**A new paradigm to agent's communication using AMQP's protocol**

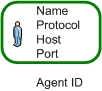
**{spajares, rilpefo, jbellver, aespinosa,....}@dsic.uvp.es**

**Abstract**

Though many networking protocol needs have been addressed, a large gap exists in common guaranteed-delivery messaging middleware. AMQP fills that gap. Advanced Message Queuing Protocol, enables complete interoperability for messaging middleware, both the networking protocol and the semantics of broker services are defined in AMQP. The AMQP model explicitly defines the server's semantics because interoperability demands the same semantics for any server implementation. To enable technology-neutral interoperability, AMQP defines an efficient wire-level format with modern features. So, AMQP, es un estándar de mensajería, y que hace uso de un protocolo binario extremadamente eficiente, a diferencia de otras aproximaciones anteriores que trabajaban con XML. Se basa en un modelo caracterizado por "routing exchanges" y "queues", las cuales permiten interacción con cualquier escenario que haga uso de las apis de mensajería. La idea fundamental, es en primer lugar, que cualquier conjunto de bytes colocados en un lado de la tubería, se pueden leer de igual manera en el otro lado, actuando esta tubería como parte de un middleware. En segundo lugar se soluciona el problema de las "Buddy Lists". En lugar de tener un directorio centralizado de contactos, se hacen uso de "queues" y "exchanges". Además como ya hemos dicho, dicho middleware de comunicación, actúa de intermediador en la comunicación entre agentes implementados en diferentes lenguajes de programación. Tres son las implementaciones destacadas para este estándar: OpenAMQ, Apache Qpid y RabbitMQ, todas ellas open source. Nos basaremos en Apache Qpid. En primer lugar detallaremos como ha sido diseñado un nuevo modelo de agentes. En el segundo apartado, contaremos aspectos relacionados con la comunicación entre nuestros agentes. Por último demostraremos por medio de diferentes benchmarks, el rendimiento obtenido para esta plataforma.

**1.- Introduction**

Se ha diseñado un nuevo modelo de agentes extensible, haciendo uso de la tecnología JAVA, en la que un agente está formado por una cadena que forma su nombre, un protocolo, y el nombre y puerto del host en el que se encuentra escuchando dicho agente inteligente.



Para poner en marcha un agente, se deberá hacer uso del iniciador "start". Y el método "execute" nos permite definir el comportamiento del agente. Además no debemos olvidar que cuando creemos un nuevo agente en la plataforma, debemos indicarle el broker QPID con el que efectuará comunicaciones internas. Evidentemente dicho broker, deberá estar lanzado previamente en una maquina y en un puerto determinado, y autentificado por medio de un user y un password. Cabe resaltar que el broker, no tiene por qué estar en la misma máquina en la que los agentes residen, de hecho las pruebas que contaremos en la sección (?), se ha trabajado con agentes dispersos en varias maquinas, y el broker en otra diferente. Por tanto podemos entender un agente como una entidad capaz de percibir el entorno, procesar tales percepciones y responder o actuar en su entorno de manera racional. Todo ello tanto para peticiones de agentes internos a nuestra plataforma, como para agentes externos a nuestra plataforma, siempre y cuando estos hagan uso del estándar FIPA.

Hasta aquí, no hay mucha diferencia con cualquier otra plataforma de sistemas multi-agente.

**2.- Agent's communication**

Existen dos diferentes tipos de comunicación por medio de mensajes, en nuestra plataforma. Por una parte está la comunicación entre agentes internos, es decir mensajes cuyo emisor y receptor, están dentro de la plataforma distribuida, y por tanto tienen asociada una cola de mensajes. En este caso, se efectuará por medio de intercambio de mensajes en colas. Por otra parte la comunicación hacia el exterior se hará vía http. Además gracias al uso de este middleware de mensajería, se ha permitido una satisfactoria convivencia entre diferentes agentes implementados en Java, C#, Python, C++, Java o incluso ADA. Es decir un agente, podrá estar implementado en cualquier lenguaje de programación. Por tanto, este middleware soporta una total independencia entre el lenguaje de un agente emisor, y el lenguaje de un agente receptor. Esto ha sido posible integrando en nuestra plataforma el broker QPID (que implementa el estándar AMQP), para la comunicación entre agentes internos.

En el caso de un agente interno, tan solo bastará con que tenga asociada una cola de mensajes para poderse comunicar con el resto de agentes, y en el caso de un agente exterior, será suficiente con que conozca al Agente BridgeAgentOutIn, este le permitirá interactuar con agente internos a nuestra plataforma.

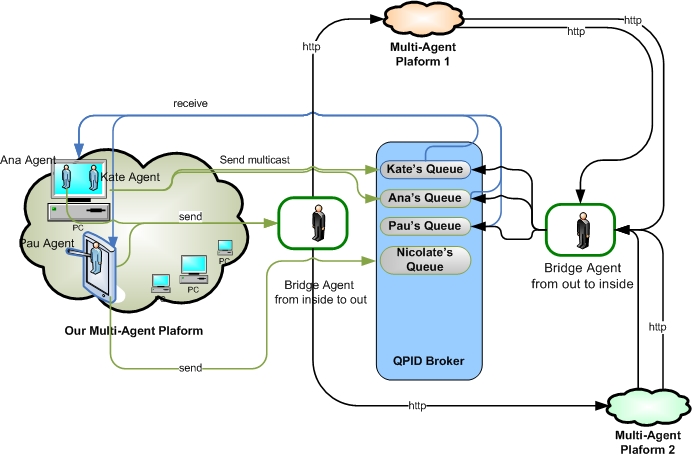
Un mensaje ha sido especificado siguiendo la normativa de FIPA-ACL, del siguiente modo: un AgentsID que identifica el agente emisor, y una lista de AgentsID, que indican el conjunto de receptores, uno o varios. El contenido, lenguaje, codificación, ontologia usada, protocolo, etc.

Como ya hemos dicho, cuando se crea un nuevo agente en nuestra plataforma distribuida, este llevará asociado una cola de mensajes, que le permitirá recibir en ella todo mensaje que vaya dirigido hacia él. De la misma forma, dado que el nombre de la cola de mensajes del agente, casa con su propio nombre, y el nombre del agente es univoco en la plataforma, cuando este quiera enviar un mensaje a otro agente, tan solo tendrá que construir un mensaje cuyo receptor sea un AgentID, con el nombre del agente destino. Además en nuestra plataforma, incorporamos el concepto de organizaciones. Un exchange en QPID, no es más que una agrupación de colas, cada cola identificada por un root key. Por tanto cada cola de un agente que pertenece a una organización, estará asociada al exchange de la organización.

Dentro de nuestra plataforma, se han diseñado dos intermediadores, BridgeAgentInOut y BridgeAgentOutIn, ambos se encargan de controlar y permitir la comunicación entre agentes internos y agentes externos, y viceversa, siempre que estos se basen en el estándar FIPA.

Dos tipos de protocolos existentes pueden ser usados en el campo protocolo de una dirección destino de un mensaje. Si el protocolo casa con "qpid" se supone que supone que el mensaje va dirigido hacia un agente interno a nuestra plataforma distribuida. Y toda comunicación entre agentes internos, se efectuara por medio de una cola de mensajes. Por contra si dicho mediador se encuentra con un protocolo "http", el mensaje será enviado al agente BridgeAgentInOut de nuestra plataforma. Este tratará de empaquetar el mensaje y enviarlo vía http, hacia otro agente exterior. Por cuestiones de eficiencia, el mediador se libera del empaquetado, creando un nuevo hilo de ejecución, encargado de ello. De este modo cualquier agente puede comunicarse con cualquier otro independientemente de el lenguaje elegido para su implementación y el lugar en el que este se encuentre, siempre que use FIPA, de una forma rápida y sencilla.

De una manera similar trabaja el agente BridgeAgentOutIn. Le llegan mensajes desde el exterior, vía http, dirigidos a un agente receptor que se encuentra en dentro de nuestra plataforma. Su tarea es el desempaquetado de un mensaje, del que debe identificar entre otras características el nombre y localización del agente destino, además del contenido, ontología, etc. Una vez desempaquetado, este mediador depositará el mensaje en la cola de mensajes, de dicho agente receptor, asociada a un determinado exchange.

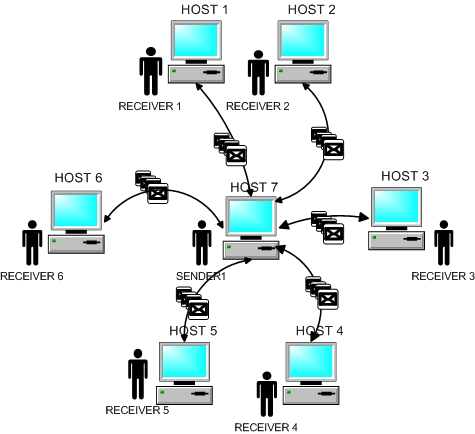


(Falta comentar la figura...) :p

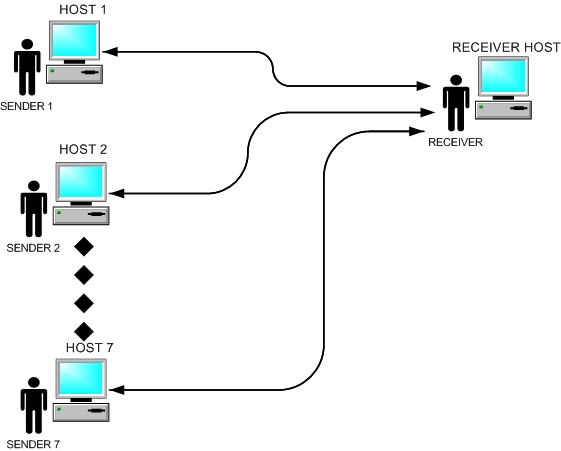
**3.- Benchmarks Description**

En esta sección detallaremos una serie de pruebas de rendimiento, efectuadas para corroborar el rendimiento de nuestra plataforma, gracias a la comunicación por medio de mensajería.

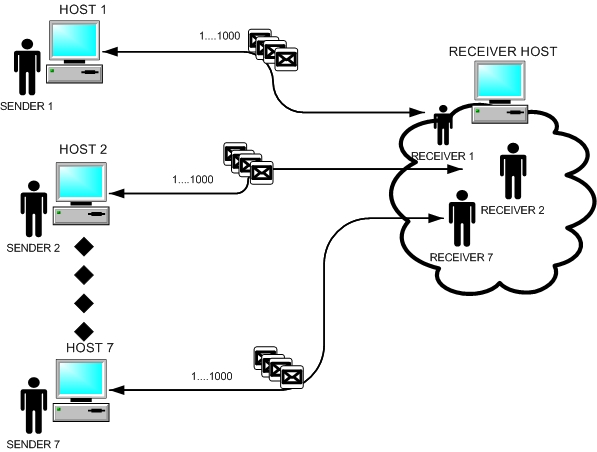
**3.1-**

****

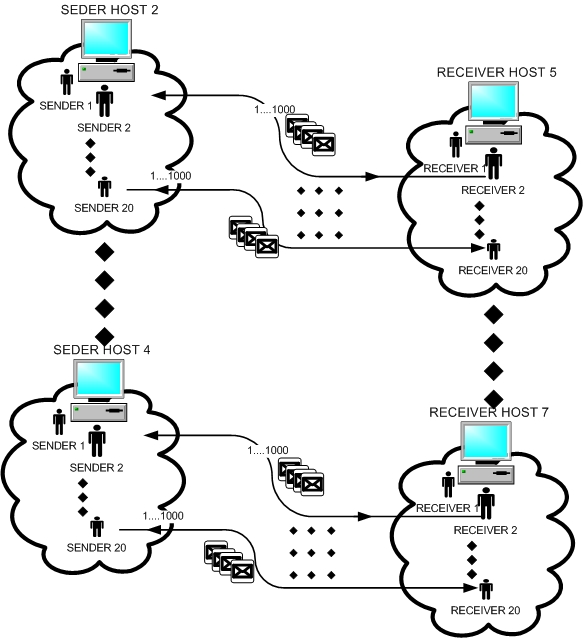
**3.2-**



**3.3-**



**3.4-**



(Comparación con Jade)

**4.- Future Work**

* En esta primera versión, no se han hecho uso de ontologías, por ello los mensajes que se envían están formados por un conjunto de bytes que no llegan a interpretarse. Se aplaza la construcción de otro agente mediador para la plataforma, que sea capaz de identificar y traducir el mensaje, según la ontología origen.
* Paralelo a este desarrollo, se ha integrado el SF y OMS de la plataforma THOMAS, en este nuevo modelo de agentes inteligentes distribuido. (Joan)

**5.- References**

***Tareas pendientes:***

***-Completar apartado benchmarks***

***-Completar con lo de Joan, OMS SF, dentro de la plataforma***

***-Traducción a Ingles***

***-Pasarlo a Latex***