Capítulo 2.
Agentes Móviles

Como se mencionó en el capitulo anterior, la nueva tecnología de agentes y en particular los agentes móviles han atraído la atención de investigadores de muy diversas área como Interacción Hombre-Computadora, Inteligencia Artificial, Bases de Datos y Computo Distribuido.

Los términos agentes de red [Sánchez 1997], agentes transportables [Gray et al. 1996] o agentes itinerantes [Chess et al. 1995], son usados pare referirse al mismo concepto - *agente móvil*-. La característica distintiva de este agente es su capacidad de *migrar*.

En este capítulo se presentan los conceptos centrales en el área, se plantea una estructura general de agentes, los lenguajes de programación existentes para el desarrollo de esta nueva tecnología, las ventajas y desventajas de este tipo de agente y las posibles aplicaciones de los agentes móviles.

Los términos *agente y agente móvil* se usarán indistintamente en lo sucesivo, a menos que se indique lo contrario.

2.1 Antecedentes

La primera ventaja de RP es el desempeño. Cuando una computadora cliente tiene trabajo que tracen en un servidor remoto, puede enviar el trabajo y supervisarlo localmente usando un agente, en vez de estar enviando continuamente instrucciones sobre la red.

La segunda ventaja de RP es la personalización. Un agente móvil permite que el cliente personalice la funcionalidad del servidor. Pueden enviarse nuevos procedimientos con poco esfuerzo. RP puede ayudar para automatizar procesos de instalación. Se puede codificar en un programa un proceso de instalación y transferirlo a un conjunto de nodos de la red. En cada nodo el programa puede analizar las características de la plataforma local de hardware / software y ejecutar la configuración correcta. RP ayuda a incrementar la flexibilidad del servidor, manteniendo limitado el tamaño y la complejidad del mismo. Cada cliente es responsable de la correcta especificación del servicio que necesita y debe ser descrito en el código enviado al servidor remoto.

Por su naturaleza, RP permite especificar que cálculos complejos se muevan a través de la red y si los servicios tienen que ser ejecutados en un servidor que es alcanzable solamente a través de una conexión lenta (por ejemplo, vía MODEM), se puede enviar el programa pare su ejecución y no se necesitaría mantener una conexión permanente con el servidor, excepto para la transmisión del resultado final.

Todos estos beneficios de la programación remota, hacen de los agentes móviles una tecnología muy atractiva para el desarrollo de nuevos sistemas. Pero que es un agente móvil.

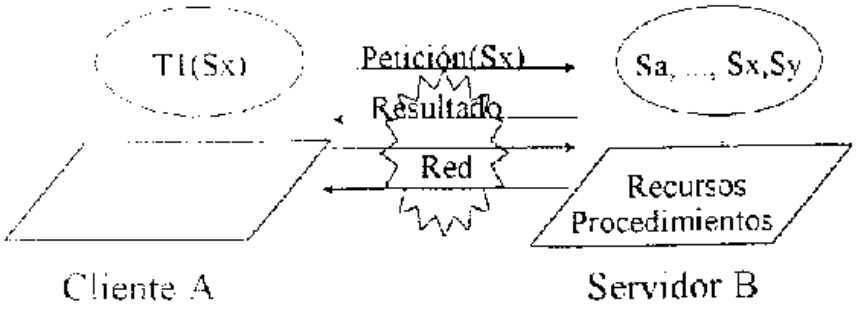
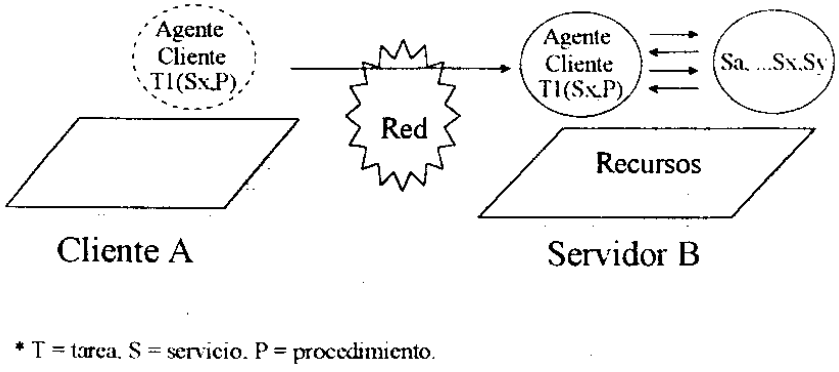


Figura 2.1 Paradigma RPC.



* T = tarea. S = servicio. P = procedimiento.

Figura 2.2 Paradigma RP.

2.2 ¿Qué es un Agente Móvil?

Existen en la literatura diversas definiciones de lo que es un agente móvil. Para [Gray 1995a], un *agente transportable* es un programa que puede migrar de una máquina a otra en una red heterogénea. Este tipo de agente debe ser portable a través de plataformas, debe ser capaz de elegir cuando y donde transportarse a sí mismo, debe ser capaz de duplicarse a sí mismo, y debe poder comunicarse con otros agentes pare intercambiar información.

Según [Nwana 1996], los agentes móviles son procesos computacionales de software capaces de moverse en redes de área amplia (WAN´s) tales como el WWW, interactuando con diferentes hosts, recolectando información en nombre de su dueño y regresando a casa después de ejecutar las tareas delegadas por su usuario. Esas tareas pueden ir desde una reservación de vuelo hasta el manejo de una red de telecomunicaciones. Los agentes móviles son agentes porque son autónomos y cooperan. Por ejemplo, pueden cooperar o comunicarse con otros agentes, intercambiando datos o información.

Para [COTAL y Kotz 1994] los agentes transportables son aquellos que soportan el movimiento de computo cliente a un sitio donde se encuentra un recurso remoto. Son capaces de suspender su ejecución, transportarse ellos mismos a otro host en la red, y reanudar la ejecución desde el punto en el cual ellos fueron suspendidos. Consumen pocos recursos de red y pueden soportar sistemas que no tienen una permanente conexión a la red, tales como computadoras móviles.

Según [Kato et al. 1997], los agentes móviles son objetos que consisten de código, datos y estado de ejecución que puede ir mas allá de dominios protegidos. [Ghezzi y Vigna 1994], definen al agente móvil como un componente que contiene al menos un hilo de ejecución, el cual es capaz de autónomamente migrar a un diferente sitio.

Para [Vitek 1996], un agente móvil es un conjunto de objetos ejecutando un calculo en nombre de un usuario. Este calculo es ejecutado en una plataforma de ejecución de agentes la cual controla la ejecución del agente. Un agente puede requerir moverse causando que su cálculo sea interrumpido y reanudado en otra

[White 1996], plantea que un agente móvil es un programa: (1) que una persona u organización enviste con su autoridad (2) que puede correr sin supervisión por un largo periodo de tiempo (por ejemplo una semana) (3) que puede conocer e interactuar con otros agentes (4) y que puede ejecutarse en diferentes sistemas de computo y en diferentes etapas de su vida.

Otras definiciones se pueden encontrar en [Chess et al. 1995], [Stone et al. 1996], [Knabe 1995], [Nwana 1996], [Sahuguet 1997], [Harker 1995], [Johansen et al. 1995a] y [Jurowiec y Radev 1995.

En resumen se puede decir que un agente móvil es un programa o proceso con las siguientes características:

1. Es autónomo o semiautónomo de manera que él decide cómo, cuándo y a dónde migrar.

2. Esta orientado a ejecutar tareas, a veces en nombre del usuario y otras basándose en los cambios de su ambiente.

3. Se envía como objeto, a través de plataformas conservando ademas de su código, los datos y su estado de ejecución.

4. Es asíncrono, debido a que tiene su propio proceso o hilo de ejecución. Por tanto, el agente se ejecuta asíncronicamente respecto a los otros procesos que se estén ejecutando en el nodo.

5. Es capaz de comunicarse con su dueño, con otros agentes y con el medio.

6. Puede operar sin conexión, es decir, que puede ejecutar sus tareas aún cuando la conexión a red no este funcionando; si el agente necesita trasladarse y la red no esta activa, el agente puede esperar o desactivarse hasta que la conexión se restablezca.

7. Puede suspender su ejecución, transportarse a otro host y reanudar su ejecución desde el punto en el cual se suspendió.

8. Es capaz de duplicarse.

9. Puede reaccionar a cambios en su ambiente, modificando su conducta debido a las acciones generadas por otros agentes, debido a su experiencia propia o por la intervención directa del programador o usuario.

2.3 Una Infraestructura para Agentes Móviles

El agente puede ser escrito en varios lenguajes de programación y puede transportar conocimiento expresado en varias formas. El agente debe ser capaz de entablar un diálogo con el lugar de reunión de agentes hasta que se ejecute o se rechace. Un agente puede ejecutarse hasta su terminación o puede elegir suspender su actividad y moverse a otro lugar de reunión de agentes y continuar su ejecución ahí.

Lugar de Reunión de Agentes (AMP o ”agente meeting places”). Un AMP ofrece servicio para los agentes móviles que entran ahí.

Una máquina es un programa residente en el servidor que implementa el marco de trabajo del agente manteniendo y ejecutando los AMPs que contiene, así como los agentes que ocupan eves AMPs. En general, la maquina es un interprete pare el lenguaje usado pare implementar el ambiente de trabajo del agente.

La máquina se comunica con el nodo a través de 3 aplicaciones APIs. Los APIs son usados pare manejar almacenamiento, transportar agentes y comunicarse con las aplicaciones externa.

2.3.4 TACOMA: una infraestructura para agentes móviles

Es esta sección se presenta una infraestructura particular para agentes móviles, llamada TACOMA. Esta infraestructura se compone de agentes, folders, portafolios y gabinetes. Los agentes TACOMA [Johansen et al. 1995b] desarrollados en la Universidad de Troms y la Universidad de Cornell, están escritos en Tcl/Horus el cual es una versión del lenguaje Tcl que proporciona comunicación y tolerancia a tallas. La abstracción más importante en TACOMA es la operación Met. La cual se use para que un agente ejecute otro agente. Este es un concepto vital usado pare la comunicación y sincronización entre agentes. La operación Met debe tener un punto de entrada, en el sitio de destino.

En TACOMA, no es posible interrumpir la ejecución de agentes. El agente que migra se ejecuta desde el comienzo del programa en vez de continuar después del punto de migración. Esto hace difícil escribir un agente que deba preservar el estado de información mientras migra a través de una secuencia de máquina.

TACOMA no proporciona mecanismos de seguridad y su componente Horas no esta disponible en la mayoría de las plataformas. Características notables de TACOMA son dinero electrónico que es usado para pagar los servicio s, agent es guardaespaldas que inicializan agentes perdidos, agentes de rompimiento que proporcionan programación y servicios de directorio. Requiere que el programador explícitamente capture la información del estado antes de migrar.

El sistema TACOMA proporciona soporte para procesos móviles, que recorren los nodos de una red pare realizar una tarea [Johansen et al. 1995a]. En los últimos tres años se han construidos prototipos de TACOMA (TromsO And Cornell Moving Agents). TACOMA versión 1.2, corre bajo HP-UX, Solaris, BSD Unix y Linux. Los agentes pueden ser escritos en TcVTk, C, Scheme, Perl o Python, se esta añadiendo soporte para Java.

En TACOMA los agentes se comunican usando archivos compartidos o ”fólderes” [Hylton et al. 1996]. Existen varios folders de interés, por ejemplo, un fólder CODE contiene el código fuente de un agente, el cual es vital para la transferencia de agentes. Similarmente, un folder DATA contiene los datos que pueden ser asociados con el agente en el folder CODE. La colección de folder asociados con un agente forman un portafolios, un agente puede acarrear un portafolios mientras se esta moviendo. Existen folders estacionarios que son necesarios para propósitos de almacenamiento de datos permanentes, para eso se usan los gabinetes. La principal diferencia entre los gabinetes y los portafolios es que lo s gabinetes son estacionarios mientras que los portafolios son móviles .

Un agente TACOMA puede causar que otro agente sea ejecutado invocando la operación meet y llamando a un agente fuente y a un

portafolios. El efecto de la operación es terminar la invocación de meet y entonces comenzar a ejecutar el agente destino con los portafolios especificado [Johansen et al¨ 1995b].

Las cuatro arquitecturas presentadas y otras como la de [Reza et al. 1996], [Kato et al. 1997], [Lange y Change 1996] y [Straber et al. 1996], tienen en común el hecho de que pare poder ejecutar un agente, se necesita de un ambiente adecuado y de un sistema o servidor de agentes que controle el acceso al ambiente. Dicho sistema debe ser capaz de monitorear las instrucciones a ser ejecutadas por los agentes y debe proveer las facilidades pare migración y recepción de agentes, así como mecanismos para que los agentes se conozcan e intercambien datos.

Para lograr un ambiente de ejecución adecuado existen ciertos requerimientos y aspectos que deben tomarse en cuenta, como se explica en la sección siguiente.

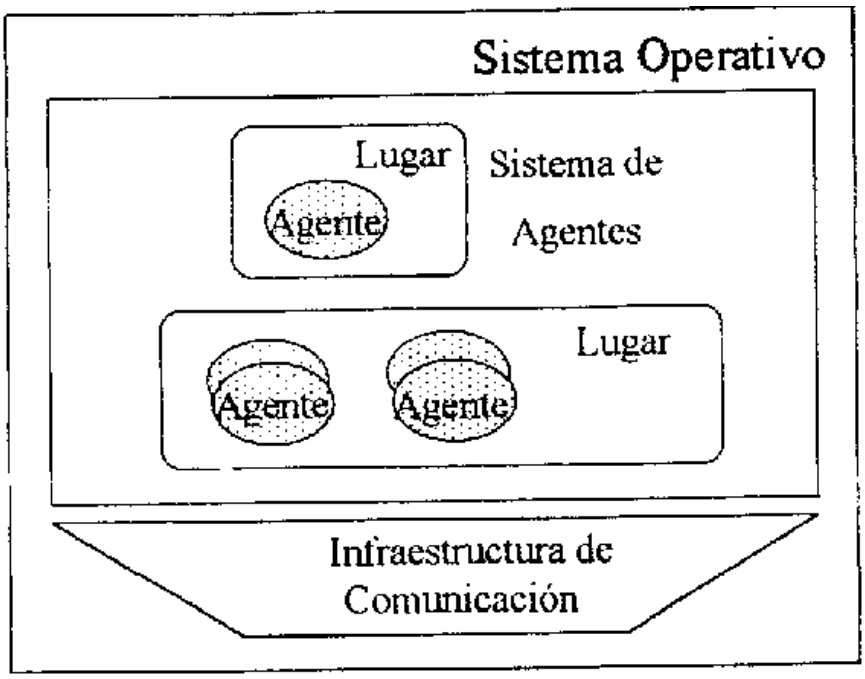


Figura 2.3 Componentes del Sistema de Agentes (Adaptada de [Cristaliz 1997])

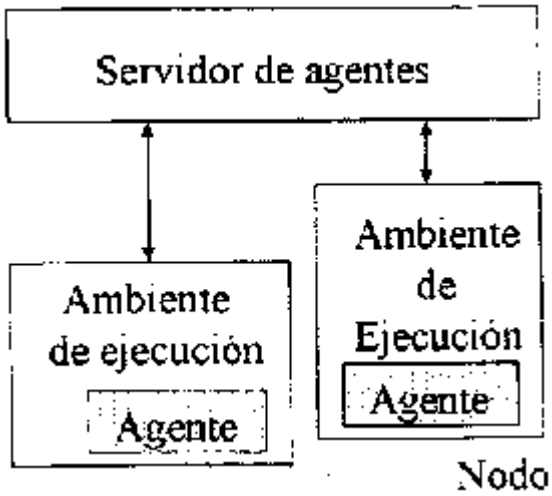


Figura 2.4 Componentes del sistema de agentes (Adaptada de [Lingnau et al. 1995]).

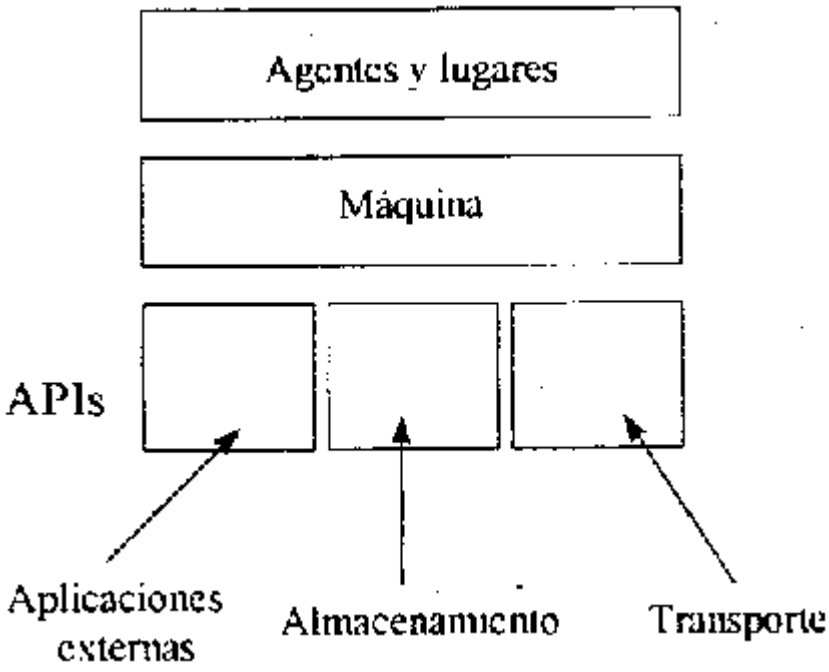


Figura 2.5 Componentes del Sistema de Agentes (Adaptada de 1Stone et al. 1996]).

2.4 Requerimientos para el Desarrollo de Agentes Móviles

La implementación de agentes móviles implica cubrir algunos requerimientos entre los que se pueden mencionar [Sahuguet 1997]:

Necesidad de un lenguaje portable. El código de un agente móvil debe ser aceptado en diferentes plataformas sin requerir una recopilación previa.

Necesidad de un lenguaje interprete. Se requiere de un lenguaje interprete muy útil para la portabilidad en el marco de una máquina virtual. La característica de interprete es también importante cuando el servidor, desea checar el código antes de ejecutarlo. Los lenguajes interpretes permiten que la ejecución sea monitoreada y algunas acciones sean abortadas cuando infringen la política de seguridad.

Necesidad de "late binding", Dada la naturaleza de los agentes, el lenguaje tiene que ser tan flexible como sea posible. Si el agente no conoce a priori a que tipo de servidor está migrando, es necesario ligado dinámico. Esto es adecuado cuando la información completa acerca de la naturaleza de los objetos solamente es conocida en tiempo de ejecución.

Necesidad de una representación común de conocimiento. Los agentes son autónomos en el sentido de que ellos acarrean todo lo que necesitan para vivir como programas estándar, esto incluye código y datos Así que es importante tener una manera estándar de representar datos y conocimiento si se desea involucrar agentes en un trabajo colaborativo.

Necesidad de una abstracción del ambiente remoto. Cuando se programa el comportamiento de un agente el usuario debe tener una representación abstracta de los recursos remotos que el agente será capaz de usar. Esto significa que debe existir una representación común de servidores remotos.

Necesidad de programas adaptativos. Los agentes deben ser capaces de enfrentarse con cualquier tipo de ambiente. La solución mas sencilla para un agente sería moverse a otra computadora si las capacidades ofrecidas localmente no son auto-suficientes. Sin embargo, los agentes deben ser capaces de manejar algunas de las restricciones locales como: almacenamiento limitado de disco, limitación de memoria y limitación de tiempo de CPU. Necesidad de compartir recursos. Una de las ideas detrás del paradigma de agentes es tener algunos servidores compartidos que proporcionen recursos para los agentes y por consiguiente sean capaces de compartir dichos recursos de una manera óptima.

Necesidad de un modelo de negocios. En este sentido hay una serie de. Cuestiones a estudiar, por ejemplo: ¿Cómo definir el precio?,qué tipo de precio manejar (pago por ver, suscripción, etc.)? ,Qué tarifas imponer (modular dependiendo de la carga del servidor o de la hora de acceso, cuotas especiales para grupos, etc.)?

Algo que se mencionó en esta sección es la necesidad de un lenguaje de programación de agentes móviles. Este debe ser capaz de expresar la construcción, transmisión, recepción y subsiguiente ejecución del agente. Su implementación se debe enfocar hacia problemas prácticos como manejo de arquitecturas heterogéneas entre computadoras y proveer capacidad auto-suficiente para ejecutar aplicaciones basadas en agentes. El lenguaje de agentes debe minimizar la sobrecarga que representa la

ejecución de agentes móviles. Finalmente como es un lenguaje para sistemas abiertos distribuidos debe proporcionar su propio modelo de seguridad y debe ser capaz de restringir y vigilar la ejecución de los agentes [Knabe 1996al.

En suma, un lenguaje de programación de agentes, debe ser un lenguaje de alto nivel, de fácil uso, orientado a objetos, que soporte movilidad y que proporcione constructores que soporten computo distribuido.

2.5 Lenguajes de Programación para Agentes Móviles

Un contexto es un objeto estacionario que proporciona un medio de recibir y manejar aglets en un ambiente seguro de aglets dañinos. A través de este contexto un aglet es capaz de conseguir información acerca de su ambiente, y enviar y recibir mensajes a ese medio y a otros aglets activos en su proximidad. Si un aglet da o intenta violar el proxy aglet y accesar datos privados, el manejador de seguridad detectará el intento y detendrá el acceso. Los aglets pueden comunicarse entre ellos pasando mensajes vía el contexto. Este proceso entabla el intercambio de un objeto mensaje entre dos aglets, y permite el paso de mensajes síncronos y asíncronos, así que habilita a los aglets para colaborar e intercambiar información de una manera estricta y relajada.

Aglets Workbench soporta movilidad e itinerario. La movilidad en Aglets Workbench es posible gracias a dos facilidades:

1. El Protocolo de Transferencia de Agentes (ATP);
2. La Interfaz Java de Transferencia y Comunicación de Agentes (J-ATCI).

El ATP es una aplicación al nivel de protocolo para agentes distribuidos basado en sistemas de información que facilita la migración de los aglets sobre la red. Basado en las convenciones de Internet, ATP usea el Universal Resource Locator (URL) para especificar direcciones de los nodos, mantiene una plataforma de protocolo independiente para habilitar la transferencia de agentes móviles entre computadoras de una red. Aunque este protocolo ha sido liberado con Aglets Workbench su dominio no es de uso exclusivo para aglets, ofrece la oportunidad de manejar agentes móviles desde cualquier lenguaje de programación y una variedad de sistemas de agentes.

Para reforzar el ATP se tiene un nivel de comunicación más alto conocido como J-ATCI. Este es un protocolo de agentes independiente que habilita a los agentes para moverse y comunicarse en una red. J-ATCI es un programa de interfaz simple y flexible que facilita a los programadores el

desarrollo de una plataforma de agentes independientes sin tener que construir protocolos de comunicación. En el Apéndice C se incluyen algunos detalles técnicos adicionales de Aglets Workbench.

2.5.5 Odyssey

Incluye un conjunto de bibliotecas Java que proporcionan la funcionalidad de agentes móviles [General Magic 1997]. Estas bibliotecas permiten crear agentes y lugares en Java. Java no proporciona un manejo de captura del estado de ejecución de un hilo Java. Así que cuando un agente Odyssey es transportado desde un sistema a otro, el hilo del agente es reinicializado en el destino, es decir que Odyssey no permite que sus agentes sean persistentes.

La clase worker de Odyssey es una subclase de la clase agente que soporta una tarea por destino. Un worker esta estructurado como un conjunto de tareas y un conjunto de destinos. En cada destino, el worker ejecuta la siguiente tarea de su lista de tareas. Un worker Odyssey puede manipular su lista de tareas en cualquier punto durante su viaje. Debido a que los agentes Odyssey son hilos Java, cada agente Odyssey y conjunto de destinos están en su propio grupo de hilo.

Odyssey corre en cualquier plataforma que soporte JDK l.l. Esta enteramente escrito en Java, y no requiere modificaciones de Java VM. Usa Remote Method- Invocation (RMI) [December 1996] de Java y mecanismos de reflexión. También use Java AWT 1.1.

Odyssey puede usar cualquier mecanismo de transporte confiable para mover un agente de un sistema de agentes a otro. Además de RMI, se tienen otros dos mecanismos da transporte, Microsoft s Distributed Component Object Model (DCOM) y el Internet- Orb Protocol (IIOP). RMI es usado para dos diferentes propósitos en Odyssey. Es el mecanismo de transporte de los agentes (aunque puede ser reemplazado por DCOM o IIOP) y es también usado como un mecanismo Finder, para determinar en que lugar se localiza un sistema de agentes Odyssey. RMI debe ser usado para la funcionalidad Finder, pero no necesariamente como el medio de transporte de agentes. El sistema de agentes de Odyssey consiste de un conjunto de clases Java para soportar agentes y lugares Odyssey.

2.5.6 MO

Implementado en la Universidad de Génova, M0 [Gray 1995a] es un lenguaje intérprete basado en piles (stacks) que implementa el concepto de mensajeros. Los mensajeros son los agente móviles en M0. Estos mensajeros o secuencias de instrucciones pueden ser enviados y ejecutados en máquinas remotas. Cada máquina proporciona un ambiente de ejecución que incluye un intérprete, primitivas de

sincronización y un diccionario de datos compartidos. Cada nodo está conectado por canales los cuales representan las rutas de comunicación entre los diferentes hosts.

M0 es un sistema de bajo-nivel que esta destinado a un rango de aplicaciones distribuidas. M0 no proporciona directamente la funcionalidad de agentes transportables pero puede ser usado como una capa menor en un sistema de agentes transportables. M0 difiere de los lenguajes mencionados anteriormente en que no fue concebido para ser un

Lenguaje en el cual las aplicaciones de código móvil estén programadas. Su meta es proporcionar una capa intermedia que soporte la movilidad de capas de nivel superior

2.5.7 Obliq

Desarrollado en DEC por Luca Cardelli, Obliq es un lenguaje intérprete que soporta cómputo distribuido orientado a objetos [Cugola et se. 1995a, Cu gola et al. 1995b]. Los objetos Obliq tienen un estado y son locales a un lugar.

Obliq permite la ejecución remota de procedimientos por medio de máquinas de ejecución las cuales implementan el concepto de nodo. Un agente móvil puede solicitar la ejecución de procedimientos de una máquina remota. El código para tales procedimientos es enviado a la máquina destino y ejecutado ahí por un nuevo agente. Los objetos Obliq están limitados por el nodo en el cual fueron creados. Cuando un agente móvil pide la ejecución de un procedimiento de un sitio remoto, las referencias a los objetos usados por tal procedimiento son automáticamente trasladados en referencias de red. Los accesos a esos objetos son traducidos en llamadas al nodo original [Cardelli 1995]3.

2.5.8 Tycoon

Desarrollado en la Universidad de Hamburgo, es un lenguaje funcional persistente [Cugola et al. 1995a, Cugola et al. l995]. Proporciona hilos de migración como Telescript, pero no tiene la riqueza de abstracciones y características que Telescript. Estos hilos se pueden mover desde una máquina virtual Tycoon a otra usando la primitiva migrate. Los hilos de migración pueden especificar el tipo de recursos remotos que accesarán en la máquina de migración de destino. Cuando un hilo arriba a la máquina de migración de destino, busca los recursos remotos requeridos, si están presentes, son limitados al hilo y entonces el hilo reanuda su ejecución.

2.5.9 MOLE

MOLE es una extensión de Java que añade la capacidad de enviar código entre intérpretes Java [Straber et al. 1996]. En Mole, los agentes son modelados como grupos de objetos. Las referencias entre agentes son referencias simbólicas. Cada agente tiene un nombre único el cual es usado para identificar el agente. Los agentes se pueden comunicar con otros agentes solamente a través de mecanismos de comunicación previamente definidos. En la primera implementación de Mole, se usan dos medios de comunicación: RPC y RMI.

Los agentes pueden comunicarse entre sí por medio del envió de mensajes. En Mole los mensajes son usados para transferir datos entre los agentes. Los mensajes pueden ser síncronos o asíncronos [Baumann et al. 1996]. La operación de envío es un mecanismo asíncrono, el cual regresa inmediatamente. Este consigue como parámetros la dirección del emisor, la dirección del receptor y los contenidos del mensaje (un objeto). El mensaje es entonces enviado al destino. Si el receptor existe en la máquina de destino, el mensaje es liberado llamando un método especial el cual ha sido implementado en cada agente. Este método es siempre ejecutado en su propio hilo por algún tiempo, y si después de un tiempo no ha sido liberado entonces es enviado de regreso

Los agentes son ejecutados en un nodo, usando los servicios proporcionados por el mismo. Existen dos tipos diferentes de agentes, los agentes de .Sistema y los agentes de usuario. Los agentes de sistema son agentes que accesan a los recursos del sistema, proporcionando abstracciones seguras y controladas de esos recursos. Estos agentes son estáticos y pueden ser inicializados únicamente por el administrador del sistema. Los agentes de usuario solamente se comunican con otros agentes y no tienen acceso directo a los recursos del sistema.

Se cuenta con un ”sitio” que maneja los agentes ejecutados en esa computadora, el cual recibe a los agentes y ofrece algunos servicios básicos como migración, comunicación y un servicio de directorio. Se tiene una ”máquina” que maneja los ”sitios” existentes en un sistema. Esta ofrece servicios como comunicación con la clase servidor. La clase servidor es responsable de que las clases desconocidas necesitadas por los agentes sean ejecutadas en un ”sitio” en la ”máquina”.

2.5.10 Facile

Desarrollado en ECRC en Munich, Facile es un lenguaje funcional concebido para programación concurrente distribuida [Knabe 1996a].

Es un lenguaje de programación que integra soporte para concurrencia y distribución. La representación natural para un agente en Facile es simplemente una función. Las funciones en Fácil pueden ser creadas en

tiempo de ejecución, pasadas a y desde funciones. Un agente puede ser estructurado como muchas funciones, con una de ellas diseñada como el punto de entrada principal, todas las otras funciones serán parte del ambiente de la función principal.

En Facile, el ligado dinámico es soportado adecuadamente. El programador puede especificar interfaces para los recursos que una función accesará durante su tiempo de vida.

La función operará sobre esos recursos únicamente a través de sus interfaces. Cada interfaz esta compuesta de un conjunto de funciones firmadas las cuales definen las operaciones que pueden ser ejecutadas. En tiempo de ejecución, cualquier recurso local que ofrece al menos un conjunto de posibles operaciones coincidentes con aquellas listadas en la interfaz puede ser vinculado a la función. Así que una función moviéndose a un nuevo nodo puede acceder cualquier recurso de ese nodo que coincide con los recursos de interfaz contenidos en el código de la función [Cugola et al. 1995a, Cugola et al. 1995b].

2.5.11 Phantom

Es un lenguaje para programación distribuida muy cercano a Modula-3. Las principales características de Phantom están fundadas en el paradigma RPC y los lenguajes son compilados estáticamente, incluyendo llamados transparentes de procedimientos remotos, soporte de concurrencia y autentificación. Phantom permite el uso de ciertas técnicas de programación tales como migración de objetos, evaluación remota, y extensibilidad dinámica [Courtney 1996]. Phantom permite enviar pequeñas partes de los programas (procedimientos individuales y funciones) a nodos remotos para su ejecución. Phantom proporciona facilidades de distribución transparente tanto de código como de datos.

Phantom es un intérprete que ejecuta manejo automático de almacenamiento, carga dinámicamente y ejecuta programas. Phantom, también proporciona un acceso transparente al código y datos de nodos remotos. En Phantom todos los valores de los programas son potencialmente accesibles desde la red. Sin embargo, los detalles de distribución son mantenidos bajo un flexible programa de control. Phantom proporciona declaraciones implícitas de tipo seguro. Usa un modelo de objetos basado en clases. Proporciona control simple de acceso y mecanismos de autentificación a nivel de lenguaje. El intérprete está implementado en ANSI C, y proporciona una biblioteca de interfaz hecha en Tk.

El lenguaje Phantom soporta un numero de características importantes: interfaces, objetos, hilos, recolección de basura y excepciones. Todas estas son características del lenguaje Modula-3. Phantom copia la sintaxis y semántica de Modula-3, y omite las características más complejas de este

lenguaje. Phantom también incluye soporte para declaraciones implícitas y listas dinámicas.

Los objetos en Phantom tienen atributos, un número de métodos y soportan herencia sencilla. Los objetos son el centro de la comunicación en Phantom. Un programa Phantom hace disponibles sus servicios a otros programas registrando los valores de sus objetos en un servicio de nombres- una aplicación en Phantom que asocia cadenas de nombres a las direcciones de los objetos. Los objetos son los únicos valores que son pasados por referencia, nunca son implícitamente migrados a nodos remotos como resultado de una asignación, llamado de procedimiento o declaración de retorno. En vez de eso, los objetos permanecen estacionarios y se pasa una referencia al nodo remoto. Si se requiere la migración de los objetos, esta debe ser ejecutada explícitamente por el programador.

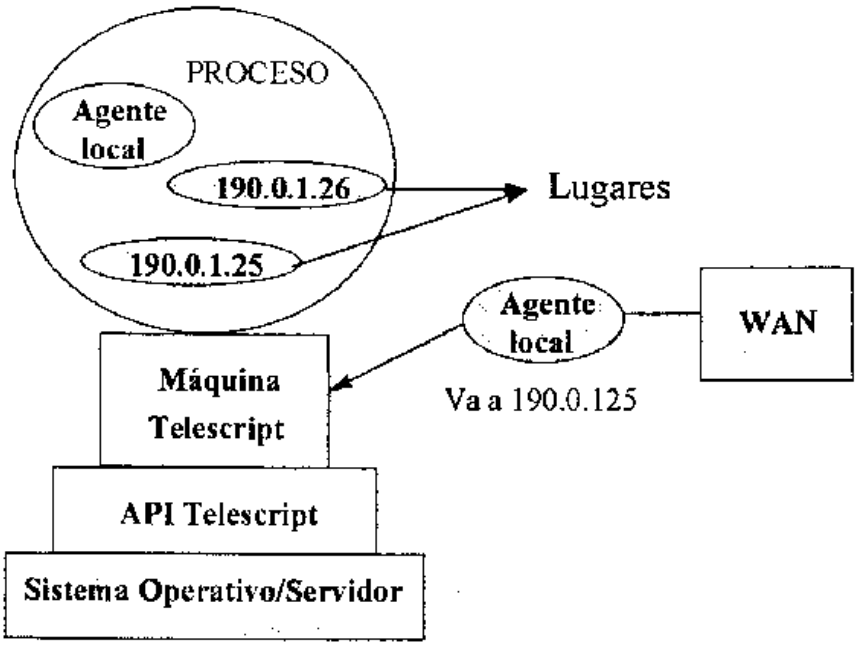


Figura 2.6 Arquitectura de una máquina Telescript (Adaptada de [Wayner 1995]).

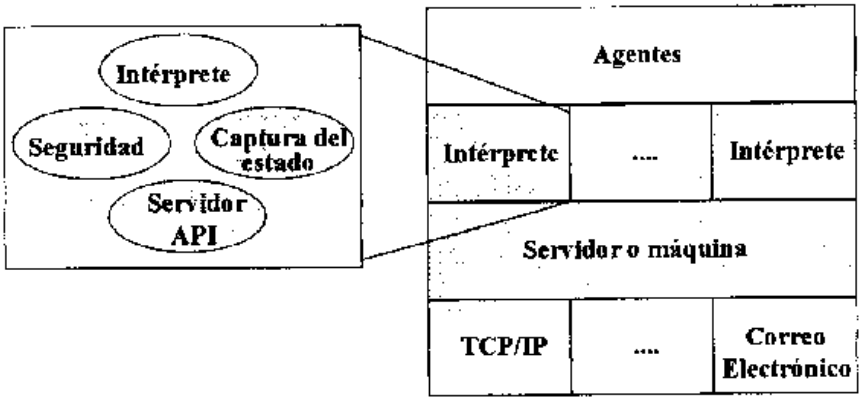


Figura 2.7 Arquitectura de Agent Tcl (Adaptada de [Gray 1995a]).

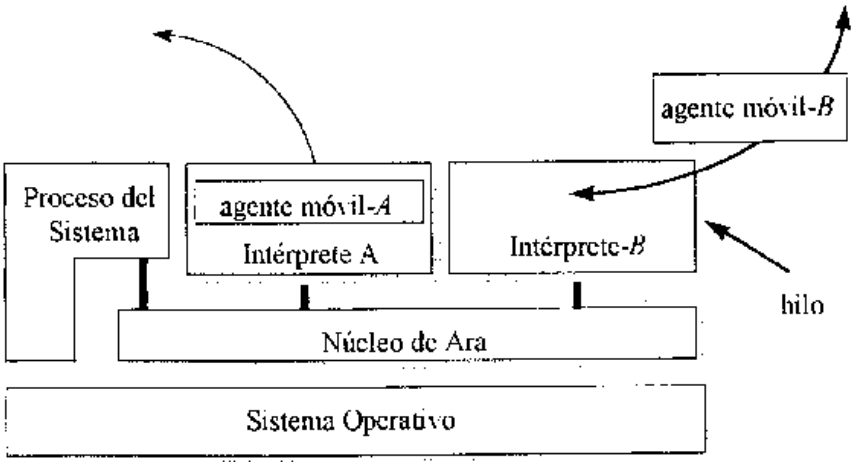


Figura 2.8 Arquitectura del Sistema Ara (Adaptada de [Peine Stolpmann 1997]).

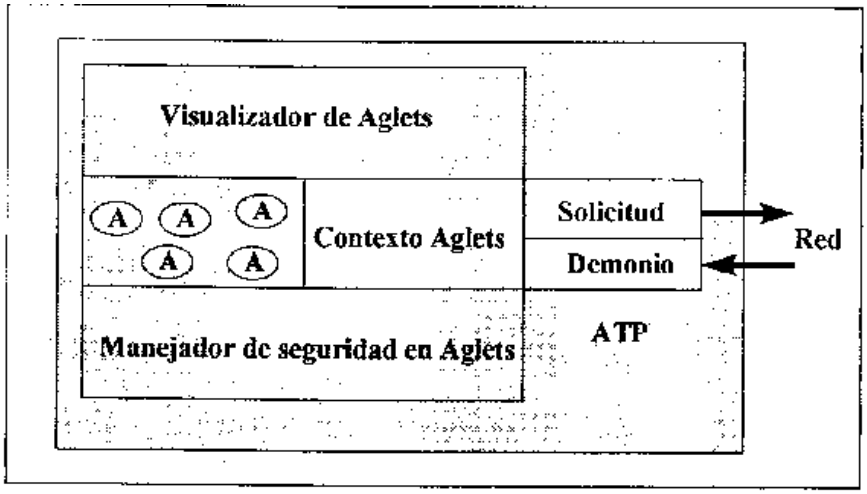


Figura 2.9 Arquitectura del Visualizador de Aglets (Adaptada de [Xu y Tao 1997]).

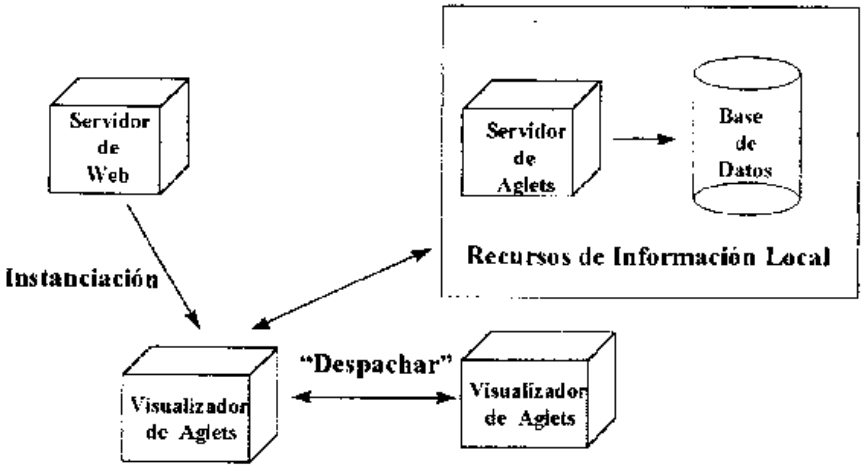


Figura 2.10 Infraestructura de Aglets (Adaptada de [Xu Tao 1997]).

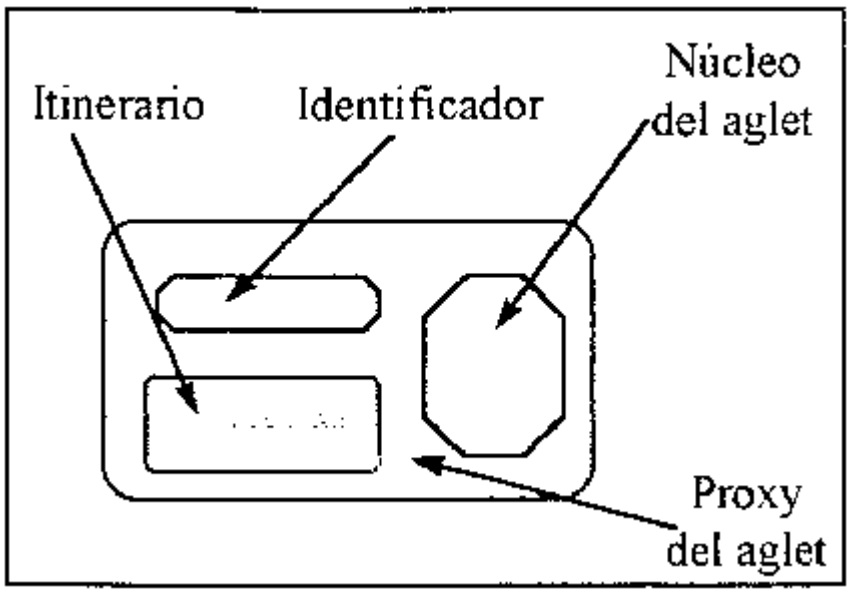


Figura 2.11 Estructura de un aglet (Adaptada de [Clements et al. 1997]).

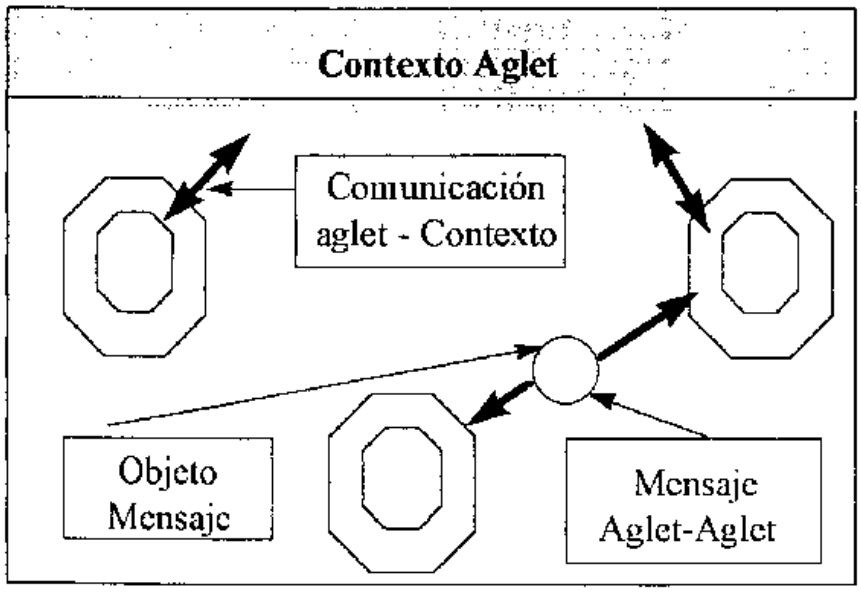


Figura 2.12 El Ambiente Aglet (Adaptada de [Clements et al. 1997]).

2.6 Tabla Comparativa

Una vez que se han presentado los lenguajes de programación y las plataformas de agentes móviles, es factible establecer una tabla comparativa. La Tabla 2.1 presenta un resumen de dichos lenguajes y

muestra de manera esquematizada las características que cada uno de ellos exhibe. Los lenguajes y plataformas están descritos tomando en cuenta los siguientes criterios: tipo de migración, tipo de ligado, estrategia de replicación usada, alcance y resolución de nombres, aspectos de seguridad, estrategia de traducción, estrategia de programación, contexto de trabajo e instrucción de migración [Cugola et al. 1995a].

Tipo de migración. Esta puede ser fuerte cuando permite que el agente migre su código y su estado de ejecución, de manera que puede interrumpir su ejecución, trasladarse a otro nodo y reanudar su ejecución ahí sin mayor problema. La migración es débil cuando el agente esta limitado a código que le llega de un agente y que liga dinámicamente.

Ligado Dinámico. Para soportar la migración, los lenguajes deben proveer mecanismos para envío de código y datos hacia otro nodo y dinámicamente ligar el código y los datos de un agente. Existen dos tipos de ligado dinámico: ligado dinámico de código remoto y ligado dinámico de recurso local. El ligado dinámico de código remoto extiende la noción de ligado de sistemas operativos en aplicaciones de red. Permite a los programadores. Implementar técnicas de "código en demanda", esto es, aplicaciones que cargan su código dinámicamente desde la red de acuerdo a diferentes estrategias. El ligado dinámico de recurso local es un mecanismo que permite que un agente accese a los recursos locales del nodo receptor. Por ejemplo, acceso a bibliotecas localizadas en el nodo de destino.

Estrategia de replicación. Cuándo el código de un agente es trasladado, ¿qué sucede si los nombres de sus variables están limitadas a recursos en su nodo fuente? Para estos casos es necesaria la replicación. Las estrategias de replicación pueden ser de dos tipos: estrategias de replicación estática y estrategias de replicación dinámica. Con la estrategia de replicación estática, algunos recursos pueden replicarse estáticamente en todos los nodos, así que se establecen nuevos valores en el proceso de ligado con las instancias locales de cada nodo de destino. En las estrategias de replicación dinámica, se hace una copia de los recursos en el nodo destino, y se establecen nuevos valores de ligado con los recursos copiados.

Resolución de Nombres y Alcance. El alcance de un identificador es el rango de instrucciones sobre las cuales se conoce el identificador . Las reglas de resolución de nombres determinan cual entidad computacional está limitada a cada uno de los identificadores en cualquier punto de un programa dado.

Seguridad. El ambiente computacional de los lenguajes de código móvil proporcionan una plataforma distribuida en la cual las aplicaciones pertenecen a diferentes usuarios y tienen diferentes derechos de acceso

al ambiente del nodo. Además, los nodos que componen la infraestructura deben ser manejados por diferentes autoridades y pueden comunicarse a través de infraestructuras de comunicación inseguras. Este escenario sugiere dos dominios de seguridad: seguridad entre ambientes y seguridad en el ambiente. Para la seguridad entre ambientes se puede usar autentificación entre ambientes y entre el ambiente y los agentes. Para la seguridad en el ambiente, se deben implantar técnicas de control de acceso que aseguren un ambiente seguro para los agentes.

Estrategia de Traducción. La elección de ejecutar un lenguaje de programación ya sea a través de interpretación directa o a través de compilación, es algo que debe estar contemplado en los lenguajes de programación de agentes móviles.

Tipo de estrategia de programación. Los agentes pueden ser programados como hilos de ejecución, funciones, procedimientos, flujos de control, procesos u objetos.

Contexto de Trabajo. La mayoría de los lenguajes de programación de agentes móviles requieren de un contexto o servidor de agentes en cada uno de los nodos en donde se va a realizar la migración de agentes, esto con el fin de tener un mecanismo que controle y supervise la ejecución de los mismos.

Tabla 2.1 Tabla comparativa de lenguajes y plataformas de agentes móviles

Lenguaje o Plataforma	**Tipo de migración**		**Ligado dinámico**		**Estrategia de replicación**		**Resolución de Nombres y Alcance**	
	Fuerte	Débil	De código remoto	De recurso local	dinámica	estática		autenti
TACOMA								
MO								
Facile								
Obliq								
Java								
Tycoon								
Telescript								

Ara								
Agent Tcl								
IBM Aglets								
Odyssey								
MOLE								
Phantom								

2.7 Aplicaciones de los Agentes Móviles

Los agentes móviles pueden ser útiles para muchas aplicaciones. [Lingnau y Drobnik 1995] sugieren:

La recuperación de información en la red puede ser soportada mucho más eficientemente si un agente representa un query que puede moverse al lugar en donde los datos están almacenados, en lugar de tener que mover todos los datos a través de la red para revisarlos y posteriormente descartar la mayoría. Esta técnica ahorra un considerable ancho de banda.

Otra área donde los agentes móviles pueden ser usados es en el manejo de red. En redes grandes, que contengan cientos o miles de computadoras conectadas, es muy difícil ejecutar operaciones de monitoreo y detección de fallas, ya que involucran grandes cantidades de datos. No es posible prefabricar programas de diagnóstico para cada eventualidad, pero será factible usar agentes móviles que observen detalladamente el sistema, enviando reportes sobre el estado de las computadoras. [Goldszmidt y Yemini 1995] introducen un. Sistema basado en "delegación a agentes" los cuales están dinámicamente ligados a procesos remotos. Por este lado, el potencial de los agentes móviles para manejo de red es de interés para las compañías de telecomunicaciones.

Comercio electrónico es otro dominio en el cual pueden funcionar los agentes móviles: los negocios en Internet son una realidad y, como ya se cuenta con sistemas para pago electrónico, las premisas comerciales accesibles vía la red crecerán rápidamente. Los agentes móviles pueden ayudar a localizar los lugares más baratos, negociar tratos o concluir transacciones de negocios en nombre de sus dueños.

Una aplicación importante para los agentes móviles es el concerniente a computo móvil. Las computadoras portátiles son cada vez más pequeñas y más poderosas, pero el acceso a infraestructura de información fija es lento debido a las restricciones en la transmisión. Para minimizar el poder

de consumo y el costo de transmisión, los usuarios desearían no tener que permanecer en línea mientras alguna consulta complicada está siendo ejecutada en su nombre por los recursos de computo fijo. Los agentes móviles ofrecen una manera prometedora para salir de este problema, los usuarios simplemente liberan un agente móvil que incorpore sus búsquedas y se desconectan, conectándose después para que los agentes devuelvan el resultado.

Para [Knabe 1995], los agentes móviles ofrecen solución a un amplio rango de problemas encontrados frecuentemente en aplicaciones distribuidas. Al mismo tiempo, hacen posibles nuevos tipos de aplicación con funcionalidades novedosas:

Comunicación heterogénea. Aunque las redes ofrecen muchos beneficios potenciales, todavía se tienen problemas de comunicación y distribución. Al tener dos sistemas juntos se revelarían los problemas de incompatibilidad en comandos y tipos de datos y a mayor número de sistemas mayor incompatibilidad. Los agentes móviles pueden resolver estos tipos de problemas sirviendo como una lengua que comparten muchos sistemas.

Comunicación reducida. La ventaja clave de los agentes es que pueden decrementar los costos de comunicación. En un sistema cliente-servidor estructurado con agentes, parte del cliente o del servidor puede moverse al otro lado de la línea de comunicación. Una vez que el cliente o servidor se ha movido, las interacciones entre los dos pueden evitar el uso de la red

Protocolos especializados. Los agente móviles permiten que los servidores usen protocolos de comunicación personalizada con los clientes. Para recibir un agente, el cliente y el servidor deben compartir algún protocolo estándar. Una vez que el agente esta corriendo, puede usar un protocolo especializado para la comunicación de regreso a su servidor de origen. Un agente en ejecución puede comunicarse repetidamente con el servidor sin intervención del usuario, permitiendo la construcción de servicios dinámicos.

Reduce la carga del servidor. Uno de los problemas con los servicios de Internet es que los componentes computacionales de un servicio típicamente deben residir en un servidor, porque los protocolos de aplicación usados como HTTP, Gopher, y Telnet se usan para el intercambio de datos no ejecutables, así que los servidores tienen la responsabilidad de ejecutar cualquier servicio relativo a computo para clientes. Esto lleva a transmisiones con requerimientos de gran ancho de banda y significa que el usuario debe esperar hasta que el servicio proporcione resultados. Estructurar un servicio con agentes puede resolver o reducir los problemas mencionados . La más importante característica de los agentes es que permiten mover computo. Un servidor

puede descargar trabajo en un cliente enviándole un agente. El cliente esta presumiblemente dispuesto a dedicar recursos tales como tiempo de CPU para la interacción del servicio, y esos recursos pueden ser usados directamente por el agente. El usuario ahorra tiempo y la carga en el servidor y la red se aligera.

Datos inteligentes. Agentes asociados con los datos pueden proporcionarle al dato una manera de "conocer" como procesarse a sí mismo.

Obviamente ninguna de estas aplicaciones requiere del uso forzoso de agentes móviles, de hecho la mayoría podrían manejarse con programas estacionarios y algún paradigma de comunicación adecuado como RPC. Sin embargo, esto puede haer que el sistema y la red se "cargue", con la consecuente incomodidad para el usuario.

2.8 Ventajas y Desventajas del Uso de Agentes Móviles

Entre las ventajas de esta nueva tecnología podemos mencionar [Nwana 1996]:

Reduce costos de comunicación. Podría haber una gran cantidad de información que necenodo ser examinada para determinar su relevancia. Transferir esta información puede consumir tiempo y atascar la red. Imagínese tener que transferir muchas imágenes solas para elegir finalmente una. Es mucho más natural tener un agente que "vaya" a esa localidad, haga una búsqueda/elección y solamente transfiera la imagen elegida de regreso a través de la red. Esto evita la necesidad de haer conexiones de red costosas entre computadoras remotas tan requeridas en llamadas de procedimientos remotos (RPC). Esto proporciona una alternativa mucho más barata en ancho de banda y en tiempo de acceso.

No se limita a recursos locales. Si el poder de procesamiento y almacenaje en una maquina local es muy limitado, es necesario el uso de agentes móviles, de esta manera se puede migrar a una computadora más poderosa y lograr ejecutar la aplicación deseada.

Coordinación más sencilla. Puede ser más simple coordinar un número de solicitudes remotas e independientes y después solamente verificar los resultados de manera local.

Permite Cómputo Asíncrono. El usuario puede activar sus agentes móviles y hacer alguna otra actividad mientras tanto y los resultados le llegarán por correo electrónico o algún otro medio, en algún tiempo posterior. Incluso puede operar aún cuando el usuario no este "conectado".

Pueden ir y venir dinámicamente y servicios mucho más flexibles pueden coexistir en unidades inferiores, proporcionando más opciones para los consumidores.

Proporciona una arquitectura flexible de cómputo distribuido. Los agentes móviles proporcionan una arquitectura de cómputo distribuido única, la cual funciona de manera diferente de las arquitecturas estáticas. Esto proporciona una manera innovadora de hacer cómputo distribuido.

Presenta una oportunidad para hacer un reestructuración radical y atractiva del proceso de diseño en general. Siguiendo esto último, se dice que los agentes móviles transforman el proceso de diseño convencional, además de que algunos productos verdaderamente innovadores deberán emerger de esta nueva tecnología.

[Sahuget 1997] menciona algunas otras ventajas del uso del paradigma de agentes:

Aprovechamiento de la asincronía. asincronía significa que dos actores de la comunicación no necesitan estar físicamente presentes al mismo tiempo (por ejemplo los usuarios del correo electrónico. Las ventajas de la asincronía son el mejoramiento del uso de las líneas de comunicación, la capacidad de realizar operaciones de recuperación de información más seguras y el hecho de que si el receptor esta ocupado cuando la comunicación se esta llevando a cabo, esta se procesará después. Esta última propiedad es muy interesante para cómputo móvil (PDA, laptops) que no esta permanentemente conectado. La estrategia estándar sería entonces: enviar el agente, desconectar y reconectar después. Con respecto al uso de las líneas de comunicación, se sabe que las sesiones basadas en comunicaciones imponen una conexión permanentemente abierta entre en emisor y el receptor, esto requiere una conexión ocupada aunque nada este pasando actualmente. Para comunicaciones de bases de datos, esto puede empeorar si las transacciones imponen algunos bloqueos, este bloqueo se mantendrá hasta que la transacción sea abortada o reanudada. Pero si un agente es despachado en una manera asíncrona. Una vez en un lugar remoto, el agente puede ejecutar un proceso asíncrono y entonces esperar por una llamada de regreso de su computadora de origen o decidir regresar por el mismo. Cuando el usuario se reconecta, recibe al agente de regreso, por todo lo expuesto se dice que cuando la tarea a ser ejecutada no es en tiempo real, este esquema parece ser muy atractivo. Como se menciono con anterioridad la asincronía permite realizar operaciones de recuperación de datos más seguras. Cuando una transacción es comprometida este es un proceso de todo o nada, quien no ha experimentado la frustración de ver su proceso de FTP interrumpido segundos antes de terminar y tener que comenzar todo otra vez. En el caso de agentes, una vez que el agente ha sido transferido y exitosamente recibído ya no hay de que preocuparse, ya que el agente puede pedirle al servidor remoto ser activado o reactivado las veces

necesarias hasta que el trabajo haya sido terminado.

Aprovechamiento de la autonomía. Un agente debe mostrar algo de autonomía. Deben comportarse como "criaturas vivas" una vez que han sido convocadas. La autonomía realmente significa que no existe la necesidad de una conexión permanente entre el agente y su nodo de origen, ya que en el caso de agentes móviles el agente acarrea junto con él su propio código. El agente es todavía más autónomo cuando tiene algún conocimiento de las preferencias del usuario. Esta propiedad de autonomía es muy importante ya que permite al agente trabajar por sí mismo y no requiere de una conexión permanentemente abierta.

Aprovechamiento de las facilidades remotas. La gran contribución de los agentes móviles es ser capaces de ejecutarse en máquinas remotas. Así que pueden aprovechar las capacidades remotas:

CPU. El agente es ejecutado en la máquina remota donde es más potente debido a la capacidad del CPU remoto. Esto es útil para dispositivos móviles (por ejemplo computadoras portátiles) con un CPU pobre o no disponible. Para máquinas cliente con CPU pobre, su debilidad puede ser resuelta a través de agentes.

Memorial Algunas operaciones pueden requerir una gran cantidad de memoria, por lo que puede ser útil tener acceso a memoria remota.

Multiprocesamiento. Como una extensión del CPU, si el nodo remoto tiene capacidades múltiples de procesamiento, estas pueden ser usadas por el agente.

MULTI- threacling Los hilos pueden ser vistos como versiones "ligeras" de paralización.

Ancho de banda. Si la PC de un usuario tiene un modem de 28.8 baudios y la red de su oficina tiene un par de conexiones T1, el usuario puede enviar su agente a la red de la oficina donde puede aprovechar un ancho de banda mayor.

Otros recursos. Otros recursos que no pueden ser hallados localmente pueden ser usados por los agentes en el nodo remoto. Por ejemplo generadores de números aleatorios, coprocesadores matemáticos, hardware dedicado.

En teoría los agentes pueden ser capaces de aprovechar una gran variedad de recursos. Sin embargo los aspectos de seguridad y económicos representan aun una gran barrera. Pasando a las desventajas la mayoría de los autores coinciden en que el punto más débil de los agentes móviles es precisamente la seguridad. Este tema será tratado con más profundidad en la siguiente sección. Otras desventajas son las que se presentan en algunos de los lenguajes de programación para el diseño de

agentes, entre las que podemos mencionar:

La migración no puede ocurrir en puntos arbitrarios o requiere la captura explícita del estado de ejecución a nivel del agente.

La comunicación entre agentes no existe o es difícil.

Los agentes deben ser escritos en un lenguaje específico y complejo

Las implementaciones solamente existen para hardware no estándar

Porciones de la implementación solamente corren en plataformas especificas de Unix.

El código fuente no esta disponible para la comunidad

Como Ejemplos de los lenguajes son los que presentan algunas de esta desventajas se podrían mencionar los siguientes: Telescript fue desarrollado en un lenguaje orientado a objetos muy complejo, requiere hardware poderoso de propósito especial, no está abierto a los investigadores y limita al programador a un solo lenguaje. TACOMA requiere que el programador explícitamente capture el estado de ejecución antes de la migración. ARA no cuenta con niveles de seguridad adecuados [Gray 1995a; Gray 1996].

2.9 Seguridad

La seguridad en las redes de cómputo han sido estudiadas y resueltas con técnicas de criptografía. La protección de la máquina y derechos de acceso (aspectos 2 y 3 en la Figura 2.13) tienen una fuerte similitud con los aspectos de protección en sistemas operativos [Vitek 1996]. La protección del AEP ha sido estudiada en el campo de los lenguajes de contenido ejecutable. El último aspecto, la comunicación entre agentes, tiene una fuerte similitud con los aspectos de protección en cómputo distribuido.

A grandes rasgos algunos de los mecanismos de seguridad que pueden implementarse son:

El uso de un lenguaje de agentes ”seguro” que no permita que otros tengan acceso a los datos ”privados”, tal como Java o el uso de espacios de direcciones aislados.

Autentificación de los agentes, por Ejemplo usando firmas digitales o técnicas de criptografía.

El manejo de ”cuentas” para el acceso a los recursos de la máquina.

El uso de mecanismos de control de recursos tales como restricciones en

tiempo de ejecución. [Hohl 1997].

Autentificación del origen del agente.

Control de acceso/itinerario.

Autentificación mutua entre los sistemas de agentes.

.

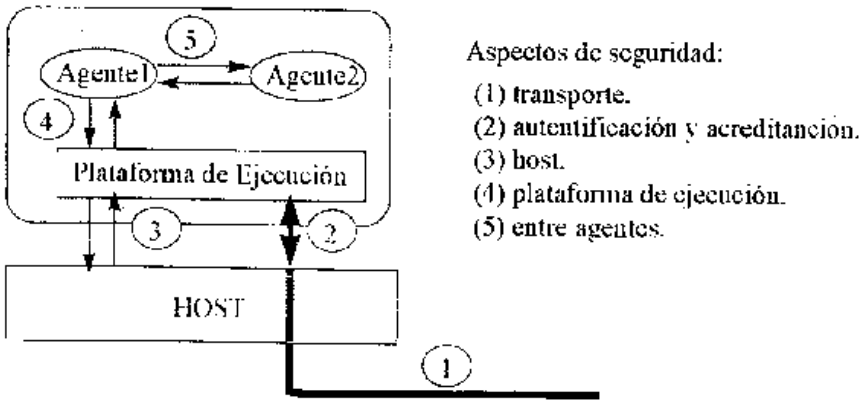


Figura 2.13 Aspectos de seguridad (Adaptada de [Vitek 1996])

Tabla 2.2 Relación entre el problema de seguridad y su posible solución (Adaptada de [Vitek 1996]).

Seguridad	Peligro	Defensa
Red	Acceso a información privada encapsulada en un agente	Técnicas de criptografía, canales seguros, autentificación
Nodo	Accesos no autorizados a recursos locales	Acceso al host condicionado por el AEP
Plataforma	Interferencia con el estado de AEP por la manera de leer, escribir o ejecutar código	Agente seguro-comunicación AEP y manejo de ”cuentas” para el acceso a los recursos de la máquina

2.10 Algunos Desafios

[Wayner 1995] lista una serie de retos a vencer en el desarrollo de agentes móviles.

Transporte: ¿Cómo haer que un agente se mueva de un lugar a otro? ¿Cómo empacarlo y moverlo?

Autentificación: ¿Cómo asegurar que el agente es quien dice ser, y que está representando a quien el dice que está representando? ¿,Cómo saber que ha navegado por varias redes sin ser afectado por algún virus?

Privacidad: ¿Cómo asegurar que sus agentes mantienen su privacidad? ¿Cómo asegurar que alguien más no ha leído a su agente personal y no lo ha ejecutado para su propia ganancia? ¿Cómo asegurar que el agente no esta muerto y su contenido es un "coredumped"?

Seguridad: ¿Cómo protegerse contra los virus? ¿Cómo prevenirse contra la entrada de virus que se ciclen y consuman todos los ciclos de CPU?

Aspectos de pago: ¿Cómo pagará el agente por los servicios? ¿Cómo asegurar que no corra disparadamente y acumule una cuenta exorbitante en nuestro nombre?

Además se tiene lo siguiente:

Aspectos de rendimiento: ¿Cuáles serán los efectos de tener cientos, miles o millones de agentes en una WAN?

Servicios de Interoperabilidad/Comunicación: ¿Cómo proporcionar servicios de tipo directorio para máquinas locales y/o servicios específicos? ¿Cómo ejecutar un agente escrito en un lenguaje de agente en una máquina de agente escrita en otro lenguaje? ¿Cómo publicar o suscribir a servicios, o soportar la transmisión necesaria para algunas otras técnicas de coordinación?

Los cuestionamientos planteados en esta última sección, muestran el gran camino que falta por recorrer en el desarrollo de agentes móviles. Esta es una área reciente en la Ciencia de la Computación que permite explorar y seguir una gran variedad de líneas de investigación.

Pérez Lezama, C. V. 1998. **Agentes Móviles en Bibliotecas Digitales**. Tesis Maestría. Ciencias con Especialidad en Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas Puebla. Mayo.