Agent and **O**bject **T**echnology **Lab**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Towards a Flexible Development Framework for Multi-Agent Systems

Agostino Poggi

AOT
LAB

Outline

- Project goals
- Overview of HDS
- Experimentation and extension
- Current and future work

Goals (1/2)

- Provide a set of software modules and tools for the lab activities of my past master course “Distributed and Agent Based Systems” and my future master course “Distributed Systems” that allow:
  - Both low and high level practice activities
  - Activities for students with limited knowledge about AI techniques

AOT
LAB

Goals (2/2)

- Realize a software framework and a distributed middleware that allow
  - The deployment of a system on a net of heterogeneous devices
  - The interaction of computational nodes through different kinds of network
  - The realization of both intelligent and traditional applications
  - An easy use for software developers with limited (or without any) knowledge on AI and agent based systems
  - The experimentation of new methodologies, techniques and solutions for (intelligent) distributed system

AOT LAB

Possible Solution: a JADE Extension

- JADE is a FIPA complaint platform
  - To be in compliance with FIPA requires to satisfy a lot of constraints
  - Now I sure that FIPA specifications will be not a standard for software interoperability …
- JADE uses FIPA ACL
  - An ACL like FIPA ACL or KQML is a very expressive language, but
  - How much are the agent based systems where the agents mainly communicate with interactions based on request - inform result pairs?

AOT
LAB

HDS Features

- Is a software framework
- Merges the client-server and the peer-to-peer paradigms
- Provides a dynamic adaptation of a system through composition filters
- Allows the distribution of processes on different kinds of computational nodes
- Allows the communication between processes of different computational nodes through different kinds of communication technologies

HDS Components

- Processes
- Message Filters
- Typed Messages
- Selectors

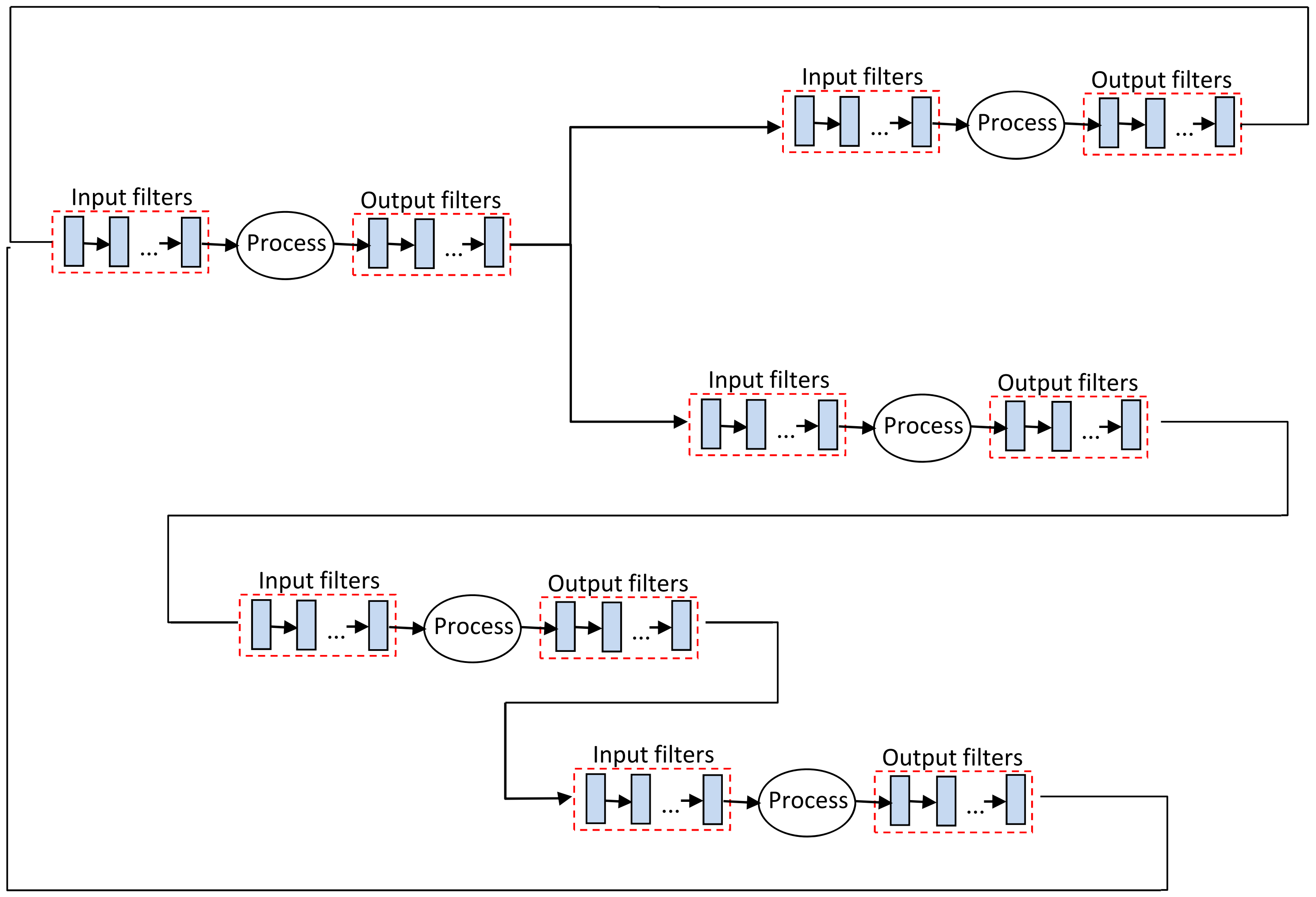
AOT
LAB

Processes

- Are divided in actors and servers
  - Actors are active objects that can start synchronous and asynchronous interactions
  - Servers are passive objects that can only respond to actors requests
- Are identifiable by an address having a mail address format
- Interact with each other through the exchange of synchronous, asynchronous or one way messages

AOT LAB

Interaction between Processes



Message Filters

- Block the reception or the sending of messages from/to some other processes
- Manipulate messages
- Send a copies or redirect messages to another process
- Can be dynamically added or removed

ACL and OOP Messages

- ACL messages
  - Separation between the speech act and the content of the messages
  - Semantic meaning of message derived from their combination
- OOP messages
  - A message usually represents a method call and its semantic meaning is defined by the class to which it belongs

AOT
LAB

Typed Messages

- Are similar to an actor message
- The content usually defines either the action that the receiver should perform or a step of a complex interaction
- The other message attributes define sender, receiver, … and the type of communication: synchronous, asynchronous or one way
- Content is defined through a very simple Java based ontology model

Selectors

- Allow the filtering of objects
- Are used for the selection of a specific message from the input queue
- Are used for requiring to the runtime lookup service the addresses of the processes that satisfy a set of constraints

HDS Design and Implementation

- Has been implemented through the Java programming language
- Its design is based on an abstract model that allows
  - The realization of a HDS software module with different solutions and software components
  - The use of different implementations of a HDS software module in the same application
- Is based on three software modules: concurrency, runtime and communication

AOT LAB

Concurrency Module

- Provides two abstract classes for the definition of actors and servers and a set of predefined processes
- Supports the concurrent execution of processes
- Supports the communication between processes in the same JVM

Runtime Module

- Manages the processes inside a single JVM
- Provides the lookup service
- Supports the execution of functions that:
    - Install sets of composition filters to drives the communication between some processes
    - Activate some connections towards other runtime nodes

AOT LAB

Communication Module

- Defines the default communication language based on three types of content: action, result, error
- Provides three derived communication languages, publish-subscribe, group communication and FIPA ACL
- Provides the typed contents for requiring the basic services (processes and functions management) through the default communication language

AOT LAB

Code Examples (1/2)

```
m = call(this.actor, new Ping());

if ((m != null) && (m.getContent() instanceof Pong)) {
  System.out.println("the actor " + this.actor + " is alive!");
}

Selector<Message> s = future(this.actor, new Ping());

System.out.println("the actor " + address + " performs some tasks ... .");

m = take(s);

if ((m != null) && (m.getContent() instanceof Pong)) {
  System.out.println("the actor " + this.actor + " is still alive!");
}

send(this.actor, new Ping());
```

AOT
LAB

Code Examples (2/2)

```
call(Namer.RUNTIME, new Register(TEMPERATURE));
while (System.currentTimeMillis() < this.end) {
  Temperature t = new Temperature(temperature();
  send(Namer.RUNTIME, new Publish(TEMPERATURE, t));
  Thread.sleep(WAITINGTIME);
}
```

```
call(Namer.RUNTIME, new Subscribe(address, TEMPERATURE));
while (System.currentTimeMillis() < this.end) {
  Message m = poll(this.end - System.currentTimeMillis());
  if (m == null) break;
  if (m.getContent() instanceof Publish) {
    Publish p = (Publish) m.getContent();
    if (p.getContent() instanceof Temperature) {
      Temperature t = (Temperature) p.getContent();
      System.out.println(“temperature is " + t.getValue()); } } }
```

AOT LAB

HDS Experimentation

- Environment for the provision of collaborative services for social networks where users
  - Are connected through heterogeneous networks and act on heterogeneous devices
  - Interact through either a Web portal or specialized clients and perform action on the basis of their rights
- Market place multi-agent application
  - Use of both client-server and FIPA ACL languages
  - Use of BDI agents and FIPA interaction protocols
- Distributed system for information retrieval
  - Dynamic construction of a domain ontology
  - Cooperative result composition

AOT LAB

HDS Extension

- Inter runtime node communication:
  - MINA, Qpid, JMS (OpenJMS and JBoss Messaging)
  - Messages routing and runtime nodes discovery
- Task composition:
  - JADE and BPMN
- Negotiation:
  - FIPA interaction protocols and voting algorithms
- Concurrency management:
  - Jetlang and Kilim
- Security
  - Encrypted and signed interactions

AOT LAB

Current and Future Work

- Selection of an appropriate ontology model and language for the definition of HDS message contents
  - High expressiveness
  - Easy to be used by software developers with limited knowledge on AI
- An automatic solution for message filters ordering

Thank you for your kind attention!

Questions?

Agostino Poggi