imAgent: un soporte para la simulación de sistemas de agentes móviles sobre Ns

Guillermo Rigotti*, Daniel Sottile, Ricardo Tártara

Resumen

En los últimos años, el paradigma de agentes móviles ha recibido una creciente atención por parte de los investigadores. Esta actividad se ha cristalizado en el desarrollo de una variedad de sistemas de soporte para agentes móviles, y trabajos de normalización que permitirán el desarrollo de sistemas comerciales. Sin embargo, en este aspecto no se ha logrado aún la difusión esperada de tales sistemas. Se argumenta que esta situación se debe principalmente a la falta de resultados cuantitativos que inciten a los desarrolladores a adoptar esta tecnología. A pesar de la significativa actividad de investigación relacionada a sistemas de agentes móviles, no se ha planteado una evaluación de la tecnología a través de mediciones reales en escenarios a gran escala, principalmente con referencia a los mecanismos básicos involucrados en la implementación de esos sistemas. Esto se produce esencialmente por la dificultad de evaluar soluciones sobre grandes escenarios que permitan comprobar la viabilidad de la solución de manera rápida y confiable.

En este artículo se presenta un soporte de simulación para sistemas de agentes móviles desarrollado para cubrir la falencia mencionada. El objetivo buscado es posibilitar la implementación rápida de aplicaciones prototipo de agentes móviles, su simulación y obtención de resultados. El mencionado desarrollo está basado en un soporte de simulación de amplia difusión y versatilidad, Network Simulator, actualmente en desarrollo en la Universidad de Berkeley.

Palabras clave: *agentes móviles, simulación*

Abstract

In the last years, the mobile agents paradigm has received significant attention in the research community. This activity has been crystallized in the development of a variety of mobile agent systems and works of normalization which will allow the development of commercial systems. Nevertheless, in this aspect the expected deployment of such systems has still not been achieved. It is argued that this situation takes place mainly by the lack of quantitative results that stimulate developers to adopt this technology. In spite of the significant activity of investigation related to mobile agent systems, an evaluation of the technology through real measurements in great-scale scenarios has not considered, mainly in reference to the basic support mechanisms of those systems. This mainly takes place by the difficulty to evaluate solutions on great-scale scenarios; these evaluations allow to verify fast and reliably the viability of the solution.

This article presents a support of simulation for mobile agent systems developed to solve the mentioned problem. The aim of this work is to make possible the fast implementation of prototype applications of mobile agent systems, enabling its simulation and easy obtaining of results. This work is based on Network Simulator 2 (Ns 2), a simulation platform that has high functionality and versatility. Ns is widely used in the scientific community to evaluate different protocols. Ns is being developed at Berkeley University.

Keywords: *mobile agents, simulation*

1 Introducción

Debido a sus notables características, el paradigma de agentes móviles ha recibido una creciente atención durante los últimos años. Los agentes móviles son a menudo descriptos como una

* UNICEN – Fac. de Ciencias Exactas-ISISTAN, Pje. Arroyo Seco, (7000) Tandil, Bs. As. Argentina, grigotti@exa.unicen.edu.ar

tecnología prometedora para realizar transacciones y obtener información en sistemas heterogéneos y ampliamente distribuidos, y se los considera muy apropiados para un mercado electrónico unificado y escalable. A pesar de que las ventajas individuales de los agentes móviles no parecerían representar una motivación suficientemente importante que lleve a su aplicación, la existencia de un framework de agentes competente facilitaría el desarrollo de un gran número de aplicaciones y servicios de red.

La comunidad de investigación involucrada en el área de agentes móviles está creciendo constantemente y cada día más sistemas avanzan en su desarrollo. Paralelamente existe un progreso en los esfuerzos por lograr una estandarización de las arquitecturas utilizadas. Dado que el buen funcionamiento de sistemas de agentes móviles a gran escala es crucial para el éxito de esta tecnología, los problemas y las necesidades actuales en esta área deben ser bien comprendidos.

Existen aspectos de comunicación y coordinación de agentes, seguridad, formación de grupos y ubicación de agentes, entre otros, que aún requieren un mayor estudio que les permita incrementar su grado de maduración.

Existen escenarios a los cuales los sistemas de agentes móviles se adaptan de manera satisfactoria, como por ejemplo la comunicación de dispositivos sobre redes wireless e Internet debido a la reducción de vulnerabilidad ante desconexiones de la red y a la baja capacidad de muchos de los dispositivos wireless, que impiden la ejecución de procesos de las características que demanda la aplicación, funcionalidad que entonces es delegada a los agentes que se ejecutan en plataformas remotas.. Sin embargo, para introducir agentes móviles en estos escenarios, se necesitan soluciones de soporte escalables debido al amplio crecimiento que puede alcanzar el uso de esta tecnología. Es evidente que las soluciones existentes no cuentan aún con una aceptación generalizada que les permita ser incluidas en los sistemas comerciales actuales.

Algunos investigadores [1] argumentan que esta situación se debe, entre otras causas, a la falta de resultados cuantitativos que inciten a los desarrolladores a adoptar esta tecnología. Varios expertos han analizado el desarrollo de sistemas de agentes móviles, pero muy pocos han planteado una evaluación de la tecnología a través de mediciones reales en escenarios a gran escala, conjuntamente con una evaluación de los mecanismos básicos involucrados en la implementación de esos sistemas. Esto se debe principalmente a la escasez de medios donde evaluar soluciones sobre grandes escenarios que permitan comprobar la viabilidad de la solución de manera rápida y confiable.

Una posible opción es el planteo analítico del caso a evaluar, pero cuando la introducción de elementos estocásticos resulta en un escenario complejo surge la simulación computacional como un medio más adecuado de evaluación.

Generalmente en un simulador es más sencillo reproducir cualquier escenario independientemente de su complejidad o características especiales, que en un sistema real. La simulación se presenta como una solución económica y viable ya que habitualmente consume recursos mínimos. Dado que no necesita desarrollarse en tiempo real pueden obtenerse resultados rápidamente y repetir el experimento tantas veces como sea necesario. Además, los riesgos de una simulación pueden considerarse nulos ya que no se compromete el funcionamiento de ningún sistema existente.

Una simulación puede generar resultados tan válidos como los que se obtendrían en pruebas sobre un escenario real. Los resultados obtenidos de la simulación generalmente son de naturaleza cuantitativa.

El objetivo principal de este trabajo fue la implementación de un soporte para el rápido desarrollo de prototipos de aplicaciones de agentes móviles, que permita comprobar a través de simulación, el comportamiento de dichas aplicaciones en diferentes medioambientes de operación. Como fue mencionado, esta alternativa resulta particularmente ventajosa en escenarios en los cuales otras

técnicas presentarían grandes dificultades debido a sus limitaciones. Para lograr el desarrollo de un prototipo adaptado a necesidades reales, se estudiaron y comprendieron las necesidades actuales de la comunidad científica del área de sistemas de agentes móviles prestando atención a las principales tendencias reflejadas en la bibliografía e investigaciones recientes.

Dado que a la fecha no se conoce la existencia de un software orientado a la simulación de casos generales de agentes móviles, pero existiendo plataformas generales de simulación de redes de computadoras, se optó por desarrollar `i`mAgent, un soporte para simulación de sistemas de agentes móviles, sobre una plataforma de simulación rica en funcionalidad y ampliamente difundida, Network Simulator (Ns) [2]. Posteriormente a la realización del simulador, se evaluaron diferentes aspectos de sistemas de agentes móviles cuidadosamente seleccionados a partir del estudio previo. En particular, la realización de experimentos basados en componentes ya existentes de soporte de agentes móviles permitió evaluar la técnica de simulación como medio de prueba de nuevos componentes en proceso de desarrollo. A los fines de validar los beneficios de la simulación en la evaluación de componentes se implementó un caso de estudio con características que en la práctica invalidarían la utilización de otras técnicas y se analizó la calidad de los resultados de experimentación obtenidos.

Los trabajos relacionados con el que aquí se presenta, a diferencia de éste, cuyo objetivo es la simulación de casos reales, evaluando combinaciones de estrategias a nivel aplicación y a nivel soporte, apuntan a la evaluación de casos particulares de aplicación de agentes móviles en ciertos escenarios, o a evaluaciones genéricas de aspectos específicos de soporte independientes de la aplicación.

Entre los primeros, [3] presenta un soporte de simulación orientado a IP Mobility [4], de manera similar a lo realizado en el presente trabajo, los autores se basan en un simulador de redes [5] que permite con mayor facilidad que Ns incorporar funciones relativas a IP Mobility. Una diferencia notoria con nuestro trabajo radica en que en aquél la entrada de datos a la simulación se realiza a través del ingreso de los parámetros relevantes utilizando una interfaz gráfica, no permitiendo de esta manera el volcado de una aplicación real en el medioambiente de simulación de una manera directa.

En [6] se propone y evalúa a través de simulación, la alternativa de utilizar agentes móviles para

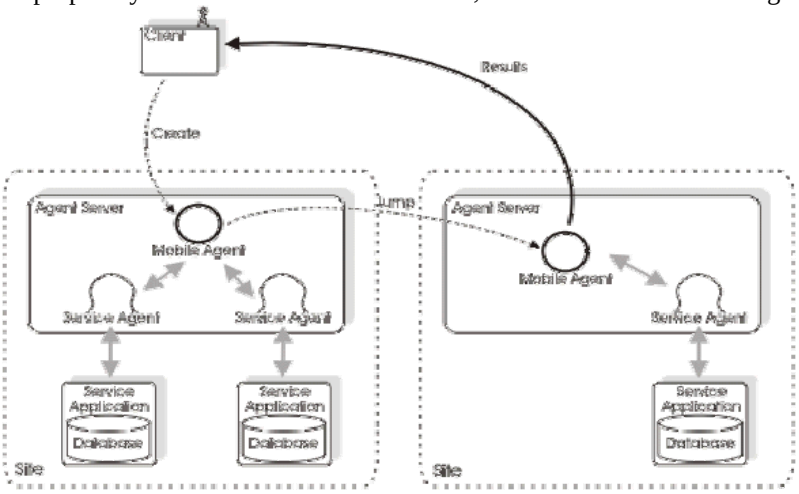

Figura 1 – Sistema de agentes móviles.

envío de mensajes entre nodos de redes wireless. Los autores utilizan un ambiente de simulación implementado específicamente para evaluar su propuesta. A nuestro criterio, trabajos de este tipo podrían utilizar `i`mAgent para comprobar resultados, sin necesidad de recurrir a métodos específicamente desarrollados para cada caso.

Entre los trabajos que evalúan aspectos de manera genérica, [7] tiene por objeto la evaluación de tres técnicas de migración de agentes móviles que a juicio de los autores cubren el espectro total de posibilidades de migración. Esta evaluación se realiza de manera genérica, es decir, sin vincularla a ninguna aplicación en particular ni confrontarla con posibles estrategias a nivel aplicación.

El resto del artículo está organizado de la siguiente manera. En la sección 2 se describe brevemente el modelo de sistemas de agentes móviles que se utilizó como referencia para el desarrollo del presente trabajo; en la sección 3 se describe el contexto en el cual fue desarrollado `i`mAgent. En la sección 4 se describe su arquitectura, completándose con la descripción de sus componentes principales y de las facilidades implementadas en las subsecciones 4.1 y 4.2 respectivamente. En la sección 5 se describe un caso particular de uso del simulador, consistente en la evaluación de la *performance* de dos algoritmos de localización de agentes. Finalmente, la sección 6 contiene las conclusiones derivadas del trabajo realizado así como las futuras mejoras a realizar. La bibliografía relevante se indica en la sección 7.

2 Modelo de sistema de agentes móviles

Un sistema de agentes móviles contiene un conjunto de sitios en los cuales residen los servidores de agentes móviles. Los agentes móviles son entidades activas, creadas por aplicaciones clientes del sistema, que se trasladan de sitio en sitio llevando a cabo funciones específicas, que pueden incluir colaboración con otros agentes móviles, orientadas a cumplir con los objetivos impuestos por la aplicación.

Los sitios se encuentran interconectados a través de una red y cada uno de ellos provee un ambiente adecuado para la ejecución de agentes. Su ubicación corresponde a un único nodo de la red, aunque varios sitios pueden compartir un mismo nodo.

Los clientes son aplicaciones que representan a los usuarios interesados en acceder a los servicios que se brindan en los sitios. Posiblemente se encuentren ejecutando sobre dispositivos de limitada capacidad de procesamiento conectados a la red de manera intermitente desde diferentes direcciones y con escaso ancho de banda, condiciones que dificultan el acceso directo a los servicios.

Dado que un cliente de estas características no tiene recursos para soportar la ejecución de agentes, éstos son creados directamente en el sitio deseado (ver Fig. 1). En ese momento cada agente recibe un identificador único dentro del sistema, que se mantiene invariable durante todo su ciclo de vida. Mediante el envío de mensajes, el cliente podrá interactuar con sus agentes para controlarlos o recibir resultados. Un agente, luego de iniciada su ejecución, podrá migrar hacia otros sitios en busca de recursos que le permitan cumplir con su tarea. Además podrá clonarse a sí mismo, creando un “child agent” en cualquier sitio, y en él delegar parte de su tarea.

Existen mecanismos que permiten a los agentes organizarse para realizar su trabajo en forma colaborativa. La comunicación entre agentes es indispensable en tales circunstancias. Los agentes podrán formar grupos de trabajo que les permitan trabajar en forma coordinada con un objetivo en común.

Los servicios son provistos por aplicaciones de tipo servidor que se encuentran en algún nodo de la red esperando por solicitudes. Los agentes de servicios son agentes generalmente estáticos que conocen los puntos de acceso a algunos de estos servicios y los representan dentro del sistema, poniéndolos a disposición de otros agentes y controlando el acceso a los mismos.

3 Medioambiente de desarrollo y operación de imAgent

Un simulador debe ser una herramienta que permita la exploración rápida y a bajo costo del comportamiento de nuevos componentes en proceso de desarrollo. Debe posibilitar la evaluación de estos componentes en pequeña y gran escala bajo un ambiente controlado y de condiciones variables, características esenciales para llegar a resultados válidos. El estudio de componentes en forma individual y en la interacción con otros componentes es indispensable para lograr comprender el comportamiento y las características de los mismos.

En un simulador es importante contar con un alto grado de flexibilidad y extensibilidad. La posibilidad de agregar nueva funcionalidad evita que el simulador se encuentre limitado en su campo de acción. Se debe prestar especial atención a la producción de resultados. El formato y el grado de detalle de los resultados obtenidos de una simulación deben satisfacer las necesidades del usuario del simulador, facilitando así su posterior análisis y comparación.

Una característica de importancia que tiene un impacto substancial en la utilidad del simulador es la disponibilidad de una amplia variedad de componentes de biblioteca. Contar con ello significa una reducción en el tiempo de desarrollo de las simulaciones, permitiendo al usuario concentrarse en aquellos aspectos de la simulación relevantes al tema en estudio.

Entre estos componentes es útil contar con generadores automáticos de escenarios que permitan al usuario cubrir mayores porciones del espacio operacional de lo que se logra mediante la construcción manual. Además, la posibilidad de repetición de pruebas sobre escenarios idénticos otorga mayor validez a la comparación de componentes de funcionalidad similar y permite lograr casos de estudio más significativos.

Todas estas características se encuentran presentes en imAgent. Su composición se basa en un conjunto de clases que representan el comportamiento básico de un sistema de agentes móviles en general, además de los componentes propios de simulación. Adicionalmente cuenta con un grupo de componentes de biblioteca listos para ser utilizados por el usuario.

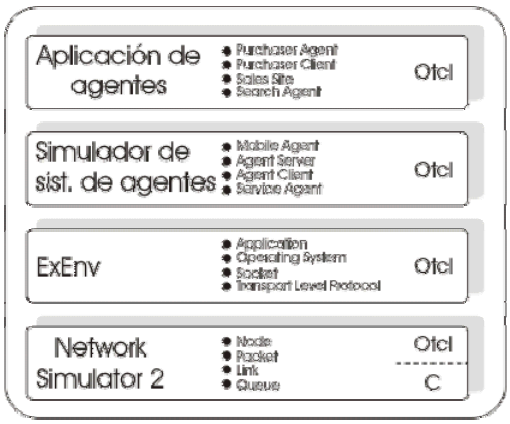

Figura 2 – Composición del simulador

imAgent funciona sobre ExEnv; un entorno de ejecución de aplicaciones desarrollado previamente sobre Ns para simplificar su desarrollo (ver Fig. 2). Aunque ExEnv fue desarrollado para facilitar el desarrollo de imAgent, debe destacarse que es posible utilizarlo para otras aplicaciones sobre Ns, permitiendo un desarrollo más rápido de las mismas, y reduciendo la brecha existente entre el desarrollo de aplicaciones reales y simuladas.

Ns es un simulador orientado a la evaluación del comportamiento de protocolos. Provee implementaciones de los protocolos más difundidos que pueden ser utilizados por aplicaciones de usuario que generan determinados patrones de tráfico para simular el comportamiento deseado y luego evaluar resultados.

En Ns los parámetros de una simulación son expresados con un programa compuesto por instrucciones del lenguaje Otcl [8] que básicamente se trata de una extensión orientada a objetos del lenguaje Tcl [9]. Este lenguaje tiene la ventaja de que no necesita un proceso de compilación antes de ser ejecutado. Eso lo convierte en el ideal cuando se necesita un tiempo de iteración reducido (cambiar el modelo y volver a ejecutar) que permita explorar distintos escenarios rápidamente. Como lenguaje dinámico tiene la desventaja de ser lento en su ejecución, pero no es una característica que afecte en gran medida el proceso de simulación ya que generalmente cada simulación se ejecutará una única vez.

La cantidad de nodos, la topología de la red y las características de los enlaces son los principales parámetros que deben ser especificados. Establecida una configuración determinada, Ns lleva a cabo la simulación basando su ejecución en un *scheduler* de eventos cuya función es la generación de estímulos dentro del escenario simulado. Dado que la ejecución de código Otcl sobre Ns no consume tiempo de simulación, es el *scheduler* el encargado de administrar el consumo de tal tiempo.

ExEnv es una solución a la falta de componentes de Ns que permitan implementar fácilmente simulaciones para evaluar componentes de más alto nivel que un protocolo de comunicaciones. Su función principal es facilitar la construcción de simulaciones compuestas por aplicaciones que utilizan una red de comunicaciones para interactuar con sus pares en la resolución de una tarea determinada. El usuario de ExEnv evita lidiar con detalles de comunicaciones como lo es la manipulación de cada uno de los protocolos tal cual los provee Ns, y puede concentrarse en la implementación de las características del escenario a estudiar.

La composición de ExEnv se basa en un conjunto de clases Otcl que representan el comportamiento básico de un entorno que posibilita la ejecución y comunicación entre aplicaciones situadas en diferentes nodos de una red. Introduce el concepto de sistema operativo y de llamadas al sistema, características inexistentes en Ns. Redefine la interfaz de interacción con los protocolos de Ns para que sean utilizados en una forma más sencilla y cercana a la realidad.

El usuario de ExEnv debe extender algunas clases para implementar el comportamiento deseado. Esta extensión generalmente involucra la implementación de las aplicaciones específicas y la definición del escenario a utilizar.

Sobre ExEnv funciona `i`mAgent, el cual basa su ejecución en el mismo scheduler de Ns. El conjunto de clases Otcl que conforman `i`mAgent representa una configuración extensible de Ns orientada a la simulación de aplicaciones basadas en el paradigma de agentes móviles.

4 Arquitectura de `i`mAgent

El diseño general de `i`mAgent se basó en los siguientes objetivos:

- Facilitar la incorporación de componentes de usuario.
- Minimizar el esfuerzo necesario para la adaptación o extensión de los componentes existentes.
- Acelerar la construcción de simulaciones mediante la utilización de componentes de biblioteca.
- Permitir la generación de resultados sin restricciones y en función de las necesidades del usuario.

Una arquitectura de eventos resulta efectiva en el cumplimiento de tales objetivos dado que provee la flexibilidad necesaria. Permite un alto grado de modularización que facilita la incorporación y evaluación de nuevos componentes creados por el usuario. En imAgent, cada funcionalidad específica se encuentra aislada en un componente especializado. El usuario dispone de los mecanismos necesarios para reemplazar fácilmente cualquier componente, adaptando el comportamiento de la simulación a sus necesidades.

Los componentes que intervienen en una simulación generan eventos informando todo lo que en ellos sucede. También existen componentes interesados en tales eventos que serán notificados ante la producción de cada evento del tipo al que previamente se hayan suscripto.

Los principales participantes de una simulación son agentes, servidores y clientes. Estos son, por naturaleza, generadores de eventos. Ellos deben publicar cada una de sus acciones y encuentran en la generación de eventos el mecanismo adecuado. Existen otros componentes que actúan como receptores de eventos. Ellos generalmente registran todo lo sucedido durante la simulación y procesan esa información.

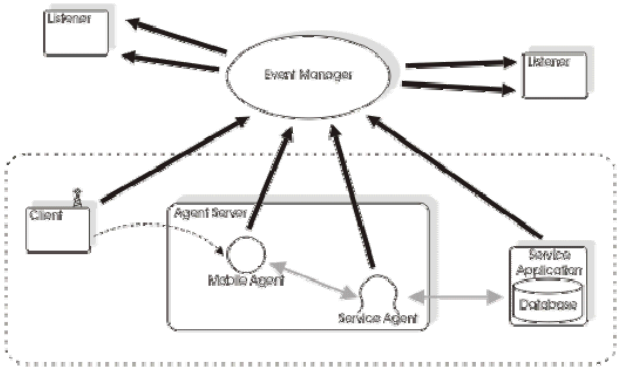

Figura 3 – Arquitectura del simulador.

Puede verse a imAgent como una caja generadora de eventos, publicando de esta manera todo lo que sucede durante una simulación. En forma externa a imAgent, los eventos son distribuidos por un administrador de eventos (Event Manager) entre los componentes interesados en recibirlos. Para una descripción más detallada de los componentes y funcionalidad de los sistema de agentes, ver [10].y [11].

4.1 Componentes

4.1.1 Agentes

Un agente móvil es un programa de computadora [12] [13] que, junto con su código y estado de ejecución, puede moverse de sitio en sitio en busca de recursos. Los agentes móviles de imAgent, representan agentes móviles reales operando sobre los nodos de una red simulada en una única computadora. El principal objetivo de su diseño fue su facilidad de programación por parte de los usuarios del simulador. La construcción de un nuevo agente prácticamente no requiere conocimientos acerca de la forma en que funciona el resto del simulador. Tampoco es necesaria la implementación de aspectos no relacionados con la tarea específica del agente. Operaciones como el envío de un mensaje entre agentes, la migración o la clonación de un agente sólo implican la invocación del comando correspondiente con los parámetros deseados. Un agente base de imAgent provee una interfaz clara y fácil de entender y utilizar. Su estructura general es sencilla. Sólo contiene unos pocos métodos más un conjunto de variables de significado especial.

Un agente que pretende migrar, crear otro agente o detener su ejecución, entre otras cosas, debe invocar comandos provistos por el sitio en el cual se encuentra. Para hacerlo debe acceder al servidor de agentes móviles que lo contiene. Esto es posible mediante una variable de referencia al servidor del sitio actual del agente. Así, cualquier comando del servidor puede ser invocado.

4.1.2 Servidor de agentes

Un servidor de agentes móviles es un componente de software que provee un ambiente con las características suficientes para la ejecución de agentes móviles. Cada sitio que desee ser visitado por agentes deberá contar con un servidor de agentes móviles. El conjunto de servidores que funcionan sobre una red constituye el soporte del sistema de agentes móviles.

`im`Agent provee una implementación básica de servidor de agentes móviles en condiciones de ser utilizada en cualquier simulación. Su construcción basada en una arquitectura interna de eventos brinda un alto grado de flexibilidad y extensibilidad. Gran parte de la funcionalidad de soporte del sistema está implementada en componentes del servidor, característica imprescindible para permitir la experimentación con componentes nuevos. La llegada de un agente, el acceso a un recurso o la llegada de un mensaje para un agente, son ejemplos de eventos que serán comunicados a todos los componentes internos del servidor que se hayan suscripto para tal fin. A partir de la notificación de uno de estos eventos, cada componente ejecutará las acciones correspondientes al caso.

Los principales componentes del servidor, que se muestran en la Fig. 4 son:
1-*Event Manager*. Es quien conoce los tipos de eventos que son de interés de cada componente. Además es el encargado de la distribución de los eventos.
2-*Agent Container*. Aloja a los agentes que llegan al servidor, provee los mecanismos de creación y migración de agentes y mantiene la información de estado correspondiente a cada agente durante toda su estadía en el servidor. Es el único componente que puede acceder en forma directa a un agente.

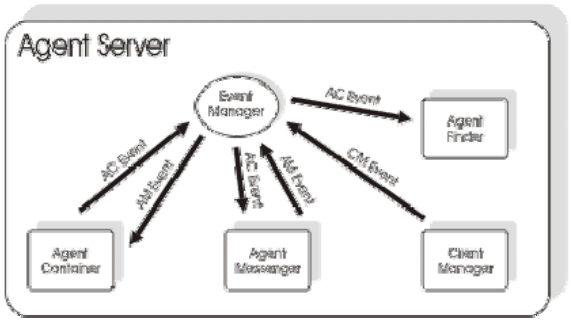

Figura 4. Componentes del servidor de agentes

3-*Agent Messenger*. Es el componente principal de comunicación, encargado del envío, distribución y recepción de mensajes entre agentes y clientes. Puede administrar convenientemente los mensajes que no puedan ser entregados en forma inmediata.
4-*Client Manager*. Administra los clientes registrados en cada sitio y su conexión y desconexión del sistema de agentes móviles.
5-*Agent Finder*. Este componente tiene la responsabilidad de ubicar agentes en la red.

Estos componentes forman parte de la implementación básica del servidor de agentes móviles. El usuario del simulador tiene la posibilidad de cambiar o agregar nuevos componentes para lograr el comportamiento específico que desee simular.

4.1.3 Cliente

Un cliente de un sistema de agentes móviles es una aplicación que utiliza el paradigma de agentes móviles para resolver un problema en particular. El cliente es quien crea y controla los agentes móviles que trabajan en su beneficio. Un sistema de agentes móviles puede tener múltiples clientes en un momento dado.
En los clientes provistos por imAgent es necesario contar con las mismas características de diseño que en los servidores. Es por ello que para su construcción también se adoptó una arquitectura de eventos. La Figura 5 muestra los componentes que integran un cliente y la forma en que interactúan.

Los componentes del cliente son:
1-*Event Manager*. En la implementación provista por imAgent, por simplicidad es el cliente mismo quien cumple el rol de Event Manager. Un nuevo componente que extienda tal funcionalidad puede ser agregado sin dificultades.
2-*AgentServer Connector*. Implementa el mecanismo de conexión del cliente al sistema de agentes móviles.
3-*ClientAgent Messenger*. En este componente el cliente delega la tarea de realizar el envío de mensajes destinados a agentes.
4-*Agent Creator*. Es el componente que conoce e implementa los mecanismos necesarios para crear un nuevo agente en un sitio especificado. El cliente, luego de establecer algunos parámetros, delega en este componente la creación de sus agentes.

4.2 Funciones implementadas

4.2.1 Creación de agentes

En su estado inicial un sistema de agentes móviles no contiene agentes. Éstos son creados posteriormente con algún fin específico.

Los agentes pueden ser creados por la aplicación cliente para optimizar el acceso a algún recurso; o bien por otro agente, quien delega en el recién creado parte de su tarea. En imAgent se contemplan ambas posibilidades.

En el caso en que el cliente necesite crear un agente, delega éste el trabajo en su componente interno *Agent Creator* al invocar la operación *createAgent*. Debe destacarse que la creación de un agente puede hacerse directamente en el sitio donde el agente debe comenzar su trabajo, lo cual resulta ventajoso cuando el cliente del sistema es un dispositivo de limitada capacidad de procesamiento que carece de los recursos necesarios para ejecutar agentes.

En el caso en que un agente sea creado por otro agente, este último se convierte en el *Root Agent* del agente creado, al cual se lo denomina *Child Agent*. El *Root Agent* ejerce la función de propietario del *Child Agent* y por ello es quien debe controlar al mismo. La creación de *Child Agents* puede verse como una estrategia adoptada por el *Root Agent* para resolver su problema. Un agente que desee crear *Child Agents* debe invocar el comando fork en el servidor del sitio en que se encuentra El subcomponente del *Agent Container* que implementa esta funcionalidad es el *Child Creator*.

Al crear un agente, hay algunos parámetros de configuración de la operación que deben ser especificados. Estos son comunes en ambos casos, la creación del agente por parte del cliente y la creación del agente por parte de otro agente. Uno de los parámetros es el conjunto de clases propias del agente más las que necesitará durante su ejecución. Otros parámetros son la dirección del sitio donde el agente será creado y los parámetros de inicialización del agente. Estos últimos permiten que el creador configure el comportamiento del agente y serán entregados al mismo en el momento de su creación.

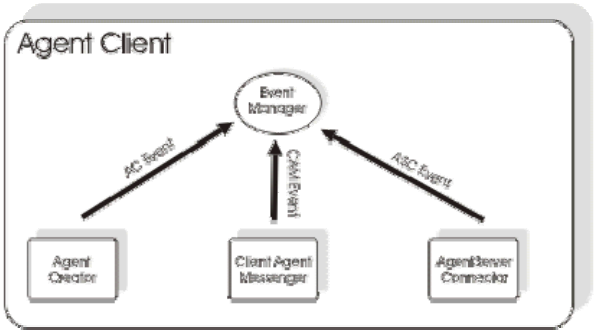

Figura 5 - Componentes del cliente

Reunidos todos los parámetros, el *Agent Creator* se encarga de crear el agente en el sitio remoto especificado. Finalizado este proceso, el creador es notificado y los resultados del mismo le son entregados. Hay dos resultados posibles dependiendo del éxito o fracaso de la operación y se los identifica como *agentSubmitted* y *submitRejected* correspondientemente.

Desde el punto de vista del servidor, la solicitud de creación de un agente móvil llega en forma de un mensaje remoto, y luego de un proceso de interpretación del mensaje, se ponen en funcionamiento los mecanismos necesarios para cumplir la tarea. El componente interno al servidor encargado de la administración de los agentes es el *Agent Container*. En este caso su trabajo se resume en la creación y puesta en ejecución del agente y en la generación de la identificación correspondiente al mismo.

En particular, en el *Agent Container* provisto por `im`Agent la primera parte de la tarea consiste en algunas verificaciones sobre la validez de la información recibida, incluyendo clases y demás parámetros. Pasada satisfactoriamente esta etapa se crea la instancia del agente, entregándosele los parámetros de inicialización enviados por el creador del agente. Las variables especiales del agente son cargadas con los valores correspondientes y luego el método start del agente es invocado representando el inicio de su primera ejecución. A partir de allí el agente podrá decidir el camino a seguir. La tarea finaliza con un mensaje de respuesta al creador del agente, conteniendo el resultado de la operación y, en el caso de un final exitoso, la identificación que recibió el nuevo agente. En forma interna al servidor, la creación del agente y el comienzo de su ejecución son publicados mediante eventos.

El comportamiento de este *Agent Container* puede ser extendido o reemplazado parcial o totalmente por el usuario del simulador sin que esto requiera mayores esfuerzos.

Existe un caso especial de creación de *Child Agents* que resta analizar. Se trata de la clonación de agentes. Básicamente consiste en un *Child Agent* instancia de las mismas clases que el *Root Agent* correspondiente y con el mismo estado interno. La operación que un agente debe invocar para clonarse en el mismo o en otro sitio es clone. En este caso sólo se debe indicar el destino donde se creará el clon y los parámetros de inicialización, ya que las clases se extraen del Root Agent automáticamente. El subcomponente del *Agent Container* que implementa esta funcionalidad es el *Clon Creator* y su comportamiento es similar al *Child Creator* salvo algunos detalles menores. Una operación de clonación que finaliza de manera exitosa se identifica con un *agentCloned* y el caso contrario con un *cloneRejected*.

4.2.2 Migración

Una de las ventajas de los agentes móviles con respecto a otras técnicas radica en la capacidad de los agentes de migrar hasta el sitio donde están los recursos que necesitan. Esto generalmente

permite optimizar el acceso a tales recursos y reducir el ancho de banda necesario para completar la tarea.

En imAgent, un agente móvil puede migrar al sitio deseado en cualquier momento de su ejecución. Para hacerlo, el propio agente debe invocar el comando jump en el servidor del sitio en que se encuentra, indicando la dirección del sitio destino. A partir de allí, el servidor se encarga de poner en funcionamiento los mecanismos necesarios para la transferencia del agente.

La migración de un agente significa la transferencia del mismo junto con su código y datos. imAgent provee soporte para migración débil (*weak migration*), o sea que, independientemente del punto en que el agente invoca al jump, en el sitio destino reanudará su ejecución desde un punto fijo en una nueva pila de ejecución. Sólo las variables que el agente indique explícitamente mantendrán su existencia y valor luego de la migración.

Este último concepto se implementó siguiendo un modelo de *Briefcase* similar al del sistema de agentes Tacoma [14]. Un agente, antes de migrar, debe recolectar todos los datos que desea conservar y almacenarlos en el *Briefcase*, al cual se accede a través de una variable especial. En el momento en que la ejecución del agente se reinicia en el sitio destino, toda la información contenida en el *Briefcase* ya habrá sido automáticamente restablecida.

Una migración que finaliza de manera exitosa resultará en la ejecución del método start del agente en el servidor destino. Una variable especial denominada *migrationCount* almacena el recuento de migraciones del agente y puede ser accedida por el mismo en cualquier momento de su ejecución.

Existen motivos, como una mala indicación de la dirección de destino, que pueden imposibilitar la migración. En este caso, luego de invocar el jump, el agente reiniciará su ejecución en el mismo sitio en que estaba pero en el método *jumpFailed*. Todas sus variables, incluso las que no hayan sido colocadas en el *Briefcase*, estarán intactas.

Desde el punto de vista del servidor de agentes, es el *Agent Migrator*, un subcomponente del *Agent Container*, quien tiene la responsabilidad de migrar al agente. Cuando recibe la invocación del jump por parte del agente debe verificar, entre otras cosas, que la migración esté autorizada. Por defecto en la implementación actual todas las migraciones se autorizan, pero en una implementación de usuario puede haber motivos que impidan la migración de un agente a un determinado sitio o en un determinado momento.

La siguiente tarea del *Agent Migrator* es construir el mensaje de migración y enviarlo al servidor destino, para luego esperar por una confirmación con el resultado de la operación. Dependiendo del valor de este resultado el *Agent Migrator* resuelve quitar al agente del contenedor o invocar su método *jumpFailed*.

El mensaje de migración será recibido por el *Agent Container* del servidor en el sitio destino. Luego que la validez de su contenido es verificada, la instancia del agente es restablecida junto con su *Briefcase*, y el método start es invocado. En ese momento, se publica un evento indicando que un agente ha comenzado su ejecución.

Según las características de cada agente, el tamaño del mensaje de migración puede variar. Una mayor cantidad de clases necesarias o una mayor cantidad de variables transportadas en el *Briefcase* aumentan el tamaño del mensaje a transmitir. Es normal que durante el ciclo de vida de un agente, éste vaya recolectando información que deba llevar a todos los sitios que visite, y por lo tanto su tamaño será cada vez mayor. Durante el diseño de una aplicación de agentes móviles es importante tener en cuenta esta situación para adoptar una estrategia que no genere agentes de gran tamaño.

4.2.3 Localización

Debido al alto grado de movilidad de los agentes que participan en un sistema de agentes móviles, en necesario contar con funciones que permitan ubicarlos. Los clientes deben poder controlar el comportamiento de sus agentes, lo que requiere el envío de directivas. Asimismo los agentes a menudo necesitan cooperar entre sí para la realización de sus funciones intercambiando información con otros agentes. Para poder llevar a cabo esta comunicación es preciso conocer la ubicación de los agentes de destino de las operaciones.

La arquitectura de `i`mAgent provee un componente de los servidores que cumple con esa función. Se trata del *Agent Finder*, el cual incluye un método principal llamado *findAgent*, cuya finalidad es ubicar agentes en la red a partir del identificador del agente. Cualquier componente interesado en conocer la ubicación de un agente debe utilizar, a través del servidor, los servicios del *Agent Finder*. Para ello existe una clase denominada *AgentFinderClient* que define los métodos *agentFound* y *agentNotFound* que debe implementar un componente que necesite ubicar agentes. Estos métodos se corresponden con los dos resultados posibles que puede provocar el *Agent Finder* al finalizar una búsqueda. Una búsqueda exitosa entrega la dirección del sitio donde se encuentra el agente.

Una particularidad del *Agent Finder* es su soporte de funcionamiento durante las migraciones de agentes. Dado que las migraciones no son atómicas, no puede considerarse que un agente estará en todo momento de su ciclo de vida ejecutando en un servidor de agentes [15]. Una migración en progreso consume tiempo en un escenario real y por lo tanto también debe hacerlo en una simulación auténtica. Durante ese lapso el agente no tiene una ubicación concreta y debe esperarse una indicación de éxito o fracaso de la migración. Eso es lo que sucede en `i`mAgent. El *Agent Finder* puede detectar los momentos en que un agente se encuentra en proceso de migración ya que está preparado para escuchar los eventos publicados por el *Agent Container* a tal efecto. Si se produce un requerimiento de ubicación de un agente durante su migración, el *Agent Finder* puede suspender la búsqueda hasta que se conozca con seguridad el sitio donde el agente ha reanudado su ejecución. Luego la búsqueda continuará en ese sitio.

La implementación del *Agent Messenger* incluida en `i`mAgent es un ejemplo de un componente del servidor que utiliza los servicios del *Agent Finder*. Una de las funciones del *Agent Messenger* es proporcionar un mecanismo de envío de mensajes destinados a agentes, y la implementación provista por `i`mAgent recurre al *Agent Finder* para lograr que el servicio de comunicación sea independiente de la ubicación del agente.

Por otro lado, un agente también puede requerir la ubicación de otro agente por motivos relacionados con su tarea específica. Para estos casos los agentes tienen acceso al servicio de ubicación a través de un comando ofrecido por el servidor.

`i`mAgent pone a disposición del usuario del simulador varias implementaciones del *Agent Finder* que podrán ser seleccionadas según las necesidades de las aplicaciones de agentes móviles. En particular se provee uno llamado *Optimal Agent Finder* que obtiene la ubicación de los agentes sin consumir tiempo de simulación. Se utiliza generalmente con fines comparativos, para analizar el comportamiento de otros algoritmos. También existen implementaciones que reflejan casos reales de algoritmos conocidos de búsqueda. En los algoritmos *Path Proxies Agent Finder* y *Forwarding Pointers Agent Finder* los servidores de agentes móviles envían mensajes de red y colaboran entre sí para ubicar un agente. Estos son objeto de análisis, como se describe en la sección 5.

4.2.4 Comunicaciones

Las facilidades de comunicación a alto nivel, son un servicio fundamental que un servidor de agentes debe proveer, a efectos de permitir la interacción entre los distintos componentes de la aplicación sin que éstos deban involucrarse en tareas de bajo nivel tales como el uso de *sockets*.

En imAgent se pueden distinguir tres tipos de comunicación de alto nivel. En una comunicación Cliente→Agente, los clientes del sistema de agentes móviles pueden comunicarse con los agentes que han creado, ya sea para enviarles directivas o con el fin de controlar su ejecución. Por otra parte, los agentes deben poder informar a sus clientes los resultados intermedios de sus tareas o anunciar la terminación de las mismas. También pueden estar preparados para responder dinámicamente a nuevas demandas del cliente. Estas situaciones evidencian la necesidad de un segundo tipo de comunicación denominado Agente→Cliente, donde los clientes son los destinatarios de mensajes de sus agentes. El tercer tipo de comunicación que se reconoce es importante para que los agentes puedan colaborar entre sí en la realización de sus funciones. Se trata de la comunicación Agente→Agente, mediante la cual un agente puede transmitir información a otros agentes, ya sea relativa a resultados intermedios o información de coordinación, que les será útil para completar una tarea con mayor rapidez o para no consumir tiempo en operaciones redundantes. Asimismo, cuando un agente requiere un servicio de aplicación específico debe contactarse con el agente que representa dicho servicio en el sistema y formular el pedido correspondiente. Los agentes de servicios también hacen uso del mecanismo de comunicación para responder a estos pedidos.

El componente del servidor encargado de encapsular estas cuestiones de comunicación se denomina *Agent Messenger*. Presenta métodos correspondientes a la funcionalidad de cada uno de los tipos de comunicación descriptos. imAgent cuenta con una eficaz implementación de un *Agent Messenger c*omo componente de biblioteca. El conjunto de los *Agent Messenger* de todos los servidores constituye el soporte principal de comunicación del sistema de agentes móviles. En todos los casos de comunicación la funcionalidad que provee este componente es el envío de cadenas de caracteres, que brindan a las aplicaciones la libertad de definir su propio protocolo de comunicación, de acuerdo al formato e interpretación de sus mensajes. La capacidad de este componente radica en que el servicio de comunicación que brinda es independiente de la ubicación del destinatario del mensaje. Esto facilita considerablemente la escritura de las aplicaciones, ya que el programador no necesita preocuparse por ubicar a un agente antes de enviarle un mensaje. De forma similar, la programación de un agente tampoco requiere conocer cuál es la dirección actual de un cliente al hacer un envío de datos.

Sin embargo, es posible que el usuario de imAgent pretenda trabajar bajo características diferentes de comunicación, para lo cual deberá crear su propio *Agent Messenger,* sin necesidad de efectuar otras modificaciones.

Cuando un agente invoca un send para enviar un mensaje a otro agente, debe especificar su propia identificación y la del destinatario, además del mensaje. El Agent Messenger utiliza para el envío de mensajes un esquema denominado Locate and Transfer, por el cual en primera instancia se debe localizar al destinatario y luego enviar el mensaje a su dirección. Esto permite el envío de mensajes de gran tamaño, ya que sería poco conveniente en la mayoría de los casos que la información enviada fuera parte de los mensajes utilizados para ubicar al agente receptor.

En el servidor donde se invoca el comando *send* se ordena inmediatamente la búsqueda del agente por medio de su *Agent Finder*. Una vez establecida la dirección del sitio donde se halla el agente, se reenvía el mensaje al servidor que se encuentra en aquella dirección. El *Agent Messenger* de este servidor advierte que el mensaje está destinado a un agente local y genera un evento con esa información, incluyendo el mensaje para el agente. El *Agent Container* escucha el evento y tras constatar que efectivamente contiene al agente destino, le entrega el mensaje informándole además qué agente se lo ha enviado.

Puede suceder que durante el proceso de búsqueda de un agente arriben al servidor otros mensajes para el mismo agente, ya sea que provengan del cliente o de otros agentes. En esos casos no se inician nuevas búsquedas ya que resultarían innecesarias, sino que se retienen los mensajes hasta conocer la ubicación del agente. Sólo en ese momento se reenvían todos los mensajes pendientes.

En el caso de un cliente que desee enviar un mensaje a un agente, debe utilizar los servicios de su componente de comunicación *ClientAgent Messenger*, que abstrae el comportamiento necesario para realizar esa función. De esta forma se permite al usuario de `im`Agent implementar esa tarea de la manera que considere más apropiada, simplemente reemplazando o extendiendo este componente. De todos modos `im`Agent ya cuenta con una implementación de este componente llamada *OriginalServer Messenger*, en condiciones de ser utilizada.

Cuando el cliente envía un mensaje a un agente debe indicar su identificación propia, la del agente destino y el mensaje. Luego el *OriginalServer Messenger* realiza el envío del mensaje al servidor origen del agente, quien inicia la búsqueda del mismo. Esto se debe a que cada servidor es responsable de los agentes que ha creado, y por lo tanto conoce cómo hacer llegar el mensaje al agente. Una vez determinado el sitio donde se encuentra el agente, se reenvía el mensaje. Al igual que en el caso anterior, el *Agent Messenger* de ese servidor genera un evento indicando que ha arribado un mensaje de un cliente para un agente local. El *Agent Container* escucha el evento y verifica que el agente aún se encuentra en el servidor, para luego entregarle el mensaje.

Una característica distinguida del *Agent Messenger* es su tolerancia a migraciones inoportunas. Considérese el caso de un servidor que ha comprobado que contiene al agente receptor de un mensaje. Instantes después puede producirse la migración del agente a otro sitio, justo antes de que el *Agent Container* pueda entregarle el mensaje. Estos casos son detectados por el *Agent Container*, que está informado de la partida del agente y genera un evento avisando que el mensaje no pudo ser entregado. Entonces el *Agent Messenger* reacciona ante este evento e inicia un nuevo ciclo de localización del agente y transferencia del mensaje.

También se puede presentar otro tipo más probable de migración inoportuna. Es el caso de un servidor que debe satisfacer un requerimiento de envío de un mensaje destinado a un agente que ha creado. Cuando el servidor obtiene la dirección del sitio en que se encuentra el agente por medio de su *Agent Finder*, reenvía el mensaje a esa dirección. El inconveniente surge porque el *Agent Finder* establece la dirección del agente en un momento preciso, pero mientras tanto el agente ha continuado su ejecución y nada impide que durante ese tiempo invoque un jump y cambie su ubicación. Esta posibilidad no representa un problema para el *Agent Messenger* ya que en cada servidor al que se reenvía un mensaje se continúa la búsqueda del agente hasta que su encuentro sea efectivo. De esta manera el conflicto se resuelve en forma transparente a los servidores involucrados en la operación. Situaciones como la descrita, donde un mensaje persigue a un agente intentando alcanzarlo son inevitables en este contexto y se conocen como "*race conditions*".

Finalmente se detalla el caso en que un agente envía un mensaje al cliente que lo ha creado. El agente invoca el send en el servidor local y especifica como parámetros su identificación y el mensaje. En el caso general este servidor no es el sitio de creación del agente que solicita el envío. Dado que el servidor origen del agente es quien conoce la información para acceder al cliente, es necesario reenviar el mensaje a este servidor. Cuando recibe el mensaje, el *Agent Messenger* del servidor que creó al agente solicita al *Client Manager*, a través del servidor, la última dirección conocida del cliente y efectúa el reenvío del mensaje. Luego el cliente recibe el mensaje junto con la identificación del agente que se lo ha enviado.

4.2.5 *Docking*

Existen casos en los que no es posible el envío inmediato del mensaje. Esto se debe en general a la naturaleza intermitente del cliente, que puede encontrarse temporalmente desconectado del sistema de agentes. En ese caso el *Client Manager* puede tener almacenada una dirección no actualizada o no tener ningún registro del cliente. El *Agent Messenger* está preparado para hacer frente a estas situaciones debido a que incluye un mecanismo llamado *Docking* Esta característica permite que la intermitencia del cliente no repercuta en sus agentes. En otras palabras, facilita la programación del agente para que no necesite lidiar con desconexiones del cliente. El agente simplemente envía un mensaje al cliente y continúa su ejecución sin importar si en ese momento el cliente se

encuentra conectado al sistema o no. Este soporte sustenta la entrega transparente de mensajes al cliente independientemente de su situación. El sistema de *Docking* implica el almacenamiento de los mensajes dirigidos al cliente durante su período de desconexión, incluyendo el tiempo que demore en actualizar su dirección. Los mensajes que se acumulen mientras el cliente esté desconectado del sistema son almacenados en un *buffer* de *docking*. La entrega de los mensajes está asociada al mecanismo de actualización de direcciones de clientes. Cada vez que el *Client Manager* completa una actualización de dirección genera un evento indicando que cierto cliente se ha conectado al sistema desde una determinada dirección. El sistema de *Docking* está capacitado para atender este tipo de eventos, y ante la reconexión de un cliente inicia el envío de todos los mensajes pendientes a su nueva dirección, respetando el orden de llegada.

4.2.6 Grupos de agentes

Existen ocasiones en las que es conveniente la creación de múltiples agentes para resolver un único problema. Algunos sistemas de agentes móviles, atendiendo a esta situación, permiten la formación de grupos de agentes. Conceptualmente un grupo de agentes (en adelante grupo) es un conjunto de agentes que trabajan en forma coordinada con un objetivo en común [16].

`i`mAgent permite utilizar grupos en las simulaciones. Cualquier agente del sistema puede crear un grupo para resolver un problema particular. En ese caso, el agente se convierte en el administrador del grupo, y los agentes creados se convierten en miembros del grupo. Los miembros son creados directamente dentro del grupo y pertenecerán al mismo durante todo su ciclo de vida. En cualquier momento el grupo podrá ser terminado y sus agentes serán eliminados.

Los miembros de un grupo conocen la identificación de su administrador, la cual está almacenada en cada agente en una variable especial denominada *adminAgentId*. Los miembros no se conocen entre sí. Ellos pueden estar realizando su tarea en diferentes sitios de la red y se comunicarán mediante un canal de eventos perteneciente al grupo. El administrador del grupo también podrá generar y escuchar eventos en ese canal.

Existe un coordinador del grupo cuya función es mantener la consistencia de la información del grupo y administrar los eventos. Éste se encuentra distribuido entre los agentes miembros y en el administrador como un componente separado, y su implementación debe ser provista por el usuario del simulador.

Cada grupo cuenta con una identificación dentro del sistema. Un mismo agente podrá crear y administrar más de un grupo a la vez. Un grupo comienza a existir a partir de que el primer miembro es creado. Para ello el servidor de cada sitio provee el comando *forkToGroup*, el cual tiene un comportamiento similar al fork salvo que además debe indicarse el identificador del grupo en cuestión. Internamente en el servidor, el encargado de crear al agente es un subcomponente del *Agent Container* denominado *Group Agent Creator*, el cual desarrolla su tarea en forma similar al *Child Creator* y genera los mismos eventos conteniendo, en este caso, el identificador del grupo.

El usuario del simulador puede implementar el comportamiento deseado de los grupos de agentes simplemente reemplazando los componentes nombrados. Finalizada la tarea asignada a un grupo, el administrador puede terminar al mismo generando un evento de finalización, el cual provoca que todos los miembros detengan su ejecución en el sitio en que se encuentran.

5 Aplicación del simulador

A los fines de validar los beneficios de la simulación como herramienta para evaluar componentes de soporte de sistemas de agentes móviles se llevó a cabo un estudio y se analizó la calidad de los resultados obtenidos.

El caso de estudio fue cuidadosamente seleccionado de manera tal que su alto grado de complejidad represente una gran exigencia para cualquier herramienta o técnica de experimentación. En estas circunstancias, un simple análisis demuestra que diversas técnicas se verían desbordadas y carecerían de características que permitan arribar de manera rápida y con bajos costos a resultados válidos. A partir de este caso de estudio también se intenta comprobar que esa limitación no se presenta cuando la técnica utilizada es la simulación.

El aspecto seleccionado a estudiar fue la función de ubicación de agentes, de vital importancia para la comunicación entre los agentes móviles en la mayoría de los casos, ya que posibilita su coordinación a nivel aplicación, indispensable para lograr una buena *performance* final.

5.1 Algoritmos a evaluar

Los algoritmos que soportan la ubicación de los agentes se basan en estrategias variadas. El comportamiento de tales estrategias depende en gran medida de las características de cada aplicación.

En nuestro caso, los algoritmos seleccionados fueron *Path Proxies*, y su variante *Compacted Chain*, ambos pertenecientes a la clase de algoritmos de *Logging*. Para mayor información referida a los mismos, ver [17].

5.1.1 *Path Proxies*

Por ser un algoritmo de *Logging*, almacena información que será luego utilizada para ubicar al agente. En este caso esa información se mantiene distribuida en los sitios que el agente visita. Cada vez que el agente llega a un sitio como resultado de una migración, el sitio desde donde partió almacena una referencia al sitio donde ha migrado el agente. Esta es la base de funcionamiento del algoritmo. En todo momento las referencias forman una cadena desde el sitio donde el agente comenzó su ejecución hasta el sitio donde el agente se encuentra ejecutando. Recorriendo esta cadena, en algún momento se localizará al agente. El algoritmo también establece la forma en que los sitios que pertenecen a la cadena son examinados. Una búsqueda comienza en el sitio de origen del agente y cada sitio es responsable de continuarla consultando al próximo sitio de la cadena.

En `i`mAgent, el componente en cada servidor encargado de la ubicación de agentes es el *Agent Finder*. La implementación del algoritmo *Path Proxies* se especializa en el componente *Path Proxies Agent Finder*. Los *Path Proxies Agent Finders* de los distintos servidores colaboran durante el proceso de ubicación dando a conocer a otros *Agent Finders* la información que mantienen.

5.1.2 *Compacted Chain*

En el algoritmo *Path Proxies* no existe un límite en la longitud de la cadena de sitios que va formando el agente, y ésta puede volverse muy extensa si el agente tiene un comportamiento altamente migratorio. En ese caso las búsquedas de agentes podrían emplear un tiempo considerable y elevar el volumen de mensajes debido a la longitud de la cadena. La variante *Compacted Chain* del algoritmo anterior incluye un mecanismo adicional de compactación de la ruta formada por las migraciones del agente.

El componente de `i`mAgent que implementa esta variante es *Compacted Chain Path Proxies Agent Finder*. Su funcionamiento responde al algoritmo *Path Proxies*, y además incorpora la implementación del mecanismo de compactación, el cual consiste en informar al sitio de origen la ubicación del agente luego de sucederse una cantidad predeterminada de migraciones de ese agente.

5.2 Pruebas realizadas

Para realizar una simulación es necesario diseñar e implementar sus características particulares. En el escenario de simulación deben recrearse los aspectos del sistema que fijan las condiciones necesarias para desarrollar la simulación. La implementación del escenario incluye la definición de la topología y la carga de la red, la implementación de las aplicaciones de usuario y las características de soporte que brindará el sistema de agentes móviles.

5.2.1 Topología

Independientemente de la aplicación de agentes móviles que sirva de base para la evaluación de los algoritmos de soporte, si la experimentación se realiza sobre una pequeña cantidad de sitios, los resultados producidos serán irrelevantes si se pretende aplicarlos a una situación real de mayor complejidad. Para permitir la distribución arbitraria de una considerable cantidad de sitios es necesario evitar el uso de una topología simplificada. Es por ello que este caso de estudio requiere la definición de una red suficientemente amplia y compleja.

La cantidad de nodos de la red se estableció en un valor máximo que no limita las posibilidades experimentales de la aplicación. Además de la ya mencionada necesidad de permitir la distribución de una suficiente cantidad de sitios, otro criterio considerado fue que la aplicación tenga la libertad de definir el grado de dispersión de sus sitios, sin verse restringida a disponerlos en ubicaciones contiguas. En vista de estos objetivos, la cantidad de nodos de la red se estableció en 256.

Para definir una red compleja, además de extensa, es necesario que sus nodos estén interconectados de manera irregular, haciendo que los caminos entre dos nodos no sean previsibles. En las topologías regulares, como anillo, estrella o jerárquica, los enlaces entre nodos tienen una estructura determinada y pueden ser predecibles en mayor o menor medida. En cambio, grandes redes irregulares como Internet brindan la arbitrariedad indispensable y son precisamente el ambiente elegido por los especialistas para experimentar con la tecnología de agentes móviles a gran escala. Por lo tanto se manifestó la necesidad de utilizar una topología que imite la distribución de nodos de Internet.

De acuerdo con lo anterior, las redes generadas están compuestas por vínculos de comunicación con capacidad de 0,5 Mbps y demora de propagación de 10 milisegundos No existen aplicaciones secundarias que ocupen una parte del ancho de banda, o sea que la aplicación de agentes móviles es la única usuaria de la red. Un estudio más exhaustivo podría considerar enlaces de características variables y la existencia de aplicaciones secundarias.

5.2.2 Aplicación

La aplicación de agentes que se simuló está representada por un conjunto de comercios que ofrecen productos para la venta. Cada comercio está representado por un sitio que provee información sobre diferentes productos, como sus características y precios. Existe además un cliente que desea adquirir dos productos ‘A’ y ‘B’ y establece el precio límite que está dispuesto a pagar por cada uno de ellos. El cliente no cuenta con capacidad para ejecutar agentes pero posee una aplicación que puede enviar agentes a recorrer los sitios comerciales en busca de información sobre los productos que satisfacen sus necesidades. Este método es útil porque el cliente puede conectarse al sistema para ejecutar su aplicación y luego desconectarse sin necesidad de esperar el resultado. Una vez encontradas las mejores ofertas, uno de los agentes se encarga de notificarlas al cliente y de informarle en qué comercios las puede encontrar. El cliente podrá recuperar esa información cuando vuelva a conectarse.

La aplicación de agentes plantea una estrategia para resolver el problema. En general existen diversas estrategias posibles para resolver este tipo de situaciones, en las que se requiere la visita a

un conjunto de sitios mediante agentes móviles en busca de información. Dos de ellas son normalmente consideradas extremas. Por un lado, todos los sitios pueden ser recorridos por un único agente que debe efectuar las migraciones necesarias. Aunque el costo de disponer de un solo agente es mínimo, el tiempo total de resolución del problema puede ser considerable dado que no se realizan tareas en paralelo. Por otro lado se puede contar con un agente en cada sitio; la información se obtiene en forma simultánea en todos los sitios sin necesidad de realizar migraciones. Sin embargo en este caso puede haber una alta tasa de transferencia de información temporal innecesaria y se estarían desaprovechando las ventajas del paradigma.

En el ambiente de investigación de agentes móviles los expertos coinciden en que ninguna de estas dos soluciones extremas ofrece resultados aceptables en términos de rendimiento para aplicaciones de agentes móviles en general. Por consiguiente la solución implementada en esta aplicación es una estrategia híbrida que combina ambas técnicas extremas. De todos modos no es objetivo de este caso de estudio la evaluación de la competitividad de la estrategia elegida, sino la valoración de la efectividad de soporte del sistema de agentes móviles ante la ejecución de una aplicación que explote toda su capacidad.

La construcción de la aplicación de agentes comprende la implementación del cliente y de los agentes involucrados en la tarea. Más específicamente, el cliente encarga oportunamente la resolución del problema a un agente llamado *Purchaser Agent*. Con ese propósito, la aplicación cliente puede crear al agente directamente en cualquier sitio. Su responsabilidad es determinar la mejor oferta de cada producto y opcionalmente efectuar la compra. Para ello utiliza como estrategia su facultad de crear *childAgents*, y así produce un conjunto de *Offer Selector Agents* en diferentes sitios. Estos agentes poseen un objetivo global factible de ser mejorado. Cada vez que se cumple el objetivo la aplicación decide si es necesario continuar trabajando para mejorarlo. Los Offer Selector Agents constituyen un segundo nivel de procesamiento en la aplicación. Tienen asignada una zona de trabajo y su función es resolver el problema en esa zona. Cada *Offer Selector Agent* crea a su vez su propio grupo de agentes para recorrer la zona señalada. Estos agentes de último nivel se denominan *Shopping Agents*. Un *Offer Selector Agent* divide su zona en una o más sub-zonas, cada una de las cuales es responsabilidad de un *Shopping Agent*.

5.2.3 Casos analizados

El caso de estudio se divide en una serie de experimentos que se desarrollan sobre el escenario mencionado. La topología de la red y la semántica de la aplicación que conforman el escenario se mantienen constantes durante todo el estudio. Esta decisión impide que cuestiones tales como la demora de propagación de la red y similares influyan en las mediciones que se obtienen. Asimismo se mantiene el enfoque del caso de estudio centrado en la evaluación del soporte del sistema de agentes móviles. De esta manera se permite que las diferencias encontradas puedan justificarse en base a los parámetros que configuran cada prueba.

El parámetro principal está relacionado con el soporte del sistema de agentes móviles que se desea evaluar. Se considera como variación de soporte a la elección de uno de los dos algoritmos de ubicación antes presentados, lo que representa el punto de partida hacia el objetivo de este caso de estudio, que pretende realizar un análisis comparativo entre ambos algoritmos bajo condiciones variables del sistema de agentes móviles. Esas variaciones ambientales se logran con la introducción de otros parámetros que permiten evaluar dicho soporte bajo diferentes condiciones de carga del sistema de agentes móviles. Ellas son la tasa de migración y la tasa de comunicación de los agentes.

La tasa de migración se refiere al dinamismo con que actúan los agentes para resolver el problema. La estrategia puede consistir tanto en pocos agentes que recorren varios sitios cada uno, como en varios agentes que migran pocas veces. En la definición de esta variable están involucrados dos factores: la cantidad de zonas, cada una adjudicada a un *Offer Selector Agent*, y la cantidad de sub-

zonas en que se divide cada zona. Como ya se explicó, cada sub-zona está asignada a un *Shopping Agent* que debe recorrer todos los sitios que la componen.

Debido a que el conjunto total de sitios donde deben moverse los agentes en busca de ofertas está determinado al momento de la simulación, la cantidad de sitios que debe visitar cada agente podrá reducirse o aumentarse según cómo se ajusten las cantidades de zonas y sub-zonas, y por consiguiente se regulará la tasa de migración.

Si ambas cantidades son suficientemente pequeñas, cada agente deberá visitar una gran cantidad de sitios y la tasa de migración será alta. Por otro lado, elevando uno de estos dos valores se disminuye la cantidad de sitios a recorrer por cada agente, lo que lleva a una reducción en la tasa de migración.

En el escenario referido existen 127 sitios de venta de productos, en cada uno de los cuales hay un servidor de agentes móviles. En ese contexto se define una serie de disposiciones estratégicas diferentes que pueden tomar los agentes para lograr su objetivo. Si bien podrían producirse innumerables distribuciones distintas, este caso de estudio especifica tres de ellas cuya importancia experimental radica justamente en la tasa de migración que resulta de la utilización de cada una. Es conveniente poder ajustar la tasa de migración de los agentes, ya que esto permite la evaluación del soporte del sistema tanto con aplicaciones que generan gran movilidad de agentes como con otras de escasas migraciones. De esta manera la eficacia de los algoritmos de ubicación de agentes podrá juzgarse teniendo en cuenta los diferentes comportamientos de los agentes. En la tabla 1 se detallan las distribuciones de las cuales resultan las distintas tasas de migración.

En cuanto a la variable restante, la tasa de comunicación de los agentes expresa la frecuencia con que éstos interactúan entre ellos, independientemente de la cantidad de veces que necesiten migrar. Con referencia a este aspecto, parte de la definición de la estrategia consiste en determinar la cantidad de ofertas relevantes que deben encontrar los *Shopping Agents* antes de informar los resultados obtenidos. Cuanto menor es esta cantidad mayor será la frecuencia con que deberán comunicarse los agentes. Otro parámetro influyente es el precio límite que el cliente está dispuesto a pagar por cada uno de los productos que requiere. Estos valores son fijados como objetivos iniciales. Recuérdese que los *Shopping Agents* recolectan y envían todas las ofertas cuyo precio sea menor o igual al límite establecido por el cliente. Teniendo en cuenta esto, si el límite es cercano al precio máximo de todos los productos que se ofrecen, se espera que los agentes encuentren muchas ofertas válidas durante su tarea, por lo que las comunicaciones también serán numerosas. Por el contrario, si el precio límite está próximo al precio mínimo de todos los productos, se estima que los agentes obtendrán menos ofertas para notificar y la tasa de comunicación será menor.

En esta simulación es importante poder especificar la frecuencia con que los agentes interactúan porque las comunicaciones entre ellos requieren la ubicación de los agentes de destino, y por lo tanto la tasa de comunicación está directamente relacionada con los algoritmos de ubicación de agentes que dan soporte al sistema de agentes móviles. La posibilidad de variación de la tasa de comunicación permite evaluar la capacidad de soporte del sistema experimentando con diferentes volúmenes de comunicaciones. Este caso de estudio identifica tres variaciones diferentes de la tasa de comunicación que se presentan a continuación. Los precios de los productos son valores enteros y su rango se mantiene en este caso de estudio entre 1 y 100 unidades.

	Alta	**Media**	**Baja**
Tasa migración	2 zonas de 4 sub-zonas c/u 8 sub-zonas de	5 zonas de 5 sub-zonas c/u 25 sub-zonas de	8 zonas de 6 sub-zonas c/u 48 sub-zonas de

	16 sitios c/u	5 sitios c/u	3 sitios c/u
Tasa comunicación	Not.: por oferta válida Precio Máx: 90	Not.: c/4 ofertas válidas Precio Máx: 50	Not.: c/8 ofertas válidas Precio Máx: 10

Tabla 1 - Parámetros que caracterizan a las pruebas realizadas

Un experimento se dedica a la evaluación de una medida específica y consiste de un conjunto de pruebas. Las medidas son cuantitativas, por lo que brindan una noción exacta del resultado de una prueba en particular.

Para facilitar la producción de resultados en un experimento se pueden utilizar eventos especiales llamados eventos de usuario. Esta clase de eventos fue ideada con la finalidad de permitir al usuario del simulador la obtención de información detallada que satisfaga sus necesidades en una simulación particular. Un evento de usuario puede ser generado en cualquier parte del código del simulador, ya sea de biblioteca o definida por el usuario, y actúa de la misma manera que otros eventos. Sólo se debe indicar en sus parámetros qué datos reunirá el evento. Cuando sea necesario, un *External Event Manager* definido por el usuario se encargará de recolectar los datos del evento y procesarlos de la manera pretendida.

Todas las pruebas de un experimento obtienen información sobre la misma medida, pero están supeditadas a diferentes configuraciones. En la configuración de una prueba están involucradas las variables de la simulación que determinan la carga del sistema de agentes móviles, como las tasas de migración y comunicación de los agentes, y aquella que especifica el soporte del sistema. Esta última variable se refiere a que todos los servidores de agentes utilizarán durante una prueba uno de los algoritmos de ubicación de agentes que se desean comparar, ya sea el algoritmo *Path Proxies* o su variante *Compacted Chain*. Es importante aclarar que la variación de otros aspectos de soporte del sistema sería contraproducente porque obligaría a incluir como objetivo la evaluación de esos atributos, y esto no forma parte del propósito concreto de este caso de estudio.

En cuanto al ajuste de las variables que determinan la carga del sistema, existe una cuestión a considerar. Si se ha determinado que una prueba se ejecutará bajo una situación de carga extrema de una de las variables, la variable restante debe mantenerse en su nivel medio. Esto se aclarará con un ejemplo. Supóngase que se decide que una de las pruebas considere una tasa de migración alta del sistema. Entonces, como parte de la configuración de esa misma prueba, debe especificarse una tasa de comunicación media de los agentes. Esto se debe a que, para ser precisos en la evaluación de una prueba, se permita obtener información relevante sobre el uso de los algoritmos de ubicación dentro de una situación específica. El mantenimiento a nivel medio de la segunda variable hace estimable que ésta no interfiera en el contexto creado por la primera. Además la conjunción de dos variables extremas hace que la prueba sea excesivamente singular y dificulta su evaluación objetiva.

Cuando en una prueba se encuentra configurada la carga que recibirá el sistema, su ejecución en el simulador se lleva a cabo dos veces, una con cada uno de los algoritmos de ubicación instalados en los servidores de agentes. De esta manera el análisis de una prueba específica aislada de las demás permite comparar el comportamiento de ambos algoritmos de ubicación manteniendo fijo el entorno completo del sistema. Ese entorno incluye, además del escenario, una tasa específica de comunicación y una tasa concreta de migración. Esto es muy útil para la evaluación ya que si se encuentran diferencias entre ambas ejecuciones, éstas estarán directamente relacionadas con la variación de soporte de la prueba, lo que permitirá extraer conclusiones cuantitativas y precisas sobre el uso de uno u otro algoritmo.

Los experimentos y pruebas realizados se presentan a continuación. En todas las pruebas se expone en primer lugar el resultado esperado, que permite estimar la validez del resultado obtenido en la simulación por simple comparación. Los resultados esperados se obtuvieron

mediante un criterio lógico en base al entorno configurado en cada prueba y constatado por investigaciones análogas.

5.2.3.1 Experimento A: Tiempo de ubicación de un agente (TUA)

Este experimento lleva a cabo la medición del tiempo que tarda en promedio el sistema de agentes móviles en efectuar la búsqueda de un agente de la aplicación. Ese intervalo comprende el tiempo desde que un pedido de ubicación es formulado en el sistema hasta que la respuesta del mismo llega al solicitante del servicio. La importancia de la medición que realiza este experimento radica en que la duración de la búsqueda de un agente de la aplicación está íntimamente relacionada con el algoritmo de ubicación de agentes que utilicen los servidores.

Con el propósito de obtener el tiempo exacto por el que se prolonga cada búsqueda se utilizaron eventos de usuario. En la aplicación de agentes mostrada, todas las búsquedas de agentes se producen como consecuencia de un intento de comunicación entre agentes. Por lo tanto las búsquedas se originan y finalizan en el componente *SCAgentMessenger*, que es quien solicita el servicio de ubicación al servidor de agentes. Esto ha permitido la inserción de eventos de usuario en dicho componente para poder obtener, en última instancia, la duración de cada búsqueda. En este caso el evento de usuario informa, además del tiempo de simulación al momento del comienzo o fin de la búsqueda, los identificadores de los agentes involucrados, el mensaje enviado y otros datos de sincronización. Algunos de estos datos se emplean únicamente como información complementaria en el análisis de resultados.

En cuanto a las diferentes búsquedas que se producen en la aplicación, fue necesario realizar una selección de las más representativas. En el contexto presentado existen relaciones entre distintos tipos de agentes que requieren que ellos puedan comunicarse, y de esa manera se producen los pedidos de ubicación.

Este experimento contiene tres pruebas. En cada una se obtiene el TUA de cada algoritmo y el porcentaje de disminución del TUA del algoritmo *Compacted Chain* con respecto al algoritmo *Path Proxies*.

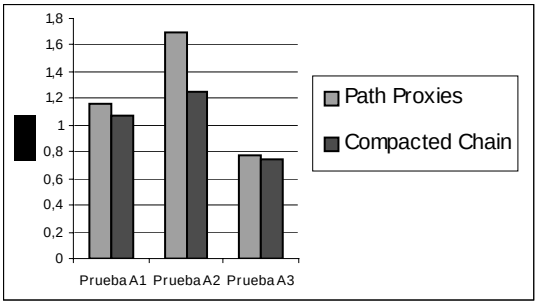

Figura 6 – Gráfico de TUA de ambos algoritmos.

La prueba A1 representa el caso de un sistema de agentes móviles promedio. Su utilidad es principalmente comparativa. En esta prueba las tasas de migraciones y de comunicaciones se mantienen ambas a nivel intermedio para crear el ambiente promedio requerido. La prueba A2 es un caso de alta carga migratoria del sistema de agentes móviles. Para lograr esto la prueba se configura con una tasa de migraciones alta y, por los motivos ya expuestos, se mantiene la tasa de comunicaciones a nivel intermedio. El aumento en la cantidad de migraciones afecta directamente al funcionamiento del algoritmo *Path Proxies* haciendo que generalmente deba recorrer cadenas más largas en la búsqueda de los agentes; esta situación aumentaría el TUA. En cambio, el algoritmo *Compacted Chain* debería lograr aprovechar significativamente las ventajas de la compactación bajo estas condiciones. La prueba A3 representa el caso de un sistema de agentes móviles con un alto volumen de comunicaciones. La prueba se configura con una tasa de

comunicaciones alta y se mantiene la tasa de migraciones a nivel intermedio. En este escenario pierde importancia la compactación adicional del algoritmo *Compacted Chain* porque las cadenas se acortarían más frecuentemente, ante cada comunicación. De esta forma, el algoritmo trabajaría sobre cadenas más cortas.

Los resultados de las pruebas del Experimento A (Ver Fig.6 y Tabla 2) demuestran que el algoritmo *Compacted Chain* aprovecha todo el potencial del mecanismo de compactación adicional cuando las búsquedas se realizan sobre cadenas más largas. Estas condiciones se presentan cuando la tasa de migraciones de los agentes del sistema es alta. Por el contrario, la utilidad del algoritmo *Compacted Chain* parece ser mínima en un escenario con gran volumen de comunicaciones y por ende, de búsquedas.

P A1 (Migr:Medio, Comunic: Medio)	TUA Path Proxies: 1,16 TUA Comp. Chain: 1,07	Porcentaje dism: 7,31
P A2 (Migr:Alto, Comunic: Medio)	TUA Path Proxies: 1,70 TUA Comp. Chain: 1,25	Porcentaje dism: 26,48
P A3 (Migr:Alto, Comunic: Alto)	TUA Path Proxies: 0,78 TUA Comp. Chain: 0,75	Porcentaje dism: 2,28

Tablas 2 - Resultados obtenidos para el experimento A.

5.2.3.2 Experimento B: Cantidad de información de mantenimiento transmitida por agente (IMA)

Este experimento efectúa la medición de la cantidad de información de mantenimiento que utiliza el sistema de agentes móviles por cada agente de la aplicación. En general, los algoritmos de ubicación de agentes pueden enviar mensajes de ubicación aún cuando no existan requerimientos explícitos de búsquedas por parte de algún componente del sistema. Los algoritmos de la categoría de *logging* normalmente utilizan esa información para realizar actualizaciones periódicas en los datos mantenidos por el algoritmo para localizar a los agentes. Ese tipo de información es el que se considera de mantenimiento.

Si bien dichos mensajes generan un tráfico adicional en la red, se espera que su inclusión produzca algún beneficio, como la disminución del tiempo consumido por búsquedas posteriores. Como ya se ha visto, el algoritmo *Compacted Chain* posee un mecanismo de envío de información periódica justamente con ese propósito. El objetivo de este experimento es evaluar el comportamiento del algoritmo *Compacted Chain* en cuanto a la cantidad de información de mantenimiento que genera por cada agente de la aplicación. El tamaño de cada mensaje de mantenimiento transmitido por este algoritmo es de aproximadamente 65 bytes. Las proporciones mostradas con respecto al algoritmo *Path Proxies* se mantendrían aún cuando se modificara el tamaño del mensaje, por lo que esta medida no influye en los resultados.

Por otro lado, el algoritmo *Path Proxies* no envía mensajes de ubicación a menos que exista un requerimiento explícito de algún componente. Es por ese motivo que las pruebas de este experimento configuradas con el algoritmo *Path Proxies* arrojan resultados nulos, ya que no existe información de mantenimiento generada por el algoritmo.

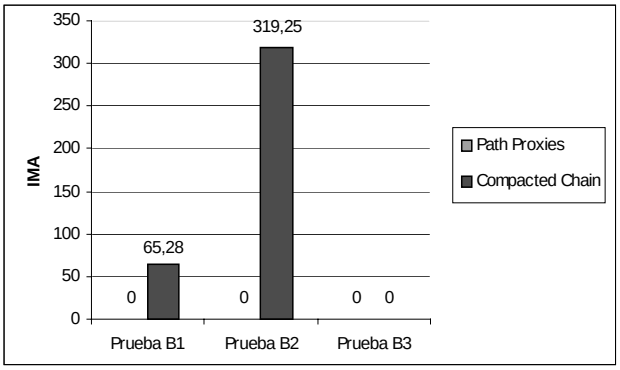

Figura 7 – Gráfico de IMA de ambos algoritmos.

La prueba B1 representa el caso de un sistema de agentes móviles promedio. Su utilidad es principalmente comparativa. En esta prueba las tasas de migraciones y de comunicaciones se mantienen ambas a nivel intermedio para crear el ambiente promedio requerido. Se espera que con una tasa de migración media los agentes migren la suficiente cantidad de veces como para activar el mecanismo de compactación del algoritmo *Compacted Chain*. De esta manera, el IMA resultante para ese algoritmo resulta ser un valor no nulo. El IMA obtenido para el algoritmo *Path Proxies* fue obviamente 0 bytes y para el algoritmo *Compacted Chain* fue 65,28 bytes Esto significa que durante la simulación con este último algoritmo se transmitieron 65,28 bytes por cada agente en mensajes de mantenimiento. Dado que en esta configuración hay 25 sub-zonas y por ende 25 agentes móviles, el total de información de mantenimiento transmitida fue de 1632 bytes.

La prueba B2 representa el caso de alta carga migratoria del sistema de agentes móviles. Se configura con una tasa de migraciones alta y se mantiene la tasa de comunicaciones a nivel intermedio. En este caso el mecanismo de compactación del algoritmo *Compacted Chain* se activa una mayor cantidad de veces debido al incremento en la tasa de migraciones. El IMA obtenido para el algoritmo *Path Proxies* fue obviamente 0 bytes y para el algoritmo *Compacted Chain* fue 319,25 bytes. Considerando que la cantidad de agentes móviles disminuyó en un 68% (8 agentes), el IMA resultante aumentó un 389,05%.

La prueba B3 representa el caso inverso a la prueba anterior. El sistema de agentes móviles debe manejar una escasa cantidad de migraciones. La prueba se configura con una tasa de migraciones baja y se mantiene la tasa de comunicaciones a nivel intermedio. Con una tasa de migraciones baja se espera que no se alcance la cantidad necesaria de migraciones para activar el mecanismo de compactación, por lo que no deberían producirse mensajes de compactación. El IMA obtenido para ambos algoritmos fue 0 bytes. Esto significa que el algoritmo *Compacted Chain* bajo estas condiciones se comporta en forma similar al algoritmo *Path Proxies*.

Como muestran la figura 7 y la tabla 3, la tasa de migraciones influye en forma determinante en la cantidad de información de mantenimiento transmitida por agente al utilizar el algoritmo *Compacted Chain*. En tanto que, como era predecible, al utilizar el algoritmo *Path Proxies* el IMA se mantiene nulo.

P B1 (Migr:Medio, Comunic: Medio)	IMA Path Proxies: 0 IMA Comp. Chain: 65,28
P B2 (Migr:Alto, Comunic: Medio)	IMA Path Proxies: 0 IMA Comp. Chain: 319,25
P B3 (Migr:Bajo, Comunic: Medio)	IMA Path Proxies: 0 IMA Comp. Chain: 0

Tabla 3- Resultados obtenidos para el experimento B.

5.2.3.3 Experimento C: Cantidad de información de ubicación transmitida por búsqueda (IUB)

Este experimento (ver Fig 8 y Tabla 4) realiza la medición de la cantidad de información de ubicación que utiliza el sistema de agentes móviles para consumar la búsqueda de un agente de la aplicación. Como *información de ubicación* se considera la originada por requerimientos explícitos de búsquedas de agentes en el sistema. Esto no incluye información de mantenimiento

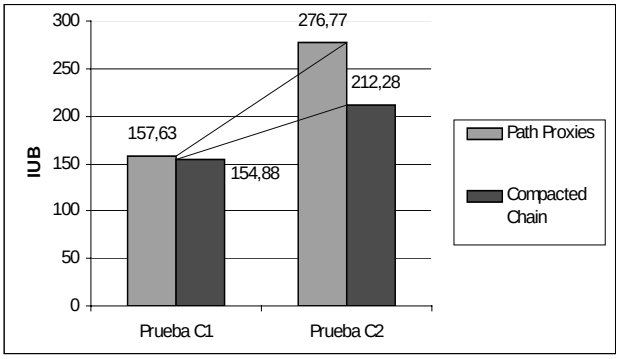

Figura 8 – Gráfico de IUB de ambos algoritmos.

La prueba C1 representa el caso de un sistema de agentes móviles promedio. Su utilidad es principalmente comparativa. En esta prueba las tasas de migraciones y de comunicaciones se mantienen ambas a nivel intermedio para crear el ambiente promedio requerido. El IUB obtenido para el algoritmo *Path Proxies* fue 157,63 bytes y para el algoritmo *Compacted Chain* fue 154,88 bytes, lo que representa una disminución del 1,74%. Este porcentaje evidencia que, debido a la compactación, el algoritmo *Compacted Chain* debió transmitir una menor cantidad de información para ubicar a los agentes.

La prueba C2 representa el caso de alta carga migratoria del sistema de agentes móviles. Para lograr esto la prueba se configura con una tasa de migraciones alta y se mantiene la tasa de comunicaciones a nivel intermedio. El IUB obtenido para el algoritmo *Path Proxies* fue 276,77 bytes y para el algoritmo *Compacted Chain* fue 212,28 bytes, representando una disminución del 23,3%. En ambos algoritmos el IUB aumentó en forma significativa afectando en mayor medida al algoritmo *Path Proxies* por carecer de un mecanismo de compactación. Por esta razón el porcentaje de disminución del IUB aumentó considerablemente.

P C1 (Migr:Medio, Comunic: Medio)	IUB Path Proxies: 157,63 IUB Comp. Chain: 154,88	Porcentaje dism: 1,74
P C2 (Migr:Alto, Comunic: Medio)	IUB Path Proxies: 276,77 IUB Comp. Chain:212,28	Porcentaje dism: 23,3

Tabla 4 - Resultados obtenidos para el experimento C.

5.2.3.4 Experimento D: Cantidad total de información transmitida por búsqueda (ITB)

Este experimento lleva a cabo la medición de la cantidad de información total utilizada por el sistema de agentes móviles para efectuar la búsqueda de un agente de la aplicación. Por información total debe entenderse tanto la información exclusivamente de ubicación como la de mantenimiento.

La prueba D1 representa el caso de un sistema de agentes móviles promedio. Su utilidad es principalmente comparativa. En esta prueba las tasas de migraciones y de comunicaciones se mantienen ambas a nivel intermedio para crear el ambiente promedio requerido. El ITB obtenido para el algoritmo *Path Proxies* fue 157,63 bytes y para el algoritmo *Compacted Chain* fue 168,59 bytes, lo que representa un aumento del 6,95%. En este escenario la información de mantenimiento transmitida por el mecanismo de compactación del algoritmo *Compacted Chain* contribuyó a aumentar el ITB a tal nivel de superar al del algoritmo *Path Proxies*.

La prueba D2 representa el caso de alta carga migratoria del sistema de agentes móviles.

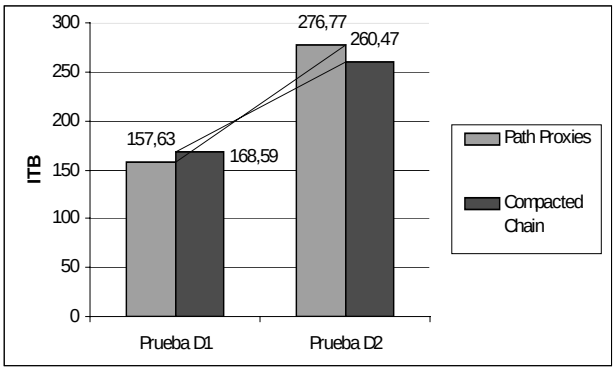

Figura 9 – Gráfico de ITB de ambos algoritmos.

Para lograr esto la prueba se configura con una tasa de migraciones alta y se mantiene la tasa de comunicaciones a nivel intermedio. El ITB obtenido para el algoritmo Path Proxies fue 276,77 bytes y para el algoritmo *Compacted Chain* fue 260,47 bytes, lo que representa una disminución del 5,89%. En este escenario la información de mantenimiento transmitida utilizando el algoritmo *Compacted Chain* contribuyó a aumentar el ITB, aunque esta vez (Ver Fig. 9 y Tabla 5) la compactación fue más efectiva y produjo un mayor ahorro en la cantidad de mensajes de ubicación por trabajar con cadenas más cortas. De esta manera, y a diferencia de la prueba anterior, el algoritmo *Compacted Chain* logró un menor ITB que el algoritmo *Path Proxies*.

P D1 (Migr:Medio, Comunic: Medio)	ITB Path Proxies: 157,63 ITB Comp. Chain: 168,59	Porcentaje dism: -6,95
P D2 (Migr:Alto, Comunic: Medio)	ITB Path Proxies: 276,77 ITB Comp. Chain:260,47	Porcentaje dism: 5,89

Tabla 5 - Resultados obtenidos para el experimento D.

5.2.4 Conclusiones obtenidas de los experimentos realizados

De los resultados obtenidos se desprende que el algoritmo *Compacted Chain* con su mecanismo de compactación logra menores tiempos de búsqueda de agentes en prácticamente todas las situaciones, y particularmente se destaca en los casos que presentan un alto grado de movilidad y búsquedas poco frecuentes. Este algoritmo podría no ser conveniente con respecto a *Path Proxies* en escenarios en los que la tasa de comunicaciones de los agentes es elevada en relación con su movilidad.

De un análisis desde el punto de vista de la carga de la red que genera cada algoritmo se concluye que *Compacted Chain* necesita transmitir una considerable cantidad de información adicional en escenarios con alta tasa de migración para mantener limitada la longitud de las cadenas, en tanto que *Path Proxies* no incurre en costos de este tipo.

El proceso de ubicación de agentes utilizando el algoritmo *Compacted Chain* suele demandar menor cantidad de mensajes que *Path Proxies* por trabajar sobre cadenas previamente compactadas. Este ahorro generalmente compensa y hasta supera el costo extra de transmisión de información de mantenimiento en mensajes de compactación.

La falta de un límite en la longitud de las cadenas al utilizar *Path Proxies* compromete su escalabilidad, en tanto que la misma situación se presenta con el algoritmo restante al considerar la carga de la red por información de mantenimiento.

Finalmente se concluye que el algoritmo *Compacted Chain* correctamente parametrizado brinda suficientes ventajas como para ser adoptado en escenarios normales; entretanto en escenarios atípicos podría ser beneficiosa la utilización de *Path Proxies*.

6 Conclusiones y desarrollos a realizar

Cumpliendo con el objetivo propuesto, se desarrolló una infraestructura que permite el rápido desarrollo de prototipos de sistemas de agentes móviles en un ambiente de simulación. Los aspectos posibles de desarrollar se refieren tanto a estrategias a nivel aplicación como a estrategias adoptadas a nivel soporte (en particular, el caso evaluado consiste en mecanismos de localización).

Cabe destacar que el sistema desarrollado cumple con dos características consideradas de importancia y que fueron impuestas como objetivos adicionales. En primer término, al estar desarrollado sobre una plataforma versátil y rica en posibilidades de generación de topologías de red e interacción entre diferentes protocolos, como Ns., permite la exploración de escenarios que se acercan a las situaciones reales En segundo término, el desarrollo de una interfaz similar a la ofrecida por *sockets* y sustentada por el servidor de agentes y por el módulo ExEnv, logró reducir la brecha existente entre el desarrollo de la aplicación real y el orientado a su simulación. El resultado de conjugar estas características fue, como se comprobó en la aplicación desarrollada, facilidad en plasmar aplicaciones en el código del simulador y una gran independencia en cuanto a los niveles de soporte a agentes y al de las aplicaciones, lo cual permite experimentar fácilmente con combinaciones de estrategias.

En cuanto a los resultados obtenidos, se mantuvieron dentro de lo inferido, lo cual demuestra el correcto comportamiento del código desarrollado. De acuerdo con la performance lograda en las diferentes pruebas realizadas, es posible afirmar que podrían extenderse las pruebas, tanto en lo que se refiere a topologías más extensas como a mayor cantidad de agentes, sin producirse un excesivo consumo de recursos.

`i`mAgent es un simulador con amplia capacidad de ser extendido, y así, la realización de cada experimento contribuirá a futuras simulaciones con el aporte de nuevos componentes. Entre los aspectos que se consideran de importancia para la mejora del soporte desarrollado, se encuentran los que se mencionan a continuación.

Un mayor grado de optimización en la performance de `i`mAgent permitirá la realización de experimentos consumiendo aún menos recursos.

Contar con herramientas visuales de construcción de escenarios y generación automática de código acelerará el proceso de implementación y hará de `i`mAgent una herramienta más atractiva.

La posibilidad de visualización animada de la simulación permitirá al usuario comprender más fácilmente el comportamiento de escenarios dinámicos complejos y eventualmente depurar aplicaciones.

Un aspecto de suma importancia que a criterio de los autores será mejorado con el uso de la plataforma, consiste en ofrecer al programador una interfaz cada vez más cercana a la que puede encontrarse en sistemas reales, para reducir de esta manera el esfuerzo de desarrollar código para simulación y código real separadamente.

Referencias

[1] Kotz, David, Gray, R. Rus, D., "Future Directions for Mobile-Agent Research", IEEE Distributed Systems Online, 3(8), August, 2002.

[2]Fall, K., Varadhan, K., "The NS Manual", The VINT Project, a collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC.

[3] Peng Sun, Sam Y. Sung, "A General Simulation Support for IP Mobility", Proceedings of the 2002 Winter Simulation Conference, E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

[4] C. Perkins, "IP Mobility Support for IPv4" (IETF RFC 3220), August 2002.

[5] SSFNet (http://ssfnet.org/)

[6] S. Bandyopadhyay, "Evaluating the Performance of Mobile Agent-Based Message Communication among Mobile Hosts in Large Ad Hoc Wireless Network", Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems Seattle, Washington, United States Pages: 69 - 73 .1999

[7] M. Erdem Türsem, M. Hadi Güne, Mustafa Yýldýz, Selahattin Kuru, "Performance Analysis of Mobile Agents Using Simulation", Department of Computer Engineering, Boðaziçi University, Istanbul, Turkey

[8] Wetherall, David, "OTcl - MIT Object Tcl - The FAQ & Manual (Version 0.96, September 95)", MIT Lab for Computer Science.

[9] Ousterhout., John K "Tcl and the Tk Toolkit",. Computer Science Division - Department of Electrical Engineering and Computer Sciences - University of California.

[10] Dejan Milojicic, et al., "MASIF: The OMG Mobile Agent System Interoperability Facility", General Magic Inc. - GMD Fokus – IBM - Open Group.

[11] Wooldridge, M., Jennings, N., "Agent Theories, Architectures and Languages: a Survey", Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages, 1994

[12] Nwana, Hyacinth S, "Software Agents: An Overview", Knowledge Engineering Review, vol. 11, n. 3, pp. 205-244. Cambridge University Press, 1996.

[13] Franklin, Stan; Graesser, Art: "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

[14] Gray, Robert, Cybenko, George, Kotz, David, Rus ,Daniela, "Agent Tcl",. Department of Computer Science - Dartmouth College.

[15]Moreau , L., "Distributed directory service and message routing for mobile agents", Department of Electronics and Computer Science, University of Southhampton.

[16] Baumann, N, Radouniklis, J, "Agent Groups in Mobile Agent Systems", IPVR (Institute for Parallel and Distributed Computer Systems).

[17] Wojciechowski,. Pawel, "Algorithms for Location-Independent communication between Mobile Agents", Swiss Federal Institute of Technology (EPFL) - Operating Systems Laboratory.