Práctica 2: Comunicación entre agentes Curso 2015/16

Arquitectura. Herramientas. Agentes sencillos.

Introducción

- La comunicación se basa en un modelo de *paso de mensajes asíncrono*. Cada agente posee un buzón (buffer) en el que se van almacenando los mensajes enviados por otros agentes.
- Los mensajes siguen el formato definido por FIPA-ACL:
  - **performativa**: El objetivo del mensaje (inform, request, …etc)
  - **sender**: el remitente del mensaje.
  - **receivers**: la lista de destinatarios.
  - **content**: el contenido del mensaje.
  - **language**: el lenguaje en el que está expresado el content.
  - **ontology**: la organización y relación de los conceptos utilizadas en el content.
  - **campos de control**: son usados para controlar las conversaciones (duración, identificador de conversación, etc).
- Los mensajes son implementados como objetos de la clase *jade.lang.acl.ACLMensaje*

Introducción

- Las performativas que se pueden utilizar es un número finito y definido. Entre éstas destacan las siguientes:
  - **inform**: Semántica de su empleo: El agente emisor cree 1) que la proposición es cierta; 2) tiene la intención que el agente receptor la crea también; y 3) supone que el agente receptor desconoce que la proposición sea cierta.
  - **confirm**: La semántica es similar al caso anterior, y además el agente emisor cree que el agente receptor no tiene claro la certeza o no de la proposición.
  - **disconfirm**: Semántica de su empleo: 1) el agente emisor cree que una proposición es falsa; 2) tiene la intención de que el agente receptor también la crea; y 3) el agente emisor cree que el agente receptor o bien cree en la proposición o no tiene clara su certeza.

Programando comunicación de agentes

- Envío de mensajes:

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Pedro", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("Weather-forecast-ontology");
msg.setContent("Today it is raining");
send(msg);
```

- Recepción de mensajes:

```
ACLMessage msg = myAgent.receive();
if (msg != null) {
   String pro = msg.getContent();
   System.out.println("El mensaje recibido fue: " + pro);
} else block();
```

Ejercicio 1

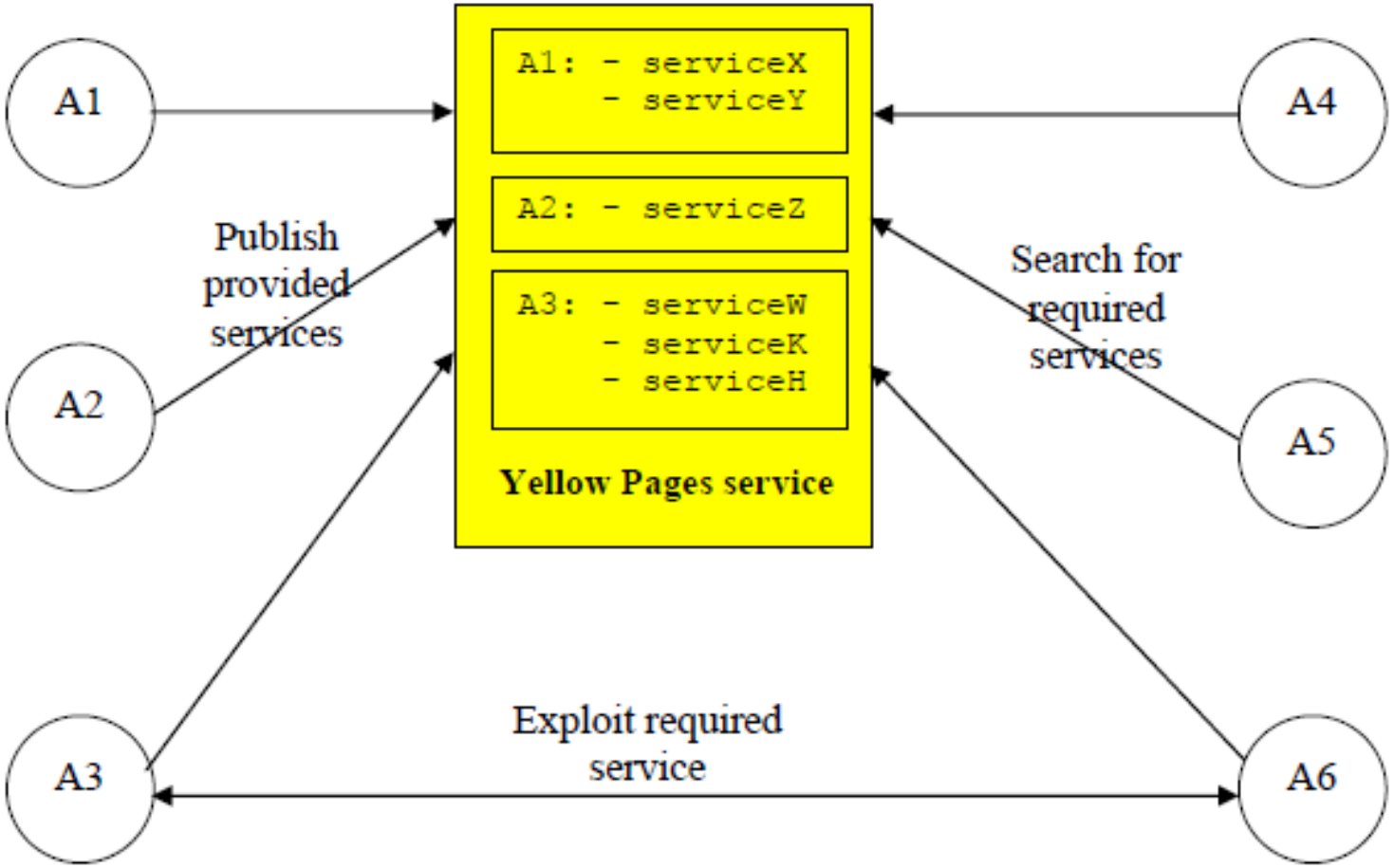
- Crea dos tipos de agentes S y R.
  - Un agente de tipo **S** debe enviar un mensaje a un agente de tipo **R** indicándole su estado. El estado de un agente es una variable de tipo String que puede tener el contenido "Ocupado" o "Libre".
  - Inicialmente los agentes de tipo **S** están libres.
  - Cada vez que un agente de tipo **S** envía un mensaje, cambia de estado (de Libre a Ocupado y de Ocupado a Libre)
  - Cuando un agente de tipo **R** recibe el mensaje imprime por pantalla el estado del agente del que ha recibido el mensaje**.**
  - Crea un comportamiento TicketBehaviour de 5 segundos para los agentes de tipo **S** de forma que en cada tick se envíe un mensaje a un agente de tipo **R** (que tiene un comportamiento cíclico).
  - Comprueba el funcionamiento ejecutando dos agentes de tipo **S** (s1 y s2) y un agente de tipo **R** (r1). Observa los mensajes que se imprimen por pantalla. Lanza el agente Sniffer y lanza primero el agente r1, posteriormente lanza los agentes s1, s2 y s3 mediante la interfaz de Jade. Observa la secuencia de mensajes que circulan de estos agentes al agente r1.

Ejercicio 1: Sugerencias

- Los comportamientos pueden escribirse como clases Java independientes o como clases privadas de la clase del agente. Este último caso permite acceder a los miembros de la clase del agente directamente.
- Existe una variable *myAgent* que forma parte implícita de todo comportamiento a través de la cual se puede acceder a todos los miembros y métodos del objeto agente que ejecuta el comportamiento.

Localizando agentes por servicios

- El agente Facilitador de Directorio (DF)



Localizando agentes

- Interacción con el agente DF.
  - Es un agente más, pero que viene definido por FIPA. El resto de agentes se comunican con el enviándole mensajes y recibiendo sus mensajes de respuesta.
  - Proporciona métodos para registrar y cancelar registros de agentes por los servicios que éstos ofrecen:
    - register()
    - modify()
    - deregister()
    - search()
  - Al registrarse se ha de proporcionar una descripción *jade.domain.FIPAAgentManagement.DFAgentDescription*

Registro de agentes

- La descripción se compone de: 1) El AID del agente; y 2) El conjunto de servicios que proporciona incluyendo para cada servicio su tipo de servicio, el nombre del servicio y las ontologías, lenguajes, etc. que ayuden a describirlo (*ServiceDescription*)

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID()); // El id del agente
ServiceDescription sd = new ServiceDescrption();
sd.SetType("Detectar");
sd.setName("d1");
dfd.addServices(sd);
try {
     DFService.register(this, dfd);
} catch (FIPAException fe) { fe.printStackTrace(); }
```

Localiza agentes

- Búsqueda en el DF: Se ha de proporcionar una descripción del servicio y el DF devuelve un array con los ids de los agentes que proporcionan el servicio solicitado.

```
DFAgentDescription dfd = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType("Detectar");
try {
  DFAgentDescription[] result =
       DFService.search(myAgent, dfd.addServices(sd));
  System.out.println("Hay " + result.length +
       " agentes de tipo Detectar");
  Agentes = new AID[result.length];
  for (int i=0; i<result.length; i++ )
       Agentes[i] = result[i].getName();
} catch (FIPAException fe) { fe.printStackTrace(); }
```

Ejercicio 2

- Crea un nuevo tipo **S** de agente que se registre en el DF.
- Un agente de tipo **S** tiene un comportamiento que consiste en dos pasos:
  - El primer paso consiste en buscar agentes que provean del servicio **R**. Imprime en pantalla cuantos ha encontrado.
  - El segundo paso envía un mensaje a todos los agentes de tipo **R**. Cuando se acaba el comportamiento, se añade otra vez este comportamiento para que se siga ejecutando implícitamente de forma cíclica.
- Crea un nuevo tipo de agente **R** que se registre en el DF.
- Un agente de tipo "R" tiene un comportamiento que consiste en dos pasos:
  - El primer paso consiste en buscar agentes que provean del servicio "S"
  - Espera recibir al menos un mensaje de cada uno de estos agentes e imprime su contenido. Después finaliza su comportamiento y se añade otra vez este comportamiento para que se siga ejecutando implícitamente de forma cíclica.

Sugerencias

- Añade un tiempo de espera (p.e. 10 seg.) entre ejecuciones de cada comportamiento y al inicio de ejecución del agente para poder observar con detenimiento los mensajes que se intercambian los agentes.
- Arranca el agente ***Sniffer*** y espía a los agentes S y R para ver la secuencia de intercambio de mensajes.
- Arranca el agente ***Introspector*** para observar un agente S y otro R y observar los comportamientos que tiene activos y el estado de sus buzones de entrada y de salida.