UNIVERSITA' DEGLI STUDI DI MILANO BICOCCA

DIPARTIMENTO DI INFORMATICA SISTEMISTICA E COMUNICAZIONE

Ph.D. course: Advanced Distributed Systems Development with Multiagent Systems

Dott. Daniela Briola

Prof.ssa Viviana Mascardi

Prof. Rafael Bordini

UNIVERSITA' DEGLI STUDI DI MILANO BICOCCA

DIPARTIMENTO DI INFORMATICA SISTEMISTICA E COMUNICAZIONE

JADE: Main Concepts

Daniela Briola

Daniela.briola@disco.unimib.it

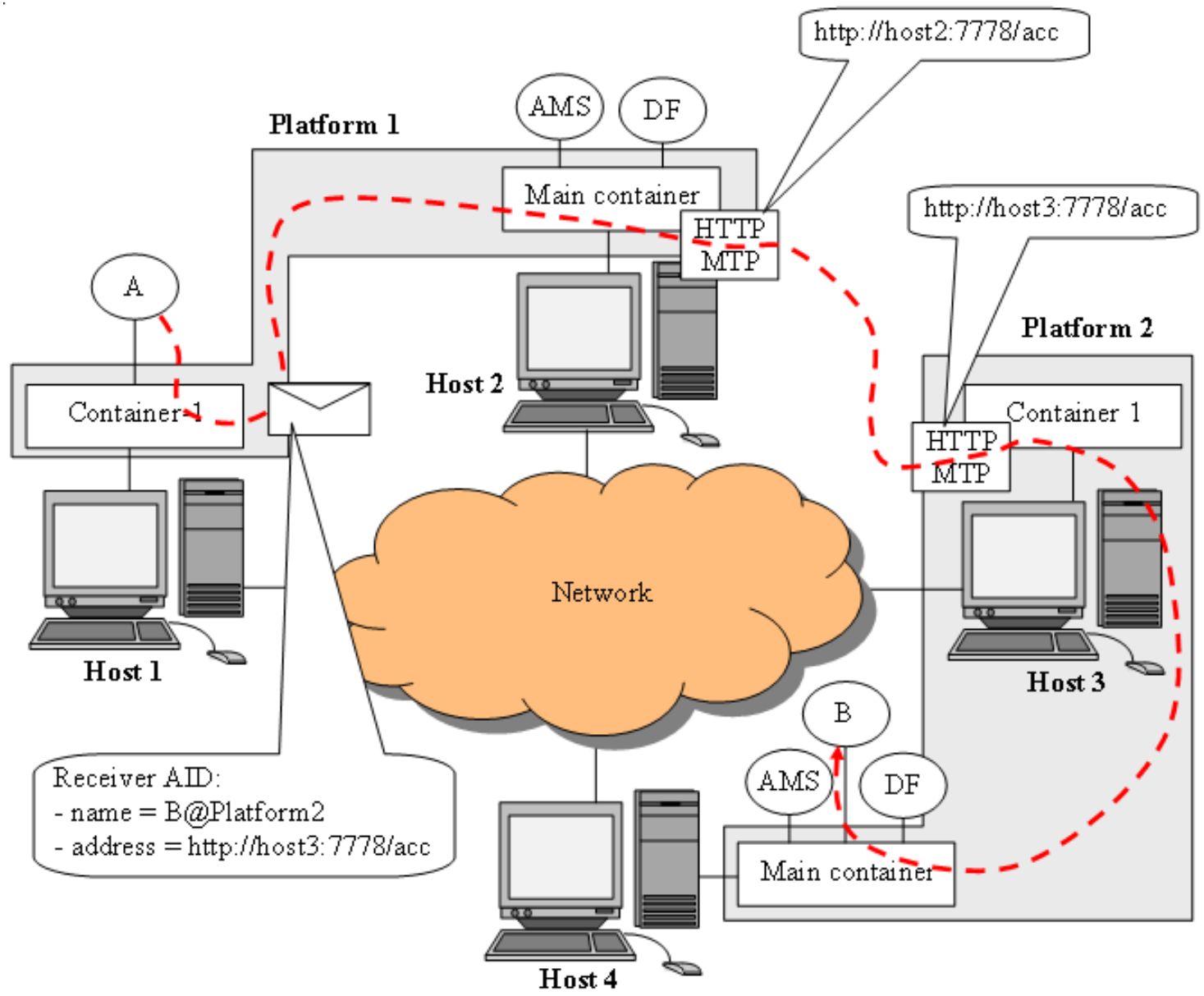
Ph.D. Course: Advanced Distributed Systems Development with Multiagent Systems

JADE

- Java based platform for developing MASs
  - No specific agent type (BDI or other) is imposed
  - A simple but powerful Agent Template is given
  - The focus is more on the MAS than on the single agent
- Based on the FIPA standard (http://www.fipa.org/)
- Widely adopted in academia and industry
- Many addons and extensions available
- It provides an environment where to execute agents, managing:
  - Agents lifecycle
  - messages exchange between agents
- It offers a set of classes to be used to implement agents
- It offers some GUIs to simply manage agents
- It helps in managing services offered by the agents
- https://jade.tilab.com/

Jade Platform

- Each Jade Platform may be divided into Containers
  - They can be used to group agents
  - A main Container (the first created) is always needed
  - Other containers may be launched and installed even on different machines: they need to register with the main one
  - You can “forget” about containers if you are ok with the main one only
- Agents live in a container (but can move between containers)
- Agents can interact (sending messages) if they are on different platforms or containers (the address of the platforms must be known)
  - JADE platform will manage the physical exchange of messages


http://host2:7778/acc
AMS
DF
Platform 1
Main container
HTTP
MTP
http://host3:7778/acc
A
Platform 2
Container 1
Host 2
Container 1
HTTP
MTP
Network
Host 1
Host 3
B
AMS
DF
Receiver AID:
- name = B@Platform2
- address = http://host3:7778/acc
Main container
Host 4

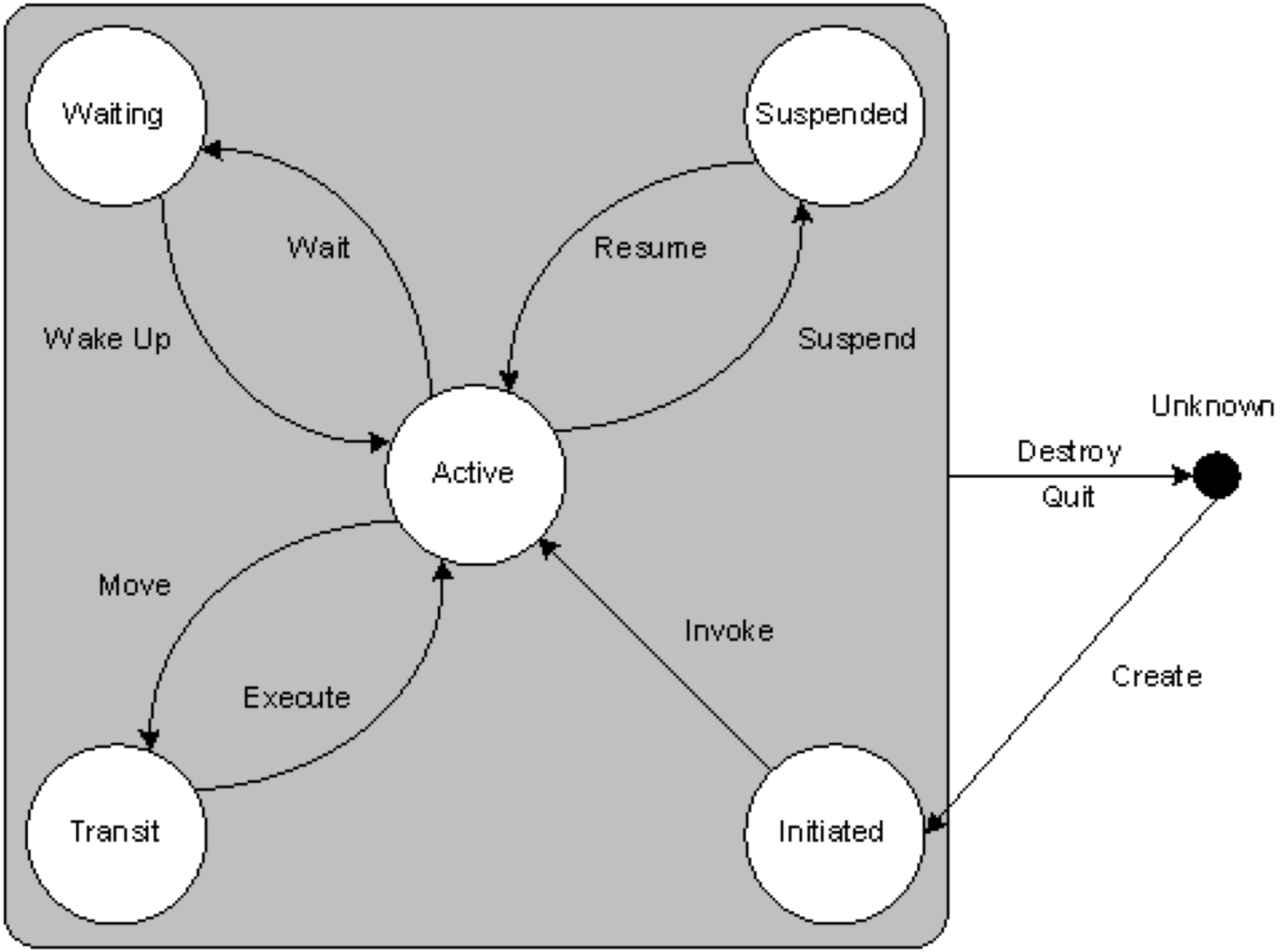
Platform agents

- On each main container there are:
  - AMS (Agents management system) Agent: it keeps the list of all the agents registered in the platform, and manage their “status”
  - DF (Directory Facilitator) Agent: it offers a service of “yellow pages” of the services offered by the agents, so that it is simple to search for them
- The AMS is unique
- DFs from different platforms can register one to the others, so that to create a network of DFs

JADE Agents

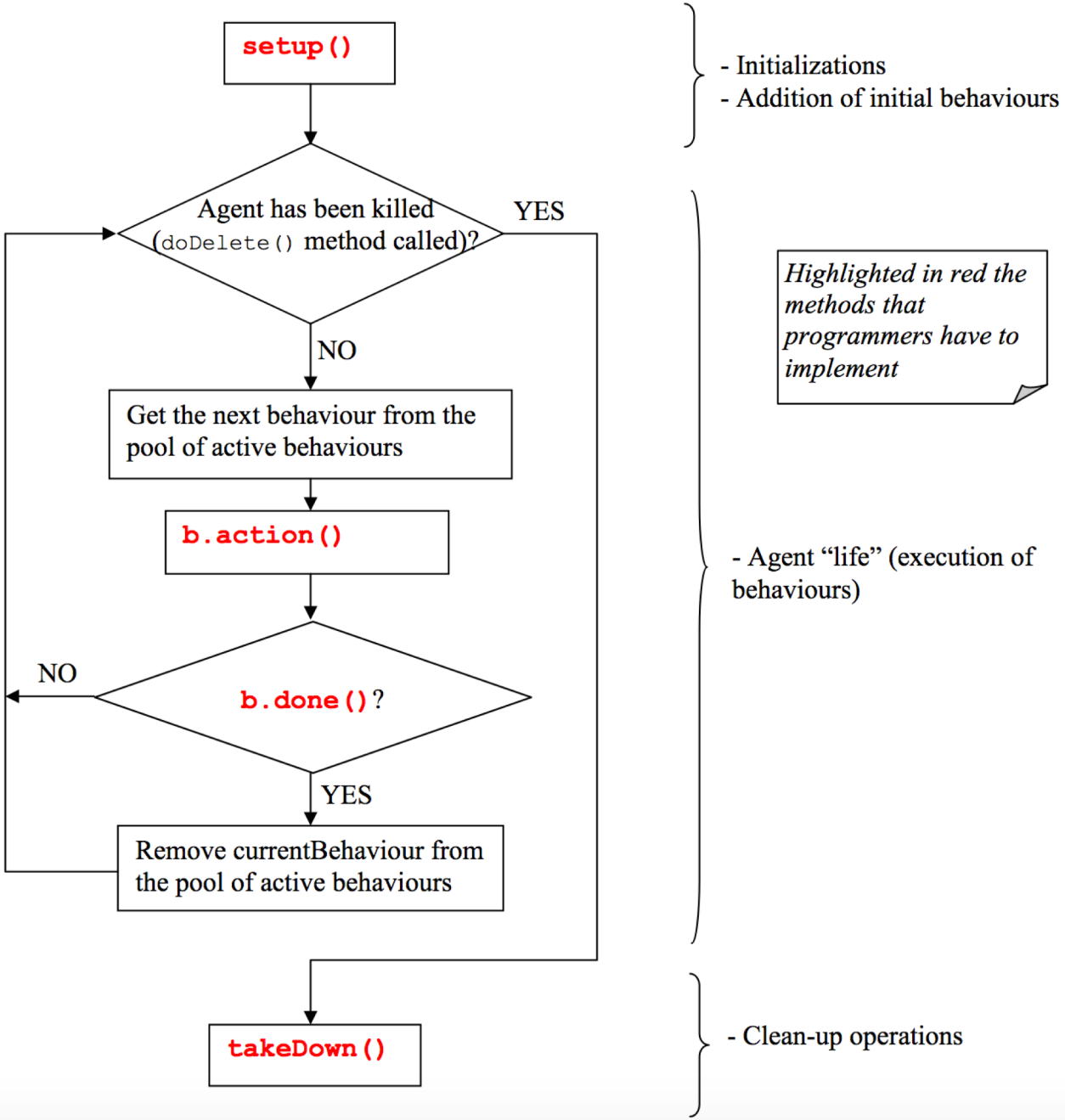
- Whatever extends the *jade.core.Agent* class, that provides:
  - Code to automatically register with the AMS
  - Code to manage the agents behaviours
  - Code to interact with the platform (DF and AMS)
  - Code to manage agent lifecycle
  - Many other basic (but crucial) activities
- Each agent has a unique name (*jade.core.AID*) in the platform:
  - <AgentName>@<PlatformName>
  - Plus addresses (those of the container it is running on)
- Each Agent needs to implement its *setup*() method
- Each agent runs on a dedicated thread
  - JADE shedules them with a round robin approach

Agent LifeCycle



Agent Behaviours

- The tasks an agent can/need to do are called “Behaviours”
- They are object of a class that extends *jade.core.behaviours.Behaviour*
- They can be combined and added/removed in any moment
- An agent has two lists, of ready and one of blocked behaviours
- The *action* method defines the job to be done in the behaviour
- The boolean *done* method is used to say if the behaviour is completed or not
- Behaviours can run concurrently and are scheduled with a RoundRobin approach. They share the same agent thread
  - An *action* method is executed in a non preemptive way: it is not interrupted when it starts… Be careful!
- You can stop a behaviour with the *block* method
- No behaviours available for execution → agent’s thread goes to sleep


setup()
Agent has been killed (doDelete() method called)?
YES
NO
Get the next behaviour from the pool of active behaviours
b.action()
NO
b.done()?
YES
Remove currentBehaviour from the pool of active behaviours
takeDown()
- Initializations
- Addition of initial behaviours
Highlighted in red the methods that programmers have to implement
- Agent "life" (execution of behaviours)
- Clean-up operations

Behaviours Types

- 4 main types:
  - "One-shot": behaviours that complete immediately and whose *action*() method is executed only once. *Done*() returns true by default
  - "Cyclic": behaviours that never complete and whose *action*() method executes the same operations each time it is called. *Done*() returns false by default
  - "Time based": behaviours that execute certain operations at given points in time. They execute every tot milliseconds (and never end) or only once after a timeout
  - Generic behaviours that embeds a status and execute different operations depending on that status. They complete when a given condition is met.
- More complex ones, like Sequential or Parallel, exist

Agent Communication

- Agents exchange messages, directly inserted in the private message queue by the platform
- Message are in the FIPA ACL language. They include
  - Sender
  - Receiver[s]
  - Performative
  - Language
  - [Ontology]
  - [Protocol]
  - Content
  - Other information to keep trace of a conversation
- Waiting for a message can be synchronous (be careful…) or asynchronous (the standard for agents)
- You can search for a specific received message using a *Template*

Let's try...

- Download Jade and put it somewhere you want
- Start Eclipse
- Create a new Java Project
- Include Jade (and the common-codec jar)
- Import the agents from the online zip
- Create a launch configuration for your project:
  - jade.Boot is the main class
  - Parameters:

  -gui -agents
  ag1:SimpleAgent;ag2:TimeAgent("500");ag3:WriteAgent("20000")
  - VM parameter: -Dlog4j.configurationFile=file:PathTo/log4j2.xml
- Open the source codes of the agent and try to understand them
- Launch the MAS (initially without ag3)