AGENTES MOVILES Y REDES ACTIVAS PARA EL DESARROLLO DE REDES DE TELECOMUNICACIONES ORIENTADAS A OBJETOS

ERICK ROBERTO GONZALEZ RETIS
JOSE DAVID HERNANDEZ MELENDEZ

UNIVERSIDAD TECNOLOGICA DE BOLIVAR
FACULTAD DE INGENERIA ELECTRONICA
CARTAGENA DE INDIAS D.T. Y C.
2007

AGENTES MOVILES Y REDES ACTIVAS PARA EL DESARROLLO DE REDES DE TELECOMUNICACIONES ORIENTADAS A OBJETOS

ERICK ROBERTO GONZALEZ RETIS
JOSE DAVID HERNANDEZ MELENDEZ

DIRECTOR
ING. EDUARDO GOMEZ VÀSQUEZ

UNIVERSIDAD TECNOLOGICA DE BOLIVAR
FACULTAD DE INGENERIA ELECTRONICA
CARTAGENA DE INDIAS D.T. Y C.
2007

AGENTES MOVILES Y REDES ACTIVAS PARA EL DESARROLLO DE REDES DE TELECOMUNICACIONES ORIENTADAS A OBJETOS

ERICK ROBERTO GONZALEZ RETIS
JOSE DAVID HERNANDEZ MELENDEZ

Trabajo de grado presentado como requisito para obtener el certificado del Minor en Telecomunicaciones

DIRECTOR
ING. EDUARDO GOMEZ VÀSQUEZ
Magíster en Ciencias Computacionales

UNIVERSIDAD TECNOLOGICA DE BOLIVAR
FACULTAD DE INGENERIA ELECTRONICA
CARTAGENA DE INDIAS D.T. Y C.
2007

Nota de aceptación

_____

_____

_____

_____

Jurado

_____

Jurado

_____

Cartagena D..T. Y C., Septiembre de 2007

Señores
COMITÉ CURRICULAR
UNIVERSIDAD TECNOLOGICA DE BOLIVAR
La ciudad

Respetados señores:

Con toda la atención me dirijo a ustedes con el fin de presentarles a su consideración, estudio y aprobación la monografía tilutala **AGENTES MOVILES Y REDES ACTIVAS PARA EL DESARROLLO DE REDES DE TELECOMUNICACIONES ORIENTADAS A OBJETOS** como requisito para obtener el titulo de Ingeniero Electrónico

Atentamente,

_____

ERICK ROBERTO GONZALEZ RETIS

Cartagena D..T. Y C., Septiembre de 2007

Señores
COMITÉ CURRICULAR
UNIVERSIDAD TECNOLOGICA DE BOLIVAR
La ciudad

Respetados señores:

Con toda la atención me dirijo a ustedes con el fin de presentarles a su consideración, estudio y aprobación la monografía tilutala **AGENTES MOVILES Y REDES ACTIVAS PARA EL DESARROLLO DE REDES DE TELECOMUNICACIONES ORIENTADAS A OBJETOS** como requisito para obtener el titulo de Ingeniero Electrónico

Atentamente,

_____

JOSE DAVID HERNANDEZ MELENDEZ

Cartagena D.T. Y C., Septiembre de 2007

Señores
COMITÉ CURRICULAR
UNIVERSIDAD TECNOLOGICA DE BOLIVAR
La ciudad

Respetados señores:

A través de la presente me permito entregar la monografía titulada **AGENTES MOVILES Y REDES ACTIVAS PARA EL DESARROLLO DE REDES DE TELECOMUNICACIONES ORIENTADAS A OBJETOS** par su estudio y evaluación, la cual fue realizada por los estudiantes ERICK ROBERTO GONZALEZ RETIS y JOSE DAVID HERNANDEZ MELENDEZ, de la cual acepto ser su director

Atentamente,

_____

ING. EDUARDO GOMEZ VÀSQUEZ
Magíster en Ciencias Computacionales

AUTORIZACION

Yo, ERICK ROBERTO GONZALEZ RETIS, identificado con la cedula de ciudadanía numero 80.895.530 de Bogota, autorizo a la Universidad Tecnológica De Bolívar, par hacer uso de mi trabajo de monografía y publicarlo en el catalogo on-line de la biblioteca.

_____

ERICK ROBERTO GONZALEZ RETIS

AUTORIZACION

Yo, JOSE DAVID HERNANDEZ MELENDEZ, identificado con la cedula de ciudadanía numero de 73.211.951 Cartagena, autorizo a la Universidad Tecnológica De Bolívar, par hacer uso de mi trabajo de monografía y publicarlo en el catalogo on-line de la biblioteca.

_____

JOSE DAVID HERNANDEZ MELENDEZ

DEDICATORIA

Hoy sobresalen los frutos de los esfuerzos que mis padres han hecho por mi, hoy mas que nunca les agradezco con todas las fuerzas de mi corazón por ser una persona a imagen y semejanza de ellos, por lo que me hace sentir un orgullo profundo de ser su hijo.

A mi madre; por su amor, comprensión y por los consejos que me han llevado a tomar buenas decisiones y salir de malos momentos, ella es quien alimenta mi alma.

A mi padre; con muchos sacrificios nos ha sacado adelante y a pesar de la distancia siempre ha estado presente sin desampararnos por lo que se convierte en un modelo a seguir.

A mi hermana; por ser “la mama” de la casa y la consentida, que aumenta el lazo familiar y nos hace unirnos más.

A mi hermano; por que siempre esta conmigo, por su nobleza, por que siempre es capaz de sacar una sonrisa y me ha enseñado que la fortaleza esta en uno mismo.

A Edith que es como mi segunda madre, la que me brinda cariño y esta pendiente de mi bienestar

El más importante a dios; por que siempre ha estado con mi familia ayudándonos a cumplir nuestros sueños, nos llena de esperanza y sabiduría.

Y por ultimo a mis amigos, que de una u otra manera han contribuido en mi desarrollo personal, acompañándome en las buenas y en las malas.

Esto es para ustedes……..

Erick Roberto Gonzalez Retis

DEDICATORIA

Ante todo le doy gracias a dios por tenerme con vida, buena salud y estar presente en mis 22 años de vida y se que permanecerá presente en toda mi vida.

Con la entrega de este trabajo de grado estoy subiendo un escalón en la pirámide de la vida que nuca cesa por siempre hay que cumplir metas y con eso vamos avanzando mas. Esta pirámide no se puede subir solo, por siempre necesitaremos ayuda, compañía, apoyo para poder avanzar siempre. Le doy gracias a todos las personas que me han ayudado, acompañado y apoyado en lo que llevo de vida para cumplir mis metas seguir mejorando en todo los aspecto de la vida.

Mi papa le doy gracia por regalarme consejos de vida para superación personal, por su confianza, y por que siempre me respalda en todo proyecto que quiero iniciar. Y lo mas importante el es mi ejemplo a seguiré

Mi mama le doy gracia por estar siempre presente para escucharme día a día mis problemas, emociones etc. Mas que una madre una amiga.

Mi abuela, mi tía Tatiana, Kevin (primo hermano) a ellos le doy gracias por apoyarme estar conmigo en las malas y en las buenas. En especial a mi abuela por criarme y quererme como un hijo. También le doy gracia a toda mi familia por apoyarme en todo los momento de mi vida y ser un ejemplo para mi de saber cómo es una familia. A mi novia y mis amigo que están siempre que los necesito dispuesto a colaborarme igual yo para ellos; y más que amigos y novia son hermanos de la vida que a puesto dios en mi camino. También le doy gracias a mi compañero de monografía por confiar en mí, por apoyarme, y ante todo por ser mi amigo. A mi director de monografía por toda la colaboración prestada en estos meses de trabajo, también le doy a todos los profesores de la universidad por su enseñarme tantas cosas para la vida y que nunca dejaran de ser mis profesores……y queda para la

José David Hernández Meléndez

CONTENIDO

INTRODUCCION

Actualmente AGENTES MOVILES Y REDES ACTIVAS y su aplicabilidad en el contexto de las redes es el objetivo principal de esta monografía. Es de apreciar que las tecnologías de redes de datos han evolucionado considerablemente a lo largo de las tres últimas décadas. Sin embargo estos avances no han modificado sustancialmente la funcionalidad básica consistente en el transporte transparente de paquetes de datos de los usuarios entre dos puntos terminales de red. En los modelos de red tradicionales los datos de los usuarios son transmitidos de forma opaca, la red es insensible a los datos que transporta y éstos son encaminados de forma transparente y sin modificación, por lo que el proceso que se realiza dentro de la red es muy limitado.

Como evolución de los modelos de red tradicionales, la comunidad científica ha propuesto un nuevo modelo identificado por el término redes activas. La idea fundamental es añadir programabilidad a las redes, que constituyen una arquitectura de red en la que los nodos de la misma pueden realizar procesamiento “a medida” sobre los paquetes que los atraviesan, las redes activas producen un cambio en el paradigma de red; de nodos capaces exclusivamente de transportar octetos de forma pasiva, a nodos capaces de procesar los paquetes en cualquier capa de la pila de protocolos.

Las redes activas introducen el concepto de procesamiento específico de los paquetes en base a código móvil que se ejecuta en los nodos de la red. Esto quiere decir que los nodos de la red no son sistemas de procesamiento especializados en un protocolo de red, como sucede en la actualidad, sino que son plataformas de ejecución genéricas en las que se puede descargar dinámicamente código específico para el procesamiento de los distintos tipos de paquetes que se desee definir.

Finalmente, la inteligencia aplicada a redes y servicios es el resultado de un progreso tecnológico que da cabida a los agentes móviles inteligentes pertenecientes al campo de la programación orientado a objetos. La necesidad de usar de manera efectiva la información disponible en una red de computadoras, como puede ser Internet, requiere de algún mecanismo de organización y acceso a tal información. Una sistema basado en agentes móviles como solución a las necesidades especificadas es una solución que existe actualmente, capaces de realizar tareas como la búsqueda, filtrado, comercio electrónico, monitorización de procesos, control de procesos y recuperación de información dentro de una red de datos, que ayudan a los usuarios a desenvolverse por la red, evitándole gastar tiempo delante de un computador y haciendo un mejor uso de este.

Esta monografía muestra las diferentes tecnologías y filosofías de agentes móviles y redes activas, por lo que se divide en dos partes. Una parte dedicada a los agentes móviles inteligentes y sus aplicaciones, y la segunda parte dedicada a la arquitectura de las redes activas y sus desarrollos científicos.

1. SISTEMA DE AGENTES MOVILES INTELIGENTES

1.1. INTRODUCCIÓN A LOS AGENTES MÓVILES INTELIGENTES

La tecnología de agentes es un prometedora tecnología en el campo de la tecnología de la información (TI). Reúne varias tecnologías, como las aplicaciones de código distribuido o la programación orientada a objetos (OOP), que tienen el objetivo común de actuar autónomamente en nombre de otra entidad (por ejemplo, una persona). Existen diferentes razones para la proliferación de agentes y de las tecnologías asociadas, como es la generalización de Internet, que da la oportunidad y proporciona los medios técnicos para crear aplicaciones y servicios totalmente distribuidos (por ejemplo, el comercio electrónico, la obtención de información). La programación y la conceptualización basada en agentes ayuda a delegar funciones que facilitan la búsqueda de información optimizando los recursos de la red.

1.2. DEFINICIÓN DE AGENTES MÓVILES

La definición del termino agente es complejo, ya que los investigadores utilizan distintas acepciones de dicho término y no existe una definición académica ampliamente aceptada, por eso de acuerdo al entorno donde se desarrolle su significado varia, algunas de las definiciones que mas se acercan a un entorno de red computacional o que se consideran que tienen las características básicas son:

- "Los agentes son sistemas computacionales que habitan en entornos dinámicos complejos, perciben y actúan de forma autónoma en ese entorno, realizando un conjunto de tareas y cumpliendo objetivos para los cuales fueron diseñados."[1]

[1]Maes, P. Agents that reduce work and information overload. *Communicalions of the* ACM, 1994

- “Los agentes inteligentes son entidades programadas que llevan a cabo una serie de operaciones en nombre de un usuario o de otro programa, con algún grado de independencia o autonomía, empleando algún conocimiento o representación de los objetivos o deseos del usuario”[2].

Desde una perspectiva más amplia los agentes móviles inteligentes son programas que realizan tareas interactivas, dirigidas por la petición y los deseos de los usuarios tipo cliente servidor (figura 1), con la finalidad de alcanzar objetivos particulares para el cual fue diseñado, las acciones y percepciones de este vienen dadas por instrucciones de programas en algún lenguaje en particular. Poseen un grado de autonomía e independencia, en virtud del cual pueden realizar una serie de tareas sin que las personas u otros agentes los dirijan en cada paso que dan en su camino, además, posee las siguientes propiedades:

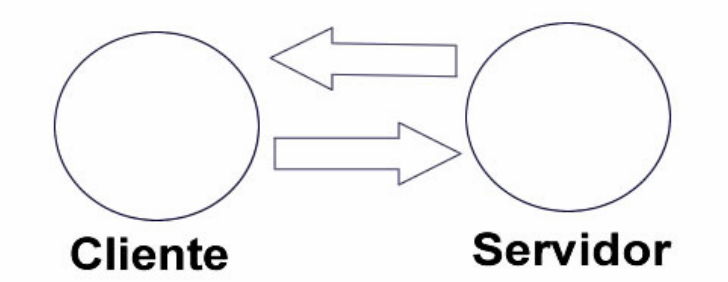

Figura 1. Modelo cliente servidor

- *Autonomía:* los agentes móviles operan sin la intervención directa de personas u otros, y tienen algún tipo de control sobre sus actuaciones y estado interno.
- H*abilidad social*: los agentes móviles interactúan con otros agentes (posiblemente humanos) por medio de un lenguaje de comunicación de agentes.
- R*eactividad*: los agentes móviles perciben el entorno o ambiente, (lo que representa la palabra físicamente, un usuario vía una interfaz de usuario, una colección de otros agentes, Internet, o quizás todos ellos

[2] IBM. Aglets, complementar con la pagina web http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm

combinados), y responde rápidamente a cambios que ocurren en dicho entorno.

- P*ro-actividad*: los agentes móviles no actúan simplemente en respuesta a su entorno, sino que son capaces de exhibir 'comportamiento dirigido hacia el objetivo', tomando la iniciativa.

1.3. CARACTERÍSTICAS DE LOS AGENTES MÓVILES[3]

La confusión respecto al término "agente" viene del hecho de que cada investigador asocia con los agentes un juego de características algo diferente, sin embargo de todas maneras, la mayor parte de las definiciones contienen algunas características comunes:

- *Capacidad de trabajar asincrónicamente y de manera autónoma*: la característica principal por la mayoría de los agentes especialmente inteligentes es que pueden funcionar con autonomía, y sin la intervención del ser humano.
- *Capacidad para cambiar su comportamiento según el conocimiento acumulado*: la capacidad de "aprender" puede ser la segunda característica más asociada con los agentes inteligentes. Por ejemplo, la "ayuda adaptativa" se ha caracterizado como una característica central en arquitecturas de interfaz inteligente.

  Similarmente, un uso frecuente de los agentes está en prevenir las necesidades de información del usuario buscando qué bases de datos mantienen un "perfil de interés" basándose en preferencias pasadas de búsqueda. Cualquier cambio de interés del usuario deberá ser tenido en cuenta por el agente.
- *Capacidad para tomar iniciativas*: un agente verdaderamente inteligente debe tener la capacidad para ejecutar tareas de acuerdo con su propia meta, separada de la del usuario. Los ambientes que conducen a esta

[3]Shoham, Y. Agent-oriented programming. *Artificial Intelligence, 1997*.

propuesta es cuando el agente “sabe” más que el usuario sobre una materia o sobre las estrategias a utilizar para resolver el problema.

- *Capacidad inferencial*: una característica frecuentemente asociada con agentes es la capacidad para realizar inferencias. Se define como la capacidad para ir más allá de las instrucciones concretas y específicas del usuario y resolver problemas utilizando alguna forma de abstracción simbólica. En algunos casos, estas abstracciones estarán en forma de reglas. En otros casos, las inferencias pueden estar basadas en razonamientos probabilísticos.
- *Conocimiento anterior de metas generales y métodos preferidos:* este nivel de comprensión es el requerido para las versiones más sofisticadas de agentes inteligentes. Un agente inteligente no solamente deber ser capaz de generar su propia meta prioritaria, sino que debe comprender también la meta del usuario para el que actúa como agente.
- *Movilidad:* es la capacidad que tiene un agente para iniciar su ejecución en cualquier sitio, e irse a otra posición llevándose datos y códigos donde continuar su ejecución. De hecho, la “movilidad” tiene dividida a toda la comunidad de agentes en dos escuelas: la primera que piensan que la movilidad no es un aspecto esencial (la comunidad de multiagentes DAI, esencialmente académicos) y los que claman que los agentes móviles son el futuro de los agentes (comunidad de programadores orientados a objetos). La verdad está en un punto intermedio. Es claro que algo que puede ser hecho con los agentes móviles, también puede hacerse con técnicas de programación convencionales, sin embargo la programación basada en agentes móviles trae un nuevo paradigma que realza la flexibilidad y eficacia del diseño y ejecución de las aplicaciones distribuidas (reducción de ancho de banda).
- *Comunicación:* es un aspecto crucial para un agente, que, por definición, tiene que comunicarse durante su vida ya sea con la entidad en nombre de la que actúa o con los otros agentes con los cuales necesita

colaborar. El nivel de interacción/comunicación depende en gran medida del nivel de conocimiento del agente, que se mueve entre “datos”, “información” y “conocimiento”. Los datos corresponden con el nivel más básico, que puede ser el contenido de una variable, un archivo, etc. La información está mucho más estructurada y puede consistir en la descripción de parte de un equipo de telecomunicación, un documento XML (eXtensible Markup Language) como un formulario de pedido en aplicaciones de agentes de comercio electrónico, o un correo electrónico. El conocimiento consiste en información estructurada y re-glas lógicas. Se pueden asociar con cada nivel de información los lenguajes de comunicación y protocolos adecuados. Por ejemplo, un agente puede usar FTP (Protocolo de Transferencia de Archivos) para enviar un archivo, o llamar al método de otro agente para obtener o transmitir información. El http (HyperText Transfer Protocol) se puede utilizar para transmitir descripciones XML. Finalmente, lenguajes ba-sados en la Speech Acts Theory , como KQML/KIF (Knowledge refo and Manipulation Language/Knowledge Interchange Format), el lenguaje de comunicación de agentes FIPA (Formation for Intelligent Physical Agents) pueden ser usados para intercambiar información.

- *Personalidad:* algunos investigadores han encontrado atractivo adjuntar características humanas a los agentes, como la personalidad. La suposición parece ser el problema a resolver y los requerimientos interactivos de comunicación pueden llegar a ser muy grandes y complejos, únicamente un humano como entidad podría gestionar el problema. De forma similar a los humanos, la comunicación de algunos tipos de información va a estar estrechamente relacionada con la personalidad.

Entonces un agente inteligente, llega ha ser una complicada mezcla de

características. Un programa de un agente inteligente, para poder realizar sus actividades de forma autónoma, necesitará ser un experto en el tema, tener conocimiento de métodos y estrategias para dividir el problema en subtareas, habilidades inferenciales y capacidades para adoptar medidas.

1.4. TIPOS DE AGENTES MÓVILES[4]

Dependiendo del contexto en el cual se ubican los Agentes móviles Inteligentes son varias las taxonomías que existen, entre las cuales se destacan las siguientes: basadas en el mapeo de las percepciones a las acciones y basadas en el tipo de aplicación.

1.4.1. Basada en el mapeo de las percepciones a las acciones

- *Agentes de reflejo Simple:* los agentes de reflejo simple operan siguiendo las instrucciones que le proporcionan un conjunto de reglas del tipo condición-acción, las cuales le permiten al agente establecer las conexiones entre las percepciones y las acciones. Mediante un chequeo entre del estado actual del mundo (ambiente) el cual es captado por los sensores, el agente busca en el conjunto de reglas condición-acción, cual es la regla que coincide con la percepción captada, para luego ejecutar la acción.

  Este agente funcionará correctamente solo si se toma la decisión adecuada basándose en la percepción del ambiente en un instante dado. Este tipo de agente es fácil de implantar pero el ámbito de aplicación es bastante limitado. En la figura 2 se observa la estructura de un agente de reflejo simple.

[4]Tomado de bibliografía [2]. Capitulo 1.5, tipos de agentes.

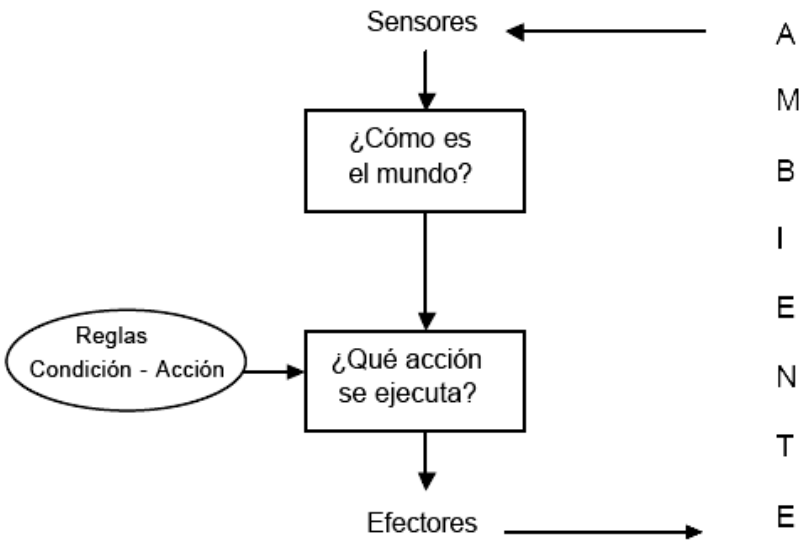

Figura 2. Agente de reflejo simple.

- *Agentes informados de lo que pasa*: como se mencionó anteriormente el agente de reflejo simple funcionará correctamente solo si se toma la decisión adecuada basándose en la percepción del ambiente en un instante dado. En un caso particular en donde el agente solo considera la situación actual, es posible que si la acción solo se basa en lo que percibe sin considerar lo que sucedió en momentos anteriores, no se alcance las metas con satisfacción total.

  Es por eso que para prevenir este tipo de casos se dota al agente de capacidad de decidir la acción a ejecutar considerando y evaluando acciones o percepciones anteriores en función de una visión global del mundo (ambiente). Específicamente en estos casos el agente necesita actualizar la información del estado interno que le permita discernir entre estados del mundo que generan la misma entrada de percepciones pero que, sin embargo, son totalmente distintos. Esto significa que para cada uno de los estados se necesitan acciones diferentes.

Para dotar al agente de capacidad de discernir se debe codificar conocimiento de dos clases, primero conocimiento acerca de cómo evoluciona el ambiente y conocimiento de cómo las acciones del agente afectan el mundo. En la figura 3 se observa la estructura del agente y como éste combina las percepciones prevalecientes con el estado interno anterior para generar la descripción actualizada del estado actual.

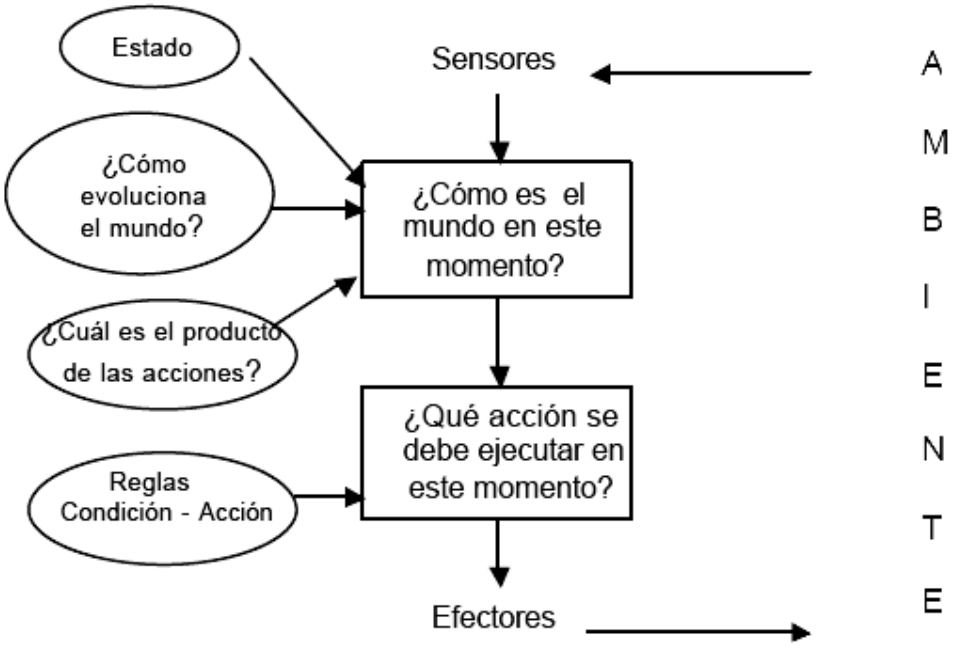

Figura 3. Agente informado de lo que pasa

- *Agentes basados en metas:* para decidir lo que hay que hacer no siempre es suficiente tener información acerca del estado que prevalece en el ambiente, es necesario también disponer de información acerca de las metas que se pretenden alcanzar. Al disponer de información acerca del estado prevaleciente en el ambiente, un programa de agente podrá combinar esta información con el resultado que producirán las posibles acciones que se emprendan y de esta manera elegir aquellas que permitan alcanzar las metas de una manera eficiente y eficaz. Es posible que la escogencia de la acción que satisfaga el alcance de la meta sea

una elección sencilla, pero en otras oportunidades el agente deberá ejecutar operaciones de búsqueda y planificación de las metas.

Es importante destacar que las decisiones acerca de cual acción se debe emprender difieren del comportamiento basado en reglas del tipo condición –acción, ya que las decisiones se deben tomar en función de satisfacer situaciones tales como, ¿qué sucederá si se toma la acción x o la acción y? O ¿esto será satisfactorio?, entre otras. En la figura 4 se esquematiza la estructura de un agente basado en metas.

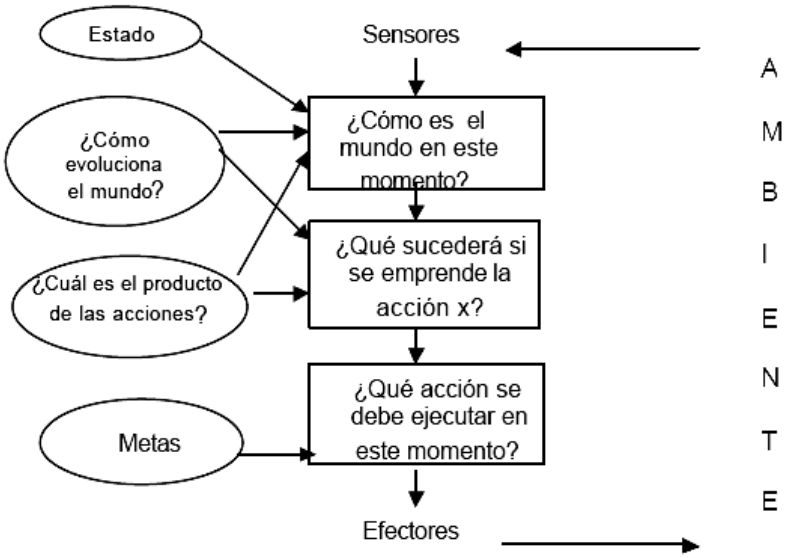

Figura 4. Agente basado en metas

1.4.2. Basada en el tipo de aplicación

- *Agentes de interfaz:* la finalidad de este tipo de agente es proporcionar información a los usuarios en los sistemas informáticos que dada su complejidad poseen una alta carga de información. Se caracteriza por su capacidad de hacer comprensible las interfaces.
- *Agentes de sistemas:* son agentes que permiten realizar inventario de hardware, interpretar eventos de red, manipular dispositivos de respaldo

y almacenaje, detectar virus, etc, todo esto sobre sistemas complejos como es el caso de los sistemas distribuidos.

- *Agentes consejeros:* este tipo de agente proporciona consejos a los usuarios en el caso de utilización de herramientas, o en sistemas de diagnostico o ayuda.
- *Agentes de navegación:* estos agentes se utilizan para navegar sobre sistemas conectados en red, algunas de sus funciones principales son el recordar sitios y direcciones de interés de manera automática.
- *Agentes de monitoreo:* los agentes de monitoreo proporcionan información de manera eficaz y oportuna a los usuarios, cuando ocurre un evento.
- *Agentes de recomendación:* este agentes utilizan bases de datos con información acerca de preferencias de un grupo de usuarios acerca de un ítem o tópico en particular (información, productos, etc.), basando su recomendación en analogías o match con otros usuarios de perfil similar a un usuario dado.
- *Agentes de información:* son agentes software que gestionan el acceso a múltiples fuentes de información, heterogéneas y geográficamente distribuidas. Una de sus principales tareas consiste en realizar búsquedas activas de información relevante, no local, para sus usuarios u otros agentes. Esta tarea incluye la recuperación, análisis, manipulación e integración de la información disponible en múltiples fuentes de información autónomas.
- *Agentes de recuperación de información:* estos agentes se especializan en búsqueda y posterior recuperación de información, ejecutándose sobre grandes bases de datos, bases de conocimiento o bases de documentos.

1.5. ARQUITECTURA DE LOS AGENTES MÓVILES

La estructura tradicional de los agentes consiste de un programa de agente que se ejecuta sobre una arquitectura en donde se consideran cuestiones para la construcción de sistemas informáticos que satisfagan las propiedades especificadas por las teorías de agente. A continuación se presentan las definiciones de una arquitectura de agente:

"Una metodología particular para construir agentes. Especifica cómo puede descomponerse el agente construyendo un conjunto de componentes modulares y cómo deben realizarse estos módulos para que interactúen. El conjunto total de módulos y sus interacciones tienen que dar una respuesta a la pregunta de cómo los datos de los sensores y el estado interno actual del agente determinan las acciones y el futuro estado interno del agente. Una arquitectura comprende técnicas y algoritmos que soportan esta metodología." [5]

"Una colección específica de módulos de software o hardware, típicamente designados por cajas con flechas que indican el flujo de datos y el control entre los módulos. Una vista más abstracta de una arquitectura es como una metodología general para diseñar descomposiciones modulares particulares para tareas particulares."[6]

El enfoque clásico en la construcción de agentes es considerarlos como un tipo particular de sistema basado en el conocimiento. Este paradigma se conoce como *Inteligencia Artificial simbólica*.

Las arquitecturas de agente, representan el paso de la especificación a la implementación. El reto será construir sistemas que satisfagan las propiedades que se han especificado teóricamente a los agentes, qué estructuras de software y/o hardware son apropiadas.

En los agentes se puede encontrar una gran variedad de arquitecturas, una primera clasificación se puede realizar según el acceso: que todas las capas del agente tengan acceso a sensores y actuadores (horizontales) o que sólo la

[5]Maes, P. The agent network architecture (ANA). *SIGART Bulletin,1991*
[6]Kaelbling, L. P. A situated automata approach to the design of embedded agents. *SIGART Bulletin,1991*

capa más baja tenga acceso a sensores y actuadores (verticales). Las arquitecturas *horizontales* ofrecerán la ventaja del paralelismo entre capas a costa de un alto conocimiento de control para coordinar las capas, mientras que las *verticales* reducen este control a costa de una mayor complejidad en la capa que interactúa con los sensores.

También se pueden clasificar las arquitecturas, según el tipo de procesamiento empleado, en *deliberativas*, *reactivas* e *híbridas*.
Se distinguen los siguientes tipos principales de arquitecturas deliberativas o simbólicas: *arquitecturas intencionales* y *arquitecturas sociales*.

Los agentes intencionales se distinguen por ser capaces de razonar sobre sus creencias e intenciones. Se pueden considerar como sistemas de planificación que incluyen creencias e intenciones en sus planes.

Los agentes sociales se pueden definir como agentes intencionales que mantienen, además, un modelo explícito de otros agentes y son capaces de razonar sobre estos modelos.

Dentro de las arquitecturas intencionales, cabe destacar aquellas que han tomado como punto de partida la teoría de agentes BDI en su implementación, representando explícitamente las actitudes intencionales de los agentes. Estos sistemas también suelen utilizar planificación para determinar qué acciones deben llevar a cabo pero, a diferencia de los agentes planificadores, emplean planes en los que se comprueban creencias, deseos e intenciones. Las creencias son el conocimiento que el agente tiene sobre sí mismo y su entorno. Los deseos son objetivos del agente a largo plazo que desea cumplir. Como normalmente no puede cumplir todos los objetivos a la vez, ya que tiene unos recursos limitados, se introducen las intenciones, que son los objetivos que en cada momento intenta cumplir el agente. Normalmente también se introduce el concepto de planes, que permiten definir tanto las intenciones como los planes

que un agente está realizando en un momento dado. Hay muchas arquitecturas de agentes que siguen el modelo BDI, como IRMA y PRS.

Los agentes sociales pueden clasificarse en dos grandes grupos: *agentes intencionales* cuya arquitectura ha sido aumentada para abordar el razonamiento sobre otros agentes, como COSY, GRATE y DA-Soc; y arquitecturas que siguiendo la IAD clásica han prestado más atención a los aspectos cooperativos (cuándo, cómo y con quién cooperar), sin modelar necesariamente las intenciones de los agentes, como Archon, Imagine, Coopera y MAST.

Las arquitecturas deliberativas pueden clasificarse como horizontales porque los estímulos recibidos del exterior son procesados en varias capas de diferente nivel de abstracción y al final el nivel superior decide qué acciones hay que llevar a cabo y las realiza directamente o se lo indica a las capas inferiores.

1.5.1. Arquitectura deliberativa

Son del tipo de la IA simbólica, basada en la hipótesis de los sistemas de símbolos físicos enunciada por Newell y Simons[7], según la cual un sistema de símbolos físicos capaz de manipular estructuras simbólicas puede exhibir una conducta inteligente. Para poder trabajar en el nivel de Conocimiento de Newell, el problema será cómo describir los objetivos y los medios para satisfacerlos, y cómo realizar la traducción del nivel de conocimiento al nivel simbólico.

Las arquitecturas deliberativas de agentes suelen basarse en la teoría clásica de planificación en inteligencia artificial: dado un estado inicial, un conjunto de operadores/planes y un estado objetivo, la deliberación del agente consiste en determinar qué pasos debe encadenar para lograr su objetivo, siguiendo un enfoque descendente (figura 5). Como ejemplo de arquitectura cuyo

[7]Newell, A. and Simon, H. A. Computer science as empirical enquiry. *Communications of the ACM,* 1976.

componente principal es un planificador cabe citar los Softbots[8].

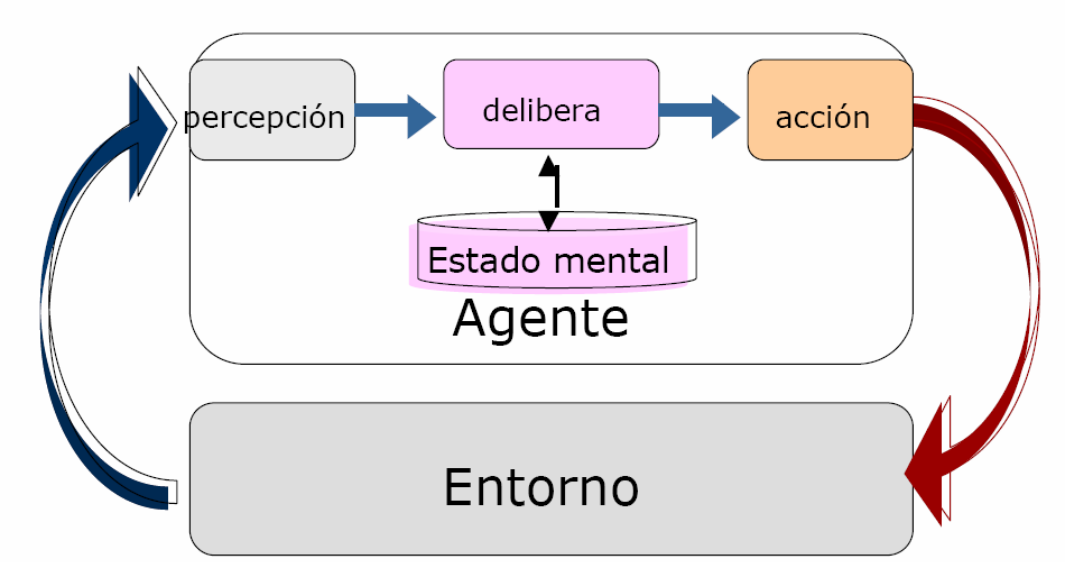

Figura 5. Planificación clásica IA

Como se menciono anteriormente, el fundamento sobre el que se basa el paradigma de la Inteligencia Artificial simbólica es la hipótesis de símbolo físico, formulada por Newell y Simon (1976). Un sistema de símbolo físico se define como un conjunto físicamente realizable de entidades físicas (símbolos) que pueden combinarse para formar estructuras, y que es capaz de ejecutar procesos que operan sobre esos símbolos de acuerdo con un conjunto de instrucciones codificado simbólicamente. La hipótesis del sistema de símbolo físico dice que tal sistema es capaz de exhibir una conducta inteligente.
Se puede decir que es un paso que va desde la noción de sistema de símbolo físico al de un *autómata de proceso sentencial*, o *agente deliberativo*.

Entonces se definirá un agente deliberativo o arquitectura de agente como uno que contiene un modelo simbólico del mundo explícitamente representado, y en el que las decisiones (por ejemplo, sobre qué acciones ejecutar) se realizan por medio de razonamientos lógicos, basándose en la manipulación simbólica. La

[8]Complementar con la pagina web http:// www2.ing.puc.cl/~dcolle/publicaciones/agentes/agentes.htm

idea de agentes deliberativos con base en el razonamiento puramente lógico es muy llamativa, para conseguir un agente que cumpla alguna teoría de agencia, es de suponer que basta simplemente con dar la representación lógica de la teoría y usarla para realizar un teorema de prueba[9]. Si se desea construir un agente de esta manera, en su planteamiento se encontraran por lo menos dos problemas importantes a resolver:

1) El problema de la transducción: traducir el mundo real en una exacta y adecuada descripción simbólica, en el tiempo necesario para que esa descripción sea útil.
2) El problema de la representación/razonamiento: cómo representar simbólicamente información sobre entidades y procesos complejos del mundo real, y cómo conseguir agentes que razonen con esa información en el tiempo necesario para que los resultados sean útiles.

El primer problema ha llevado a la realización de trabajos sobre visión artificial, reconocimiento del habla, aprendizaje, etc. El segundo problema ha conducido a trabajos sobre la representación del conocimiento, razonamiento automatizado, planificación automática, etc. Sin embargo la cantidad de investigadores que se han dedicado a resolverlo no han podido llegar a culminar el trabajo. Algunos problemas aparentemente triviales, tales como el sentido común, tienen una resolución sumamente difícil[10]. El problema parece ser la dificultad de un teorema de prueba, aún en lógicas muy simples, y la complejidad de los algoritmos de manipulación de símbolos en general. Así, la idea de construir "agentes como probadores de teoremas", que sería un punto de vista de agencia extremadamente visionaria, aunque que sea muy atractiva en teoría, parece que, por ahora, es casi imposible de llevar a cabo en la práctica. Quizás lo más inquietante para la IA simbólica es que muchos algoritmos de manipulación de símbolos interesantes son intratables. Resulta difícil construir algoritmos de manipulación de símbolos útiles que puedan garantizar que terminen con resultados útiles en un límite de tiempo que sea

[9]Shardlow, N. Action and agency in cognitive science. Department of Psychlogy, University of Manchester, 1990.
[10]Guha, R. V. and Lenat, D. B. Enabling agents to work together. *Communications of the ACM,1994*

aceptable. Estos algoritmos parecen ser necesarios para agentes que deban operar en cualquier mundo real, con un dominio de tiempo limitado.

1.5.2. Arquitectura Reactiva

Los numerosos problemas que lleva asociado utilizar una representación simbólica del conocimiento, han conducido al estudio de modelos más efectivos de representación del conocimiento. Las arquitecturas reactivas se caracterizan por no tener como elemento central de razonamiento un modelo simbólico, y por no utilizar razonamiento simbólico complejo.

Las arquitecturas reactivas cuestionan la viabilidad del paradigma simbólico y proponen una arquitectura que actúa siguiendo un modelo estímulo-respuesta. Las arquitecturas reactivas no tienen un modelo del mundo simbólico como elemento central de razonamiento y no utilizan razonamiento simbólico complejo, sino que siguen un procesamiento ascendente, para lo cual mantienen una serie de patrones que se activan con ciertas condiciones de las percepciones y tienen un efecto directo en las acciones. Esta discusión entre mantener una representación explícita del modelo o no, no es una discusión específica del campo de los agentes sino de la inteligencia artificial en general; de hecho las primeras arquitecturas de agentes reactivos se basan en los planificadores reactivos. Las principales arquitecturas reactivas son:

- Reglas situadas: la implementación más sencilla de reactividad consiste en definir el comportamiento con reglas del tipo situaciones percibidas entonces acciones específicas.
- Arquitecturas de subsunción (subsumption) y autómatas de estado finito: Esta arquitectura se basa en el hecho de que se puede generar un comportamiento inteligente sin utilizar propuestas del modelo simbólico, y en el hecho de que la inteligencia es una propiedad emergente de ciertos sistemas complejos. Las arquitecturas de subsunción manejan

jerarquías de tareas que definen un comportamiento. Suelen estar organizadas en jerarquías de capas, de menor a mayor nivel de abstracción. La mayor aplicación de este tipo de arquitecturas se ha centrado en el desarrollo de controladores en robótica. Los robots se pueden considerar como agentes reales (no software) que actúan en un entorno cambiante. Precisamente, la necesidad de actuar en un entorno impredecible y altamente cambiante dificulta la adopción de una arquitectura deliberativa, ya que las necesidades de replanificación y de continua adaptación del plan a la realidad hace inservible dicha arquitectura.

- Redes neuronales: la capacidad de aprendizaje de las redes neuronales también ha sido propuesta en algunas arquitecturas formadas por redes que son capaces de realizar una función concreta, como, por ejemplo, evitar colisiones en la red.

Las arquitecturas reactivas pueden clasificarse como verticales porque los estímulos recibidos del exterior son procesados por capas especializadas que directamente responden con acciones a dichos estímulos y pueden inhibir las capas inferiores.

1.5.3. Arquitectura Híbrida

Dado que algunos investigadores opinan que para la construcción de agentes no es del todo acertado utilizar una arquitectura totalmente deliberativa, o totalmente reactiva, se han propuesto sistemas híbridos que pretenden combinar aspectos de ambos modelos. Una propuesta puede ser la construcción de un agente compuesto de dos subsistemas: uno deliberativo, que utilice un modelo simbólico y que genere planes en el sentido expuesto anteriormente, y otro reactivo, centrado en reaccionar a los eventos que tengan lugar en el entorno y que no requiera un mecanismo de razonamiento complejo. Por su propia naturaleza, estas arquitecturas son propicias para una

estructuración por capas, que puede ser: (a) vertical, sólo una capa tiene acceso a los sensores y actuadores; (b) horizontal, todas las capas tienen acceso a los sensores y a los actuadores. De la misma forma que las arquitecturas de subsunción, las capas se organizan jerárquicamente con información sobre el entorno a diferentes niveles de abstracción. La mayoría de las arquitecturas encuentran suficiente tres niveles jerárquicos:

1. *Reactivo (de más bajo nivel):* se toman decisiones acerca de las acciones por hacer en base a los estímulos recibidos del entorno en tiempo real. Suele estar implementado como arquitecturas de subsunción.

2. *Conocimiento (nivel intermedio):* se olvida de los datos que recopila el agente y se centra en el conocimiento que él posee del medio, normalmente con la ayuda de una representación simbólica del medio.

3. *Social (capa de más alto nivel):* maneja aspectos sociales del entorno, incluyendo tanto información de otros agentes, como deseos, intenciones, etc. El comportamiento global del agente viene definido por la interacción entre estos niveles.

Esta interacción de niveles jerárquicos cambia de una arquitectura a otra. Uno de estos ejemplos es la arquitectura TOURINGMACHINES[11], donde cada nivel está continuamente sugiriendo qué acción realizar y existe un sistema de control para garantizar el correcto funcionamiento del agente.

La arquitectura consiste en un subsistema de acción y otro de percepción, que interactúa directamente con el entorno del agente, y tres capas de control, incrustadas en una estructura de control, que media entre las capas (figura 6).

[11]Ferguson, I. A. *TouringMachines* Una arquitectura para agentes móviles dinámicos y racionales. Tesis PhD, 1992.

Cada capa es un proceso independiente, que produce actividad y se ejecuta concurrentemente.

La *capa reactiva* genera guías de acción en respuesta a los sucesos que ocurren demasiado rápido como para que otras capas puedan tratarlos. Se implementa como un conjunto de reglas de situación-acción, al estilo de la arquitectura de subsunción.

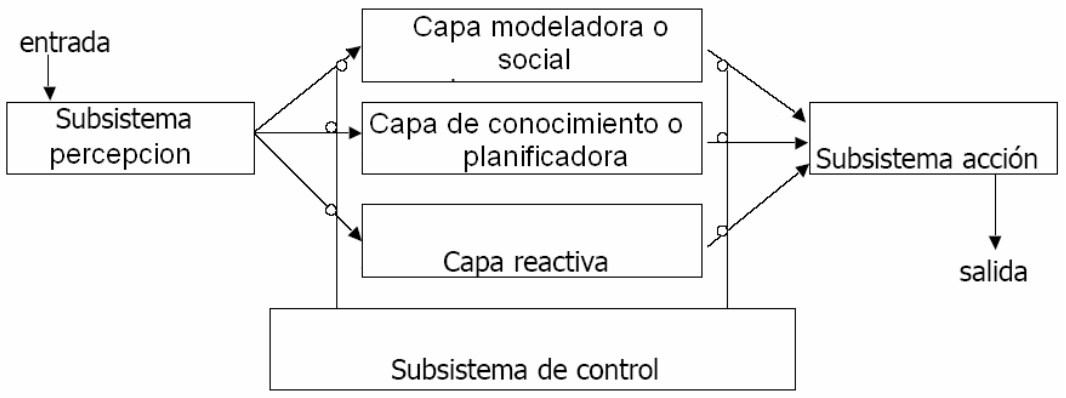

Figura 6. Arquitectura Touringmachines

La *capa planificadora* construye planes y selecciona las acciones a ejecutar a fin de lograr las metas del agente. Esta capa consiste en dos componentes: un planificador, y mecanismo de atención. El planificador integra la ejecución y generación de planes, y usa una librería de planes parcialmente elaborados, junto con un mapa topológico del mundo, a fin de construir planes que lleven a cabo la meta principal del agente. El propósito del mecanismo de atención está en limitar la cantidad de información con la que el planificador debe tratar, y mejorar su eficiencia. Esto se hace posible filtrando la información irrelevante de su entorno.

La *capa de modelado* contiene representaciones simbólicas del estado cognitivo de otras entidades en el entorno del agente. Estos modelos se

manipulan para identificar y resolver conflictos de meta, situaciones donde un agente no puede lograr sus metas, como resultado de interferencias inesperadas.

Las tres capas se comunican entre sí mediante el traspaso de mensajes, y se encuentran incrustadas en una estructura de control. El propósito de esta estructura es mediar entre las capas, y en particular, tratar con propuestas conflictivas de acción desde capas diferentes. La estructura de control para ello utiliza reglas de control.

La arquitectura INTERRAP (figura 7) también actúa del mismo modo, contiene 3 capas verticales con 2 pasos; una capa de comportamiento (reactivo), una capa de planificación (conocimiento) y una capa de cooperación (interacción social), con más de una base de conocimiento que también se organiza por capas.

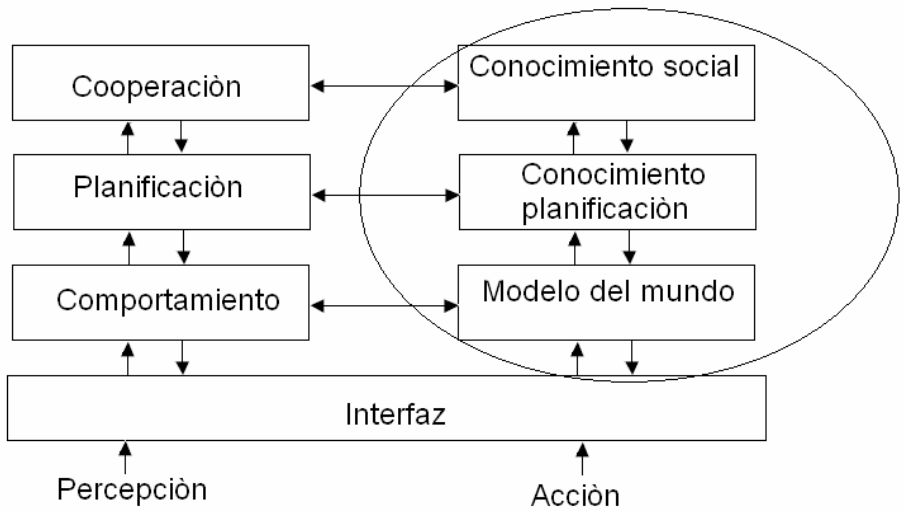

Figura 7. Arquitectura Interrap

1.5.4. Arquitectura BDI (Beliefs-Desires-Intentions)

La arquitectura BDI (Belief, Desire, Intention) caracterizada porque los agentes que la implementan están dotados de los estados mentales de Creencias, Deseos e Intenciones, ha sido el modelo más difundido y el más estudiado dentro de los modelos de razonamiento de agentes. Hay varias razones para que esto haya ocurrido, pero quizás la más convincente es que el modelo BDI combina elementos interesantes: un apreciable modelo filosófico de razonamiento humano fácil de comprender, un número considerable de implementaciones, como por ejemplo sistemas de control de procesos, procesos de decisión en negocios, etc. Y el desarrollo de una semántica lógica abstracta y elegante, la cual ha sido aceptada por la comunidad científica.
Las intenciones del agente juegan un importante papel en el razonamiento práctico como:

- Dirigen el razonamiento basado en medios y fines.
- Restringen las deliberaciones futuras.
- Persisten.
- Influencian las creencias sobre las que se basará el futuro razonamiento práctico.

Cada cierto tiempo el agente deberá replantearse sus intenciones, abandonado aquellas que considera que no va a alcanzar, aquellas que ya ha alcanzado y aquellas cuya justificación ha desaparecido

El agente tipo BDI presenta diferentes comportamientos o cualidades que son:

- *Comportamiento dirigido por objetivos:* es el agente que no reconsidera suficientemente a menudo sus intenciones (atrevido).

- *Comportamiento dirigido por eventos (reactivo):* el agente que continuamente reconsidera sus intenciones dedicando así un tiempo insuficiente a su consecución (precavido).

Es difícil encontrar un equilibrio entre ambos comportamientos ya que cada uno posee diferentes características que lo identifican en un determinado entorno, en entornos estáticos el comportamiento dirigido por objetivos es más adecuado y en los entornos dinámicos es necesario considerar cierta reactividad.

Algunos de los componentes que posee un agente BDI son:

- Conjunto de creencias actuales que tiene el agente acerca de su entorno.
- Función de revisión de creencias que actualiza las creencias en base a las percepciones.
- Función de generación de opciones (deseos) a partir de sus creencias e intenciones.
- Conjunto de opciones actuales.
- Función filtro correspondiente al proceso de deliberación, determina las nuevas intenciones en función de sus creencias, deseos e intenciones.
- Conjunto de intenciones actuales.
- Función de selección de acciones que determina la acción a ejecutar a partir de las intenciones actuales.

En la figura 8 se esquematiza la arquitectura de un agente BDI.

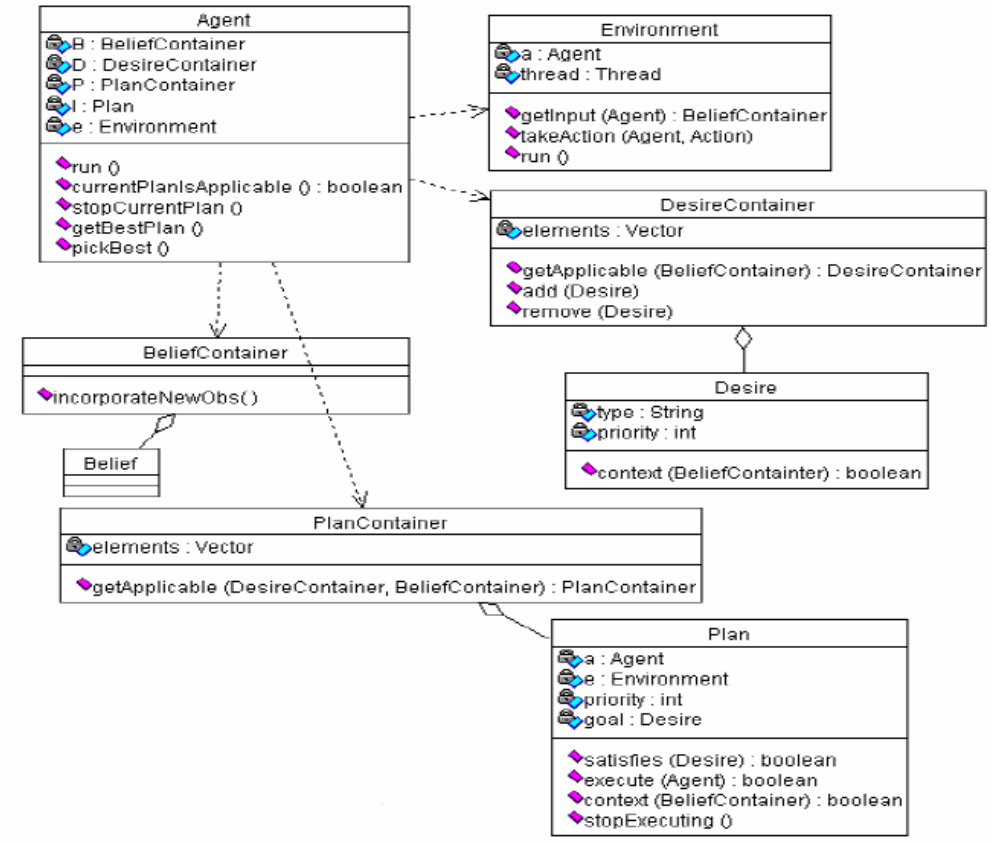

Figura 8. Arquitectura BDI

2. LENGUAJES PARA AGENTES MOVILES [12]

Mientras la tecnología de agentes siga avanzando es de esperar en el medio la aparición de diversas herramientas de software para el diseño y construcción de sistemas basados en agentes. Por lenguaje de agente, se entiende como un sistema que permite programar sistemas de hardware y/o software en términos o en base de algunos de los conceptos de agente anteriormente expuestos. Dentro del cual se puede distinguir dos tipos principales de lenguajes de programación:

1) *Lenguajes de agentes de propósito general*: lenguajes destinados a programar agentes genéricos utilizables en cualquier aplicación.

[12] Tomado de bibliografía [3]. Capitulo 1.6, comunicación entre agentes.

2) *Lenguajes de agentes específicos*: lenguajes para un tipo de agentes específicos, por ejemplo los lenguajes para agentes móviles Telescript o Agent-Tcl.

Se pueden distinguir los siguientes *niveles* en la programación de agentes:

- *Lenguajes de programación de la estructura del agente*: permiten programar las funcionalidades básicas para definir a un agente: funciones de creación de procesos y funciones de comunicación entre agentes.
- *Lenguajes de comunicación de agentes*: definición del formato de los mensajes de intercambio, de las primitivas de comunicación y de los protocolos disponibles.
- *Lenguajes de programación del comportamiento del agente*: permiten definir el conocimiento del agente: conocimiento inicial (modelo de entorno, creencias, deseos, objetivos), funciones de mantenimiento de dicho conocimiento (reglas, planes, etc.) y funciones para desarrollar habilidades (programación de servicios).

2.1. LENGUAJES DE PROGRAMACIÓN DE LA ESTRUCTURA DEL AGENTE

Este nivel de programación normalmente sólo es utilizado por los desarrolladores de una plataforma de desarrollo de agentes. Los lenguajes empleados suelen ser lenguajes de propósito general (C, C++, Java, Lisp, Prolog, etc.) o lenguajes específicos (p. ej. April dentro del proyecto IMAGINE, sobre Prolog/C y CUBL (Concurrent Unit Based Language) dentro del proyecto DAISY, sobre CLOS/C++).

2.1.1 Agentes móviles por medio de JAVA

Java es un lenguaje de programación orientado a objetos que fue desarrollado por Sun Microsystems a principios de los años de 1990. Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

El lenguaje en sí mismo toma mucha de su sintaxis de otros lenguajes como C y C++, pero tiene un modelo de objetos más simple, El lenguaje Java se creó para que cumpliera con cinco objetivos principales:

1. Debería usar la metodología de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos (Applets, Servlets, Jini).
3. Debería incluir por defecto soporte para trabajo en red (sockets, datagramas, Java RMI, FTP, http, URL).
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura (Verificador de bytecode, Java Security, Java Card).
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos (C++, Visual FoxPro, Delphi).

Características JAVA hace viable la implementación de agentes móviles, pero quizás la segunda ha contribuido más a su desarrollo, lo que significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware (figura 9). Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como “bytecode” (específicamente Java bytecode) que son instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está “a medio camino” entre el

código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran librerías adicionales para acceder a las características de cada dispositivo. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT (Just In Time).

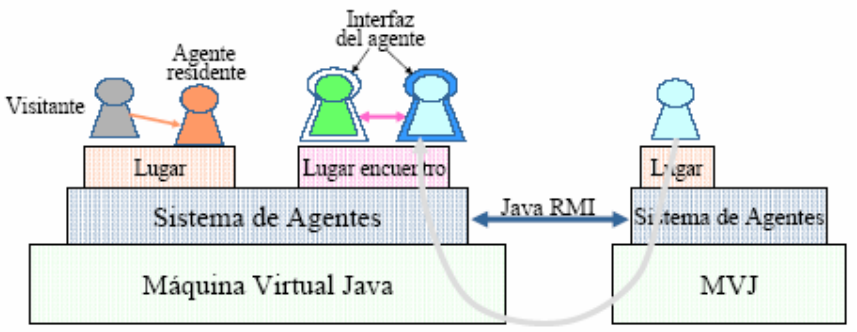

Figura 9. Agentes móviles con Java

Una de las aplicaciones de agentes móviles basados en java son los Aglets[13], desarrollados por IBM. Proveen las capacidades básicas requeridas para la movilidad. Un aglet refleja el modelo de applet en Java pero brindándole la propiedad de movilidad. Un aglet también puede ser un agente móvil porque soporta las ideas de ejecuciones autónomas y ruteo dinámico sobre sus funciones. Tiene las siguientes características:

- Un esquema global único para agentes (Modelo de navegación /seguridad)
- Un itinerario de viaje, para la especificación de patrones complejos de viajes con múltiples destinos y manejos de fallas automáticos (Modelo de navegación)

[13]Maestras P, Inteligencia artificial ;Agentes inteligentes, curso Phd, 2005

- Un mecanismo white−board permitiendo que múltiples agentes colaboren y compartan información asíncronamente (Modelo de comunicación)
- Un esquema de transmisión de mensajes que soporta una unión asíncrona desahogada tan bien como una comunicación síncrona entre agentes (Modelo de comunicación)
- Una carga de clases dinámicamente que permite que el código Java de los agentes y la información de su estado viajen a través de la red (Modelo de navegación)
- Un contexto de ejecución que proporciona un ambiente independiente del sistema actual sobre el cual se están ejecutando (Modelo computacional)

Desafortunadamente el modelo de ciclo de vida de un aglet es muy simple y algunas de sus desventajas se mencionan a continuación:

- Soporte inadecuado de control de recursos
- No tiene protección de referencias
- No provee ayuda para la preservación y reanudación del estado de ejecución.

Los aglets utilizan el Protocolo de Transferencia de Agentes independiente de la plataforma para transferir agentes entre redes de computadoras.

2.2. LENGUAJES DE COMUNICACIÓN DE AGENTES

En este tipo de lenguajes de comunicación se puede distinguir dos clases:

- *Procedimentales*: se basan en el intercambio de directivas procedimentales, es decir, un agente recibe un mensaje que implica la ejecución de un procedimiento, pudiendo transmitir además de comandos, programas enteros que permiten a los agentes alcanzar sus

objetivos. Las ventajas de estos lenguajes procedimentales es que son sencillos y su ejecución es eficiente y directa. Sin embargo existen desventajas. Estos lenguajes requieren que se tenga información sobre los agentes que recibirán los mensajes, información que a veces no está disponible para el agente emisor. También existe el problema de que los procedimientos son unidireccionales. Mucha de la información que los agentes requieren compartir debe estar disponible en ambas direcciones (transmisión-recepción, y viceversa). Lo anterior puede complicarse cuando un agente recibe varios "*scripts*" provenientes de múltiples agentes que trabajan simultáneamente y que pueden interferir entre sí. La mezcla de información procedimental es más complicada que la de información de tipo declarativa. Suelen usarse lenguajes de intérpretes de órdenes (*scripts*) tales como Perl, Tcl, Apple Events, Telescript etc. Estos lenguajes permiten un rápido □reformativ aunque no suelen ser fácilmente escalables ni reciclables.

- *Declarativos:* la comunicación tiene lugar utilizando enunciados declarativos, es decir intercambio de declaraciones (definiciones, suposiciones, etc), por lo que necesita que el lenguaje sea lo suficientemente expresivo y compacto como para comunicar diferentes clases de información. Varios Proyectos ARPA han hecho posible que la iniciativa Knowledge Sharing Effort (KSE) haya permitido definir los componentes del lenguaje de comunicación de agentes KQML (Knowledge □refo Manipulation Language) (Genesereth, 1994) (ARPA, 1992). El otro estándar ha sido desarrollado por FIPA (Foundation for Intelligent Physical Agent) y se denomina ACL (Agent Communication Language) Como resultado del esfuerzo por crear un lenguaje que permitiera la interoperación entre agentes autónomos.

2.2.1. Especificaciones Telescript

Telescript es un lenguaje basado en el entorno para construir sociedades de agentes “Multiagentes” que ha sido desarrollado por la compañía General Magic: quizás fue el primer sistema comercial de agente. Hay dos conceptos principales en la tecnología Telescript: lugares y agentes.

Los lugares son las localizaciones virtuales ocupadas por agentes, un lugar puede corresponder a una máquina, o a un conjunto de máquinas (figura 10). Los agentes son los proveedores y consumidores de bienes en las aplicaciones de mercado electrónico soportadas por Telescript. Los agentes son procesos de software móviles: son capaces de trasladarse de un lugar a otro; su programa y su estado pueden codificarse y transmitirse a través de una red a otro lugar, y allí comenzar de nuevo su ejecución. Los agentes son capaces de comunicarse entre sí: si ocupan lugares diferentes, pueden conectarse a través de una red; y si ocupan el mismo lugar, entonces pueden encontrarse.

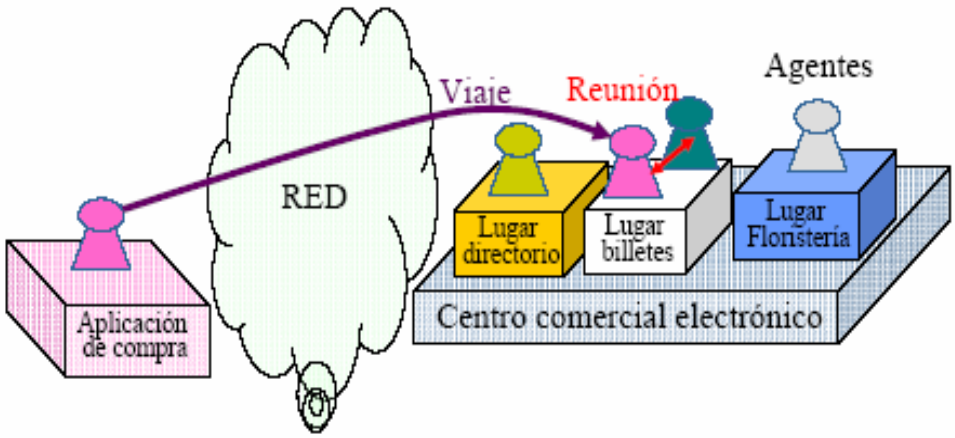

Figura 10. Modelo de agentes de Telescript

Cuando el agente *viaja* usa un *ticket*, donde se especifica los parámetros de su *viaje*:

- El destino del agente
- El tiempo en cual el *viaje* se completará

Los agentes de Telescript se comunican con otros agentes de diferentes modos:

- Si ocupan diferentes lugares, entonces se pueden conectar mediante una red
- Si ocupan el mismo lugar, entonces pueden *quedar* con otros

Por otro lado, los agentes de Telescript disponen de unos permisos asociados, donde se especifica qué pueden hacer (por ejemplo, limitaciones de viaje), y que recursos puede usar el agente. Los recursos más importantes son el dinero (medido en *teleclicks*, que corresponden a dinero real), el tiempo de vida (medido en segundos) y el tamaño (medido en bytes).

Tanto los agentes como los lugares son ejecutados por un motor (figura 11), que actúa como un intérprete para el lenguaje Telescript el cual es esencialmente una máquina virtual del estilo de JVM. Al igual que los sistemas operativos puede limitar el acceso a un proceso, mantiene lugares, planifica la ejecución de agentes, dirige la comunicación y el transporte de agentes, y finalmente, los motores proporcionan enlaces a otras aplicaciones mediante APIs.

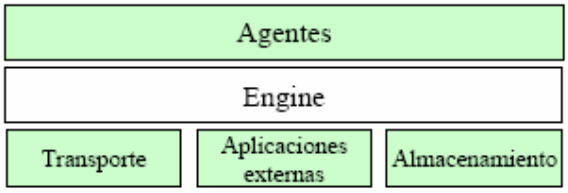

Figura 11. Motor Telescript

Otro componente es el conjunto de protocolos de Telescript (figura 12). Estos protocolos tratan de forma primaria la codificación y decodificación de agentes, gestionan la comunicación entre motores, permite transportar agentes, ofrece servicios de autentificación y seguridad, además pueden ejecutarse sobre varios protocolos como TCP/IP. El componente final es un conjunto de herramientas de software para soportar el desarrollo de aplicaciones telescript.

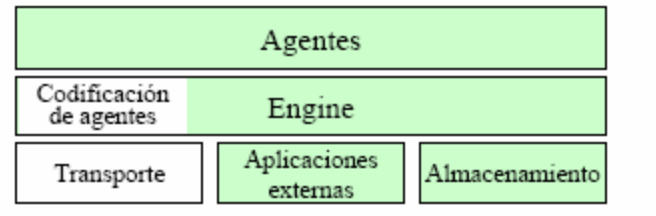

Figura 12. Protocolos telescript

Los agentes y los lugares se programan usando el lenguaje Telescript. Dicho lenguaje tiene las siguientes características:

- Es un lenguaje orientado a objetos
- Es interpretado
- Viene en dos niveles: alto (el lenguaje visible para los programadores) y bajo (un lenguaje semi-compilado parecido a Java bytecodes
- Contiene una clase proceso, de la cual agente y lugar son subclases
- Es persistente, significa que, por ejemplo, si una máquina que aloja se apaga y se enciende de nuevo, el estado del proceso sigue ejecutándose en la máquina.

Aunque **Telescript** fue un lenguaje pionero, que atrajo la atención de muchos desarrolladores, fue rápidamente reemplazado por Java finalizando la década de los 90

2.2.2. ACL Agent Communication language

ACL esta compuesto por tres componentes: vocabulario, lenguaje de contenido llamado KIF (Knowledge Interchange Format) y lenguaje de comunicación llamado KQML (Knowledge □refo Manipulation Language). Un mensaje en ACL seria una expresión en KQML que consiste en una directiva de comunicación y un contenido semántico en KIF expresado en términos del vocabulario, en donde el vocabulario de ACL es un diccionario de palabras apropiado para áreas comunes de aplicación. Cada palabra en el diccionario tiene una descripción que es usada por las personas para entender su significado y una anotación formal (escrita en KIF) que es usada por los programas.

KIF y KQML son el resultado de las investigaciones desarrolladas en el marco del proyecto DARPA Knowledge Sharing Effort (KSE). Ha sido definido como un lenguaje de cálculo de predicados de primer orden, con extensiones propias para resaltar la expresividad. Con este enfoque ofrece medios para incluir en sus expresiones datos simples, constricciones, disyunciones, reglas, negativas, metainformación. Sin embargo, en su diseño es independiente del contexto, por lo que cada mensaje debería incluir información implícita sobre el emisor, el receptor, el camino o vía, la historia de lo comunicado, etc. Para aumentar la eficiencia del sistema de comunicación, la misión de KQML es actuar como una capa lingüística que se encarga de tomar en consideración el contexto en el que se produce la comunicación, y liberar a KIF de trabajar con la información contextual.

2.2.3 Especificaciones FIPA Y KQML

FIPA (“Foundation for Intelligent Physical Agents”) fue creada en 1996 para producir software estándar para sistemas basados en agentes, heterogéneos e

interactuando entre ellos. FIPA ha elaborado una serie de especificaciones que pueden ser usadas para el desarrollo de sistemas basados en agentes.
Las distintas especificaciones FIPA van progresando a través de un ciclo de vida (figura 13), desde un estado preliminar (borrador en construcción cuyo identificador empieza por P), experimental (especificación estable que sólo admite pequeños cambios, cuyo identificador empieza por X) hasta el estado estándar (especificación implementada con éxito en varias plataformas; su identificador empieza por S). Cualquier especificación puede ser reprobada (el identificador empieza con D), previo al estado de *obsoleta* (su identificador empieza por O), cuando se considera innecesaria por cambios en la tecnología o en otras especificaciones y ha quedado por tanto obsoleta.

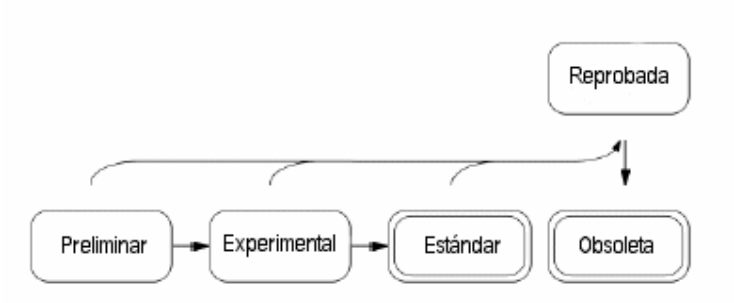

Figura 13. Ciclo de vida de las especificaciones FIPA

Las especificaciones FIPA se pueden organizar en cinco materias, según el dominio de agentes al que va dirigido:

- Aplicaciones
- Arquitecturas Abstractas
- Comunicación entre Agentes
- Gestión de Agentes
- Transporte de Mensajes de los Agentes

Sin embargo solo se trataran las especificaciones referentes a los mensajes escritos en un Lenguaje de Comunicación entre Agentes. Estas a su vez están

organizadas en Protocolos de Interacción para el intercambio de mensajes, actos comunicativos basados en la teoría de los actos de habla y lenguajes de contenido (figura 14).

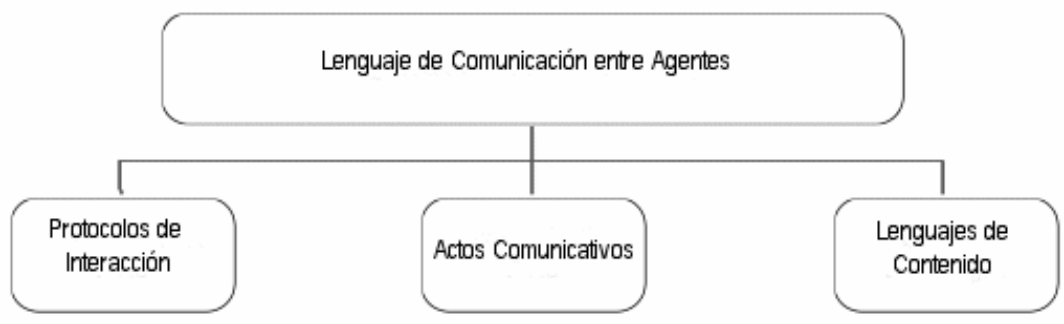

Figura 14. Especificaciones de ACL

Elemento	Categoría del Elemento
performative	Tipo de acto comunicativo
sender	Participante en la comunicación
receiver	Participante en la comunicación
reply-to	Participante en la comunicación
content	Contenido del mensaje
language	Descripción del contenido
encoding	Descripción del contenido
ontology	Descripción del contenido
protocol	Control de la conversación
conversation-id	Control de la conversación
reply-with	Control de la conversación
in-reply-to	Control de la conversación
reply-by	Control de la conversación

Tabla 1. Elementos de un mensaje en ACL de FIPA

FIPA propone distintas especificaciones para representar el contenido de los mensajes en distintos lenguajes: SL (“Semantic Language”), CCL (“Constraint Choice Language”), KIF (“Knowledge Interchange Format”) y RDF (“Resource Descripction Framework”).

Por otro lado KQML[14] (“Knowledge refo and Manipulation Language”) es un lenguaje y protocolo para el intercambio de información y conocimiento, surgido de los trabajos para el desarrollo de técnicas y metodologías para la construcción de bases de conocimiento a gran escala que sean compartidas y reutilizables. La sintaxis de KQML consiste en una notación en forma de listas de paréntesis balanceados (similar a LISP). KQML es un lenguaje basado en la teoría de actos de habla. Fue concebido como un formato de mensajes y como un protocolo que maneja los mensajes para permitir a un programa identificar, conectarse e intercambiar información con otros programas. En términos lingüísticos se puede decir que KQML se enfoca principalmente a la parte pragmática de la comunicación. Puede ser usado como un lenguaje para programas de aplicación que interactúen con un sistema inteligente o para dos o más sistemas inteligentes que compartan conocimiento en la resolución cooperativa de problemas. Por el momento, es uno de los modelos para comunicaciones entre agentes más usado. Las Tres características más importantes de KQML son:

1. Los mensajes de KQML son indiferentes al contenido de lo que transportan, esto es, los mensajes en KQML no comunican únicamente oraciones en un lenguaje, sino que comunican una actitud acerca del contenido (por ejemplo, afirmación, solicitud, pregunta).
2. Las primitivas del lenguaje, que se llaman reformativas, indican las acciones u operaciones permitidas que los agentes pueden utilizar cuando se comunican (actos de habla).
3. El entorno en el que los agentes se comunican con KQML puede ser enriquecido con un tipo de agentes especiales llamados reformativas, que son una clase especial de agentes que coordinan las interacciones entre los otros agentes.

KQML es un lenguaje que se divide en tres capas o niveles en los cuales codifica la información (figura 15):

[14]RIZO Ramon,LLORENS Faraón, PUJOL Mar. Arquitecturas y comunicación entre agentes. Universidad de Alicante

- *La capa de contenido* se relaciona con el contenido real del mensaje escrito en el lenguaje de representación propio de cada agente. Un mensaje en KQML puede tener cualquier lenguaje de representación incluyendo lenguajes expresados como cadenas en ASCII y aquellos expresados utilizando una notación binaria. Cualquier implementación de KQML ignora la parte del contenido del mensaje excepto para determinar dónde termina.

- *La capa de comunicación* codifica un conjunto de características del mensaje, las cuales describen los parámetros de la comunicación de bajo nivel, tales como la identidad del agente que envía el mensaje y la del que lo recibe, así como un identificador único asociado con la comunicación.

- *La capa de mensaje* se utiliza para codificar el mensaje que una aplicación desea transmitir a otra. Esta capa forma el corazón del lenguaje y determina las clases de interacciones que se pueden tener con un agente que hable KQML. La función principal de la capa de mensaje es identificar el protocolo que se va a usar para entregar el mensaje (síncrono o asíncrono) y proporcionar una reformativa que el transmisor le agrega al contenido. Como el contenido es indiferente a KQML, en esta capa también se incluyen características opcionales que describen el lenguaje del contenido, la ontología que se está asumiendo y alguna clase de descripción del contenido. Estas características hacen posible que las implementaciones de KQML analicen, redirijan y entreguen el mensaje apropiadamente aún cuando su contenido sea inaccesible.

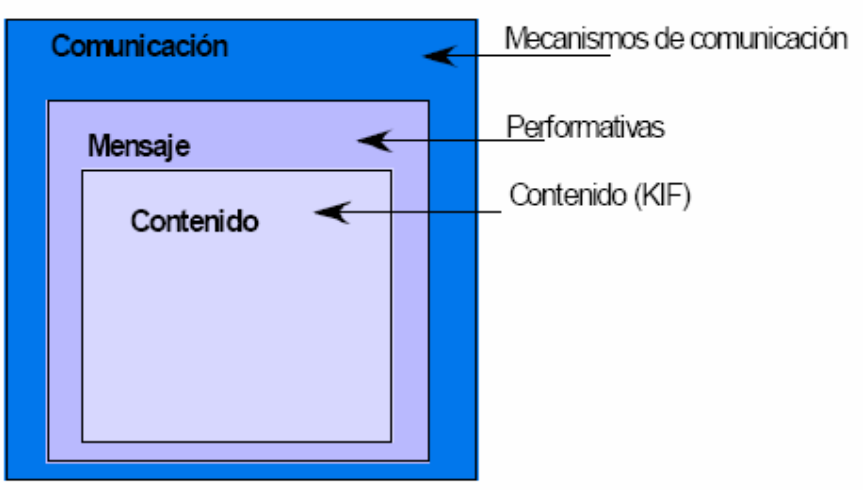

Figura 15. Capas del lenguaje QKML

Todo mensaje en KQML se inicia con una directiva de comunicación (reformativa o expresión realizativa) que indica el tipo de comunicación y un conjunto de argumentos opcionales, llamados parámetros, entre los que se encuentra el contenido real del mensaje así como otros que describen el propio contenido. Existe un conjunto de reformativas estándar con un significado al que toda implementación de KQML debe adherirse. Estas reformativas estándar se dividen en tres grupos:

- *Las preformativas de discurso* (ask-if, ask-all, ask-one, tell, deny, achieve, advertise, subscribe, entre otras), empleadas en el contexto de un intercambio de información y conocimiento entre dos agentes.
- *Las preformativas de intervención y mecánica de la conversación* (error, sorry, ready, next, discard, rest, standby), cuyo papel es intervenir en el curso normal de una conversación.
- *Las preformativas de red y de facilitación* (register, forward, broadcast, recommend-one, recruit-all, entre otras), los cuales, estrictamente hablando no son actos del habla, pero permiten a los agentes encontrar otros agentes capaces de procesar sus mensajes.

Los nombres de los parámetros empiezan con ":" y deben estar seguidos por el correspondiente valor del parámetro. Los posibles parámetros con su correspondiente descripción se pueden ver en tabla 2

Palabra clave	Descripción
:sender	emisor (actual) de la performativa
:receiver	receptor (actual) de la performativa
:from	el origen (emisor virtual) de la performativa de :content cuando se hace un forward
:to	el destinatario final (receptor virtual) de la performativa de :content cuando se hace un forward
:in-reply-to	etiqueta esperada en la respuesta al mensaje previo (la misma de :reply-with del mensaje previo)
:reply-with	etiqueta esperada en la respuesta al actual mensaje
:language	lenguaje de representación en el que está escrito :content
:ontology	nombre de la ontología asumida en :content
:content	información con la que la performativa expresa una actitud

Tabla 2. Parámetros reservados para las preformativas KQML

El protocolo básico está definido por la siguiente estructura:

```
(<KQML-performativa>
      :sender     <nombre>
      :receiver   <nombre>
      :languaje   <nombre>
      :ontology   <nombre>
      :content    <expresión>
      ...
)
```

A continuación se puede observar un ejemplo de un mensaje KQML donde un agente que se identifica como *jorge* solicita a un *facilitador* que envíe un mensaje proveniente del mismo *jorge* y escrito en KQML a otro agente *ana*, asumiendo la ontología *ontologia-fdl*. A su vez el mensaje que *jorge* quiere hacer llegar a *ana* a través de *facilitador* indica que *jorge* desea que *ana* haga cierto en su entorno lo que está en el contenido del mensaje (*estado=suspendido*) y que le responda con un mensaje identificado por *id1*. El

parámetro *content* define el nivel de contenido; los parámetros *from, to, sender y receiver* especifican información del nivel de comunicación; y *language* y *ontology* forman parte del nivel de mensaje.

```
(forward
      :from jorge
      :to ana
      :sender jorge
      :receiver facilitador
      :language kqml
      :ontology ontologia-fdl
      :content (achieve
            :sender jorge
            :receiver ana
            :reply-with id1
            :content (estado=suspendido)
      )
)
```

Es posible decir que cada mensaje KQML es una pieza de dialogo entre emisor y receptor, y KQML proporciona el soporte para una amplia variedad de tipos de diálogos. El conjunto de reformativas es extensible y no todas son necesarias. Un agente puede elegir manejar sólo unas pocas reformativas de las descritas, e incluso implementar algunas adicionales. Sin embargo, cualquier implementación de las reformativas reservadas debe ser utilizada de la manera descrita.

2.2.4. Lenguajes de programación del comportamiento del agente

Permiten la programación de los agentes dentro de la cual definen su estructura, conocimiento y habilidades. Se pueden distinguir de varios tipos:

- *Lenguajes de descripción de agente*: los agentes se derivan de una clase de agente genérica, permitiendo la definición de los elementos básicos del modelo de agente tales como base de conocimiento, grupos de agentes, habilidades del agente, servicios ofrecidos, planes para alcanzar objetivos, etc. La descripción se traduce a un lenguaje ya ejecutable. Se puede citar a MACE ADL, AgentSpeak y MAST/ADL.

- *Lenguajes de programación orientados a agentes*: siguiendo el paradigma de Shoham de la programación orientada a agentes (AOP), se han definido algunos lenguajes que tienen en cuenta el estado mental del agente para programar las funciones de transición entre estos estados mentales, que consisten en creencias, capacidades y obligaciones. Entre estos lenguajes se destacan el AGENT-0, PLACA y Agent-K.

- *Lenguajes basados en reglas de producción*: la programación de la base de conocimiento de los agentes se realiza en algunos sistemas empleando reglas de producción como, por ejemplo, MAGS (basado en OPS5), DYNACLIPS (arquitectura de pizarra basada en CLIPS) y RTA, que permite definir las conductas del agente con reglas.

- *Lenguajes de especificación*: emplean una especificación (normalmente lógica) del agente que se ejecuta directamente para generar su conducta, pudiendo verificar propiedades de la especificación.

3. SISTEMAS MULTIAGENTES (MAS)

Normalmente los agentes no actúan en solitario, sino que se localizan en entornos o plataformas con varios agentes, donde cada uno de ellos tiene sus propios objetivos, toma sus propias decisiones y puede tener la capacidad de comunicarse con otros agentes. Dichos entornos se conocen con el nombre de sistemas multiagentes o agencias.

Los sistemas multiagentes dan un mayor nivel de abstracción con respecto a la informática distribuida tradicional, estando más cercanos a las expectativas del usuario y permitiendo al programador una mayor flexibilidad para expresar el

comportamiento de los agentes. Como características principales de los MAS cabe destacar las siguientes:

- Cada agente del sistema tiene un punto de vista limitado (no tiene la información completa).
- No existe un control global para todo el sistema.
- Los datos están descentralizados.
- La computación es asíncrona.
- Permite interoperación con sistemas existentes.

Sin embargo, este tipo de diseños suele plantear una serie de problemas, como la elección del tipo de comunicación entre los agentes, del tipo de plataforma o del método de desarrollo, y por ultimo los criterios para la seguridad del sistema.
Los agentes con una arquitectura basada en capas no constituyen por sí mismos un sistema multiagente. Existen dos tipos fundamentales en este tipo de construcciones:

- Capas verticales: sólo un subagente tiene acceso a los sensores y sólo uno puede actuar sobre el entorno.
- Capas horizontales: cada subagente puede sentir y actuar.

Ambas capas han sido mencionadas en capítulos anteriores por ende no es imprescindible hacer énfasis de nuevo. Sin embargo la O.M.G.[15] (Object Management Group) define un sistema multiagente como: "una plataforma que puede crear, interpretar, ejecutar, transferir y terminar agentes".

3.1. INTEROPERABILIDAD ENTRE SISTEMAS MULTIAGENTES

Los agentes que forman un sistema pueden comunicarse entre ellos. El proceso para transferir un agente de un sistema a otro se realiza en tres fases:

[15]Complementar con la pagina web http/:www.omg.org/

1. Iniciación de la transferencia:

- El agente identifica el destino deseado, realiza una petición de viaje al sistema y si es aceptada recibe el permiso para ejecutar la transferencia.
- El sistema suspende la ejecución del agente e identifica el estado y las partes del agente que serán enviadas.
- Se realiza la conversión en serie del código y del estado del agente (seriación) y se codifica según el protocolo establecido.
- El sistema hace la autenticación del agente.
- Se realiza la transferencia.

2. Recepción del agente.

- El sistema destinatario acredita al cliente.
- Se realiza la decodificación del agente y la conversión de serie a código y estado del agente.
- El sistema crea la instancia del agente, restaura su estado y continúa la ejecución.

3. Transferencia de otras clases (sistemas orientados a objetos).

- Este proceso es necesario cuando el agente se mueve de un sistema a otro, cuando el agente se crea remotamente o cuando necesita otros objetos. La transferencia de las clases puede realizarse completamente junto con el viaje del agente o hacer peticiones de carga cuando sea preciso.

La normalización en el proceso de interconexión de agentes móviles se aplica a dos niveles:

1. Interoperación entre lenguajes de programación.
2. Interoperación entre sistemas escritos en el mismo lenguaje.

La primera resulta muy compleja de alcanzar y continúa aún en proceso de investigación, mientras que el segundo caso está siendo normalizado por CORBA.

Para alcanzar el grado de interconexión deseado las MASIF de CORBA definen los siguientes conceptos:

- Lugar: contexto dentro de un sistema donde puede ejecutarse un agente; por lo tanto, un agente viaja de un lugar a otro, ya sea en el mismo sistema o en otro distinto.
- Localización: va asociada con un lugar, indicando el nombre y la dirección que ocupa en la agencia.
- Localidad: propiedad de cercanía al destino, ya sea en el mismo computador o en la misma red.
- Infraestructura de comunicación: provee servicios de transporte al sistema.
- Autoridad: persona o entidad en nombre de la cual actúa un agente o un sistema de agentes.
- Región o dominio: conjunto de sistemas de agentes que tienen la misma autoridad y que no tienen que ser del mismo tipo.

En algunos casos pueden ocurrir circunstancias distintas en el proceso de comunicación entre dos agentes que se encuentran en distintos sistemas.
Los sistemas comparten el mismo perfil para la gestión de agentes (figura 16): no existen problemas de interoperación y el agente puede viajar de un sistema a otro para hacer una comunicación local con el otro agente.

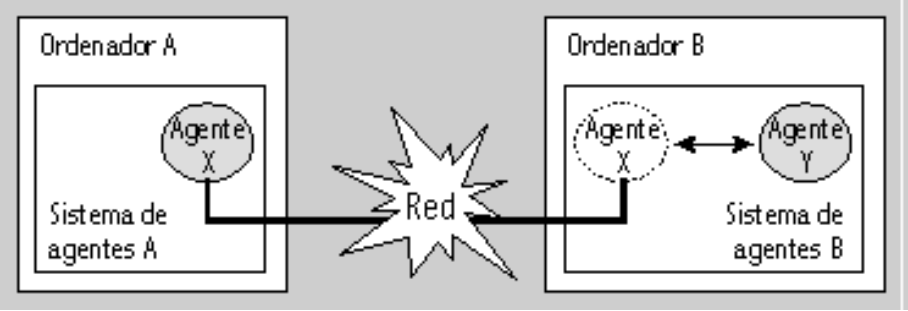

Figura 16. Sistemas compatibles

Los sistemas no siguen la misma norma, pero puede realizarse una comunicación local (figura 17): en el ordenador destino existe otro sistema de agentes (C) compatible con el sistema origen (A), el agente se desplaza al sistema compatible y realiza la comunicación como RPC por ejemplo con el otro agente.

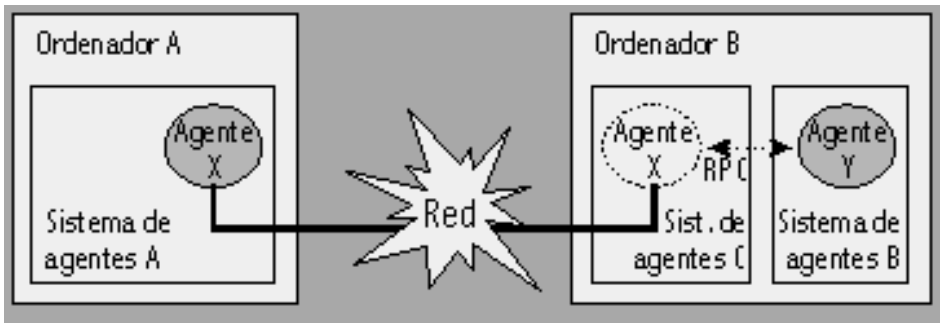

Figura 17. Sistemas incompatibles con localidad

Por ultimo se puede encontrar con sistemas que no son compatibles y no puede alcanzarse la propiedad de localidad (figura 18), y se debe realizar una comunicación normal entre los agentes.

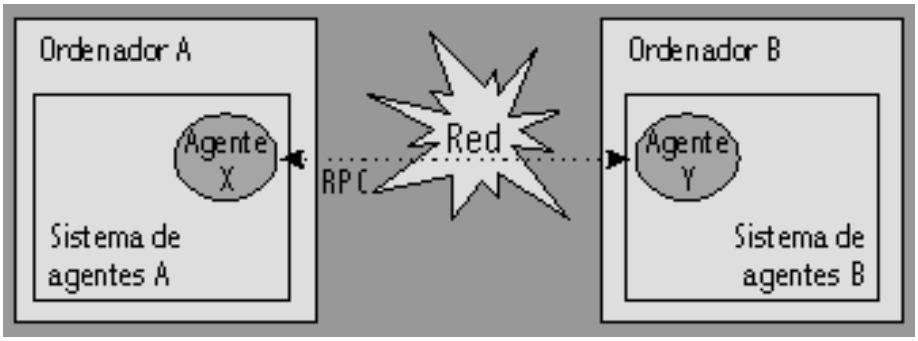

Figura 18. Sistemas incompatibles

3.2. AGENTES MÓVILES Y CORBA

CORBA (*Common Object Request Broker Architecture),* arquitectura común de intermediarios en peticiones a objetos, es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

CORBA fue definido y está controlado por el *Object Management Group* (OMG) que define las APIs, el protocolo de comunicaciones y los mecanismos necesarios para permitir la interoperabilidad entre diferentes aplicaciones escritas en diferentes lenguajes y ejecutadas en diferentes plataformas, lo que es fundamental en computación distribuida.

En un sentido general CORBA "envuelve" el código escrito en otro lenguaje en un paquete que contiene información adicional sobre las capacidades del código que contiene, y sobre cómo llamar a sus métodos. Los objetos que resultan pueden entonces ser invocados desde otro programa (u objeto CORBA) desde la red. En este sentido CORBA se puede considerar como un formato de documentación legible por la máquina, similar a un archivo de cabeceras pero con más información.

CORBA utiliza un lenguaje de definición de interfaces (IDL) para especificar los interfaces con los servicios que los objetos ofrecerán. CORBA puede especificar a partir de este IDL la interfaz a un lenguaje determinado, describiendo cómo los tipos de dato CORBA deben ser utilizados en las implementaciones del cliente y del servidor. Implementaciones estándar existen para Ada, C, C++, Smalltalk, Java y Python. Hay también implementaciones para Perl y TCL.

Al compilar una interfaz en IDL se genera código para el cliente y el servidor (el implementador del objeto). El código del cliente sirve para poder realizar las llamadas a métodos remotos. Es el conocido como *stub*, el cual incluye un

proxy (representante) del objeto remoto en el lado del cliente. El código generado para el servidor consiste en unos *skeletons* (esqueletos) que el desarrollador tiene que rellenar para implementar los métodos del objeto.

CORBA es más que una especificación multiplataforma, también define servicios habitualmente necesarios como seguridad y transacciones. Y así este no es un sistema operativo en si, en realidad es un middleware.

3.3. SEGURIDAD

El control de la seguridad es un grave problema, ya que un agente es un programa que viaja de un computador a otro mediante la red, al igual que un virus. Los sistemas deben hacer enfasis en la seguridad de los computadores y de los agentes móviles. Los aspectos de seguridad típicos que deben ser controlados son:

- Protección del computador contra los agentes.
- Protección contra otros agentes.
- Protección de los agentes contra el computador
- Protección de la red.

Los ataques más comunes que pueden realizarse a un sistema de agentes móviles son:

- Inundar el sistema con peticiones, tanto legales como ilegales.
- Escuchar la red para obtener información privada.
- Modificar, borrar o sustituir cualquier elemento transferido por la red.
- Grabar y retransmitir ilegalmente una comunicación.
- Falsificar la identidad "enmascaramiento" de un agente o sistema de agentes para tener acceso a la información o a ciertos servicios.

- Utilización abusiva de algún recurso para que no pueda ser utilizado por otro usuario.
- Colocar un Caballo de Troya; agente o sistema de agentes para recibir información confidencial o denegar acceso a los recursos.

Tanto los sistemas como los propios agentes móviles deben reforzar las tareas de seguridad para evitar, de un modo fiable, los ataques descritos anteriormente. Pueden tener varias políticas de seguridad que permitan:

- Comprobar la autentificación de los participantes en cualquier comunicación, en cierto modo saber quienes requieren el acceso.
- Restringir o garantizar las operaciones que puede ejecutar un agente.
- Gestionar privilegios de acceso a los recursos y establecer límites de consumo.

La mayoría de los sistemas operativos de los principales fabricantes proporcionan un completo juego de herramientas de seguridad, y hay algunas reglas básicas que aplicar a todos ellos. Para poder hacer esto posible de manera segura es necesario proveer los medios para garantizar la autenticación, integridad y confidencialidad de los datos durante el proceso de la comunicación:

- *Autenticación:* tanto el agente o sistema emisor como el receptor deben ser identificados para evitar accesos a información o a recursos reservados.
- *Confidencialidad*: Dado que los agentes y su información viajan a través de un medio potencialmente hostil como Internet, los mismos son susceptibles de intercepciones, por lo que es fundamental el cifrado de los mismos. De este modo, la información no podrá ser interpretada por nadie más que los agentes destinatarios de la misma.
- Integridad: comprobar que los datos no han sido modificados o alterados durante la transferencia.

- Detección de reproducción: evitar la duplicación de un agente durante una comunicación.

Es importante resaltar que los agentes no deben incluir en su viaje ninguna clave criptográfica, sino que ha de utilizarse un algoritmo “denominado autenticador en las MASIF” que verifique la validez del agente y del sistema origen, las autoridades de los sistemas y agentes involucrados en la comunicación y cuáles son las autoridades consideradas como seguras.

Aunque hay una gran variedad de políticas de seguridad que pueden utilizarse para evitar los ataques, un proceso de comprobación típico antes de iniciarse el viaje de un agente incluye los siguientes aspectos:

- El sistema debe verificar la autoridad propietaria del agente.
- Durante la creación de la petición, el propietario define las preferencias de seguridad para el agente.
- Al crearse la instancia del agente, se incluye información sobre su autoridad y la de su sistema.
- El sistema origen codifica la información.
- Los sistemas origen y destino crean un canal de comunicaciones seguro.
- El sistema destino descodifica la información y realiza las comprobaciones necesarias.

Es importante tener en cuenta que un agente tiene la potestad de restringir o permitir a otros agentes el acceso a sus métodos, a sus datos o a los recursos que controla, según su comportamiento o sus objetivos.

4. REDES ACTIVAS

El rendimiento de los modernos sistemas de computación distribuida depende en gran medida de los servicios de red que se utilizan para mover información de un nodo a otro. Sin embargo, curiosamente, la evolución de estos servicios está siendo más lenta que la evolución del resto de los elementos que integran el entorno en el que se construyen los sistemas de computación. Esto hace referencia tanto al hardware implicado (computadores, Swicth, routers,) como al software utilizado (sistemas operativos, protocolos, lenguajes de programación).

Esta lenta evolución no es por la falta de necesidades ni a la ausencia de ideas innovadoras. El problema principal se encuentra en que el cambio o creación de nuevos protocolos de red es una tarea lenta y dificultosa. Esto se debe a que es necesaria la estandarización de la propuesta de protocolos realizada para garantizar la interoperatividad de los sistemas. Así, pueden transcurrir años desde que surge la necesidad de cambio y el momento en que se instaura una solución. Cuando un protocolo ha sido aceptado, su desarrollo presenta muchas dificultades ya que no existe una forma automática para incluir el nuevo protocolo con los ya existentes. Unido a esto aparece el problema de la compatibilidad con versiones existentes. Por tanto, estos tres periodos de tiempos unidos hacen que el contar con una solución viable suponga una evolución muy lenta de los servicios de red. En este sentido se prevé que la implantación de IPv6[16] puede llevar quince años aproximadamente.

Un nuevo planteamiento que permite solucionar los problemas anteriormente mencionados es la propuesta de *redes activas*. Las redes activas representan un paso fundamental en la evolución de las redes de conmutación de paquetes, desde los dispositivos tradicionales de encaminamiento de paquetes hacia una

[16]Complementar con la pagina web http://imasd.elmundo.es/imasd/ipv6/queesipv6.html

funcionalidad más general que soporta el control y la modificación dinámica del comportamiento de la red. Las redes son "activas" desde dos puntos de vista. Para los routers y switches de la red, se plantea dotarlos de nuevas funcionalidades para que puedan actuar sobre ella; por ejemplo, realizando cálculos sobre el flujo de datos de usuario que pasan a través de ellos. Desde el punto de vista de los usuarios, éstos pueden programar la red cargando sus propios programas para la realización de tareas específicas; por ejemplo, el usuario puede especificar a un router que ejecute un determinado algoritmo de compresión durante el procesado de sus paquetes. No obstante, la idea subyacente y fundamental en las redes activas es estandarizar un modelo de comunicación en lugar de protocolos de comunicación individuales. De esta forma se pretende lograr un consenso en la comunidad de investigación para estandarizar un modelo computacional en los nodos activos (conjunto de instrucciones y recursos disponibles) que proporcione un interfaz de programación de aplicaciones común.

4.1 EVOLUCIÓN DE LAS REDES ACTIVAS

Los conceptos de las redes activas tienen su origen en la iniciativa llevada a cabo por la comunidad científica investigativa DARPA durante los años de 1994 y su aplicación a las redes en 1995. Durante este periodo se identificaron varios problemas en las redes actuales: la dificultad de integración de nuevas tecnologías y estándares en una infraestructura de red compartida, rendimiento pobre debido a las operaciones redundantes en varios niveles de la pila de protocolos, y la dificultad de acomodar nuevos servicios en el modelo arquitectural existente.

Para tratar de solucionar estos problemas surge el concepto de redes activas. Las redes activas proporcionan una nueva forma de pensar en cuanto a los entornos de red. En las redes tradicionales las aplicaciones se sitúan en los extremos finales y en los nodos intermedios los paquetes son encaminados

independientemente de su contenido. Las redes activas rompen con este modelo permitiendo que las redes sean programables y puedan realizar cómputos en los datos de usuario que están pasando a través de ellas[17]. Algunas de las aplicaciones más inmediatas de las redes activas son: firewalls, web proxies, comunicaciones móviles, fusión de información, Actualmente, existen dos planteamientos para la construcción de redes activas: uno discreto y otro integrado. En el primero se separa el mecanismo para inyectar programas en los nodos activos del procesamiento de los paquetes que atraviesan el nodo. Existirá, por tanto, un mecanismo para que los usuarios incluyan sus programas en los nodos activos y se marcará en los paquetes cuál es el programa que los va a procesar. Otro planteamiento diferente es incluir el propio código con el que se van a procesar los paquetes dentro de los mismos. De esta forma, los paquetes, denominados cápsulas, van a contener los datos y los programas.

4.2 ENTORNOS ACTUALES

Existen diversos entornos de redes activas, entre los que se pueden destacar:

- Smartpackets de BBN
- Active Network Transport System (ANTS), del MIT
- Liquid Software de la Universidad de Arizona
- SwitchWare, de la Universidad de Pennsylvania
- Netscript de la Universidad de Columbia
- Liane, de Georgia Techs

Otros entornos basados en agentes móviles y susceptibles de entrar en el ámbito de las redes activas se estudiaron anteriormente por ejemplo CORBA.

[17]D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. “A Survey of Active Network Research.” IEEE Communications Magazine, Vol. 35, 1997

4.3 ESTÁNDARES UTILIZADOS

Existen actualmente diversas recomendaciones y estándares relacionados con las redes activas; sin embargo, tienen aún poca incidencia en el desarrollo de soluciones generales y públicas. Entre las iniciativas más destacadas se tiene:

- Interfaces programables abiertos del OPENSIG, IEEE P1520
- Arquitecturas de control múltiple, en el Multiservice Switching Forum
- Diversos RFCs del IETF
  RFC 1987. GSMP, General Switch Management Protocol.
  RPC : ANEP Protocol (en estudio).

4.4. API DE RED

La programación de la red se hace en base a la utilización de una *Application Program Interface de red* (API de red). En nomenclatura de redes activas, una API de red define una máquina virtual que interpreta un lenguaje específico. El API de red para IP incluye el lenguaje que define la sintaxis y la semántica de la cabecera IP y su efecto en los routers de la red. En las redes tradicionales la maquina virtual es fija, y el poder expresivo del lenguaje limitado. Si es posible ver la cabecera IP de una red tradicional como la entrada de datos a una máquina virtual, se puede ver que en una red activa la entrada de datos a la máquina virtual son, además de los datos, los programas que porta el propio paquete.

Poder expresivo del lenguaje: de todos los aspectos que determinan como es el API de red, el más crucial es el lenguaje utilizado para programar la red.
El grado de programabilidad del API de red puede ser muy diverso. Las posibilidades van desde la selección de una simple lista de parámetros seleccionados de un conjunto predeterminado de posibilidades, a la utilización de un lenguaje completo equivalente Turing capaz de describir cualquier tipo de computación. Entremedias existe una infinita gama de posibilidades basadas o

en restringir de distintas formas los lenguajes de propósito general o en la utilización de lenguajes de propósito específico.
Un aspecto esencial del lenguaje de programación es la facilidad de acceso a los recursos del nodo por ejemplo, colas de salida o tablas de encaminamiento, para esto es determinante cual es el conjunto de abstracciones asociados a éstos que proporciona el lenguaje.

La elección de un determinado lenguaje de programación está muy relacionada con la arquitectura de seguridad de la red activa. Los nodos se pueden programar, pero cualquier plataforma de redes activas debe ofrecer ciertos niveles de seguridad que garanticen que un programa inyectado en el nodo, ya sea de forma intencionada o sin intención, no haga cosas "prohibidas" que pongan en peligro el funcionamiento del nodo. Al hablar de cosas prohibidas es a programas que se meten en un bucle infinito, que acceden a zonas de memoria protegidas, o que hacen un uso abusivo de los recursos.

La ventaja de utilizar un lenguaje restringido es que limita los posibles comportamientos del nodo simplificando el análisis de que el comportamiento programado es correcto y por otro lado acota de algún modo el efecto que un determinado paquete puede hacer en un nodo. En esta línea alguna de los lenguajes propuestos no admiten la existencia de estructuras de control del tipo bucles o bifurcaciones, o no admiten la utilización de punteros
.
Alexander[18] ha identificado una serie de propiedades que debería tener un lenguaje de programación válido para redes activas siendo las más relevantes las siguientes: fuertemente tipado, disponer de un recolector de basura, posibilidad de carga dinámica, un mecanismo que permita definir diferentes interfaces para acceder a un mismo módulo y ofrecer buen rendimiento.

Almacenamiento temporal en el nodo: otra característica importante del API de red es si permite que un programa incluido en un determinado paquete activo,

[18] S. Alexander. *"A Generalized Computing Model of Active Networks".* Ph.D. U. de Pennsylvania. 1998.

deje residente información, por ejemplo información de estado, que pueda ser utilizada posteriormente por otros paquetes (por ejemplo que pertenecen al mismo flujo) que atraviesen el nodo posteriormente. Normalmente esta memoria es denominada *soft-state* pues tiene asociado un tiempo de vida pasado el cual si no se ha accedido a dicha información de alguna forma (para leerla o modificarla), dicha información se borrará del nodo sin que exista ningún tipo de notificación. Obviamente cuando se ofrece esta posibilidad, deben incluirse mecanismos que protejan de accesos no autorizados a dicha información de estado, y mecanismos que acoten la cantidad de memoria que puede utilizar un determinado usuario o flujo. La existencia o no, de memoria del tipo soft-state hace que sea muy diferente el que es posible programar en una red activa.

Composición dinámica de servicios: la meta última de las redes activas es el hacer mas fácil el desarrollo de nuevos servicios de red. En concreto, debería proporcionar cierto soporte al proceso de creación de servicios. De aquí que una importante característica que debe incorporar el API de red es la posibilidad de poder componer servicios a partir de bloques básicos. Estos bloques básicos son usualmente denominados componentes. Un API de red debería contener un mecanismo de composición para poder construir servicios compuestos en base a determinadas componentes. Existen trabajos que apuntan soluciones en esta línea siguiendo distintas aproximaciones, como por ejemplo el sistema LIANE[19] que plantea un sistema de composición basado en eventos, o el Netscript[20] donde se utiliza un modelo de computación de flujo de datos.

4.5. SISTEMA DE DISTRIBUCIÓN DE CÓDIGO

Existen dos enfoques principales para realizar la descarga de código en el nodo: uno denominado discreto o fuera de banda y otro llamado integrado o en banda.

[19] Kenneth L. Calvert, Samrat Bhattacharjee,. Directions in Active Networks. IEEE Communications Magazine, 1998.
[20] Complementar con la pagina web http://www.cs.columbia.edu/~dsilva/netscript.html

1. *Enfoque discreto:* se mantiene el formato de paquete existente y se proporciona un mecanismo que soporta de forma separada la descarga de programas (*extensiones activas*), independizando esta descarga de programas, del proceso realizado sobre los paquetes. Cuando llega un paquete a un nodo se examina su cabecera y se lanza el programa adecuado para procesar dicho paquete. Esta separación puede resultar interesante en los casos en los que la selección de los programas que se deben inyectar en la red es realizada por administradores de red, en lugar de por usuarios finales. En este enfoque, el nodo programable no es tan "activo" como puede ser el enfoque integrado, pero es más fácil de implementar y desarrollar en las redes actuales.

2. *Enfoque integrado*: en este enfoque, los paquetes pasivos de las redes tradicionales son reemplazados por *paquetes activos* que incluyen tanto los datos como el programa activo, relativamente pequeño, que será ejecutado en los nodos cuando los vaya atravesando el paquete activo. En realidad, en el paquete activo puede ir realmente el código o sólo una referencia a dicho código, de ahí las dos posibilidades siguientes:

    - *Código embebido*: en el que el código asociado al paquete activo va incluido en el propio paquete activo. Este enfoque es muy adecuado cuando el código asociado al paquete activo tiene un tamaño muy pequeño o cuando se trata de paquetes activos que se mandan muy pocas veces.

    - *Carga bajo demanda*: en el que los paquetes activos identifican el proceso que se debe ejecutar en el nodo, y este se carga la primera vez que se recibe un paquete activo de ese tipo.

De esta forma el proceso que trata un tipo de paquetes activos sólo se cargará una vez por sesión o mientras se esté mandando paquetes activos de ese tipo. En ambos casos, enfoque discreto o integrado, deberán existir mecanismos que permitan la carga dinámica de código. Este mecanismo de carga dinámica permite que el código se cargue en el nodo mientras este se encuentra operativo, sin que sea necesario interrumpir temporalmente su funcionamiento.

4.6. SEGURIDAD

En una red activa existen dos tipos de funcionalidades encargadas de proporcionar seguridad al sistema. Por un lado existe una funcionalidad, usualmente denominada seguridad (security) que se encarga de evitar que usuarios no autorizados puedan causar daños intencionados al sistema, incluso hasta el punto de dejar un nodo o la red no operativa. Estos daños pueden consistir en dejar a un nodo sin recursos por ejemplo memoria, cambiar las tablas de encaminamiento, modificar una extensión activa o el código embebido en un paquete generado por otro usuario. Generalmente, los mecanismos utilizados para proveer esta funcionalidad se basan en utilizar mecanismos de autorización y autenticación, firmas digitales y listas de control de acceso.

Por otro lado, existe una funcionalidad (denominada unas veces *safety* y otras *robustness*) que se encarga de evitar que usuarios autorizados, ya sea de forma intencionada o no, puedan causar daños al sistema o a otros usuarios. Estos daños pueden consistir en enviar a un nodo un programa que se mete en un bucle infinito, acceder al espacio de memoria que está siendo utilizado por otro usuario o por el propio nodo, etc. Los mecanismos utilizados para lograr este tipo de seguridad están en muchos casos relacionados con las propiedades del lenguaje utilizado para programar la red, como es la utilización de lenguajes fuertemente tipados y con recolector de basura. También se ha propuesto la utilización de técnicas formales que permitan probar la corrección

de un programa antes de que éste sea introducido en la red. Uno de los principales retos para definir una arquitectura de seguridad satisfactoria es el coste de proceso de los mecanismos de seguridad asociados, que en muchos casos la hace inviable en una red activa en producción.

Dado que este aspecto constituye un factor crítico en las redes activas, en la arquitectura de red activa propuesta por DARPA y presentada en la sección 8 se plantea como objetivo de diseño que, deben siempre estar limitadas las consecuencias que puedan tener las acciones realizadas por un usuario, aunque éste disponga del nivel más alto de autorización.

4.7. GRANULARIDAD DEL CONTROL

Este atributo se refiere a cuanto tiempo se mantendrá modificado el comportamiento del nodo por ejemplo un cambio en el algoritmo de encaminamiento del nodo. Una primera posibilidad es que este cambio del comportamiento del nodo afecte a todos los paquetes que atraviesen el nodo hasta que no llegue otro paquete que modifique de nuevo dicho comportamiento. Otra posibilidad es que este cambio afecte únicamente al paquete que lleva el programa, granularidad por paquete. Y entre estas dos posibilidades existe otra intermedia que hace que el cambio en el comportamiento se restrinja a todos los paquetes pertenecientes a un determinado flujo. Siendo un flujo un conjunto de paquetes que comparten una característica común como por ejemplo tener una determinada dirección origen.

4.8. VENTAJAS DE LAS REDES ACTIVAS

La programación de la red puede resultar útil a distintos niveles, siendo sus principales potencialidades las siguientes:

- Acelerar la evolución de las redes, al permitir que nuevos protocolos y servicios sean introducidos en la red sin la necesidad de largos procesos, de estandarización y despliegue en la red.
- Desde el punto de vista de los proveedores de servicio de red, las redes activas permiten reducir el tiempo necesario para desarrollar y desplegar nuevos servicios de red.
- Permitir a los usuarios crear nuevos servicios o adaptar los ya existentes de acuerdo a las necesidades concretas de sus aplicaciones. Asimismo es posible imaginar usuarios que adapten los servicios, en base a opciones proporcionadas por “terceras partes” que vendan servicios de red mejorados.
- Facilitar la experimentación de nuevos protocolos o nuevos servicios de red. Desde el punto de vista de los investigadores, las redes activas ofrecen una plataforma en la cual experimentar con nuevos servicios de red de forma real, mientras que ésta continúa operando normalmente.

Un aspecto importante a reseñar es, que la utilidad de las redes activas está relacionada con la velocidad de la red. En redes con poco ancho de banda, el procesamiento de todos los paquetes es factible. Pero cuando el ancho de banda aumenta, la capacidad de computación disponible por byte decrementa, hasta tal punto, que el único procesamiento que se puede realizar por paquete es el relacionado con la función en encaminamiento. De aquí que sólo una pequeña fracción de los paquetes puede ser procesados, y es necesario que en cada paquete se indique si se necesita un procesamiento rápido asociado al encaminamiento o lento. La mayoría de los paquetes sólo necesitarán un procesado rápido (denominado usualmente *Fast-path processing* en terminología inglesa).

Con esta idea en mente, los investigadores en redes activas se han planteado como requisito de diseño, la construcción de un nodo activo que además de realizar procesamiento a medida sobre algunos paquetes que lo atraviesan,

sean capaces de encaminar datagramas IP tradicionales, a una velocidad comparable a la conseguida por los encaminadores IP "pasivos" existentes en la actualidad.

5. ARQUITECTURA DE REDES ACTIVAS

En esta sección se presentan las principales ideas de la arquitectura de red activa que está siendo desarrollada por DARPA (Defense Advanced Research Projects Agency) dentro del programa de redes activas[21]. Esta arquitectura tiene como propósito definir cuales son los principales componentes de una plataforma de red activa y los interfaces entre éstos. Uno de sus requisitos es permitir que exista más de una API de red. Esto tiene varios objetivos. Por un lado, actualmente han sido propuestas varias APIs de red y todavía no existe la experimentación suficiente como para decidir cual es la mejor o incluso si debe existir una única API de red. Por otro lado, permite un marco en el que incorporar el "procesamiento rápido" de paquetes que se ha comentado previamente. Y por último, resuelve problemas de incompatibilidad, dado que la funcionalidad de IPv4 o IPv6 puede ser vista simplemente como otra API de red.

La funcionalidad de un nodo de la red activa se divide entre los entornos de ejecución (EE) y el sistema operativo del nodo (NodeOS). En la figura 19 se muestra la organización general de estos componentes que a continuación se analizan con más detalle.

[21] Complementar con la pagina web http://www.darpa.mil/ito/research/anets

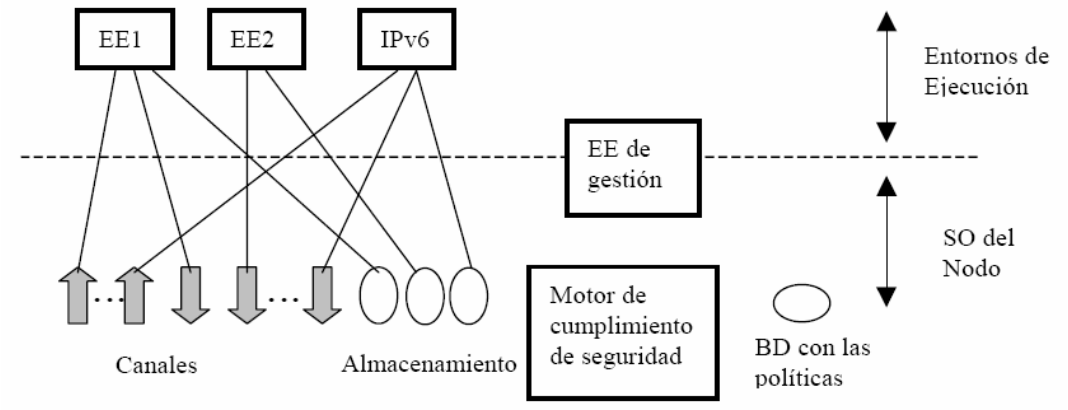

Figura 19. Entornos de ejecución

5.1. ENTORNO DE EJECUCIÓN

Cada EE exporta un API de red o máquina virtual que los usuarios pueden programar o controlar enviándole paquetes. En general, la ejecución de estas instrucciones puede causar que haya una actualización en la información de estado del EE o incluso del NodeOS, y puede causar que el EE transmita uno o más paquetes, ya sea inmediatamente o pasado un cierto tiempo.

Posibles EEs residentes en un nodo podrían ser: ANTs que sería un EE de propósito general o el Smart Packets que sería un EE específico de gestión de red. El EE oculta al NodeOS los detalles de la interacción con los usuarios. Cada nodo activo dispone de un entorno de ejecución de gestión a través del cual se controlan ciertos aspectos de la configuración del nodo. Por ejemplo, el EE de gestión puede permitir la modificación de la base de datos con la política de seguridad del nodo (ver figura 22) o puede permitir el arranque de otros EEs.

5.2. SISTEMA OPERATIVO DEL NODO

El NodeOS es el nivel intermedio que opera entre los EEs y los recursos físicos (transmisión, computación y almacenamiento). El NodeOS proporciona la funcionalidad básica sobre la cual los entornos de ejecución construyen las abstracciones presentadas al usuario. El NodeOS gestiona los recursos del nodo activo y media entre los distintos EEs cuando se solicitan dichos recursos. El NodeOS aísla a un entorno de ejecución de los efectos que puede tener el comportamiento de otros entornos de ejecución.

Los usuarios y otras entidades de la red se representan por una abstracción denominada "principal". Esta abstracción se utiliza en temas de seguridad y de contabilización de uso de recursos. Las políticas de seguridad se definen con relación a los principales y el NodeOS es responsable de que se cumplan tales políticas. Cuando un EE solicita un servicio del NodeOS, está petición se acompaña del identificador (y posiblemente una credencial) del principal, en cuyo nombre se está realizando la petición. Dicho principal puede ser el propio EE u otro (por ejemplo, un usuario) en cuyo nombre actúa el EE. El NodeOS presenta esta información al motor de cumplimiento de la seguridad que verifica la identidad (ver figura 22) y autoriza o no al principal a obtener el servicio solicitado.

Procesamiento de los paquetes: el NodeOS implementa la abstracción de *canal de comunicaciones* sobre la cual se envían y se reciben paquetes. Existen dos tipos de canales: los "*anchored"* y los *"cutthrough*". Los canales "anchored" están anclados a un determinado EE y son la abstracción sobre la cual los EEs envían y reciben paquetes.

El flujo lógico de los paquetes a través de un nodo activo es el siguiente: cuando un nodo activo recibe un paquete desde un enlace físico, le clasifica en

base al contenido del mismo (normalmente con la cabecera), y de acuerdo con esto: el paquete será descartado o bien asignado a un determinado canal.
El NodeOS utiliza unos patrones para asociar cada paquete recibido a un determinado canal. Dichos patrones son proporcionados por el EE cuando crea dicho canal (por ejemplo paquetes con una determinada combinación de protocolo IP y puerto TCP).

Además de los canales "anchored" también pueden existir los canales "cuthrough" que por el contrario no están asociados a ningún EE. Cuando un NodeOS recibe un paquete sobre un canal de este tipo, dicho paquete no será interceptado ni procesado por ningún EE, y la única operación que se realizará con dicho paquete será la de almacenamiento-reenvío, típica de una red *pasiva*. Cómo se puede observar esta abstracción se utiliza para implementar el procesado rápido de paquetes ya antes mencionado.

5.3. PROTOCOLO DE ENCAPSULAMIENTO (ANEP)

El Active Network Encapsulation Protocol (ANEP)[22] proporciona los mecanismos para que un usuario pueda especificar a que EE particular debe dirigirse el paquete que está generando. Un campo en la cabecera ANEP es un identificador del EE que debe ejecutar dicho paquete. En la actualidad estos identificadores de EE son asignados por la Autoridad de Números Asignados para Redes Activas.

Es interesante señalar que un paquete que no contiene una cabecera ANEP puede ser procesado por un EE siempre que dicho EE cree el canal "anchored" adecuado. Este podría ser el caso de un paquete que ha sido generado en un sistema final no activo y se desea que se procese en un EE. Un ejemplo de aplicación podría ser un servicio para mejorar el control de congestión de TCP implementado en un EE.

[22] S. Alexander, A. D. Keromytis. Active Network Encasulation Protocol (ANEP). 1997

5.4. ÁREAS DE APLICACIÓN DE REDES ACTIVAS

El control dinámico de la red activa permite que los servicios "se ajusten a medida", de las aplicaciones y de las condiciones cambiantes de la red. Este ajuste tiene como objetivo mejorar el rendimiento percibido por las aplicaciones, en comparación con las soluciones punto-a-punto.
Los esfuerzos de mejorar el rendimiento de las aplicaciones de la red de datos convencional usando las redes activas cubren una amplia variedad de áreas, entre estas áreas las que parecen más prometedoras son las siguientes.

5.4.1. ADAPTACIÓN DINÁMICA

Un área interesante de investigación está relacionada con la adaptación dinámica a las condiciones cambiantes de la red.
Principalmente, las redes activas pueden aportar mejoras de rendimiento significativas en situaciones donde sea crucial la respuesta rápida ante cambios en la información local (en un nodo o conjunto de nodos de la red).
Por ejemplo, la calidad de servicio obtenida por una aplicación se puede degradar de forma significativa, en situaciones de congestión en la red o de enlaces con una tasa alta de pérdida de paquetes. La aproximación tradicional para resolver este tipo de situaciones ha sido que la fuente se vaya adaptando a las condiciones de la red, pero este planteamiento tiene una serie de limitaciones como es el tiempo que necesita la fuente para detectar el cambio y reaccionar en consecuencia. Al introducir en los nodos inteligencia sobre como adaptarse a los cambios en la red, se hace que la adaptación se produzca antes con el consiguiente aumento de rendimiento.

En esta área ya se han obtenido resultados interesantes en el Protocolo Boosters[23]. En este trabajo se muestra como un protocolo se puede adaptar a condiciones tales como un aumento en la tasa de pérdidas en la red añadiendo dinámicamente funcionalidad de corrección de errores alrededor del área de la

[23] D. Feldmeier, A. McAuley, J. Smith, D. Bakin,W. Marcus and T. Raleigh. *"Protocol booster".* Revista IEEE, 1998.

red donde se están produciendo las pérdidas. Otros trabajos en esta área, relacionados con control de congestión son: el de Zegura[24] que ha propuesto una estrategia inteligentes para el descarte de paquetes en situaciones de congestión para preservar la calidad de vídeo MPEG en esos periodos, y el de Faber[25] que ha planteado como introduciendo en la red inteligencia para actuar en situaciones de congestión, TCP puede mejorar su rendimiento en un 18%.

5.4.2. GESTIÓN DE RED

La manera tradicional de realizar la gestión de una red, consiste en recolectar información de los nodos gestionados, en base a solicitar el valor de unas variables determinadas y comprobar si se detectan anomalías. Este enfoque concentra la inteligencia en las estaciones de gestión, lo que puede provocar cuellos de botella. Además esta aproximación limita seriamente la posibilidad de seguir la pista a problemas detectados, en escalas de tiempo razonables. El principal grupo que ha realizado aportaciones en esta área ha sido BBN con su proyecto Smart Packets[26]. Este grupo ha planteado que la programación de la red hace posible que los nodos gestionados sean así mismo nodos programables. De esta forma, los centros de gestión pueden enviar programas a los nodos gestionados. Este enfoque aporta tres ventajas. Primero, la información de gestión que vuelve al centro de gestión puede ajustarse en tiempo real a los intereses que en un momento dado tenga dicho centro de gestión, reduciendo por tanto, el tráfico y la cantidad de información que debe analizar dicho centro de gestión. Segundo, muchas de las reglas de gestión empleadas por el centro de gestión pueden ser incluidas en programas que se envían al nodo gestionado, permitiéndole identificar y corregir problemas automáticamente sin la intervención del centro de gestión. Tercero, permite hacer más cortos los ciclos de monitorización y control.

[24] Ellen W. Zegura. *"An Architecture for Active Networking"*. High Performance Networking, 1997.
[25] T. Faber. Using Active Networking to Enhance Feedback Congestion Control Mechanisms. IEEE Networks,1998.
[26] Complementar con la pagina web http://www.nettech.bbn.com/smtpkts/smart.ps.gz

6. PROPUESTAS TECNOLOGICAS SOBRE REDES ACTIVAS

6.1. PROYECTO ANDROID, MODELADO Y SIMULACIÓN DE UN NODO ACTIVO[27]

Este proyecto es realizado en la Universidad Politécnica de Madrid, con la colaboración de un grupo de investigadores integrados por: Pablo Mage, Amalio F Nieto, Mercedes Garijo pertenecientes al departamento de Ingeniería de Sistemas Telematicos.

El proyecto ANDROID (Active Network Distrubuted Open Infraestructure Development), el cual implementa una infraestructura de red programable, escalable y gestionada, basada en la alternativa de redes activas. En la infraestructura de red activa se distinguen dos tipos de nodo: los routers activos (ARs) y los servidores activos (ASs), esta investigación describe el diseño y la implementación del modelo de simulación de nodo AS, usando como herramienta de modelado el OPNET modeler.
En el proyecto ANDROID es patrocinado por el programa IST de la Comisión Europea, que demostró que es posible obtener una infraestructura programable, escalable y gestionada, basando sus esfuerzos en la arquitectura de las Redes Activas. El grupo de investigadores de la Universidad Politécnica de Madrid participó en el diseño e implementación de modelos de simulación, modelos diseñados para predecir comportamientos de los elementos activos de la infraestructura de red. A continuación se describirán los conceptos básicos, características y trabajos de investigación en Redes Activas. Posteriormente se realiza una descripción de la implementación del modelo de simulación del nodo AS.

Las Redes activas son una novedosa propuesta en la cual programas “personalizados” son ejecutados dentro de las redes convencionales IP. Estos

[27] Tomado de bibliografía [4], Modelado y Simulación de un Nodo Activo.

programas pueden ser ejecutados en nodos intermedios denominados nodos activos.

Como resultado de la ejecución de los programas “personalizados” se puede modificar el comportamiento o el estado de los nodos visitados. Una arquitectura común de nodo activo ha sido definida dentro del programa de Redes Activas de DARPA. En esta arquitectura cada nodo activo contiene su propio sistema operativo (NodeOS) y varios entornos de ejecución (EEs). El NodeOS es responsable del control y manejo de los recursos y de la seguridad del nodo activo. Además, el EE es el responsable de facilitar las condiciones necesarias para la ejecución de programas activos sobre el nodo activo. Los usuarios debidamente habilitados pueden interactuar con el nodo activo para solicitar la ejecución de un programa activo (servicio activo).

6.1.1. Infraestructura de la red activa en el proyecto ANDROID.

A. El Proyecto ANDROID.

En el proyecto IST ANDROID (IST 1999-10299), se implementó una infraestructura de red programable, escalable y gestionada, basando sus esfuerzos en la alternativa de las Redes Activas.
La responsabilidad del grupo de trabajo de la Universidad Politécnica de Madrid estuvo enmarcada en el campo del modelado y simulación del sistema real que se implementó por parte de los otros socios del proyecto ANDROID. Dentro del entorno de trabajo de este proyecto se desarrolló una simple Red Activa, con el fin de probar de forma real una infraestructura de gestión para este novedoso tipo de red. Con el propósito de obtener resultados sobre Redes Activas más grandes y complejas, se recurrió a la simulación fuera de línea como un mecanismo para predecir resultados. En esta actividad de simulación, se obtuvieron los siguientes modelos de red:

- Modelo de la Red Activa en ANDROID, modelado del prototipo real de Red Activa que utilizó en este proyecto.
- Modelo del Servidor activo, es el nodo que tiene la mayor responsabilidad en el desempeño de la Red activa.
- Modelo de una granja de servidores activos, por medio de este modelo es posible predecir el comportamiento de un grupo de servidores activos, determinando de esta manera la capacidad de servicio de dicha granja.

B. Arquitectura de Red Activa en el Proyecto ANDROID.

La arquitectura de Red Activa en el proyecto ANDROID posee dos clases de nodos activos: Los Servidores Activos (ASs) y Encaminadores Activos (ARs). Los ARs además de poseer las funcionalidades básicas de un Router convencional, adicionan un entorno de ejecución (EE) que le permite ejecutar programas activos, que podrán operar sobre el contenido de los paquetes que fluyen por el router. Dichas operaciones determinan el trato que se le debe dar a dichos contenidos. La arquitectura de los ASs, se basa en arquitectura ALAN (Application Layer Active Networking)[39]. En él se ejecuta el programa activo. Este programa activo es implementado con el objeto de prestar algún tipo de servicio al cliente.

La arquitectura ALAN consta de clientes (browsers) y servidores (Servidores Web) desplegados en Internet. La conexión entre los servidores y clientes se hace a través de DPS (Dynamic Proxy Servers), y las entidades programables/ descargables son denominadas proxylets (figura 20). Los proxylets son almacenados (como archivos JAR[28]) en servidores Web y son obtenidos vía una URL. Existe una implementación denominada 'FunnelWeb' y una infraestructura ha sido desarrollada a través del proyecto ALPINE[29].

[28]Complementar con la pagina web http:/ http://es.wikipedia.org/wiki/Jar
[29] A. Ghosh, M. Fry, and C. Crowcroft, An Architecture for Application Layer Routing 2000, Pag 71-86.

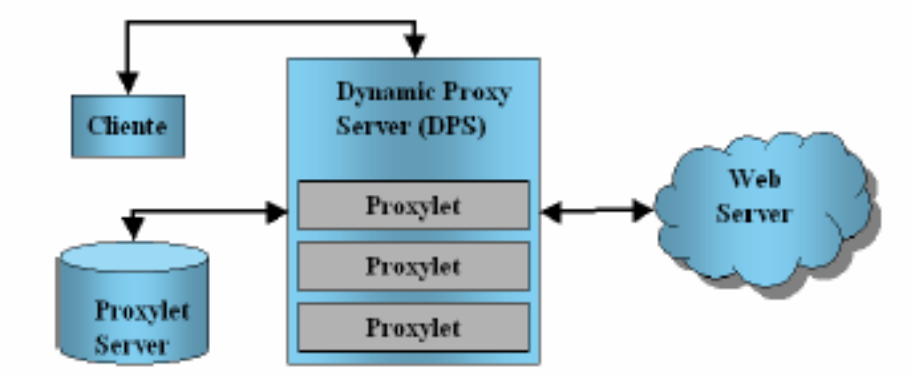

Figura 20. Arquitectura ALAN

C. Desarrollo de una CDN en ANDROID

.

Para mejorar la búsqueda y ubicación de los proxylets en la red se recurre a aplicar la filosofía de las redes CDN (Content Distribution Network).

En el proyecto ANDROID se implementa una arquitectura similar a la CDN (figura 21). La CDN en este contexto proporciona búsqueda y rápida localización de los proxylets. Por medio de la CDN se distribuyen los proxylets requeridos en el sistema, junto a los datos de información de ese proxylet (metadatos del proxylet). Esta CDN consta principalmente de tres elementos:

1. ISVs (Independent Software Vendors), los cuales son los proveedores de las implementaciones de servicios.

2. Los Clientes, los cuales buscan y solicitan un proxylet determinado.

3. PBN (Proxylet Broker Network), la cual es una red conformada por servidores dedicados, que en el contexto de este proyecto son llamados PBs (Proxylet Brokers), los cuales proporcionan las funcionalidades de almacenamiento, búsqueda y recuperación de proxylets.

Los PB interactúan con los ISVs para intercambiar información y descargar los proxylets y los metadatos de proxylets. Un PB está ubicado en cada sitio de la red donde existe al menos un AS. En este proyecto se tienen "sitios" conectados a través de una VPN (Virtual Private Network). De esta manera los ASs proporcionan un soporte a nivel de aplicación para servicios activos a usuarios participantes en la VPN.

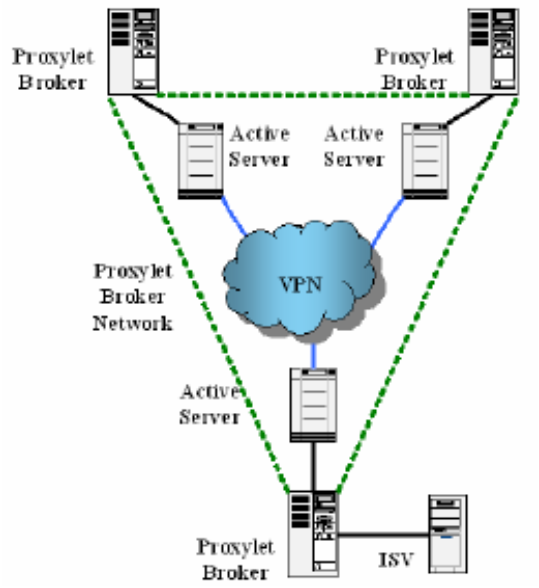

Figura 21. Infraestructura CDN en el proyecto ANDROID

D. El Sistema de Gestión en ANDROID.

La plataforma de Gestión que se implementó se basa en un sistema de gestión distribuida soportado por políticas, eventos, y programación activa. De esta manera se permite una monitorización y control de los nodos y servicios activos de manera automática. Las decisiones de configuración se basan en la información local disponible y en un conjunto de reglas, que determinan el comportamiento que se desea imprimir al elemento o al sistema gestionado. El conjunto de estas reglas constituye las políticas de gestión. Las políticas son escritas en XML (eXtensible Markup Languaje). Las políticas que se

implementan se orientan sobre tres frentes principales: políticas para la gestión de los ARs, políticas para la gestión de los ASs, y políticas para la gestión de la infraestructura de soporte.

E. Escenario de Prueba en ANDROID.

El objetivo del escenario de prueba en el que se trabajó en el proyecto ANDROID fue implementar una sesión de video conferencia usando la infraestructura descrita anteriormente (Figura 24). En este escenario de prueba (figura. 22) se observan 2 sitios, que toman parte de la VPN. Para controlar el funcionamiento de la VPN se cuenta con un sistema de gestión de VPN. Este sistema es el encargado de configurar de manera dinámica a los ARs y a los routers convencionales con el fin de establecer la VPN. Esto significa que los túneles IPSEC se configurarán o borrarán de acuerdo a los requerimientos de los clientes. Cada "sitio" de esta VPN consta básicamente de un AR, un servidor que contiene la herramienta que genera el tráfico de video (VIC), un AS donde se ejecuta el proxylet usado en esta aplicación (Trascoding Active Gateway - TAG), y dependiendo de su posición en la red, se requerirá el uso de un router convencional.

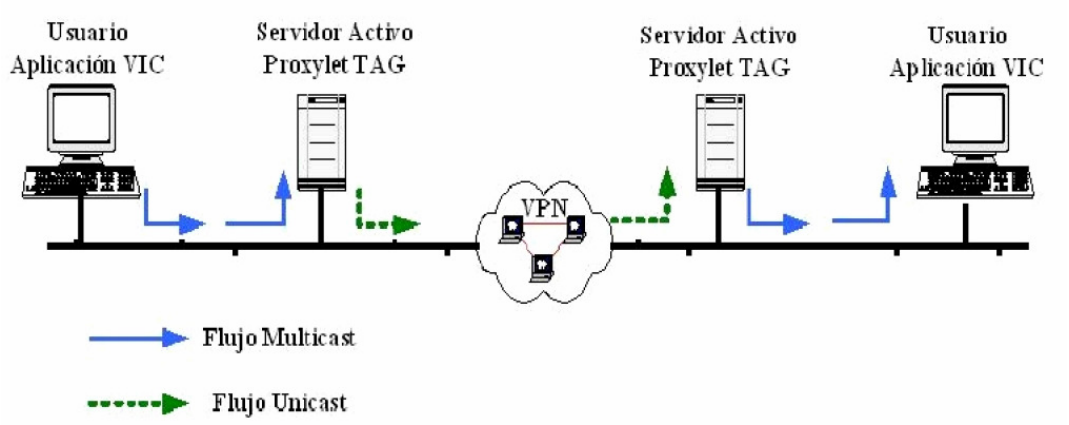

Figura 22. Escenario de prueba en ANDROID

El tráfico de video originado por la herramienta VIC es enviado hacia la red. En el trayecto este tráfico es detectado por un Router activo (AR), el AR enruta este tráfico hacia el AS (figura 25) donde es procesado por el proxylet (TAG) en un sitio local. Este tráfico multicast es encapsulado en paquetes unicast. Dinámicamente se inicia el proceso para establecer una VPN. Con la VPN establecida, el tráfico unicast es reenviado hacia el AR el cual refleja este tráfico hacia los diferentes sitios de la VPN. En los sitios de destino los paquetes unicast son tomados para desencapsular los paquetes multicast por parte del proxylet (TAG) donde posteriormente es distribuido localmente en forma de tráfico multicast.

6.1.2. Modelado del servidor activo (AS)

El Servidor Activo (AS) (figura 23) es el elemento de esta red que proporciona un entorno de ejecución seguro para los proxylets que el usuario requiere. El AS dentro del sistema interactúa de manera directa con el AR, y de forma indirecta con los usuarios y el PB. Para realizar su propia gestión y proporcionar seguridad, se proveen los servicios del MID (Management Information Distribution), DS (Directory Service) y el RM (Resource Manager). Estos servicios corresponden, cada uno, a un módulo en el modelo de nodo del AS (figura 26).

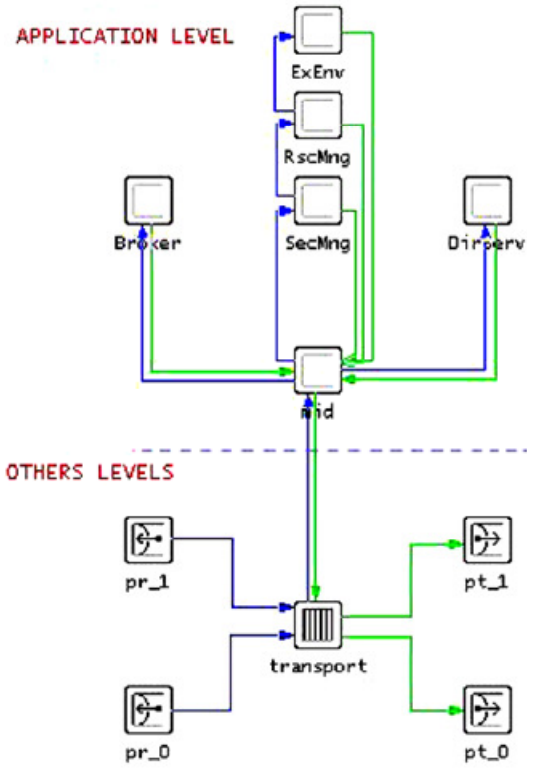

Figura 23. Modelo del servidor activo

A continuación se describen los módulos que componen el modelo del AS.

A. Infraestructura del Servidor Activo.

Con base en los servicios de gestión relacionados con el AS, se implementan una serie de módulos, que proveerán la funcionalidad de estos servicios en el modelo desarrollado. El Servidor Activo en el modelo de nodo esta compuesto por los siguientes módulos (figura 23):

- Management Information Distribution (MID)
- Broker
- Directory Service (DS)
- Security Manager (SM)

- Resource Manager (RM)
- Environment Execution (EE)
- Transport
- Transceivers

B. Management Information Distribution (MID)

Su función básica es el manejo del flujo de eventos entrantes y salientes circulantes, para establecer la correcta intercomunicación entre los diferentes módulos que componen el Servidor Activo. Cuando un evento llega a este módulo, se busca la política relacionada con este evento, y dependiendo de dicha política el evento es redireccionado al módulo respectivo.

C. Broker.

Este módulo es responsable de almacenar los metadatos del proxylet, y del proxylet, usando para el primer caso al DS, también atiende las consultas realizadas por usuarios que estén autorizados, acerca de información relacionada con un determinado proxylet y por ultimo hacen la distribución de los metadatos del proxylet a todos los elementos que pertenecen a la CDN.

D. Directory Service (DS).

El DS es usado para almacenar los metadatos del proxylet, también se asume que el almacenamiento del código del proxylet es también manejado por este módulo.

E. Security Manager (SM)

Su función básica es la validación de los eventos que fluyen hacia el entorno de ejecución. Al igual que los módulos anteriores el SM posee un almacén de políticas. Cuando un evento llega a este módulo se carga la política respectiva con el fin validar las operaciones a realizar o el proxylet implicado.

F. Resource Manager (RM).

El RM es el encargado de verificar la disponibilidad de recursos que se requieren para el mantenimiento de la ejecución de un proxylet. Si existen los recursos suficientes, este módulo se comunica con el entorno de ejecución (EE) para iniciar o continuar con la ejecución del proxylet.

G. Environment Execution (EE).

El EE es el responsable del control del ciclo de vida del proxylet durante su ejecución. En esta implementación se modela un proceso padre (figura 24). Este proceso padre genera y controla procesos hijos mediante una petición por demanda. Un proceso hijo representa el ciclo de vida de un proxylet. El modelo de proceso de un proceso hijo se observa en la figura 25.

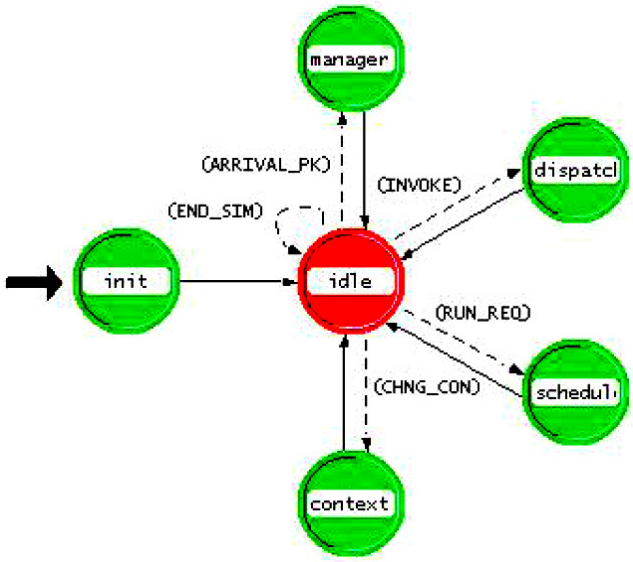

Figura 24. Modelo del proceso EE

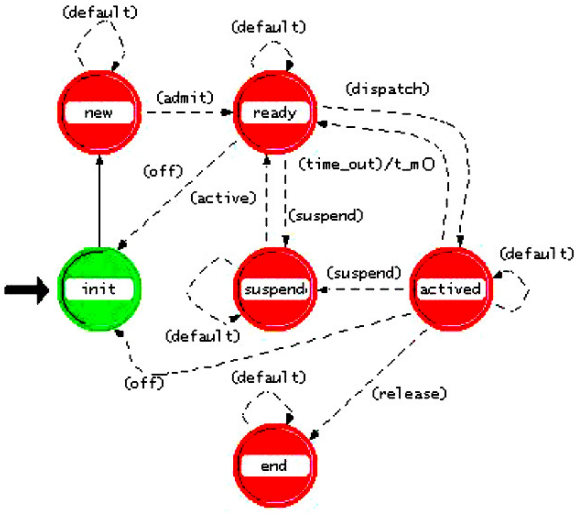

Figura 25. Modelo del proceso Proxylet

6.2. MECANISMOS DE SEGURIDAD EN REDES ACTIVAS SOBRE ARQUITECTURA SARA[30]

Enmarcado en el contexto del proyecto europeo IST-GCAP (2000-2001) se ha desarrollado en la Universidad Carlos III, una plataforma básica de redes activas denominada SARA (*Simple Active Router-Assistant)* que opera sobre redes IPv4 e IPv6. En esta plataforma, la descarga de código de una aplicación activa (AA) en un nodo activo se hace de forma dinámica cuando llega el primer paquete activo que hace referencia a dicha AA, y se realiza desde uno o varios servidores de código administrados por el proveedor de red. Esta aproximación asegura que en la red activa solamente se ejecutarán aquellas AA que han sido previamente validadas antes de su habilitación por el proveedor o administrador de la red activa, y no el código inyectado por cualquier usuario anónimo como en otras plataformas.

Otro elemento importante en esta arquitectura es el concepto de Router-Asistente (figura 26), desarrollado por primera vez en la plataforma SARA. En esta arquitectura el router delega las funciones de procesado activo en un asistente presente en una red local de alta velocidad como si de un

[30] Tomado de bibliografía [5]. Bagnulo Marcelo,Calderón Maria, Alarcos Bernardo, Larrabeiti David, Departamento de Ingeniería Telemática. Universidad Carlos III de Madrid, Área de Ingeniería Telemática, Universidad de Alcalá de Henares

coprocesador externo se tratara. Esta decisión de diseño tiene implicaciones sustanciales, en primer lugar, permite que la ejecución de aplicaciones activas en la red tenga un impacto mínimo en las prestaciones ofrecidas por el router que debe poder transportar también paquetes convencionales (no activos) con el máximo rendimiento, por otro lado permite dimensionar la capacidad de proceso del nodo activo sustituyendo únicamente el Asistente si fuera preciso. Por último, permite que el router esté blindado frente a potenciales errores en la programación de las AA que podrían afectar al Asistente pero no al router.

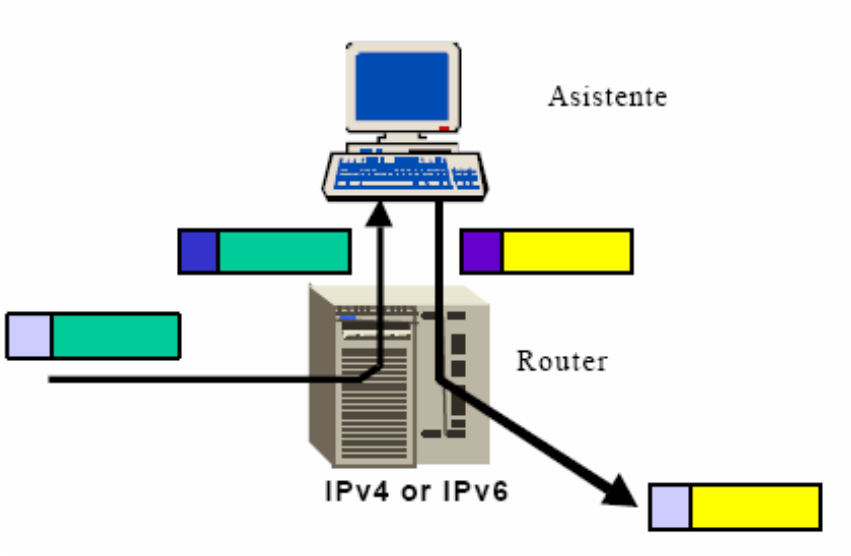

Figura 26. Arquitectura Router-Asistente

La plataforma desarrollada ofrece servicios activos transparentes. Este concepto tiene varias vertientes; por un lado es totalmente transparente para los paquetes tradicionales (paquetes pasivos) el hecho de que coexistan en la red con los paquetes activos y que existan en la red determinados nodos con soporte de redes activas. Por otro lado la topología de red activa es transparente para los sistemas finales, esto significa que estos últimos no están obligados a conocer la ubicación física de los nodos activos para poder hacer uso de sus servicios. Serán los propios nodos activos los que se encarguen de capturar los paquetes activos que los atraviesan. Esta funcionalidad se implementa haciendo uso de la opción *router alert* de IP.

En la actualidad SARA[31] es una plataforma de redes activas de alto rendimiento consistente en un prototipo de experimentación que soporta las funcionalidades básicas de una plataforma de este tipo.

Si bien el esquema de funcionamiento de las redes activas ofrece una importante flexibilidad, que permite ofrecer una diversa gama de servicios de forma dinámica, presenta también, por su propia estructura de funcionamiento, una serie de riesgos de seguridad, algunos de los cuales pueden ser muy graves y poner en riesgo el correcto desempeño de toda la red, por lo que intenta proponer una solución a los mismos mas adelante.

A continuación se describirá el intercambio básico de paquetes activos en una red activa con arquitectura SARA para luego analizar los riesgos existentes y los requisitos de seguridad que estos imponen así como las condiciones de borde impuestas por restricciones de otro orden como puede ser la escalabilidad del sistema. En una segunda instancia se presentará una solución que pretende cumplir con los requisitos detectados para luego evaluar la implementación de la misma utilizando tecnología existente y así dar paso a las conclusiones extraídas.

6.2.1. Intercambio básico de paquetes activos en SARA

A continuación se hace referencia a los elementos que participan en el Intercambio de paquetes:

- *Origen*: ordenador del usuario de la red activa que genera tráfico en la misma y utiliza las facilidades activas que esta ofrece.
- *Destino*: Ordenador al cual está dirigido el tráfico generado por *Origen*.
- RACT: router activo (router + asistente) capaz de interpretar paquetes activos que circulan en la red y obtener el código necesario para procesarlos.

[31] Complementar con la pagina web http://matrix.it.uc3m.es/~sara

- *Servidor de código:* suministra el código ejecutable utilizado por los routers para procesar las distintas clases de paquetes que circulan en la red.

Para hacer uso de las prestaciones ofrecidas por la red activa para el procesamiento particular de una clase de paquetes se emplea el siguiente mecanismo que es básico: *Origen* debe solicitar dicho servicio a la red (ver figura 27). Esta solicitud se realiza mediante el envío de un paquete activo (ACT[1]), dirigido hacia el destino final deseado (*Destino*), y que adicionalmente indica a los routers activos del camino el código necesario para el procesamiento de los paquetes de esta clase. Cuando un router activo recibe un paquete de este tipo, verifica la disponibilidad local del código solicitado. En caso que no posea el código necesario para procesar los paquetes, solicita el mismo al Servidor de Código (CODREQ [2]). El Servidor de Código envía entonces la información solicitada al router activo (COD[3]), quien puede ahora procesar el paquete activo y encaminarlo hacia el próximo salto (ACT[4])). Los siguientes routers activos del camino realizarán un procedimiento análogo hasta que el último lo encaminará hasta *Destino*, quien ignora la información concerniente al tratamiento del paquete y extrae la información para las capas superiores.

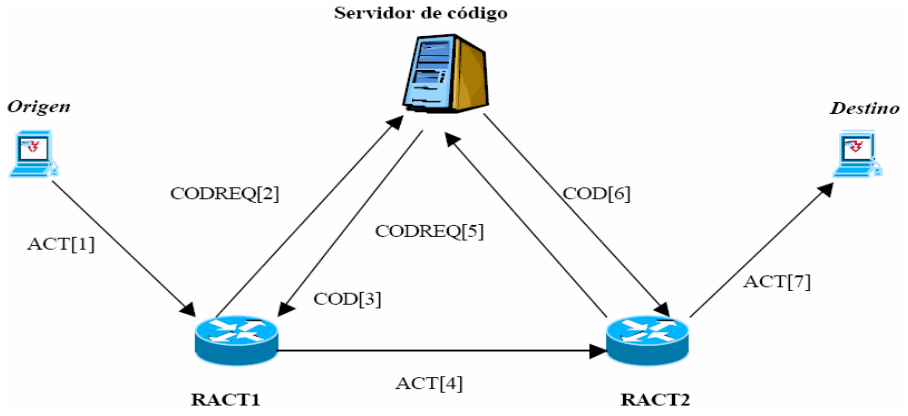

Figura 27. Entorno de trabajo

Una vez que se ha establecido el camino y todos los routers contienen el código necesario para procesar los paquetes como lo ha solicitado *Origen*, este debe informar a la red la intención de continuar utilizando este código de procesamiento de paquetes. Para ello, *Origen* debe enviar periódicamente paquetes de refresco (Refresh) del código en uso. Los routers activos verifican el código solicitado y extienden su tiempo de vida en el router.

6.2.2. Análisis de riesgos y requisitos de seguridad

Es claro que el principal tema a resolver se vincula al control de acceso. Ya sea el control de acceso de los distintos *Origene*s a los códigos solicitados así como el control de cuales routers activos tienen permisos para acceder a los distintos códigos solicitados. También resulta de principal importancia controlar que servidores de código pueden introducir código en los routers activos. Algunos de los requisitos para brindar seguridad son:

Autenticación e integridad: el riesgo más evidente que parece amenazar a la arquitectura descrita, es la posibilidad que alguna parte no deseada pueda cargar código en los routers activos de la red, por lo que parece imprescindible que los mensajes que contienen el código (COD) a ser introducido, sean autenticados por parte del router activo de forma de estar seguros que fue el servidor quien generó dichos paquetes. Otro riesgo existente es la posibilidad que un intruso altere el código contenido en el paquete (COD) mientras viaja hacia el router, por lo que también resulta necesario que dicho paquete posea una verificación de integridad asociada a la autenticación. Adicionalmente, por razones de servicio, resulta deseable que sólo partes autorizadas sean capaces de ejecutar ciertos códigos, ya sea porque los mismos son potencialmente peligrosos para la red, por ejemplo requieren muchos recursos del router, por lo que también es necesario poder identificar a *Origen* cuando solicita la ejecución de un código particular (ACT). Análogamente al caso

anterior, resulta necesaria una verificación de la integridad de la solicitud realizada.
Finalmente, es posible pensar que algunos de los códigos existentes en el servidor no sean de dominio público por lo que sería deseable que sólo routers activos autenticados pudieran acceder al mismo

Confidencialidad: como se menciono anteriormente, parecería interesante que el código intercambiado no pudiera ser accedido por terceras partes que tienen acceso a la red, por lo que dicha comunicación (COD) debería ser confidencial. Los otros intercambios podrían ser confidenciales por razones menos críticas, como la privacidad de las intenciones de los clientes de los distintos servicios pero este puede ser un requisito opcional para el resto de los paquetes

No repudio: es posible imaginar que los operadores de las redes activas deseen cobrar de forma diferenciada en función de la forma de procesamiento de paquetes que sea solicitada a los routers activos mediante los paquetes ACT, por lo que dichos paquetes tendrán implicaciones comerciales y contractuales, específicamente jugará el rol de solicitud de servicio. Por lo tanto es interesante poder garantizar el no repudio de dichos paquetes por parte de *Origen*, ya que la tarificación del servicio se realizará en función de esto.

Retransmisiones: el proceso de carga de código en los routers activos, es exigente en memoria y en capacidad de procesamiento, por lo que es posible imaginar un ataque a los routers de la red simplemente retransmitiendo paquetes válidos de un origen auténtico, lo que aumentaría la demanda en el router e incluso podría saturarlo. Por ello es importante que los paquetes que pueden cargar código (ACT) posean una validez temporal de forma que luego de un lapso ya no sean más válidos y sean descartados por todas las partes. Asimismo sucede con las solicitudes de código al servidor (COD) ya que se

podría imaginar un ataque al mismo solicitando múltiples veces código ya pedido

Conocimiento nulo de usuarios por parte de los routers activos: resulta imprescindible diseñar el sistema de forma escalable, considerando una cantidad importante de potenciales usuarios, por lo que no resulta posible considerar una administración de permisos de usuarios local en cada router activo. Es necesario por consiguiente, la utilización de un servicio de directorio que contenga toda la información de usuarios y permisos, donde no sea necesaria la información de usuarios en los routers.

Transparencia frente a los routers activos que forman el camino: resulta importante que los distintos *Orígene*s no deban conocer los routers que formen el camino hasta el destino, para brindar transparencia de la red e independencia de la topología de la misma, así como asegurar la escalabilidad, por lo que los paquetes activos enviados (ACT) no deben depender de los routers por los que deban transitar.
Este requisito es particularmente fuerte parta un sistema de seguridad donde se requiere autenticación y confidencialidad con una parte a la cual no se conoce.

Eficiencia y velocidad en el procesamiento: el objetivo principal es la transmisión de la información por lo que la velocidad en el procesamiento de los paquetes debe ser óptima y si bien los requisitos de seguridad son relevantes, la eficiencia en el uso es la razón de ser.

6.2.3. Arquitectura de seguridad

Para el funcionamiento de la solución propuesta, resulta necesario contar con los siguientes elementos:

Origen: un par de claves asimétricas, pública (PubO) y privada (PrO) y un certificado digital que asocie dichas claves a *Origen.*

RACTs: par de claves asimétricas, pública (PubR1 y PubR2) y privada (PrR1 y PrR2) y certificado digital que asocie dichas claves al router correspondiente.

Servidor de Código: par de claves asimétricas, pública (PubSC) y privada (PrSC) y un certificado digital que asocie dichas claves al Servidor de Código

.

Protocolo de seguridad en el intercambio básico de paquetes activos

ACT[1]: este paquete es especialmente crítico y debe ser autenticado, no repudiable, íntegro y con validez temporal para evitar retransmisiones, por lo que se propone que el mismo contenga un sello temporal (timestamp) que establezca la hora de creación del mismo y se firme digitalmente el contenido activo del paquete con la clave privada de *Origen* (PrO) de forma de asegurar autenticidad, integridad y no repudio. Adicionalmente, este paquete debe transmitir una clave simétrica (K), previamente generada por *Origen*, que será luego utilizada por los routers para autenticar los mensajes de Refresh. Dicha clave debe ser transmitida de forma secreta y como *Origen* solo conoce al Servidor de Código (no puede conocer a los routers por requisito establecido previamente) K se encripta con la clave pública del servidor (PubSC).

Contenido de ACT

- Dirección origen: *Origen*
- Dirección destino: *Destino*
- Identificación de código deseado
- Clave simétrica K
- Timestamp

Firmado por *Origen*

Confidencialidad: clave simétrica K encriptada por PubSC

CODREQ[2]: el router activo analiza el paquete activo, pero por el requisito impuesto de no conocimiento de *Origen* por los routers, este no puede verificar si el paquete no ha sido alterado. Verifica si posee o no el código solicitado y reenvía el paquete recibido al Servidor de Código encapsulándolo en otro paquete e indicando si posee o no el código solicitado. En caso de que el router ya posea el código, este intercambio es utilizado para obtener autorización de acceso al código por parte de *Origen*.

Contenido de CODREQ

Dirección origen: RACT1
Dirección destino: Servidor de Código ACT
Indicación si desea o no el código

COD[3]: cuando el Servidor de Código recibe la solicitud (CODREQ), este verifica la firma de *Origen* utilizando su clave pública (PubO) para luego comprobar si *Origen* posee permisos para realizar la presente solicitud. En caso afirmativo, comprueba el timestamp, de forma que el momento de creación del paquete esté dentro de un entorno aceptable del momento presente. Adicionalmente verifica que el router activo solicitante posea los permisos para ejecutar el código solicitado. Una vez realizadas las verificaciones, desencripta la clave K utilizando PrSC. Genera entonces el paquete COD en el cual incluye el código en caso que el router activo así lo solicite y un timestamp. Firma luego la información utilizando PrSC para luego encriptarlo con la clave K. Finalmente adjunta la clave K encriptada con la clave pública del router (PubR1) de forma que sólo éste pueda leer la información enviada.

Contenido de COD

Dirección origen: Servidor de código
Dirección destino: Router activo 1

Código solicitado
Clave simétrica K encriptada con PubR1
Timestamp
Firmado por SC

Confidencialidad para Router Activo 1 (encriptado por K, clave K encriptada con PubR1)

ACT[4]: una vez que el Router activo recibe COD, este desencripta la clave K utilizando su clave privada (PrR1) para luego desencriptar el resto del contenido del paquete utilizando la clave K. Posteriormente verifica la firma digital del servidor utilizando PubSC. Una vez superadas estas corroboraciones verifica que el tiempo degeneración del paquete (timestamp) se encuentre dentro de un entorno admisible del instante actual. En caso afirmativo, obtiene el código del paquete en caso que lo hubiera solicitado y asocia la clave K a dicho código, para luego cursar el paquete activo hacia el próximo salto. Los siguientes routers activos del camino realizarán un procedimiento análogo al descrito anteriormente hasta que finalmente el último de los routers encaminará el paquete hacia *Destino*. Este simplemente descartará la cabecera de seguridad, ya que no es capaz de comprenderla y la información contenida no le presenta interés alguno.

Paquetes de Refresh: una vez que se ha notificado a todos los routers activos del camino el código a utilizar, es necesario enviar paquetes que indiquen la extensión de la validez del código utilizado en el tiempo de forma que este permanezca en los routers. Dichos paquetes son generados por *Origen* y serán dirigidos a *Destino*; contienen la identificación del código deseado, una marca temporal y están firmados por *Origen*. Para que el proceso de verificación de firma sea menos exigente para los routers, se utiliza la clave K para autenticar los paquetes de refresh, adjuntando el hash del contenido del paquete concatenado con K.

Contenido de Refresh

Dirección origen: *Origen*

Dirección destino: *Destino*

Identificación de código deseado

Timestamp

Autenticación: hash del contenido del paquete concatenado con la clave simétrica.

Cuando los routers activos reciben el paquete de Refresh, verifican la firma comparando el hash contenido en el paquete con el hash resultante del contenido del paquete concatenado con la clave asociada al código, la cual obtienen de sus bases internas. Si además la marca temporal del paquete es correcta, extienden el tiempo de vida del código dentro del router y envían el paquete hacia el próximo router del camino. Una vez concluido el trayecto, *Destino* recibe el paquete, quien descarta la cabecera de seguridad.

CONCLUSIONES

Debido a la importancia del desarrollo de nuevos sistemas que utilizan la tecnología de agentes móviles, se realizo de manera favorable un estudio de esta tecnología, presentando sus características, ventajas, desventajas y posibles áreas en las que se aplican dichos agentes, también se estudiaron los ambientes y lenguajes de programación que permiten el desarrollo de este tipo de tecnología. Por otro lado, se describieron y se analizaron una serie de lenguajes que se están proponiendo como medio para la creación de agentes móviles como FIPA y KQML, los cuales cumplen con algunas de las características para la construcción de agentes móviles. Obteniendo con ello conocimiento nuevo, actual y sobre todo de gran ayuda para nuevos proyectos de desarrollo de software, brindando información de las arquitecturas que brindan la oportunidad de diseñar agentes móviles en base al comportamiento humano y la inteligencia artificial.

Cuando se revisan los diversos modelos y paradigmas de la inteligencia artificial (IA), llama la atención el hecho de que la gran mayoría de ellos surgieron y han sido utilizados en los últimos cuarenta años. Lo interesante de la situación actual, es el giro que ha tomado la aplicación de estos modelos y paradigmas en los sistemas distribuidos.

El elemento principal de este potencial de los esquemas de IA para su aplicación en el ámbito computacional es que estos esquemas permiten definir sistemas que abarcar las dos facetas de cualquier actividad, como son el actuar y el decidir. Al poder generarse sistemas que tomen decisiones y que puedan efectuar acciones en consecuencia de estas decisiones, estamos hablando de verdaderos sistemas automatizados o autómatas, que permiten liberar a los usuarios humanos de un sin fin de tareas que realmente no requieren de su intervención.

Por ultimo, Se presento una panorámica general de la tecnología de redes activas y se mencionaron algunas de sus potenciales aplicaciones.

Desde el punto de vista de los proveedores de servicio de red, ven que las redes activas es la tecnología que revolucionara los equipos y las plataformas de computo en las que actualmente nos desenvolvemos como es la Internet y las redes de telecomunicaciones brindando mayor seguridad y un menor ancho de banda, lo que permite reducir el tiempo necesario para desarrollar y desplegar nuevos servicios optimizando los recursos de red, y desde el punto de vista de los investigadores, las redes activas ofrecen una plataforma en la cual se puede experimentar con nuevos servicios de red de forma real, mientras que ésta continúa operando normalmente.

Por ende la utilidad de las redes programables puede resultar ventajoso en los distintos niveles, siendo las siguientes sus principales características: acelerar la evolución de las redes permitiendo que nuevos protocolos y servicios sean introducidos en la red sin la necesidad de largos procesos de estandarización y despliegue en la red, además permitir a los usuarios crear nuevos servicios o adaptar los ya existentes de acuerdo a las necesidades concretas de sus aplicaciones.

Sin embargo, las redes activas son un área investigación prometedora que en estos momentos se encuentra en una fase de prueba debido a que todavía no se ha realizado la suficiente experimentación y son muchas las líneas de investigación abiertas.

LISTA DE TABLAS Y FIGURAS

Pág.

GLOSARIO

Dominio: Conjunto de caracteres que identifican un sitio de Internet accesible por un usuario.

FTP: Los servidores FTP son grandes cajones de ficheros distribuidos y organizados en directorios. Contienen programas (normalmente de dominio público o shareware), ficheros de imágenes, sonido y video.

Hardware: Elementos físicos que componen o complementan a un ordenador o red.

HTML: Acrónimo de "Hipertext Markup Language". Lenguaje que permite escribir las páginas web a las que se accede a través de navegadores WWW.

HTTP: Protocolo de transporte de hipertexto, usado para acceder a un servidor HTTP, transferir y manejar páginas y documentos WWW.

Inteligencia Artificial: La IA o inteligencia artificial es la simulación de los procesos de la inteligencia humana por máquinas, especialmente sistemas de ordenadores.

IRMA: la Arquitectura de Máquina de Recursos Inteligentes (IRMA) (Bratman, 1988).

Jar: por sus siglas en inglés, **J**ava **AR**chive es un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Existen tres operaciones básicas con este tipo de archivos: ver contenido, comprimir y descomprimir.

Linux: Sistema operativo de fuente libre y compatible con UNIX, desarrollado por multitud de programadores de todo el mundo y coordinados por Linus Torvalds, quien ideó el proyecto.

Nodo: es un punto terminal de una red, o cualquiera de sus intersecciones, en una red de computadores cada una de las máquinas es un nodo, y si la red es Internet, cada servidor router o switch constituye también un nodo.

Pragmática: es el estudio del modo en que el contexto influye en la interpretación del significado. El contexto debe entenderse como *situación*, ya que puede incluir cualquier aspecto extralingüístico. La Pragmática toma en consideración los factores extralingüísticos que determinan el uso del lenguaje, esto es, todos aquellos factores a los que no se hace referencia en un estudio puramente gramatical.

Protocolo: Conjunto de reglas que gobiernan las comunicaciones entre sistemas de telecomunicación.

RFC: Una **R**equest **F**or **C**omments, que se traduce como "petición de comentarios", cada RFC tiene un título y un número asignado, que no puede repertirse ni eliminarse aunque el documento se quede obsoleto. Cada protocolo de los que hoy existen en Internet tiene asociado un RFC que lo define, y posiblemente otros RFCs adicionales que lo amplían. Por ejemplo el protocolo IP se detalla en el RFC-791, el FTP en el RFC-959, y el HTTP el RFC-2616.Las RFC se redactan en inglés según una estructura específica y en formato de texto ASCII.

RPC: *Remote Procedure Call*, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

Router: Un dispositivo (o programa de software) que maneja la conexión entre dos o más redes. Los ruteadores se encargan de buscar la dirección de destino de los paquetes que pasan por ellos y deciden hacia cual ruta enviarlos.

Semántica: El término semántica, se refiere a los aspectos del significado o interpretación de un determinado código simbólico, lenguaje o representación formal.

SOCKET: Un receptáculo en una placa madre de una PC diseñado para almacenar el microprocesador.

Softbots: están basados en la Inteligencia Artificial y destinados a facilitar operaciones frecuentes como la mantención de una agenda o la navegación por la WWW y recurren a los conocimientos acumulados en materia de realidad virtual y diseño de interfaces.

SOFTWARE: Denominación que reciben los programas informáticos.

SWITCH: es un dispositivo de *propósito especial* diseñado para resolver problemas de rendimiento en la red, debido a anchos de banda pequeños y embotellamientos. El switch puede agregar mayor ancho de banda, acelerar la salida de paquetes, reducir tiempo de espera y bajar el costo por puerto. Opera en la capa 2 del modelo OSI y reenvía los paquetes en base a la dirección MAC.

TCP/IP: Siglas de "Transmission Control Protocol/Internet Protocol" -- TCP/IP (Protocolo de Control de Transmisión/Protocolo Internet). Es el conjunto de protocolos de la comunicación básica en Internet.

UNIX: es un sistema operativo multiusuario y multitarea..

XML: eXtensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML).

BIBLIOGRAFIA

Libros y Artículos Científicos de Referencia

1. BARBARA M., Antoni, HESSELBACH S., XAVIER. Inteligencia de red. Barcelona: Ediciones UPC, 2002.

2. SILVESTRE., Jiménez, ESMERALDA Ramos. Agentes inteligentes. Caracas: Lecturas en ciencias de la computación, ISSN 1316-6239 2000.

3. MOLINA L., José M., Agentes y sistemas multiagente. Madrid: CEDITEC, 2004.

4. MAGE Pablo.,GARIJO Mercedes.,NIETO Amalio .,"Modelado y Simulación de un Nodo Activo", Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid [citado 15 de agosto del 2007].

5. BAGNULO Marcelo, CALDERÓN Maria, ALARCOS Bernardo, LARRABEITI David, "Mecanismos de Seguridad en Redes Activas sobre Arquitectura SARA", Departamento de Ingeniería Telemática. Universidad Carlos III de Madrid, Área de Ingeniería Telemática, Universidad de Alcalá de Henares.

6. JULIAN. V., BOTTI. V. Agentes inteligentes el siguiente pasó en la inteligencia artificial. May-jun 2000,[citado el 12 agosto 2007], Dpto. sistemas Informáticos y Computación Universidad Politécnica de Valencia.

7. GIRET. A, BOTTI V. Metodología Multi-Agente para Procesos Industriales. Feb 2003, [citado el 14 de agosto de 2007], Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia.

8. SEDANO. M[1], ALARCOS. B[1], CALDERÓN. M[2], LARRABEITI. D[2]. Caracterización de los enlaces de Internet utilizando tecnología de Redes Activas, [citado el 22 de agosto de 2007], [1]Área de Ingeniería Telemática., Dpto. Automática Escuela Politécnica. Universidad de Alcalá; [2]Dpto. de Ingeniería Telemática. Universidad Carlos III.

Paginas Web

Departamento de Informática, Univ. Carlos III de Madrid
http://giaa.inf.uc3m.es
Centro de Difusión de Tecnologías, Univ. Politécnica de Madrid
http://www.ceditec.etsit.upm.es
Agentes y Sistemas Multiagente Aspectos a abordar Arquitecturas
www.dirinfo.unsl.edu.ar/~sma/Teorias/teo2ag4.pdf
Agentes Móviles y CORBA
www.informatica.us.es/~ramon/tesis/CORBA/Seminario-MASIF/
Aglets para agentes móviles
www.maestrosdelweb.com/editorial/aglets-para-agentes-moviles/
Alfonso Jiménez: agentes moviles
www.alfonsojimenez.com/buscar?q=agentes+moviles
Agentes móviles y sus principales características
www.maestrosdelweb.com/editorial/agentes-moviles-y-sus-principales-caracteristicas/

Tecnología de agentes y sus aplicaciones

www.mipagina.cantv.net/vmendillo/Tesis/Agentes.htm

ARAMCEL: Arquitectura basada en Agentes Mviles para Comercio

www.energiaycomputacion.univalle.edu.co/edicion22/22art9.pdf

Alfonso Jiménez | Telescript

www.alfonsojimenez.com/2007/05/18-telescript

Java - Wikipedia, la enciclopedia libre

http://es.wikipedia.org/wiki/Java

Java Technology

http://java.sun.com

Pragmática - Wikipedia, la enciclopedia libre

http://es.wikipedia.org/wiki/Pragmática

Semántica - Wikipedia, la enciclopedia libre

http://es.wikipedia.org/wiki/Semántica

Introducción a las redes activas

www.edicionsupc.es/ftppublic/pdfmostra/TL02705M.pdf

Mecanismos de seguridad en redes activas sobre arquitectura SARA

enjambre.it.uc3m.es/~auras/papers/Seguridad%20en%20SARA-jitel01.PDF

Caracterización de los enlaces de Internet utilizando tecnología

enjambre.it.uc3m.es/~auras/papers/caract_enlace_jitel01.PDF

Redes activas" con IPv6

www.idg.es/comunicaciones/mainart.asp?artid=117863

TESIS MARIFELI SEDANO RUIZ

www.gsi.dit.upm.es/~marifeli/tesis.html

DARPA Active Networks

www.sds.lcs.mit.edu/darpa-activenet/

DARPA Active Networks Conference and Exposition 2002

www.informatik.uni-trier.de/~ley/db/conf/dance/dance2002.html

GE Active Networking Home

www.crd.ge.com/~bushsf/AVNMP.html