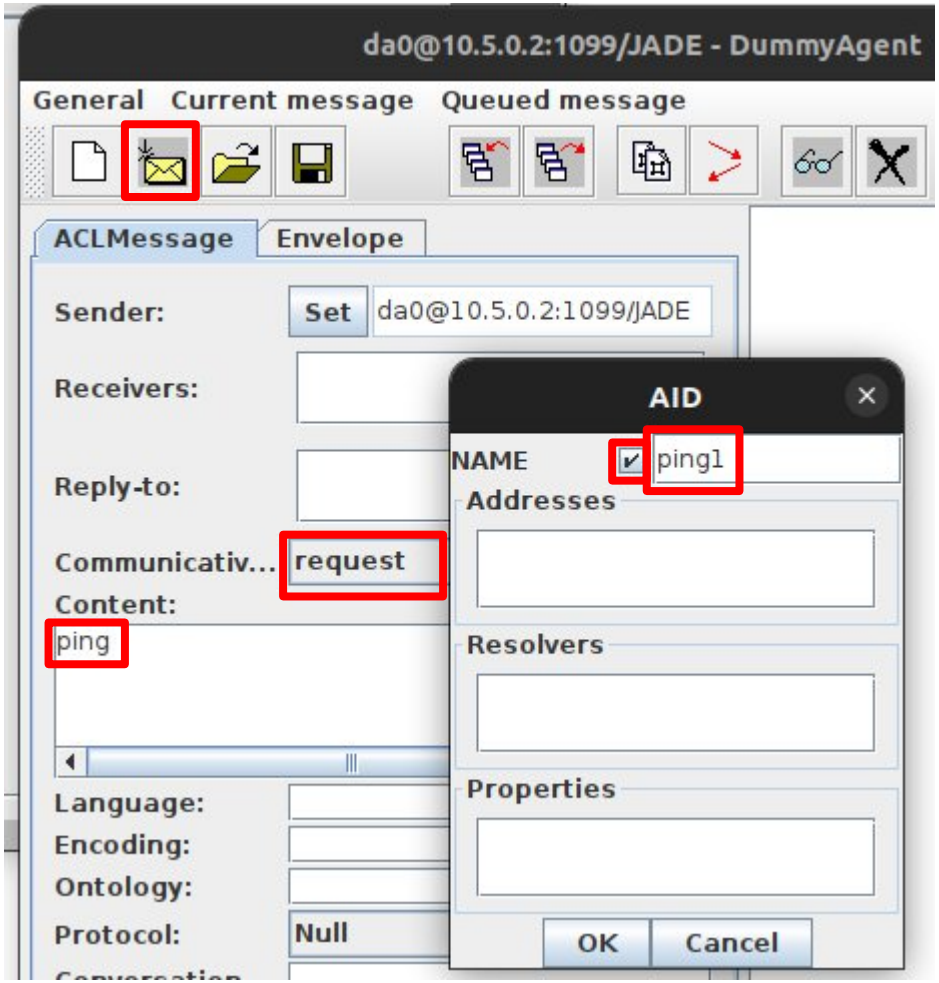


7. In the sniffer GUI, add the new ***da0* Dummy Agent** to be sniffed

3. Sniffer and Dummy agents: Exercise

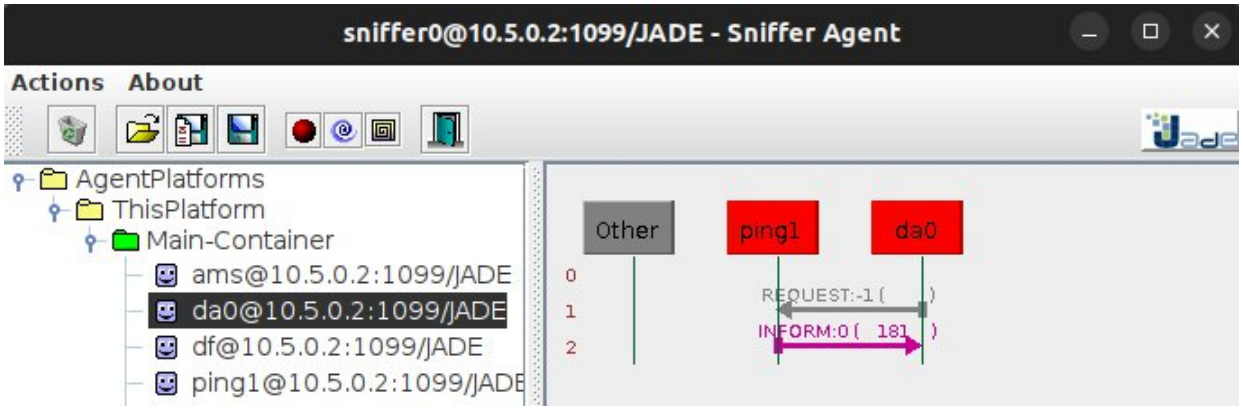
Next, we will use the Dummy Agent to send a ping message to the **ping1** Ping Agent

8. Right click the Receivers box and select **Add**
9. Check the **NAME checkbox** and set the receiver to be the **ping1** agent
10. Set the performative to **Request** and the content to **ping**
11. Press the **Send the current ACL message** button



3. Sniffer and Dummy agents: Exercise

- In both the Dummy Agent and the Sniffer Agent, you should see how the Dummy Agent sends a Request message to the Ping Agent, and gets an Inform reply
- This is expected as the sent message is the one expected by the Ping Agent



- Now, you can try sending improperly formatted messages to the Ping Agent to see how it responds differently

4. More resources

- FIPA Protocols
- JADE Javadoc
- JADE Guides
- JADE Maven Setup for Beginners