Agentes *Ping* y *Pong* con **JADE**

Juan A. Suárez Romero

22 de febrero de 2005

Resumen

El objetivo de este proyecto es medir el rendimiento de los agentes empleando la plataforma **JADE**, que permite comunicar agentes mediante FIPA ACL. Se presenta la filosofía de funcionamiento de dicha plataforma, y cómo se ha medido su rendimiento en base a unos agentes de tipo *ping* y *pong*.

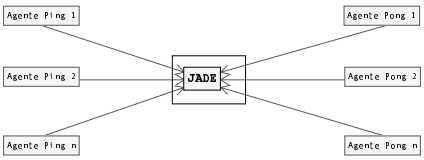

Figura 1: Relación entre los agentes y la plataforma

0.1. Objetivos del proyecto

El objetivo del proyecto es medir el rendimiento de la plataforma **JADE** en función del número de agentes que contiene. La idea consiste en crear un agente que haga un ping y otro agente que conteste con un pong a dicho agente, y realizar varios ping y pong para medir el tiempo medio de ida y vuelta (RTT, *Round Trip Time*) de los mensajes. Se irá incrementando el número de parejas para ver cómo afecta al RTT de los mensajes. Así, para n parejas el RTT será la media de los RTTs de los mensajes de cada pareja.

En orden a que las medidas sean correctas, cada pareja sólo comenzará a tomar las medidas cuando detecte que todos las parejas han sido creadas. No obstante, aunque no comiencen las medidas, si comenzarán a transmitir y recibir los mensajes tan pronto como se vayan creando, y continuarán así hasta el final de las pruebas.

0.2. Descripción de la plataforma JADE

JADE[3] es una herramienta basada en Java para la comunicación de agentes empleando los estándares de FIPA[1].

Una plataforma **JADE** está formada por uno o más contenedores. Los contenedores no son más que máquinas virtuales Java en las cuales se ejecutan los agentes. Así, como mínimo existe un contenedor dentro de una plataforma **JADE**, que es la que provee de los diversos servicios necesarios para cumplir los estándares de FIPA (por ejemplo, el servicio de Páginas Amarillas), y en la cual se pueden también ejecutar agentes. Además podemos añadir más contenedores simplemente ejecutando más máquinas virtuales Java, bien dentro del mismo ordenador o en otro sistema, y enlazándolas con el contenedor principal. Los agentes pueden viajar de un contenedor a otro (por ejemplo, en función de las necesidades de recursos).

0.2.1. Comunicación de los agentes

Cuando se ejecuta un agente en **JADE**, se hace dentro de un contenedor, que provee de las facilidades necesarias para la comunicación a dicho agente. Cada agente se registra en la plataforma con un nombre (se le asigna uno por defecto si no se indica ninguno), el cual se emplea para poder referirse a ellos. Además, de el nombre con el que se registra, se le añade un sufijo para indicar la plataforma en la que se encuentra; de esta forma podemos comunicarnos con agentes en otra plataforma.

Se puede ver la relación entre los agentes y la plataforma en la figura 1.

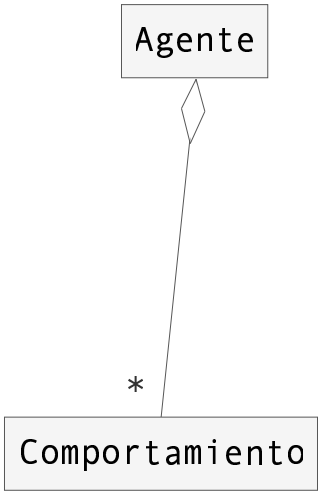

Figura 2: Estructura de un agente en JADE

0.2.2. Estructura de los agentes

En **JADE** los agentes descienden de la clase Agent. Para la ejecución de sus tareas, los agentes implementan *Comportamientos* (figura 2), que son ejecutados por la plataforma **JADE**. Así, el control principal lo posee más bien la plataforma, que ejecuta los agentes según sus necesidades.

0.3. Diseño del programa

0.3.1. Agentes *Ping*

Como hemos comentado al principio, los agentes *ping* son los encargados de medir el rendimiento del sistema enviando y recibiendo los mensajes ping y pong. Sin embargo, no comienzan sus medidas hasta que todos ellos estén listos. Para ello los agentes comparten una clase, *Contador*, que permite su sincronización. Así, cuando un agente está listo para realizar sus medidas, incrementa este contador y comienzan a enviar y recibir los mensajes pong. Cada cierto número de envíos, comprueban si el Contador ha llegado al tope, esto es, al número total de agentes que tiene que haber. Si este es el caso, entonces es cuando comienza la medida. Una vez que ha finalizado, y para evitar que la máquina quede descargarda y desvirtuar así la medida, cada agente decrementa el contador y sigue enviando y recibiendo mensajes sin medirlos, y comprobando de vez en cuando si el contador ha llegado a cero. Este caso ocurrirá cuando todos los agentes hayan decrementado respectivamente una vez el contador, esto es, cuando hayan finalizado sus medidas. En este momento es cuando el programa finalizará. Este proceso se encuentra esquematizado en las figuras 3 y 4. Nótese que en esta última las fases de `Iniciar medida` y `Finalizar medida` sólo tienen

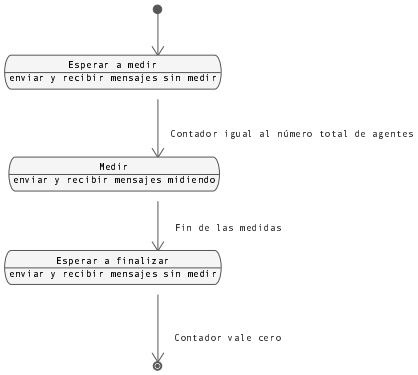

Figura 3: Fases de un agente *ping*

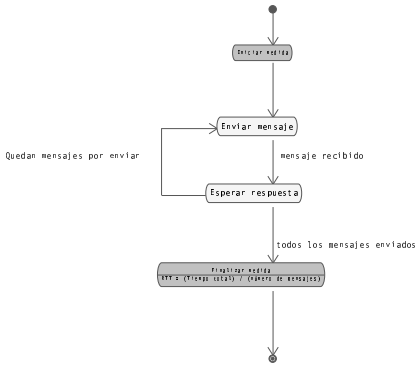

Figura 4: Fases en el envío de un mensaje

sentido cuando se está midiendo los tiempos.

Como vemos, existen claramente tres fases en los agentes: esperar a que todos los agentes estén listos, realizar las medidas, y esperar a que todos los agentes hayan finalizado sus medidas. Puesto que las tareas de los agentes se definen como Comportamientos, hemos empleado el patrón de diseño Estado[2] para crear tres comportamientos: PingStartingBehaviour, PingMeasuringBehaviour y PingEndingBehaviour. Aunque en **JADE** se puede indicar que un comportamiento está formado por tres subcomportamientos que se ejecutan secuencialmente, la forma que tiene de ejecutarlos nos ha llevado a implementar nosotros mismos esta secuencialidad. Así, cuando el estado PingStartingBehaviour detecta que todos los agentes están listos, se sustituye a si mismo por el siguiente comportamiento, PingMeasuringBehaviour, el cual continua su ejecución. Este finalmente hace lo mismo con el último comportamiento, PingEndingBehaviour.

0.3.2. Agentes *Pong*

Estos agentes son más sencillos de de diseñar. Su único objetivo es permanecer a la escucha de mensajes, y cuando se reciba uno se conteste al agente

Figura 5: Actividad de un agente *pong*

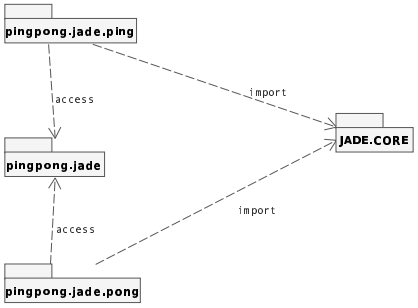

Figura 6: Paquetes que forman el proyecto

emisor. Se puede ver la actividad de estos agentes en la figura 5.

0.3.3. Funcionamiento global

El programa está formado por tres paquetes: *pingpong.jade.ping*, que contiene las clases que forman al agente *ping*, *pingpong.jade.pong*, que contiene las clases que forman al agente *pong*, y *pingpong.jade*, que contiene otra serie de clases útiles (ver figura 6).

Cada agente *ping* recibe un identificador numérico, que va desde 1 hasta el número total de agentes *ping* que se creen. De igual forma se hace con los agentes *pong*, de forma que el agente *ping i* sólo se comunicará con el agente *pong i* correspondiente. Así, primero se crean los agentes *pong* y posteriormente los agentes *ping*.

A la hora de hacer las medidas, hemos posibilitado que o bien todos los agentes residan dentro de la misma máquina virtual, como que estén en máquinas separadas: la plataforma en una máquina virtual, todos los agentes *ping* en otra y todos los agentes *pong* en otra. Por ello se han creado tres clases nuevas: *Cliente*, que lanza tantos agentes *ping* como se indique, *Servidor*, que lanza tantos agentes *pong* como se indique, y *ClienteyServidor*, que lanza tantas parejas *ping* y *pong* como se indique dentro de la misma máquina virtual. Se puede ver en la figura 7.

Independientemente de si se ejecutan las pruebas en una o en más máquinas virtuales, el proceso siempre es el mismo: lanzar primero los agentes *pong*, luego los agentes *ping* y realizar las medidas, tal y como se ve en la figura 8.

En la figura 9 podemos ver una secuencia de funcionamiento de un agente *ping* en conjunción con un agente *pong*.

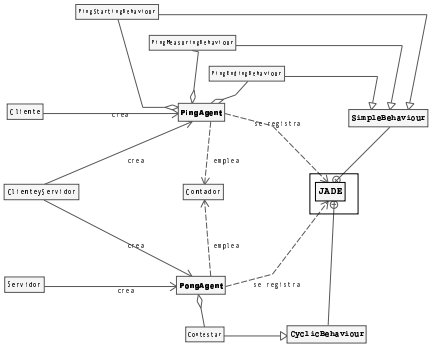

Figura 7: Diagrama de clases del proyecto

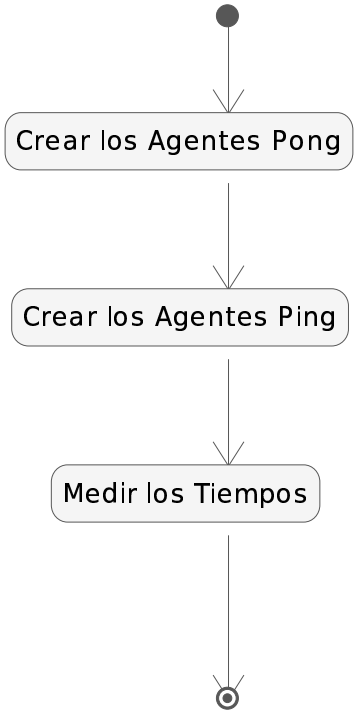

Figura 8: Esquema de funcionamiento

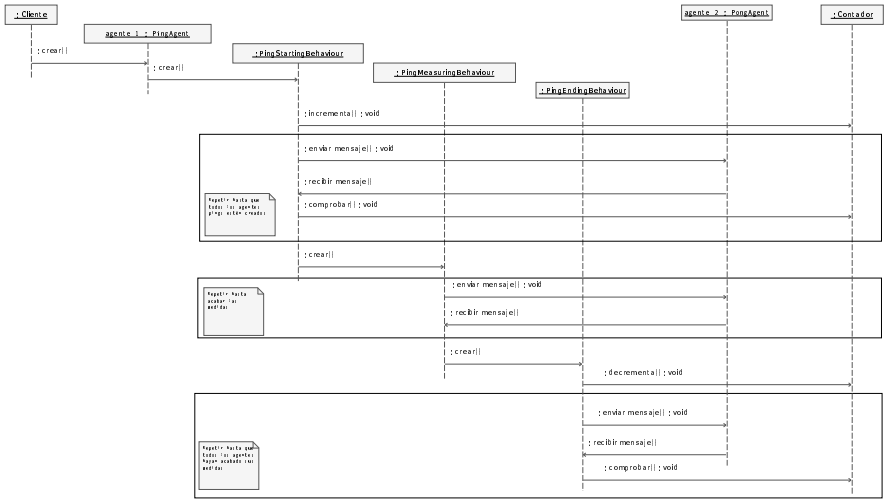

Figura 9: Funcionamiento de un agente *ping*

0.4. Carga de la máquina

En la sección anterior hemos descrito los agentes que hemos empleado para medir el rendimiento de la plataforma **JADE**. En la descripción hemos indicado que los agentes *ping* envían un mensaje y quedan esperando la respuesta; de igual forma los agentes *pong* esperan un mensaje y contestan con una respuesta. En ambos casos el proceso se repite continuamente.

En un sistema real, los agentes no quedan simplemente esperando por la respuesta a un mensaje, ni la respuesta es un simple mensaje, sino que mientras esperan realizan alguna tarea, y la respuesta suele ser fruto de alguna operación que se realiza.

Por ello se ha creado paralelamente a las clases *PingAgent* y *PongAgent* (que se corresponde con los agentes mencionados antes) las clases *PingAgentLoad* y *PongAgentLoad*. La diferencia entre los agentes *PingAgent* y *PingAgentLoad* es que éste último ejecuta una tarea entre el envío del mensaje y la espera por una respuesta (ver figura 10), mientras que la diferencia entre los agentes *PongAgent* y *PongAgentLoad* es que éste último calcula la respuesta en función de una tarea (ver figura 11). En ambos casos se ha aportado una clase denominada *FullLoad* con un método que carga la máquina durante el tiempo estipulado (en nuestro caso durante 1 segundo).

De nuevo, puesto que empleamos comportamientos, hemos creado los comportamientos *PingMeasuringBehaviourLoad*, *PingStartingBehaviourLoad* y *PingEndingBehaviourLoad*.

De igual forma, para poder ejecutar estos nuevos agentes se han creado, a semejanza de antes, las clases *ClienteLoad*, *ServidorLoad* y *ClienteyServidorLoad*.

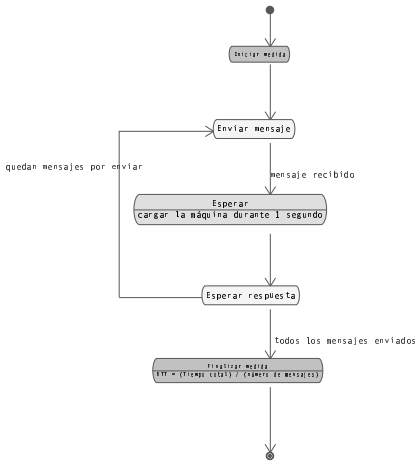

Figura 10: Tarea del agente *PingAgentLoad*

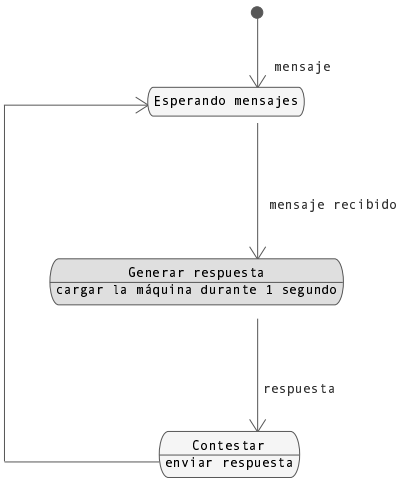

Figura 11: Tarea del agente *PongAgentLoad*

Bibliografía

[1] FIPA. Foundation for Intelligent Physical Agents. http://www.fipa.org.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[3] TILAB. JADE - Java Agent DEvelopment Framework. http://jade.tilab.com.