UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO DE UNA RED INTELIGENTE POR MEDIO DE UN SISTEMA DE ONTOLOGÍAS

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

CRISTIAN YOSHIDA HONMA

PROMOCIÓN
2004 - I

LIMA – PERÚ
2009

ESTUDIO DE UNA RED INTELIGENTE POR MEDIO DE UN SISTEMA DE ONTOLOGÍAS

SUMARIO

En la actualidad es imposible tener una red de computadoras sin una correcta administración y un mantenimiento elaborado. Es por esta razón que se busca tener un control inteligente sobre los recursos y servicios a los que los usuarios deben acceder. El presente trabajo pretende mostrar el concepto de ontologías por la cual el sistema de alguna forma emulará un entendimiento de las intenciones del usuario o usuarios. Para esto se verán una serie de herramientas como XML y Java, donde Jini es una de sus derivaciones.

Agradezco a mis padres, ya que gracias a ellos estudié la carrera de ingeniería en la mejor universidad del Perú.

INDICE

PRÓLOGO

El propósito del trabajo es realizar un estudio detallado de la problemática actual de las redes de computadoras. La necesidad de establecer una comunicación rápida y eficaz en el intercambio de información por medio de redes informáticas representa todo un reto en nuestra época, cada vez es más común ver redes de computadoras, no solo en el ambiente laborar, también se ve aplicaciones educativas y lúdicas.

En este trabajo se expondrá uno de los problemas más comunes en la búsqueda de información y asignación de servicios a través de una red informática.

Primero tenemos el caso de la búsqueda de información. En la actualidad existen motores de búsqueda que básicamente comparan la palabra o conjunto de caracteres que ingresamos y las busca en un banco de datos o en algún otro recurso disponible, sin embargo dichos motores de búsqueda no son capaces de entender el significado o el propósito de dicha palabra o conjunto de caracteres ingresados por el usuario. Con un sistema de ontología se pretende diseñar un sistema que permita entender propiamente las intenciones del usuario, emulando así lo que se conoce como inteligencia artificial.

De la misma manera, tenemos el caso de la asignación de recursos dentro de una red. La forma en que se asigna los recursos disponibles muchas veces está basada en algún algoritmo establecido, sin embargo, dichos algoritmos tienen sus limitaciones. Con el sistema de ontologías se intentará lograr que dicha asignación de recursos se haga de manera "inteligente" ahorrando recursos y proporcionándonos un mejor rendimiento.

CAPITULO I
PLANTEAMIENTO DE LA SITUACION INICIAL DE LA RED Y SUS NECESIDADES

En este capítulo se tratará de describir el funcionamiento y las limitaciones de las redes actuales que son las que comúnmente están implementadas en las empresas.

1.1 Red de servicios internos y sus desventajas

La necesidad de una red informática que permita intercambiar información y aplicaciones entre los usuarios se ha hecho muy común en los últimos años, casi no se puede encontrar una empresa que no tenga implementada una red informática con una arquitectura organizada y establecida para sus usuarios.

Sin embargo, nos enfrentamos a un problema, las redes que usamos actualmente obligan a sus usuarios a esforzarse por entenderlas y, si bien es cierto que los servicios y aplicaciones están presentes en ella, los usuarios cuentan con herramientas limitadas para buscarlas, localizarlas y luego usarlas como es debido. El sistema ofrece un servicio pero la concepción de éste puede ser diferente a la idea que tiene un usuario.

Pongamos un ejemplo de un servicio en red para la búsqueda de información en la Internet, como los buscadores clásicos como Google o Yahoo. Cuando uno inserta una palabra para ser buscada mediante estos motores de búsqueda, es común que nos encontremos con resultados diferentes a los que buscábamos, esto se debe a que los motores son incapaces de entender el significado de las palabras y por lo tanto, de las intenciones y objetivos del usuario, por lo que éste deberá introducir más datos para que la búsqueda sea más específica.

De la misma manera, en redes más complejas, un usuario entrará desde su terminal a solicitar un servicio o una aplicación que de ante mano está establecida en el sistema, sin embargo, con las redes actuales muchas veces se le da acceso a un servicio o aplicación que nada tenía que ver con las necesidades o requerimientos que dicho usuario tenía en mente o estaba buscando. Es por eso que muchas veces son los mismos usuarios que

tienen que aprender a buscar por ellos mismos la manera de acceder a dichos servicios y aplicaciones, esto se debe a que el método usado en las redes actuales no se basa en contextos o significados, sino mayormente a ciertas rutas o algoritmos de búsqueda establecidos o configurados por los desarrolladores de software o administradores de red.

Otro punto son los cambios, es común que las empresas vayan creciendo tanto en los recursos y servicios de su sistema como el número de usuarios que dependan de ellos. Esto constituye un problema para los diseñadores de redes y los desarrolladores de softwares, porque estos cambios obligan a reconfigurar tanto el diseño de la red como la manera de usar y de proveer los recursos y servicios del sistema. Muchas veces constituye una inversión fuerte de tiempo y dinero. Una red o un sistema puede ser diseñado pensando en la escalabilidad para futuras agregaciones, sin embargo, llegado el momento es difícil predecir como o cuando se llevarán a cabo estos cambios.

1.2 Diferencia entre una red estática y una dinámica

Con lo expuesto anteriormente se puede deducir que las redes actuales que comúnmente se usan en nuestra vida diaria y en las empresas caen en la categoría de redes estáticas. Estas redes están diseñadas e implementadas de una manera ya definida desde el inicio de su planeamiento, lo que significa que los desarrolladores deben de tomar en cuenta todos los posibles requerimientos y como trabajar con los recursos y el posible número de usuarios que tengan acceso al sistema. En esta clase de redes estáticas los usuarios por lo general deben tener ciertos conocimientos previos para poder hacer uso de los servicios, pero no sólo para las aplicaciones, también deben ser capacitados para poder localizarlos y saber como solicitarlos cuando tengan la necesidad de usar alguno de los servicios disponibles en el sistema.

En una red dinámica lo que se busca es que esta pueda realizar ciertas funciones de manera autónoma, sin necesidad de recurrir a un administrador de red o a una persona encargada de realizar los cambios respectivos en el sistema de manera manual. Otra de las características de una red dinámica es su interacción con los usuarios o clientes del sistema, el objetivo es lograr que el sistema pueda entender exactamente las intenciones y necesidades del usuario, sin que este último tenga que entrar en detalles o que tenga que invertir una gran cantidad de tiempo en ello. Se debe lograr un sistema de tal forma que éste sea el encargado de buscar los servicios más apropiados para la necesidad del usuario,

que se le asigne dicho servicio o aplicación y luego que pueda ser guardado nuevamente cuando el usuario ya no lo necesite. De esta forma se pueden distribuir de manera más eficiente los recursos de la red sin que los usuarios inviertan su tiempo en búsquedas. Otra de las características de una red dinámica es su resistencia a las fallas, lo que significa que debe contar con los mecanismos para auto diagnosticarse y auto repararse sin necesidad de que una persona tenga que intervenir manualmente. Obviamente el diseño y la implementación de una red dinámica trae como consecuencia mayor complejidad, mayor tiempo para investigación y desarrollo y mucho más consumo de recursos, sin embargo, considerando el crecimiento de las empresas y su necesidad de contar con redes confiables y efectivas, este tipo de redes dinámica representarán una gran ventaja.

CAPITULO II
REDES SEMANTICAS Y ONTOLOGIAS

En el presente capítulo se darán algunos conceptos con respecto al área de las redes semánticas y las ontologías

2.1 Visión de una Red Semántica

La World Wide Web (www) o la Red de Expansión Mundial ha evolucionado de tal forma que ha cambiado la manera en que las personas se comunican. Originalmente las computadoras personales eran usadas para operaciones numéricas, actualmente su uso predominante es el procesamiento de información, principalmente aplicados a las bases de datos, procesamiento de textos y en algunos casos, juegos en línea.

Hoy en día la mayoría de las redes presentan contenidos aptos para el uso de las personas comunes. Incluso las bases de datos que se generan automáticamente usualmente son presentadas sin la información estructural del proceso de su creación. Otros usos típicos de las redes hoy en día también incluyen la búsqueda y reunión de información de manera exacta, catálogos para compras por Internet y filtrado de contenidos no aptos.

Estas actividades generalmente muchas veces no pueden ser realizadas por las limitaciones de la mayoría de herramientas y softwares. A parte de la existencia de conexiones establecidas entre los documentos y archivos, una de las herramientas principales y necesarias sin duda es un motor de búsqueda que nos permita encontrar la información requerida de manera rápida y exacta.

Algunos de estos motores de búsqueda son los conocidos Altavista, Yahoo y Google, es evidente que sin éstos sería imposible localizar la cantidad de páginas o enlaces necesarios con la información que buscamos. Sin embargo, estos motores presentan muchos problemas, por ejemplo: son poco precisos ya que es común que al buscar algo en google.com introduciendo una palabra "X" nos dé enlaces a páginas que no tengan mucho

que ver con lo que buscamos, sin embargo, contienen la palabra "X" que introducimos en el buscador.

Esto se debe a la alta sensibilidad del vocabulario que es usado en estas herramientas, lo cual no es satisfactorio, porque significa que palabras semánticamente iguales pero con diferente significado generan los mismos resultados. Otro problema común es cuando la información que buscamos está dispersa en varios documentos, en estos casos debemos realizar varias entradas para acumular la documentación relevante, y luego debemos extraer la información parcial manualmente y unirla.

Una alternativa para solucionar esta clase de problemas es representar el contenido de la red en una forma que sea más accesible, para que sea procesado por el hardware y usar técnicas "inteligentes" para tomar ventajas de estas representaciones. Para ello existen las conocidas Redes Semánticas, las cuales gradualmente se convertirán en el siguiente paso en la evolución de las redes actuales. La visión de una Red Semántica consiste en que el significado de la información juegue un papel de mayor importancia que en las redes actuales.

Actualmente el desarrollo de las Redes Semánticas está teniendo una gran acogida en las investigaciones gubernamentales, por ejemplo: el gobierno de los Estados Unidos ha establecido el DARPA Agent Markup Language (DAML) project.

El manejo de información, recursos y conocimientos en una organización ha ido creciendo al pasar el tiempo. Esta necesidad ha emergido como la clave de las actividades en los grandes negocios. Sin embargo, la mayor parte de la información está disponible en estructuras débiles, como por ejemplo: texto, audio y video, en otras palabras se presentan las siguientes limitaciones:

- **Búsqueda de Información:** Las compañías usualmente dependen de los motores de búsqueda basadas en teclas o serie de caracteres, cuyas limitaciones ya se comentaron.
- **Extracción de información:** Se debe considerar el tiempo y el esfuerzo humano que se debe invertir en la búsqueda y organización de documentos con la información relevante. Los agentes actuales aún no son suficientemente eficientes para esta clase de tareas.
- **Mantenimiento de Información:** Actualmente se presentan problemas como la inconsistencia en la terminología y el fracaso para remover información desactualizada o información innecesaria.

- **Observando la Información:** Muchas veces es difícil ocultar información y permitir que otros la vean, especialmente cuando se tiene una Intranet.

Para solucionar esta clase de problemas las Redes Semánticas tienen los siguientes objetivos:

1) La información es organizada en espacios conceptuales de acuerdo a su significado.

2) Herramientas automáticas ayudarán en el soporte y mantenimiento por medio de la inspección de inconsistencias y extracción de la nueva información

3) El motor de búsqueda por medio de palabras o caracteres será reemplazado por un sistema de búsqueda relacionado al significado de la información de manera más amigable para el usuario.

4) Soporte para las respuestas sobre muchos documentos.

5) Determinan quienes pueden ver cierta parte de la información, e incluso que partes del documento.

2.2 Tecnología usada en las Redes Semánticas

Actualmente las grandes necesidades se encuentran en el área de integración, estandarización, desarrollo de herramientas y la adopción de usuarios. Obviamente, el progreso tecnológico lleva ha obtener Redes Semánticas más avanzadas, algunas de las cuales pueden ser logradas con la tecnología actual.

Como se ha mencionado anteriormente, el contenido de la Red está en un formato para ser entendido por los seres humanos, sin embargo, los programas tendrán ciertas dificultades para entender estos formatos.

Por ejemplo: una página de Internet en HTML tendrá la siguiente forma:

<h1>Horarios de Servicios Médicos </h1>

El hospital ofrece los siguientes horarios de servicio. No se atienden los días feriados. Para realizar la consulta primero se deberá coordinar con la recepcionista, Srta. Daniela, luego el doctor especialista, Elías J. indicará su disponibilidad.

<h2>Horarios</h2>

Lunes 11am-9pm

Martes 11am-9pm

Miércoles 11am-9pm

Jueves 11am-9pm
Viernes 11am-9pm
Más informes

Este tipo de información es fácilmente entendible por un ser humano, sin embargo, una computadora tendrá problemas para establecer el significado exacto de las palabras de las partes textuales. Por ejemplo: los motores de búsqueda actuales podrán identificar las palabras “horarios” o la frase “Servicios Médicos”. Pero este motor tendrá problemas en reconocer algunos datos como las horas exactas de atención, o entre diferenciar a una secretaria con un doctor. Además para “Más informes” uno tiene que hacer clic en el link.

El método usado por las redes semánticas es resolver el problema desde el punto de vista de la página web. Si el lenguaje HTML es reemplazado por un lenguaje más apropiado, entonces será más eficiente y fácil de entender por los programas. Además de que la información esté en formatos entendibles por las personas, también debe contener información explicando el contenido. Por ejemplo una solución podría ser:

<Sitio>
<Servicio>Servicio Médico</Servicio>
<Equipo>
<Doctor>Elías J.</Doctor>
<Recepcionista>Srta. Daniela</Recepcionista>
</Equipo>
<Horarios>
<día1>Lunes</día1><hora1>11am</hora1>
<día2>Martes</día2><hora2>11am</hora2>
<día3>Miércoles</día3><hora3>11am</hora3>
<día4>Jueves</día4><hora4>11am</hora4>
<día5>Viernes</día5><hora5>11am</hora5>
</Horarios>
</Sitio>

Con este formato, la información se hace más entendible por los programas. El término **Metadata** se refiere a esta clase de información: Datos acerca de Datos. La Metadata captura parte del significado de los datos, de ahí el significado de las Redes Semánticas.

2.3 Definición de una Ontología

El término "Ontología" proviene de una de las ramas de la filosofía, llamada el estudio de la naturaleza de la existencia. La identificación de las cosas que existen y como describirlas. Como ejemplo se puede poner la observación de que el mundo que nos rodea está hecho de objetos específicos que pueden ser agrupados en clases abstractas basadas en ciertas propiedades comunes.

En general "Una Ontología" describe formalmente un dominio. Típicamente una ontología consiste en una lista finita de términos su respectiva relación entre ellos. Estos términos denotan conceptos importantes del dominio, por ejemplo en una universidad se tiene varias personas con funciones diferentes como profesores, empleados y estudiantes, y no solo eso, también se tiene que considerar que existen diferentes cursos y especialidades, cada una con sus propias características. En la figura 2.1 se puede observar la gráfica de lo descrito anteriormente.

Estas relaciones generalmente incluyen jerarquías de clases y estas a su vez contienen sub-clases con la información necesaria. En un contexto de la red, las ontologías proveen el entendimiento necesario para comprender un dominio. Este entendimiento es necesario para superar las diferencias en cuanto a la terminología.

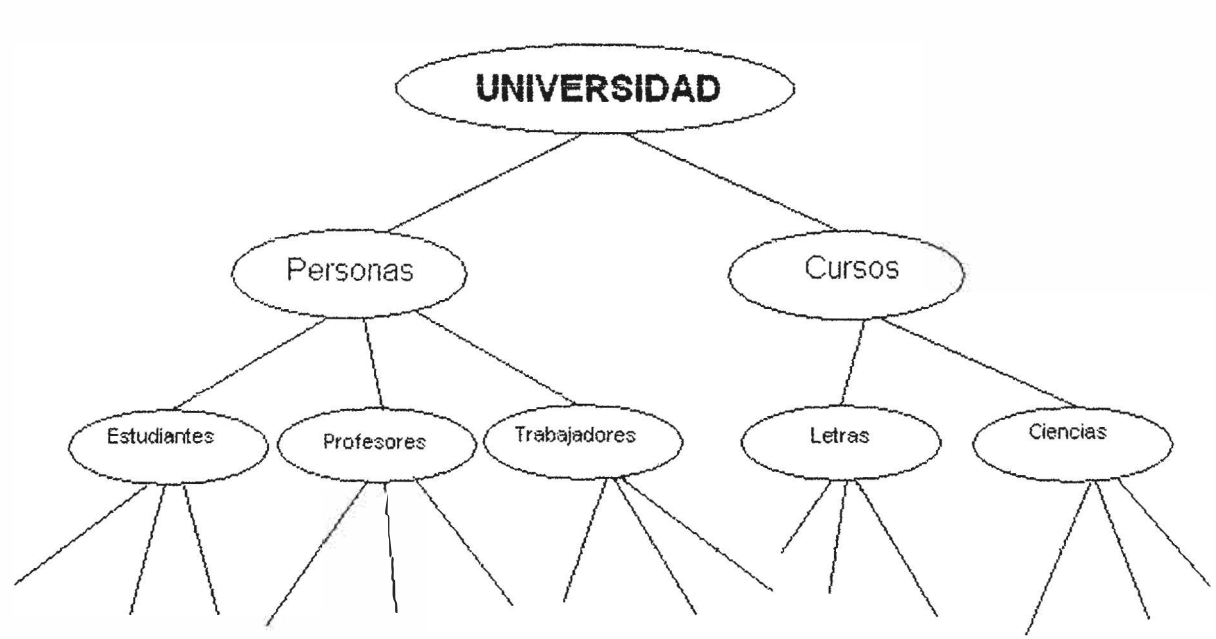

Figura 2.1: Diagrama Jerárquico de lo que sería la organización de una universidad

Las Ontologías son bastante útiles para la organización y la navegación en los sitios de la Red. Por ejemplo: muchos sitios en la Red de hoy en día tienen un menú en el lado izquierdo de la página para mostrar los diferentes campos y categorías, organizados generalmente de manera jerárquica, los usuarios pueden hacer click en los links para poder acceder a las subcategorías.

De la misma manera, las Ontologías pueden ser muy útiles para mejorar el nivel de búsqueda y la exactitud de las búsquedas en la Red. Los motores de búsqueda pueden buscar dentro de las páginas por alguna referencia a un concepto preciso en lugar de recopilar una lista de todas las páginas que contienen dicha palabra o frase buscada, que en la mayoría de los casos puede tener significados ambiguos y pueden ser demasiadas.

Adicionalmente, la búsqueda en las redes puede explotar la generalización y la especialización de la información. Si una entrada falla para encontrar un documento con la información relevante, el motor de búsqueda puede sugerir al usuario una entrada más generalizada. En el campo de la Inteligencia Artificial (AI de sus siglas en inglés) desde hace muchos años que se emplea el uso de las Ontologías.

2.4 Lenguajes de Ontologías

Actualmente los lenguajes de Ontologías más importantes son:

- XML provee una superficie de sintaxis para documentos estructurados pero no impone una coacción semántica en el significado de los documentos
- XML Schema es un lenguaje para restringir la estructura de un documento XML
- RDF es un modelo de datos para objetos o recursos y sus respectivas relaciones entre ellos. Este lenguaje provee una semántica simple para este modelo de datos, los cuales a su vez pueden ser representados con la sintaxis XML.
- RDF Schema es un lenguaje de descripción de vocabulario para describir propiedades y clases de los recursos RDF, con una semántica para la generalización de jerarquías de dichas clases.
- OWL es un lenguaje de descripción de vocabulario para describir propiedades y clases, como la relación entre clases, cardinalidad, ecualidad, características de las propiedades y clases enumeradas.

El correcto desempeño de una red semántica sólo se podrá llevar a cabo cuando los

niveles de interoperabilidad sean establecidos. Dicha interoperabilidad está definida por la W3C , ver figura 2.2.

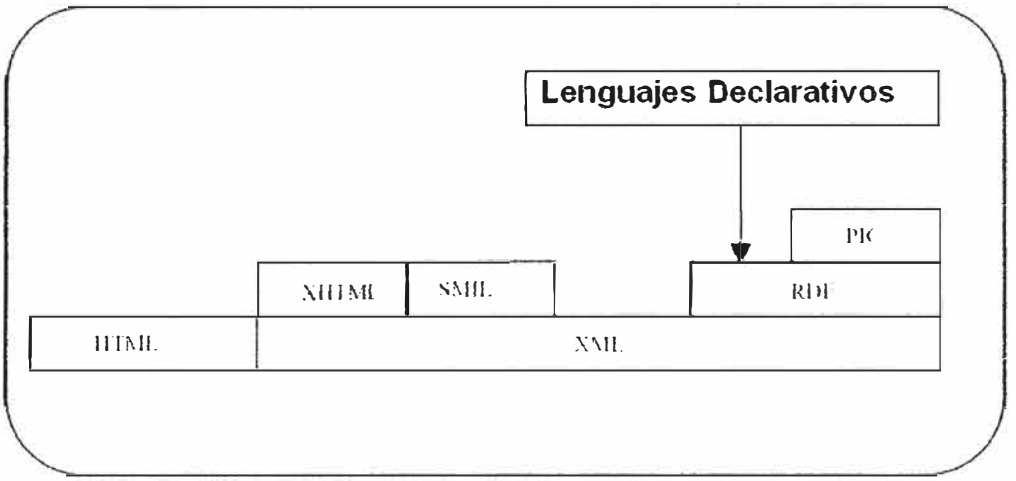

Figura 2.2: Capas de los lenguajes de la Red

2.5 Lenguaje XML

En la actualidad HTML (Hypertext markup language) es el lenguaje estándar en que las páginas de Internet son diseñadas. Así mismo el lenguaje HTML fue derivado desde el SGML (Standard generalized markup language) . Los lenguajes conformados por SGML son llamados aplicaciones de SGML, por lo que HTML se considera una aplicación, desarrollada porque SGML en sí se consideraba muy compleja para las páginas de Internet. Otra de las aplicaciones de SGML es XML (Extensible markup language) la cual fue creada para superar las limitaciones del HTML.

XML es un lenguaje que tiene marcas para identificar estructuras de documentos arbitrarios, lo cual es opuesto a HTML, que está hecho para una clase especifica de documentos de hipertexto. Un documento XML consiste en un arreglo de "etiquetas" que encierran un determinado número de atributos y valores. El vocabulario usado para dichas etiquetas y sus combinaciones posibles no están fijadas, pero pueden estar definidas por aplicaciones de XML. Un ejemplo sería:

```
<class-def>
    <class name="Planta"/>
    <subclass-of>
        <NOT><class name="animal"/></NOT>
    <subclass-of>
```

```
</class-def>
<class-def>
<class name="Arbol"/>
        <subclass-of>
                <class name="Planta"/>
        <subclass-of>
</class-def>
<class-def>
        <class name="Rama"/>
                <slot-constraint>
                        <slot name="is-part-of"/>
                        <has-value>
                                <class name="Arbol"/>
                        </has-value>
                </slot-constraint>
        </class def>
```

De esta línea de comandos se puede ver un modelo básico de programa escrito en XML el cual está etiquetado como un árbol, donde cada etiqueta corresponde a un nodo en el modelo, y cada sub-etiqueta se le conoce como un "hijo" en el árbol. Hay que recalcar que este modelo es sólo una de las posibles sintaxis de la ontología del ejemplo. Se pueden definir muchas otras versiones alternativas en XML de la misma información semántica, por ejemplo una alternativa podría ser:

```
<class-def>
        <name>Rama</name>
        <slot-constraint>
                <name>is-part-of</name>
                <has-value>tree</has-value>
        </slot-constraint>
</class-def>
```

Como se puede ver, la idea es la misma, simplemente se cambia la sintaxis, pero una herramienta de búsqueda compatible con ésta podrá encontrar la información solicitada de manera exacta en cualquiera de las dos alternativas mostradas.

2.6 Lenguaje RMD

El RMD fue creado para proveer a la meta-data de un estándar, para las descripciones en la red. Uno de los problemas del XML es que no provee entendimiento del significado (semántica) de los datos. Por ejemplo: no hay un entendimiento asociado con el uso de las etiquetas, depende de cada aplicación y como éstas interpretan las etiquetas.

El RDF es más un modelador de datos que un lenguaje, su constructor es básicamente una tripleta de objeto-atributo-valor llamado sentencia o "statement". Se puede decir que se tiene un objeto "O", cuyo atributo se define como "A" y tiene un valor "V". Esta relación comúnmente se le da la notación A(O,V), como por ejemplo, cuando se tiene un libro de título "Historia del Perú" y se le quiere dar un precio para poder venderlo se puede usar la siguiente notación:

PrecioLibro('Historia del Perú',"65Soles")

También existe otra forma de representar dichas tripletas que vendría a ser la siguiente: [O]-A->[V] que se podría ver como una especie de cadena, por ejemplo, se tendría lo siguiente:

Nombre('http://www.sitio.com',"Juan Pérez")

Autorde('http://www.sitio.com','http://www.libro.com')

Precio('http://www.libro.com',"40 soles")

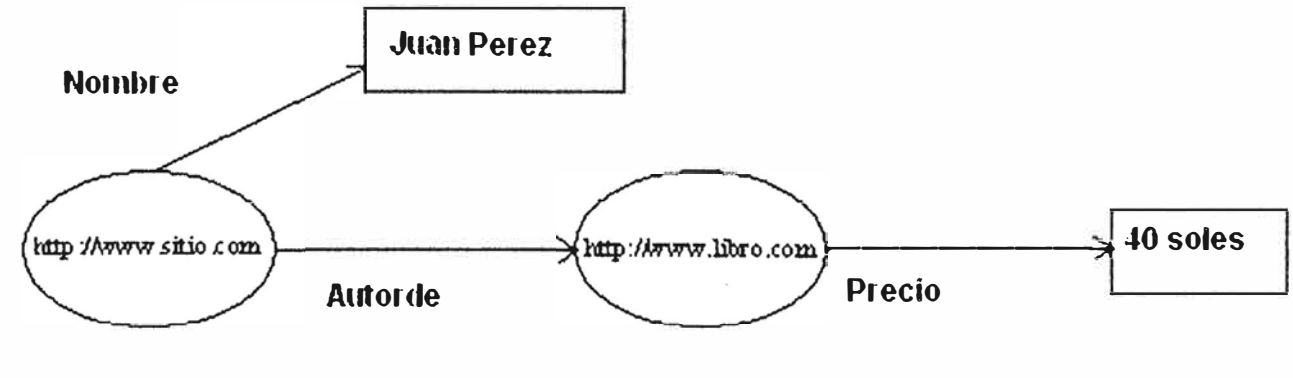

Figura 2.3: Diagrama Objeto-Atributo-Valor

Se puede hacer un diagrama sobre la tripleta Objeto-Atriibuto-Valor. Para un mejor entendimiento de dicho ejemplo se puede ver en la figura 2.3.

Hay que recalcar que el rol del RDF es proveer el modelo de datos de la tripleta Objeto-Atributo-Valor (OAV) para la meta-data. A parte del OAV, no existe otro modelamiento de datos. Así mismo al igual que XML, el modelo de datos de RDF no provee mecanismos para declarar apropiadamente los nombres que serán usados. RDF Schema provee un vocabulario de definiciones para RDF de la misma forma en que XML Schema lo hace para XML.

Las dos construcciones más importantes del RDF Schema son subClassOf y subPropertyOf. En el caso de que los objetos son instancias de una o más clases, se debe de indicar esto por medio la propiedad type, subClassOf permite las especificaciones de la organización jerárquica de dichas clases mientras que SubPropertyOf hace los mismo para las propiedades. La interacción entre las propiedades pueden ser especificadas usando los constructores domain y range.

2.7 Jini

Jini es una API desarrollada por Sun Microsystems. El objetivo es convertir la red en un sistema flexible y fácil de administrar, en el cual se puedan encontrar rápidamente los recursos disponibles tanto por clientes humanos como computacionales. Un sistema Jini consiste en un sistema distribuido basado en la idea de grupos federados de usuarios y de recursos requeridos por otros usuarios. Los recursos pueden ser implementados tanto por dispositivos hardware y software.

Jini supone que la infraestructura de red sobre la que se monta tiene el ancho de banda y es lo suficientemente fiable para funcionar, por lo que no aporta mecanismo para mejorar estos dos puntos. También se asume que los dispositivos Jini tienen capacidad de procesamiento y memoria suficientes.

2.8 Diseño de una Red Inteligente con Jini y XML

Tratar de desarrollar una red inteligente, que pueda satisfacer las necesidades de los usuarios no es una tarea fácil, es por ello que se recurre a ciertas herramientas de tecnología de Red, estas son Jini y XML. Mientras que Jini es un modelo universal de

computación en red para las plataformas de Java, XML es un formato universal para el intercambio de datos en la red. Tanto Jini como XML se complementan bien y ofrecen muchas posibilidades para el futuro.

La arquitectura Jini, como se verá con más detalles en el capítulo siguiente, representa una arquitectura perfecta para construir un sistema distribuido federado, sin embargo, para que los diferentes sistemas que compongan la red se puedan entender se hará uso del lenguaje XML.

XML será usado para representar los datos que están involucrados en un evento, lo cual nos permite que nuestro sistema también pueda comunicarse con otros sistemas que estén basados también en XML, de esta manera se podrá permitir futuras interacción entre nuestro sistema con otros sistemas. La ventaja de usar XML, al ser universal, es que no habrán confusiones sobre la estructura y los tipos de datos que son transferidos entre diferentes sistemas.

CAPITULO III
ARQUITECTURA JINI

La tecnología Jini (Java Intelligent Network) es un servicio de arquitectura orientada que define un modelo de programación que utiliza y expande la tecnología Java para permitir la construcción de sistemas seguros y distribuidos que consisten en redes de servicios confiables al cliente. La tecnología Jini puede ser usada para construir sistemas de redes adaptativas que son escalables, flexibles y con la capacidad de evolucionar en medios dinámicos.

3.1 Objetivos y Definiciones

Un sistema Jini es un sistema distribuido basado en la idea de federar grupos de usuarios y de los recursos requeridos por dichos usuarios. El objetivo general es convertir la red en una herramienta flexible y de fácil administración en cuyos recursos pueda ser encontrados por sus usuarios o clientes. Los recursos pueden ser implementados como dispositivos de hardware, programas de software o una combinación de ambas. Este método se centra en hacer que la red sea una entidad más dinámica y que refleje la naturaleza dinámica del sistema agregando o retirando servicios de manera flexible. El sistema Jini consiste de las siguientes partes:

- Un arreglo de componentes que proveen de una infraestructura a los servicios federados en el sistema distribuido.
- Un modelo de programación que soporta y fomenta la creación de servicios distribuidos confiables.
- Los servicios que pueden ser parte del sistema

Los objetivos finales de este sistema son:

- Permitir a los usuarios compartir servicios y recursos a través de la red.

- Proveer a los usuarios un acceso fácil a los recursos sea cual sea su ubicación en la red.
- Simplificar la tarea de construir, mantener y alterar la red de dispositivos, programas y usuarios.

El sistema Jini expande el medio de aplicación de Java de una simple máquina virtual a una red de máquinas. Las aplicaciones de Java proveen una buena plataforma de computación porque tanto como el código y los datos pueden moverse de máquina a máquina. El resultado es un sistema en que la red soporta una configuración fluída de objetos que pueden trasladarse de sitio a sitio según cómo y cuándo se les necesite y pueden ser llamados a cualquier parte de la red para que cumplan sus funciones.

La arquitectura del sistema Jini está dirigida al grupo de trabajo, esto quiere decir, que los miembros del grupo federado por el sistema deben estar de acuerdo con las nociones básicas de confianza, administración, identificación y políticas.

Servicios: Un servicio es una entidad que puede ser usada por un usuario, programa o incluso por otro servicio. Los miembros de un sistema Jini tienen que federar para poder compartir accesos a los servicios que pueden ser combinados para el mejor rendimiento de una tarea. Los servicios se comunican entre ellos usando un protocolo de servicios, el cual es un arreglo de interfaces escritas en un lenguaje de programación en Java.

Servicio Look Up: Los servicios son encontrados y resueltos por el Look up, el cual es un mecanismo del sistema que mapea las interfaces indicando la funcionalidad de los servicios que se requieren.

Método de Invocación Remota de Java (RMI): Se usa para lograr la comunicación entre servicios. RMI es fundamentalmente una extensión del lenguaje Java que permite que los datos pasen de objeto a objeto a través de la red, pero no sólo eso, también permite que pasen objetos completos, incluyendo el código.

Seguridad: Básicamente el diseño de seguridad de Jini es tener las nociones de "principal" y de la "Lista de Control de Acceso". El acceso a los servicios es requerido por el "principal" y la lista de control de acceso es la que se encarga de darle acceso o no.

Leasing: Se encarga de garantizar el acceso a los servicios sobre un periodo de tiempo. El leasing es negociado entre el que va usar y el que va a proveer el servicio. Si el "léase" no es renovado al acabar su periodo, entonces se corta la comunicación entre el usuario y el servicio.

Transacciones: Es una serie de operaciones tanto para un servicio único o para múltiples servicios, las cuales están provistas por un protocolo de servicios.

Eventos: Un objeto puede permitir que otros objetos registren interés en ciertos eventos y recibir notificaciones de la ocurrencia de dichos eventos.

3.2 Componentes

Los componentes del sistema Jini pueden ser clasificados en tres partes: Infraestructura, Modelo de Programación y Servicios. Se muestra dicha clasificación en la figura 3.1.

	Infraestructura	Modelo de Programación	Servicios
Java	Java VM RMI Seguridad	Java Apls Java Beans ™ ...	JNDI Enterprise Beans JTS ...
Java + Jini	Discovery/Join Distributed Security Lookup	Leasing Transacciones Eventos	Administración de Transacciones Servicio JavaSpaces ™ ...

Figura 3.1: Segmentación de la Arquitectura Jini

El sistema Jini puede ser visto como una extensión hacia la red de estas tres categorías que hacen que funcione de manera eficiente la tecnología Java en una sola computadora.

3.2.1 Infraestructura

La infraestructura de Jini incluye lo siguiente:

- Un sistema de seguridad distribuida, integrada en el RMI, esto expande la plataforma de seguridad de Java al sistema distribuido.
- Los protocolos de descubrimiento y unión, los cuales permiten a los servicios formar parte de otros miembros de una federación.
- El servicio Lookup, que sirven de repositorios para los demás servicios. Las entradas en el servicio del lookup son objetos escritos en el lenguaje de programación de Java, dichos objetos pueden ser descargados como parte de la operación del lookup y actuar como proxis locales para los servicios requeridos.

Los protocolos de unión y descubrimiento definen el modo en que un servicio de cualquier tipo se transforma en parte del sistema Jini.

3.2.2 Modelo de Programación

La infraestructura se encarga de habilitar y hacer uso del modelo de programación. Cuando un evento se une o abandona un servicio lookup, los eventos son señalados y los objetos que han registrado interés en dichos eventos reciben notificaciones cuando un nuevo servicio se vuele disponible o cuando servicios antiguos cesan su actividad. Los servicios constituyen un arreglo de interfaces por las cuales se definen los protocolos de comunicación que pueden ser usados por los mismos servicios y por la infraestructura para comunicarse entre ellos.

Estas interfaces unidas, crean la extensión distribuida del lenguaje estándar de la programación en Java que viene a ser el modelo de programación Jini. Las interfaces para armar el modelo de programación en Jini son:

- La interfaz Leasing, que define la forma en que se localizan y se liberan los recursos usando un modelo renovable basado en un periodo de duración.
- La interfaz de eventos y notificaciones habilitan la comunicación basada en eventos entre los servicios.

- La interfaz de transacción permite que las entidades cooperen de tal manera que los cambios realizados en el grupo ocurran atómicamente o que ninguno de ellos ocurra.

3.2.3 Servicios

Estos servicios hacen uso de la infraestructura para comunicarse, descubrirse y anunciar su presencia entre ellos. Los servicios aparecen como objetos escritos dentro de la programación del lenguaje Java.

Los servicios tienen una interfaz que define la operación que puede ser requerida para dicho servicio, algunas de esas interfaces son usadas por programas, mientras que otras son usadas para interactuar con el usuario final. El tipo de servicio determina su interfaz y también el arreglo de métodos usados para acceder a dicho servicio.

3.3 Arquitectura de Servicios

Los servicios forman la base interactiva del sistema Jini, tanto en le nivel de programación como en el de las interfaces.

3.3.1 Protocolos de Descubrimiento y de Lookup

El trío de los protocolos de descubrimiento, de unión y de lookup. Los dos primeros ocurren cuando un dispositivo es conectado al sistema.

1) El protocolo de descubrimiento se acciona cuando un servicio esta buscando un servicio de lookup para poder registrarse.
2) El protocolo de unión ocurre cuando un servicio ha localizado el lookup y desea unirse.
3) El protocolo de Lookup ocurre cuando un cliente o usuario necesita localizar e invocar un servicio descrito por su tipo de interfaz o por otros atributos.

Este proceso de descubrimiento está descrito en la figura 3.2.

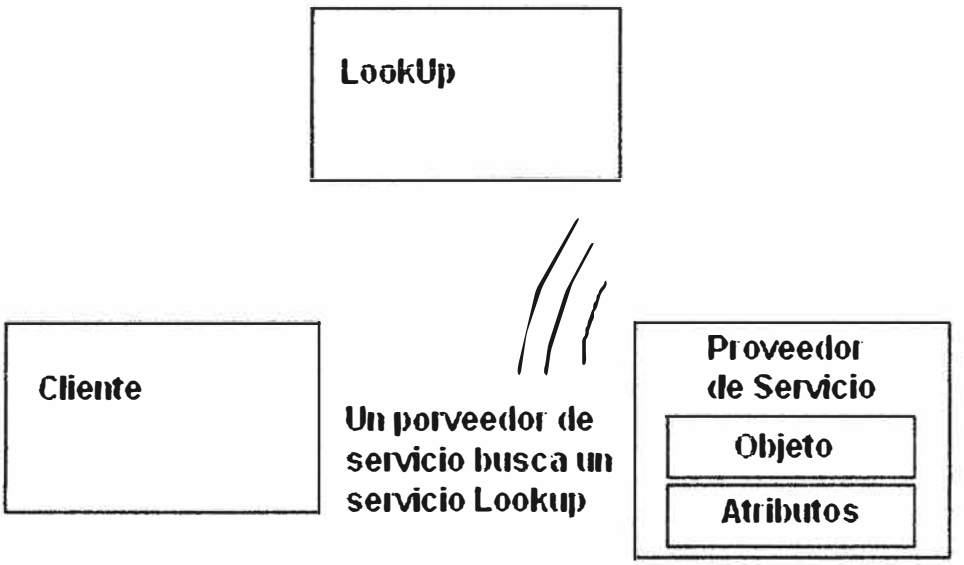

Figura 3.2: Proceso de Descubrimiento

Un proveedor de servicio es el que origina el servicio para un dispositivo o software. Lo primero que hace el proveedor es localizar el servicio Lookup por medio de una petición multicast en la Red Local. Cuando el objeto servicio requerido es localizado, este es cargado en el servicio Lookup. Dicho objeto contiene la interfaz en lenguaje Java para el servicio, incluyendo los métodos que los usuarios y las aplicaciones invocarán para ejecutarlo junto con otros atributos descriptivos. Este proceso está descrito en la figura 3.3.

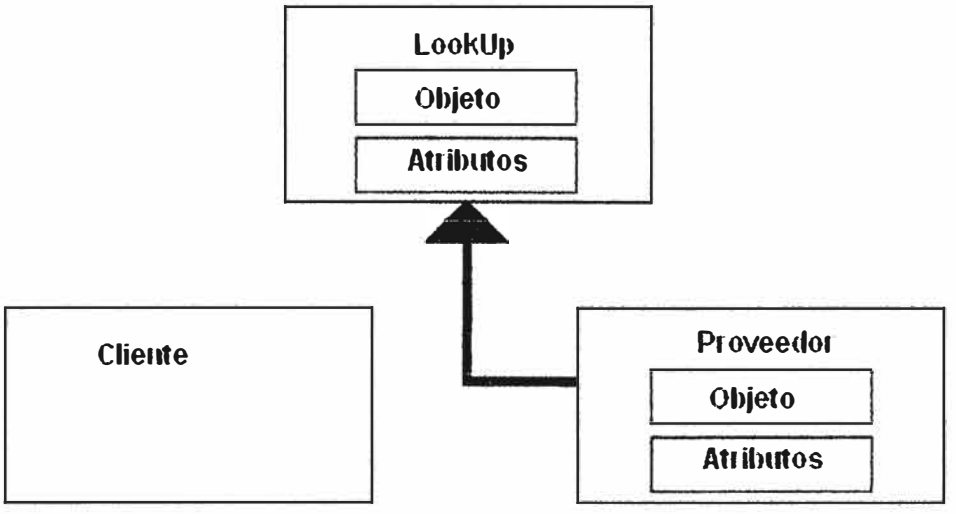

Figura 3.3: El Proveedor del Servicio registra el objeto del servicio y sus atributos en el LookUp

Los servicios deben de ser capaces de encontrar el servicio LookUp, sin embargo, un servicio podría delegar esta tarea a un tercero, con esto el servicio está listo para ser usado

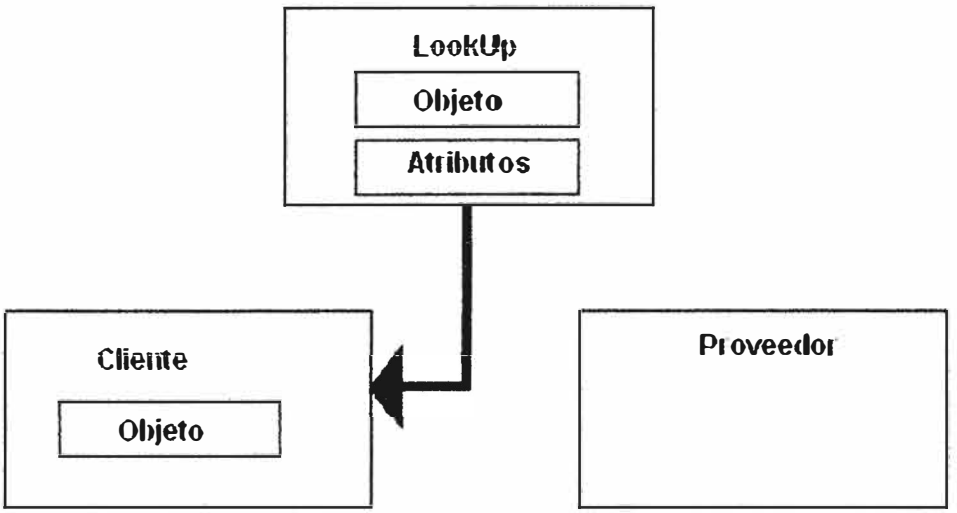

Figura 3.4: *Una copia del objeto es transferido al cliente y es usado por este para comunicarse con el servicio*

El cliente localiza el servicio apropiado por medio de su tipo, según lo que este escrito en su interfaz, además de usar la descripción dada sobre los atributos usados en la interfaz del servicio Lookup, luego el objeto es cargado en el cliente, tal como se muestra en la figura 3.4.

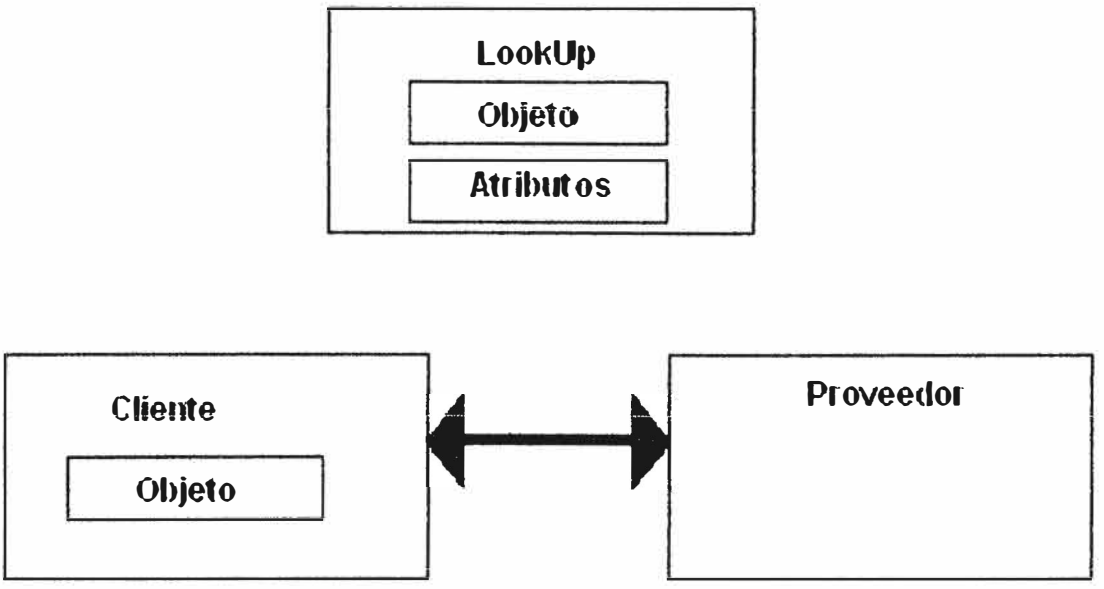

Figura 3.5: *El Cliente ahora interactúa directamente con el proveedor*

Los métodos de los objetos pueden implementar un protocolo privado entre estos y el proveedor del servicio. Las diferentes implementaciones de la misma interfaz pueden usar diferentes protocolos. La capacidad de poder mover objetos y códigos del proveedor de servicios al Lookup y del Lookup al cliente, dan mucha libertad al proveedor de servicios en los patrones de comunicación entre el servicio y su cliente. El movimiento del código también asegura que el objeto tomado por el cliente y el servicio requerido siempre estén sincronizados. El cliente sólo conoce que está tratando con una implementación de interfaz en lenguaje Java, de esta forma el código que implementa la interfaz puede hacer lo necesario para proveer el servicio, tal como es muestra en la figura 3.5. Debido a que el código proviene directamente del mismo servicio, éste puede aprovechar los detalles de la implementación del servicio que son sólo conocidas por el código.

El cliente interactúa con el servicio por medio del arreglo de interfaces escritas en programación Java, estas interfaces definen el arreglo de métodos que pueden ser usados para interactuar con el servicio. Esta forma de encontrar un servicio asegura que el programa que busca el servicio conocerá como usarlo ya que su uso estará definido por el arreglo de métodos en su tipo.

Las interfaces programables pueden ser implementadas como referencias de RMI hacia el objeto remoto que implementa el servicio, como una computación local que provee todos los servicios localmente o alguna combinación. Estas combinaciones, llamadas "Proxis Inteligentes", implementan algunas de las funciones del servicio de manera local y las llamadas por medio remoto para una implementación centralizada del servicio.

Una interfaz de usuario también puede ser almacenada en el servicio LookUp como un atributo de un servicio registrado. Una interfaz de usuario almacenada en el servicio LookUp por medio de Jini es una implementación que permite que el servicio sea directamente manipulado por el usuario del sistema. Las interfaces de usuarios son formas especializadas de la interfaz de servicio que habilitan un programa para luego retirarse y dejar que pueda ser libremente manipulado por la persona que lo requiere.

En situaciones en que no se pueda encontrar el servicio LookUp, el cliente puede usar una técnica llamada "Peer Lookup". En este caso el cliente puede enviar el mismo paquete de identificación que es usado por el servicio lookup para pedir que los proveedores del servicio se registren. Los proveedores de servicio se registrarán con el cliente como si estuvieran en el servicio LookUp, luego los clientes podrán seleccionar los servicios necesarios y descartar el resto.

3.3.2 Implementación del Servicio

Los objetos que implementan un servicio pueden ser diseñados para correr en un mismo espacio con otros objetos especialmente cuando hay ciertas localizaciones o requerimientos básicos de seguridad, a esto se le conoce como un "Grupo de Objetos" o "Object Group". Un Grupos de Objetos siempre estará en una dirección o máquina virtual cuando estos objetos están corriendo, los objetos que no se encuentren en dicho grupo son aislados.

Un servicio puede ser implementado directamente o indirectamente por un hardware especializado, dichos dispositivos pueden ser contactados por el código asociado con la interfaz del servicio.

Desde el punto de vista del cliente del servicio, no debe haber distinción entre servicios que están implementados por objetos en diferentes máquinas, los servicios que son descargados en las direcciones locales y los servicios implementados en el hardware. Todos estos servicios aparecerán disponibles en la red y una clase de implementación puede ser remplazada por otra clase de implementación sin que el cliente tenga conocimiento.

3.4 Puente JINI/J2EE para Servicios Telefónicos IP a gran escala

Los servicios actuales de telefonía IP son relativamente estáticos y poco escalables, por lo que se requiere que sean más dinámicos. Para lograr eso se puede usar Jini haciendo el servicio de telefonía IP más fácil de manejar. Sin embargo, muchas de las aplicaciones globales con una lógica compleja y con miles de usuarios potenciales son desarrolladas en plataformas J2EE.

Java Platform, Enterprise Edition o Java EE (J2EE), es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java, con arquitectura de niveles distribuida, basándose ampliamente en componentes de software modulares, ejecutándose sobre un servidor de aplicaciones.

J2EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos.

Por un lado J2EE está diseñado para sistemas de gran escala mientras que Jini esi es para aplicaciones de escala media. J2EE provee un servicio centralizado donde Jini ofrece

una serie de federaciones con una administración dinámica.

3.4.1 Características

Los principales problemas de una aplicación de voz es que son extremadamente sensibles a los retardos, mucho más que a los errores. En la capa de transporte, UDP puede ser usado para cargar los paquetes de voz, mientras que el TCP puede ser usado para transportar el control de señales, las demoras o retardos se ocasionan generalmente por la re-transmisión y el mecanismo de three-handshake.

El protocolo RTP (Protocolo de Transporte en Tiempo Real) es un protocolo compensativo para la deficiencia en tiempo real, en redes de paquetes por medio de operaciones en UDP, y provee mecanismos para aplicaciones en tiempo real para procesar paquetes de voz. A parte de esto el RTP provee una retro alimentación para la mejora de la calidad y el manejo de la red en tiempo real.

Las redes integradas APIs (Interfaces de Programación de Aplicaciones) para la plataforma Java (JAIN) es un arreglo de redes de APIs para la plataforma Java. La arquitectura JAIN divide el sistema de software relacionada a la red en diferentes capas y coloca APIs de adelantamiento abierto (Forward Open) entre las capas que están adyacentes, esta arquitectura también está orientada a través de redes, señalizaciones y capas de servicio con o sin cableado y basadas en paquetes. Los APIs de estándar abierto permiten a diferentes compañías proveer soluciones para diferentes porciones del sistema y permitir a los usuarios escoger los productos de distintos vendedores a diferentes niveles basados en sus requerimientos, éstos también se encargan de ocultar la complejidad de la red y los protocolos de señalización para los desarrolladores de aplicaciones de alto nivel, lo cual da una gran oportunidad para los nuevos servicios de telefonía IP.

También tenemos OSS/J que es un arreglo de APIs estándares para sistemas de soporte de operaciones (OSS). Los OSS cubren el software usado por el proveedor de servicios para crear y administrar los servicios en su red. Los APIs de OSS estándares deben promover la interoperabilidad entre los productos de diferentes vendedores.

La implementación actual de Jini usando carga de clases dinámica no puede ser desarrollada en pequeños dispositivos, debido a sus limitaciones en recursos. La arquitectura Jini provee soluciones basadas en proxys para solucionar esta clases de problemas.

3.4.2 Puente JINI/J2EE

La finalidad de usar Jini es la de descubrir los servicios J2EE en las federaciones Jini, a través de un programa independiente que corre fuera del servidor de aplicación J2EE, el cual podría considerarse un Jini/2EE proxy para las aplicaciones J2EE, las cuales pueden ser accedidas por el cliente Jini.

La arquitectura del puente Jini/J2EE se muestra en la figura 3.6.

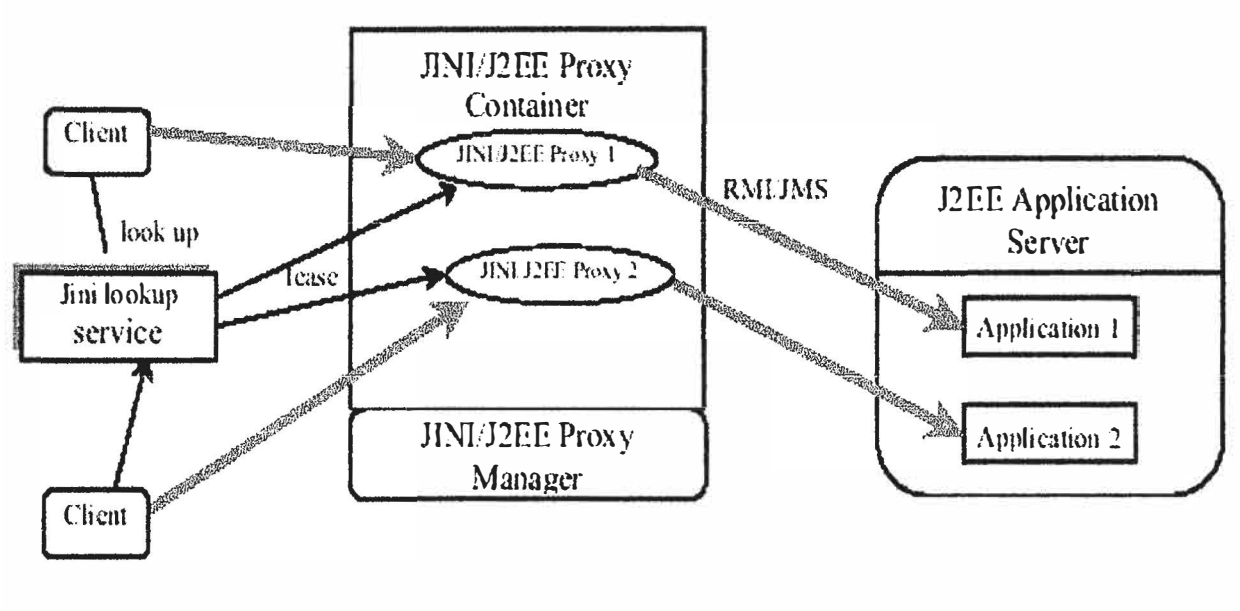

Figura 3.6: Arquitectura Jini/J2EE

Todas las aplicaciones J2EE tienen un proxy y corren en el puente, un proxy Jini/J2EE representa una aplicación J2EE en una federación Jini y puede comunicarse con su aplicación J2EE por medio del Método de Invocación Remota (RMI) o por el Servicio de Mensajes de Java(JMS). El puente tiene dos partes:

- **Contenedor del Proxy Jini/J2EE:** Contiene los recursos, organiza y provee el runtime para los proxies Jini/J2EE. Este contenedor puede localizar y liberar recursos de los proxy Jini/J2EE.
- **Administrador de Proxy:** Es el responsable del manejo del proxy, es capaz de comunicarse con las aplicaciones J2EE que corren en un servidor J2EE, permitiendo la creación y destrucción de los proxies, según el estado de la aplicación J2EE.

3.4.3 Mecanismo de Descubrimiento

Para que el puente y la aplicación se puedan encontrar entre ellas, se desarrollan dos tipos de mensajes; los mensajes Query y los mensajes de Respuesta. En la figura 3.7 se puede observar el proceso de descubrimiento.

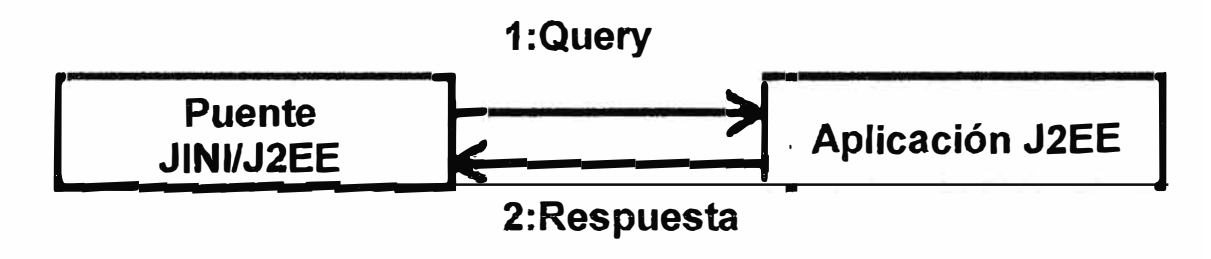

Figura 3.7: Proceso de Descubrimiento

El proceso de descubrimiento se inicia en el puente Jini/J2EE, el cual envía un mensaje tipo Query a la aplicación J2EE, consultándole si éste puede proveerle el servicio. Si la aplicación accede, éste le proporciona el Proxy Jini. El administrador de Proxy del Puente es el encargado de manejar el mensaje query enviado y de recibir la respuesta, mientras que la aplicación J2EE es la encargada de procesar el mensaje query y enviar la respuesta.

3.4.4 Modelo de Mensajes

Para que el puente hable con la aplicación J2EE, la información de los mensajes Query y de Respuesta necesitan estar envueltos en un sistema de mensaje J2EE como JMS. Se pueden usar diferentes modelos de mensajes.

Para los mensajes Query se usa el modelo publicar/suscribir (publish/subscribe messaging model), debido a que este tipo de mensajes debe ser enviado a través de varias aplicaciones que deseen proveer los servicios Jini. Se puede mostrar este modelo en la figura 3.8.

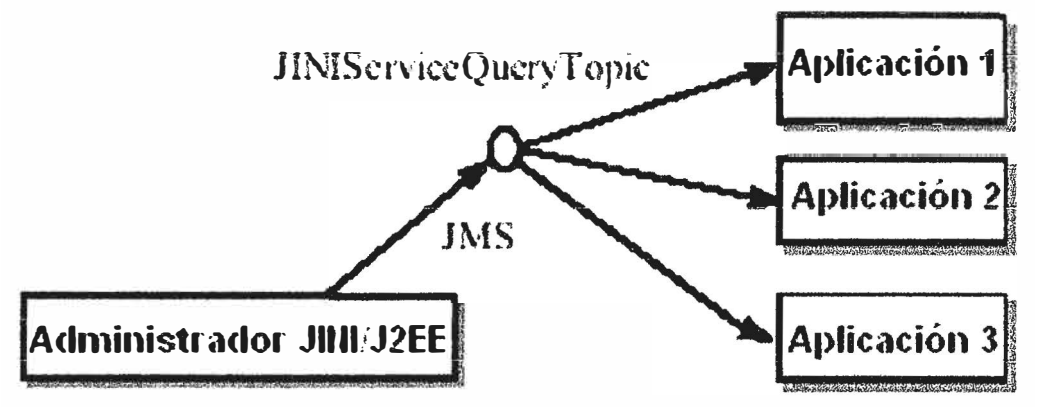

Figura 3.8: Modelo Publicar/suscribir

El administrador del puente Jini/J2EE envía un mensaje Query periódicamente a ciertos tópicos, cuyos JNDI son JiniServiceQueryTopic. Luego de que la aplicación J2EE es desplegada, su mensaje Jini/J2EE se subscribe este tópico y el mensaje Query puede ser recibido. De esta manera cualquier aplicación que va ha ofrecer un servicio Jini puede ser informada de que el proxy del puente está activo.

Sin embargo, como el mensaje de respuesta sólo es dirigido hacia un solo punto, el administrador del proxy Jini/J2EE, se debe usar un sistema de mensajería de punto a punto.

Una vez que la aplicación recibe el mensaje Query, este envía un mensaje de respuesta a una cola determinada con el nombre de JINIServiceRequestQueue. Es entonces que el administrador del puente escucha lo que le dice dicha cola y decide si el mensaje debe ser recibido.

3.4.5 Recuperación y Manejo del Proxy Jini/J2EE

Un administrador de Proxy Jini/J2EE, es responsable de mantener el tiempo de funcionamiento de todos los proxis que corren dentro del contenedor, para ello el administrador debe coordinar con la aplicación J2EE. Si la conexión se pierde, el administrador debe destruir el proxy representado y luego preguntar al contenedor si desea soltar todos los recursos almacenados en el proxy.

Todos los proxis deben ser configurados para un tiempo de expiración cuando son creadores en el contenedor, cuando éstos expiran, el puente los desactiva. El administrador publica mensajes Query periódicamente y estos mensajes disparan los message-driven del Jini/J2EE para enviar una respuesta.

Para un proxy que esté corriendo en ese momento, su tiempo de expiración es extendida según si recibe el mensaje de respuesta. Esto significa que si una aplicación desaparece, no habrá un mensaje de respuesta por un periodo largo de tiempo y el proxy expirará.

3.4.6 Implementación:

Para la implementación de un administrador proxy de Jini/J2EE se debe de contar con:

- **Un Administrador de Proxy Jini/J2EE:** Que es responsable de la creación del proxy y de su interacción con el contenedor.
- **QueryTask:** El propósito del QueryTask es el de establecer una conexión con un Tópico específico en una aplicación.
- **Hilo Response Listener:** Es usado para pasar los mensajes de respuesta a través de los servicios Jini.
- **Incoming Response**: Es usada para analizar los mensajes de respuesta de las aplicaciones J2EE y tomar una acción de acuerdo al contenido del mensaje y del estatus de su proxy representado.
- **ProxyRec**: Esta clase contiene la información de un Proxy, el cual está corriendo en el Contenedor.
- **Hilo de Expiración del Proxy**: Este hilo es usado para mantener una tabla del Proxy en el administrador. Dicha tabla guarda la información de todos los proxies activos en el Contenedor.

En la figura 3.9 se muestra el diagrama de clases de un administrador de Proxy JINI/J2EE.

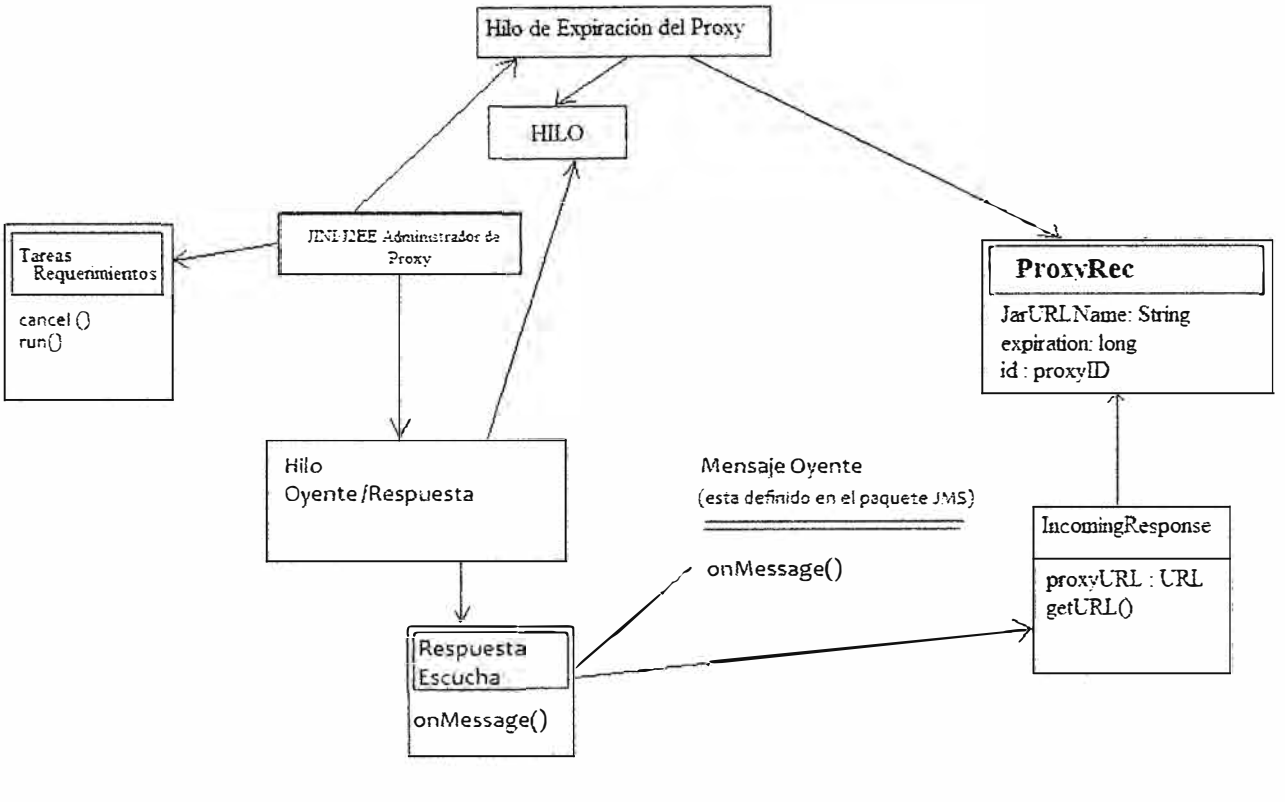

Figura 3.9: Diagrama de clases de un administrador de Proxy JINI/J2EE

CAPITULO IV

APIS DE REDES INTEGRADAS PARA LAS PLATAFORMAS JAVA (JAIN)

Los APIs de Redes Integradas para las plataformas Java, conocido como tecnología JAIN, permiten la integración de la Internet y de los protocolos de Redes Inteligentes (IN), que son referidos como Redes Integradas. Debido a que permiten aplicaciones de Java para tener un acceso seguro a los recursos dentro de la red, se tiene la oportunidad de entregar muchos más servicios que los disponibles sin esta tecnología.

4.1 Objetivos

El objetivo principal de la tecnología JAIN es el de crear una cadena de valores para los servicios de proveedores diferentes y ajenos a nuestra empresa o sistema. La iniciativa de JAIN pretende integrar redes basadas en conexiones alámbricas e inalámbricas, así como redes basadas en paquetes. La adaptación de los protocolos específicos de red al modelo JAIN es cubierta en el Grupo Experto de Protocolos (PEG). La tecnología JAIN puede ser descrita como un modelo de 3 capas, tal como se muestra en la figura 4.1.

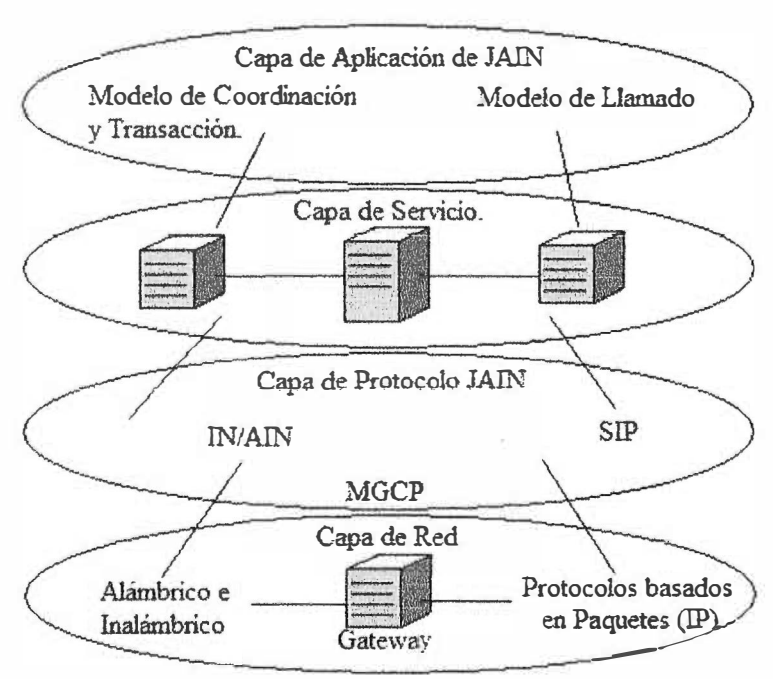

Figura 4.1: Modelo de Capas de JAIN.

Luego de esto, el protocolo visto por el PEG es cubierto por una llamada de control, coordinación y modelo de transacción para ser usado por los servicios solicitados, estas tareas son realizadas por el Grupo Experto de Aplicaciones (AEG).

4.1.1 Objetivos Industriales

El servicio JAIN debe proveer servicios de cobertura y de seguridad para el acceso por vías telefónicas e Internet, con lo que se alterarán algunas estructuras del negocio según lo siguiente:

- **Servicio de Portabilidad:** En la actualidad se cuenta con el problema de que las interfaces que se usan tienen propietarios, con JAIN se busca que estas interfaces sean reformadas en una interfaz uniforme de Java.
- **Cobertura de Red:** Debe proveer las facilidades para que todas las aplicaciones puedan correr en los accesos telefónicos y redes con protocolos basados en paquetes.
- **Seguridad de Acceso a la Red:** Se debe de permitir ciertas aplicaciones que provengan del exterior de nuestra red pero ésta debe ser controlada por motivos de seguridad tanto interna como externa.

Se debe considerar que el mercado de las telecomunicaciones y de la Internet está formado por muchos proveedores o propietarios, todos estos con una gran variedad de servicios. Con la incursión de Java a la red se tiene la oportunidad de integrar muchos servicios que actualmente están disponibles.

4.2 Tecnología Usada

La tecnología usada en JAIN se puede dividir en dos grupos:

- **El Grupo Experto de Protocolo (PEG):** Este grupo especifica las interfaces y las señales de los protocolos IP.
- **El Grupo Experto de Aplicación (AEG):** Indica los APIs requeridos para la creación de servicios dentro de Java que se expanden a través de los protocolos cubiertos por PEG.

Estos dos grupos unen las Redes Inteligentes con las tecnologías de la Internet para proveer un estado apropiado para los servicios y aplicaciones. Algunos ejemplos de estos servicios son: Free Pone, Follow-Me. Time-of-Day y muchos otros servicios de los teléfonos celulares y de la Internet.

JAIN debe incluir los siguientes servicios:

- **Creación de Servicios:** Permite el desarrollo de nuevos bloques de construcción del servicio y el ensamblaje de los servicios de dichos bloques, usando herramientas comerciales a su disposición.
- **Servicio de Ejecución Lógica(SLEE):** Permite la administración y el mantenimiento de dichos servicios durante su ciclo de vida.
- **Servicio de Políticas:** Se encarga de establecer las reglas del comportamiento y uso de los servicios en el sistema.

4.2.1 Arquitectura

En JAIN la arquitectura define una librería de software, un juego de herramientas, un ambiente de creación de servicios y un ambiente para la ejecución lógica de los servicios que construirán la próxima generación de servicios integrados para las redes inteligentes.

La Arquitectura de JAIN está constituido por tres capas: Capa de Servicio, Capa de Señalización y la Capa de Red. También incluye un ambiente de Creación de Servicios (SCE) tanto para los servicios de red confiables como para las aplicaciones de otros propietarios. Los servicios confiables residen dentro del núcleo de las redes públicas, mientras que los servicios escritos por terceros u otros propietarios sólo acceden a las funciones dentro del núcleo de las redes públicas. Mediante un servicio que provee la interfaz, estas aplicaciones de terceros son mantenidas lejos para no comprometer la integridad de estas redes.

En la figura 4.2 se puede observar como el Ambiente de Creación de Servicios y los posibles sistemas de operaciones (OSS/BSS) se encuentran fuera de las tres capas, mientras que el servidor de aplicación se encuentra en la Capa de Servicio y la plataforma de JAIN dentro de la Capa de Señalización. Por último tenemos la Capa de Red que contiene los Protocolos de Red.

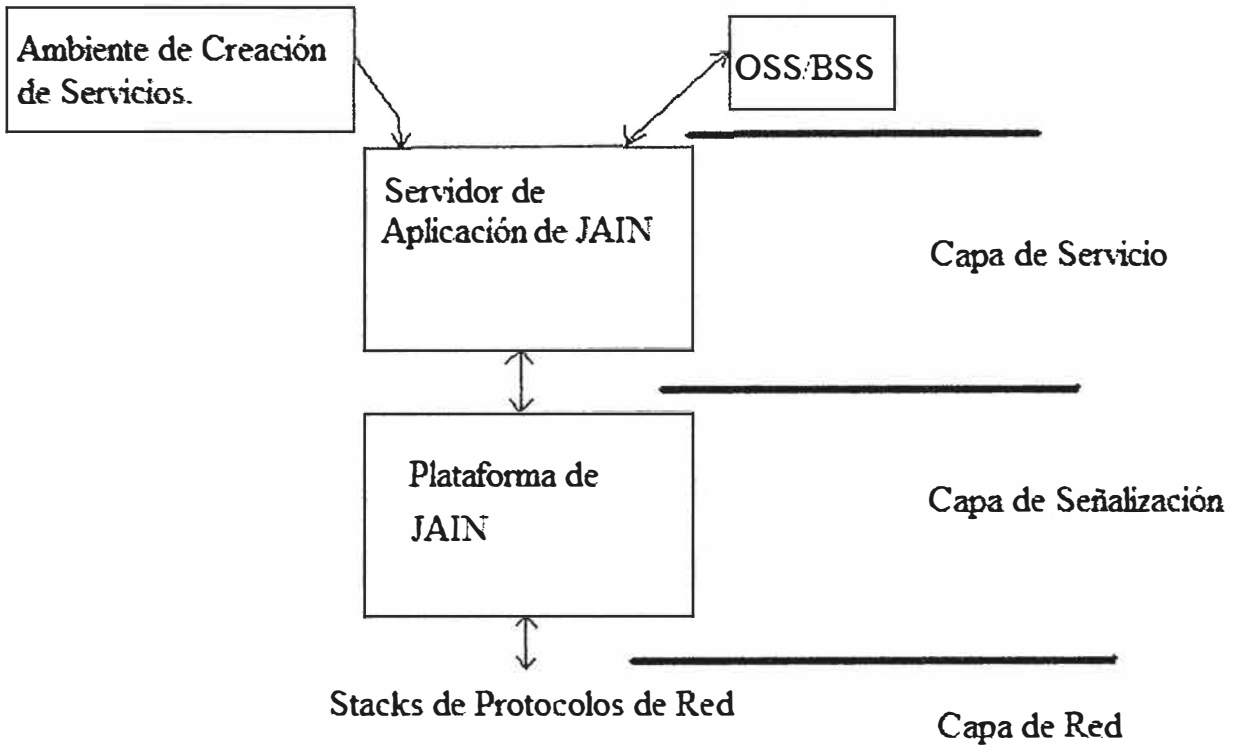

Figura 4.2 : *Arquitectura JAIN*

4.2.2 Interfaces Estándar para la Señalización y Servicios

Se ha definido una serie de estándares para las interfaces que reciben señalizaciones y protocolos de los servicios. La interfaz de control de llamada/seción está definida por el API de control de llamadas de JAIN (JCC) y el Coordinador de Transacciones de JAIN (JCAT).

El objetivo de los APIS JCC/JCAT es el de proveer aplicaciones con un mecanismo consistente para el control de llamadas y los procesos de transacción. EL JCC/JCAT incluye las facilidades requeridas para observar, iniciar, consultar, procesa y manipular llamadas, estas llamadas pueden ser multimedia y sesiones multi-partitas.

JCC es un modelo unificado de llamadas que incorpora las características básicas de la tecnología JAVA y de sus APIs al igual que el Servicio de control de llamadas de Parlay, que ayuda en el soporte de aplicaciones complejas de proceso de llamadas.

La coordinación y las transacciones incluyen facilidades para aplicaciones adicionales que podrían ser solicitadas antes, durante o después del proceso de

llamada, generalmente para agregar nuevas características como una Red Privada Virtual o sesiones que usen bastante multimedia.

4.2.3 Topología de Red

La Topología de red JAIN provee de portadores con la habilidad de desplegar servicios de nueva generación de red en dispositivos dentro o al borde de las Redes Inteligentes, incluyendo dispositivos finales que usen tecnología Java. Obviamente, es completamente necesario que esta topología pueda soportar todos los protocolos usados en la red.

Uno de los puntos principales de los componentes de la arquitectura JAIN es el de mover la capa de señalización lejos de los switches propietarios en los servidores abiertos de los controles de llamadas, también conocidos como agentes de llamadas, controles de gateway para la media o switches suaves. La Señalización es el hilo entre los switches de telecomunicaciones convencionales y los switches suaves, la habilidad de adaptar los componentes de señalización entre redes es gracias a los portadores y los proveedores del servicio de red.

En la figura 4.3 se puede observar una representación de dónde son definidas los APIs de JAIN dentro de una plataforma de comunicación. La arquitectura de los switches suaves está centrada en el mapeo de las interfaces de control de llamada/sesión en los protocolos subyacentes de los APIs. Como los switches suaves realizan señalización en las redes IP, la mayoría son equipados con SIP, MGCP o protocolos subyacentes H323. Muchos switches suaves también incluyen protocolos SS7 para indicar interfaces para las redes telefónicas existentes.

Los APIs de JAIN trabajan en 4 categorías:

- Interfaces de Protocolos o Conexiones.
- Interfaces de Sesión o Control de Llamadas.
- Interfaces Lógicas de Servicios.
- Acceso a los proveedores de Servicios.

Los componentes de JAIN no necesariamente residen en un solo servidor, debido que el Gate Keeper, el control del Gateway y las funcionalidades de la Señalización del Gatewat son típicamente implementadas como una aplicación en todos los elementos de

señalización en la red. Esta característica provee ventajas en cuanto a escalabilidad, rendimiento, confiabilidad, manejabilidad y flexibilidad.

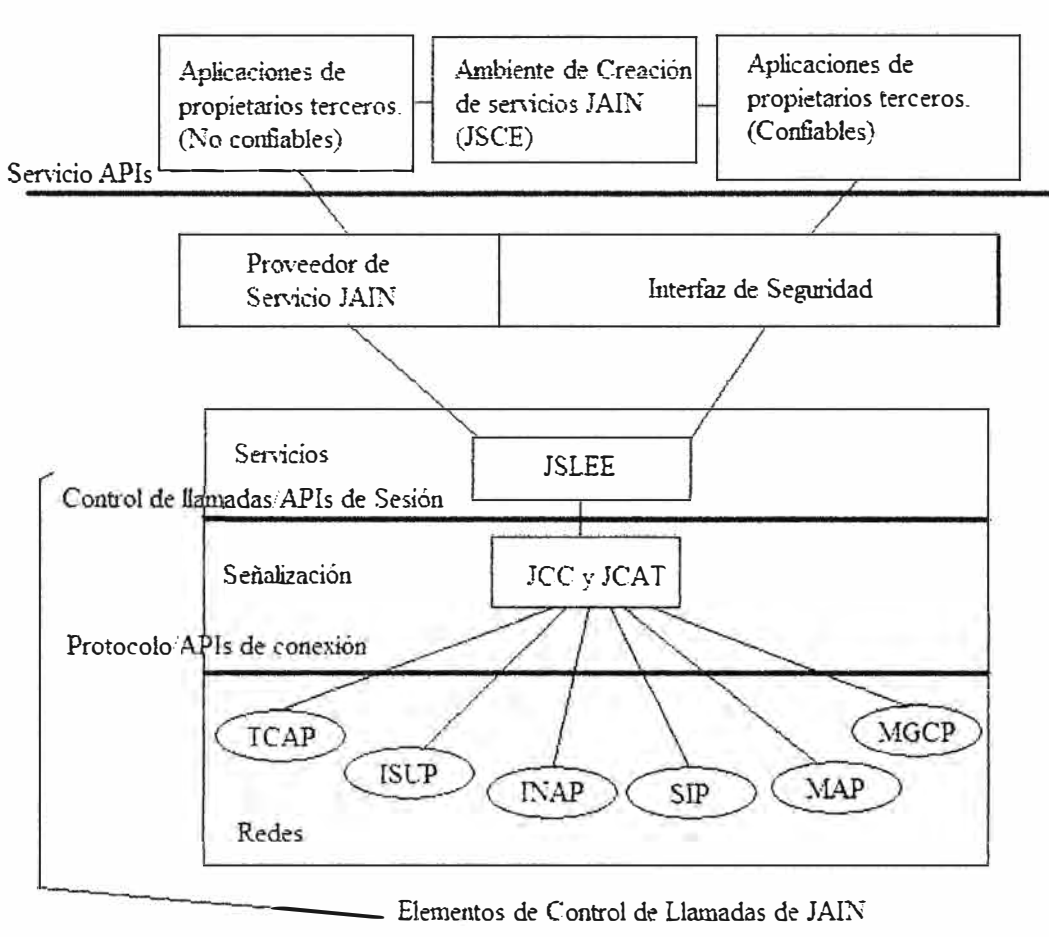

Figura 4.3: APIs de JAIN

CAPITULO V
APLICANDO JINI PARA ESTABLECER UN AMBIENTE COMPUTACIONAL INTEGRADO.

Un ambiente computacional integrado consiste en un ambiente con las facilidades para acceder a la información donde sea y cuando sea, ya sea por medios estacionarios o portátiles. Mientras que el hardware conforma los ladrillos para la construcción de un ambiente computacional integrado, éste también requiere de un software robusto para su correcto funcionamiento y para que sus múltiples aplicaciones puedan ser integrados en una sola unidad cohesiva.

5.1 Objetivos

Un sistema computacional integrado al ambiente es un concepto en el que se debe agregar dispositivos y tecnologías, mayormente invisibles e imperceptibles para los usuarios, a un ambiente común y corriente tanto como en un lugar de trabajo como en un domicilio. Un ambiente como este debe ser un conjunto de dispositivos pequeños y servicios que reaccionen con su entorno, coordinando entre ellos usando una serie de servicios de red para asistir a los usuarios en sus tareas, y para proveerlos con un acceso ubicuo a la información necesaria.

La visión de esta clase de entornos es la de permitir que los dispositivos y las tecnologías se desvanezcan y puedan ser parte de nuestro ambiente natural, esta clase de integración permite a los usuarios tener acceso a dichas tecnologías sin muchas complicaciones y sin necesidad de ser expertos en el manejo de estas, permitiendo que estos puedan concentrarse en tareas más importantes.

La computación integrada al ambiente tiene el potencial de cambiar la forma en que realizamos nuestras tareas diarias, muchas de las aplicaciones incluyen un sistema de casa inteligente mencionado en el capítulo 3 como ejemplo de aplicación. Algunos ejemplos de

estos proyectos a gran escala son los proyectos Oxygen del Instituto Tecnológico de Masachuset y el Proyecto Aura del instituto de la interacción humana con la computación de la Universidad de Carnegie Mellon. También hay grandes corporaciones como el proyecto Blue Planet de la IBM y el proyecto Cooltown de la HP.

Un buen ambiente computacional integrado tiene que tener las siguientes características:

- **Acceso Ubicuo:** Se refiere a un ambiente en donde los usuarios están rodeados de computadoras y aplicaciones y en donde pueden acceder a la información en cualquier momento.
- **Comprensión del Contexto:** Estos ambientes deben de entender lo que el usuario quiere decir o cuales son sus necesidades. Debe adaptarse según su localización, tiempo, dispositivos y sus respectivos usuarios.
- **Comportamiento Inteligente:** Debe ser capaz de encontrar la mejor solución para satisfacer las demandas y necesidades de los usuarios.
- **Interacción Natural:** Una misma función debe ser accedida por medio de muchas interfaces , como la voz, interfaces inalámbricas o reconocimiento de gestos.

Debido a que este ambiente tiene una variedad de dispositivos que necesitan interactuar de manera coordinada y precisa, es necesaria una tecnología robusta que pueda integrar todos estos elementos. Como se ha visto en el capítulo 3, Jini representa una red auto configurable y auto diagnosticable permitiendo la creación de servicios centrados en la red que son altamente adaptables a los cambios.

5.2 Jini en un ambiente computacional integrado

Como se mencionó anteriormente, los ambientes computacionales integrados deben soportar movilidad, adaptación a las rutinas con recursos variables debido a la movilidad, poder lidiar con fallas en el sistema y los cambios que los usuarios necesitan, todo esto con la mínima intervención del ser humano.

Pero principalmente, el ambiente debe de proveer soporte para poder resolver el problema de los cambios en el sistema exponiéndolos en lugar de esconderlos, dando soporte para la composición dinámica y la extensión de aplicaciones y servicios como parte de la rutina y proveyéndole de una clara separación entre los datos y la funcionalidad.

Podemos dar como ejemplo un sistema inteligente implementado en un museo. En la mayoría de los museos se puede ver un sistema de audio que sirve de guía a los visitantes, este servicio de audio guía puede ser dado en forma de audífonos que los visitantes llevarán puestos en su visita al museo. Sin embargo, como no todos los visitantes tienen interés en escuchar lo mismo, en algunos instantes el audio guía puede tornarse un fastidio. Esto se debe a que se usa una misma grabación para todos los visitantes, y el sistema no puede detectar cual es el interés de cada uno de los visitantes y sólo describe una parte relativamente pequeña que podría no ser del interés del visitante. En un hipotético ambiente computacional integrado en un museo, los visitantes estarían equipados con equipos portátiles que coordinarían para asistirlos basados en contextos, intereses y otros factores proporcionados por los mismos visitantes.

5.2.1 Acceso Ubicuo usando Jini

El sistema Jini tiene ciertas características que ayudan a cumplir los requerimientos necesarios para esta clase de ambiente. Uno de estos requerimientos es la infraestructura del área de trabajo que permita accionar pequeños dispositivos inteligentes para descubrir, unir, comunicarse y colaborar con cualquier otro en cualquier lugar y a cualquier hora. De esta forma el sistema provee de un acceso ubicuo a los servicios ofrecidos por éste.

En un sistema Jini, tanto los componentes de software como los de hardwares son vistos como servicios y permite que otros dispositivos se unan a la red. Como se vio en el capítulo III, Jini soporta un "Servicio de Descubrimiento" por medio de su servicio LookUp y su protocolo de descubrimiento, lo que permite a los servicios el poder descubrir y unir las redes para que estas interactúen con otros servicios conectados a la red con cierta facilidad. Debido a que el código es descargado a los clientes que usan el servicio durante la rutina, esto elimina la necesidad de los drivers para los dispositivos. A si mismo, Jini puede ser usado con casi cualquier protocolo de comunicación, sea de cableado o inalámbrico, para poder establecer servicios de comunicación.

5.2.2 Comprensión de Contexto por medio de Jini

Otra característica que se puede lograr gracias a Jini, es la de comprensión del contexto, lo cual permite proveer un mejor servicio a los usuarios basado en la información

del contexto, en términos de entendimiento de localización, tiempo, dispositivos y sus respectivos usos. Los dispositivos móviles en el ejemplo del museo podrían proveer de información relevante con respecto al contexto de lo que cada visitante pueda estar interesado. Ya que Jini permite la formación espontánea de redes de servicio y que pueden ser usadas con cualquier protocolo de comunicación, la información acerca del contexto podría ser fácilmente obtenida a través de interacciones entre los dispositivos inteligentes de Jini.

5.2.3 Infraestructura para un comportamiento inteligente

Luego también se puede lograr un comportamiento inteligente en el sistema gracias a Jini, ya que el ambiente debe tener la habilidad de adaptarse dinámicamente al comportamiento del usuario para que pueda proveerlo de la información que realmente necesita. En el ejemplo del museo, el sistema guía de audio debe de adaptarse a los patrones variantes del comportamiento del usuario. A pesar de que Jini no ofrece capacidad inherente de personalización y otros atributos relacionados a una clase de "inteligencia", este si provee del soporte para la infraestructura requerida para la implementación de componentes inteligentes a través de aplicaciones lógicas.

5.2.4 Interacción Natural

En esta clase de ambiente se debe de tener muchas interfaces que puedan proveer la misma funcionalidad y que puedan ser dadas por voz, interfaces inalámbricas o reconocimiento de gestos. Para esto Jini puede establecer múltiples interfaces para los usuarios y asociarlas todas en un simple servicio por medio del adaptador ServiceUI. Esto provee un mecanismo para adjuntar cualquier adaptador de interfaz a un servicio permitiendo al servicio Jini ser accesible para los usuarios con diferentes interfaces variables.

5.2.5 Seguridad y Confiabilidad

Todo sistema de red debe ser confiable y seguro, en cuanto a la confiabilidad Jini provee mecanismos como el "leasing" que ofrecen soluciones a fallos parciales en la red

sin la necesidad de una administración manual de esta. En cuanto a la seguridad, Jini extiende Java en un ambiente distribuido, de esta manera provee a la red protección contra códigos maliciosos que corran en JVM. A parte también soporta servicios clásicos como autenticación, confidencialidad e integridad.

5.3 Jini y los dispositivos móviles

A pesar de que la arquitectura Jini orientada a servicios acopla tanto dispositivos de hardware como software en una comunidad de red, existen ciertos requerimientos que cada participante debe tener para poder unirse e interactuar exitosamente con la red de Jini. Estos requerimientos incluyen capacidad de almacenamiento, la habilidad de participar en descubrir y unir protocolos, y también la habilidad de descargar , ejecutar y exportar código. Para hacer que Jini funcione en dispositivos móviles, se requiere una cantidad considerable de recursos que no siempre son accesibles, sin embargo, se han desarrollado algunas tecnologías para resolver esta clase de problema.

5.3.1 Arquitectura Subrogada de Jini

Provee ciertos medios para dispositivos que tienen capacidades limitadas para participar en una red Jini usando la ayuda de otros componentes capaces de Jini, los cuales tienen los recursos para participar en las federaciones de Jini por medio de ciertos requerimientos. El Host Subrogado es una máquina que cumple con todos los requerimientos de las federaciones de Jini y tiene recursos suficientes para soportar la arquitectura subrogada. El subrogado (el objeto proxy del dispositivo) y el dispositivo están en contacto entre ellos a través de un adaptador de interconexión residente en la máquina del host y un protocolo interconectado. Cualquier protocolo, ya sea por cable o inalámbrico, puede ser usado para las comunicaciones entre el host y el dispositivo.

5.3.2 ServiceUI

El SeriviceUI da soporte a las interacciones naturales en un ambiente computacional integrado implementado con Jini, en donde los servicios deben de poder soportar múltiples interfaces para presentar la misma funcionalidad a través de diferentes interfaces.

5.4 Diferentes alternativas a Jini

Las características como el servicio de descubrimiento, auto-diagnostico y auto-configuración hacen de Jini una herramienta poderosa para la creación dinámica de redes y facilitan su uso en entornos inteligentes. Sin embargo, Jini no es la única clase de tecnología que se puede adecuar para este tipo de aplicaciones. Tanto Microsoft como HP han diseñado y desarrollado nuevas tecnologías para facilitar la interacción entre dispositivos. Los más representativos son los UpnP (Universal Plug and Play) de Microsoft y los JetSend de HP.

5.4.1 UPnP de Microsoft

Es una arquitectura de computación distribuida dirigida al manejo de algunos componentes en la computación integrada. Es una fuente abierta, con una arquitectura abierta y es que tiene una plataforma y lenguaje
independiente. Esta tecnología sigue los estándares industriales para las redes como HTTP, SOAP y XML.

Esta nueva tecnología provee una infraestructura para la formación de comunidades dinámicas de dispositivos y al igual que Jini, también elimina la necesidad de la instalación de diferentes clases de drivers para los dispositivos y soporta la configuración "cero" y los descubrimientos automáticos.

En UPnP, cualquier dispositivo que desea ofrecer servicios, debe advertir su presencia o estado a través de los SSDP (Protocolo Simple de Descubrimiento de Servicios), de la misma manera los clientes usaran el mismo método para solicitar los servicios que requieran. Tanto los servicios como sus clientes interactuarán usando mensajería SOAP.

A pesar de que UPnP usa estándares industriales y funciona bien con los dispositivos, en la actualidad este no soporta los servicios de software. Esta falta de soporte le da a Jini la ventaja en ser preferido sobre el UpnP

5.4.2 JetSend de Hewlett Packard

JetSend es una tecnología potente de comunicación independiente de la media de

punto-a-punto desarrollado por HP. Esta tecnología permite a los dispositivos conectarse entre ellos al intercambiar información, en contexto apropiado, sin la necesidad de conocer ninguna clase de información acerca del funcionamiento y características del otro dispositivo con el que se intercambia la información, lo cual es muy útil porque al igual que Jini, elimina la necesidad de instalar drivers en los equipos. Sin embargo, los dispositivos que estén involucrados deben tener el software de JetSend instalado para que puedan inter-operar e intercambiar información.

La tecnología JetSend trae consigo un concepto llamado e-material o "superficie" que todo dispositivo involucrado necesita poseer. Estos dispositivos pueden comunicarse uno con otro por medio de cualquier protocolo. En JetSend se puede ver una arquitectura simple pero robusta, la cual es independiente de las máquinas y protocolos, de esta forma se elimina la necesidad de que los dispositivos tengan los drivers.

Sin embargo, al igual que UPnP de Microsoft, todo esto está limitado a la comunicación entre dispositivos e impone la necesidad de que el software de JetSend tenga que estar instalado en cada uno de los dispositivos, dicha característica limita sus opciones para ser escogido como tecnología para la implementación de un ambiente computacional integrado.

La tabla de la figura 5.1 muestra los puntos de comparación entre las tecnologías mencionadas con Jini.

Características	Jini	UPnP	JetSend
Independencia de Protocolos	Sí	No	Yes
Auto-Configuración	Sí	Sí	Yes
Auto-reparación	Sí	Sí	Yes
Soluciona problemas de latencia en la red.	Sí	Sí	Yes
Eliminación de distinción entre Hardware y Software	Sí	No	No

Figura 5.1: Tabla comparativa de las características de las posibles tecnologías a ser usadas en un ambiente computacional integrado.

5.5 Implementaciones de Jini en el mercado

A pesar de que Jini es una tecnología relativamente nueva, se le considera como una tecnología confiable para ambientes distribuidos e integrados. Algunos de estos ejemplos en el mercado actual son:

- Eko Systems Inc, es una compañía que se dedica al desarrollo de productos para el cuidado de la salud, y ha escogido Jini para crear un sistema de monitoreo electrónico para los pacientes en los hospitales. Los requerimientos para este sistema es que sea simple de operar, confiable, que tenga la habilidad de recolectar y analizar los datos de los pacientes en tiempo real, para crear de manera electrónica un historial médico y poder mostrar los resultados obtenidos a través de una variedad de equipos médicos. Se le considera confiable ya que Jini tiene la capacidad de acepta cualquier dispositivo en su red sin necesidad de la instalación de drivers y la manera de cómo soluciona los fallos en su sistema.
- El proyecto Jgrid está siendo desarrollado por el grupo de Parallel and Distributed System de la universidad de Veszprem en Hungría. Su objetivo es el de desarrollar una infraestructura tipo "malla" orientado a servicios donde los componentes pueden unirse y separarse del sistema en cualquier momento, y que los usuarios puedan acceder a la "malla" con facilidad. El proyecto hace uso de las propiedades de Jini como el descubrimiento de servicios, leasing y la visión orientada a servicios para crear un sistema "malla" confiable y dinámico.
- La compañía PsiNaptic ha desarrollado una plataforma para computación integrada usando Jini, que consiste en un hardware y software modular que puede aumentar la habilidad de un objeto para interactuar en un ambiente computacional integrado y se le conoce como la solución "Psinode".
- Zucotto Systems ha desarrollado una versión comercial de la arquitectura subrogada de Jini llamada "Edge Zucotto Xpresso" que está diseñada para habilitar dispositivos, desde simples a complejos, para hacer uso y proveer servicios Jini.

La tecnología Jini ha ganado popularidad en el área de la tecnología confiable para el servicio de las necesidades en la computación integrada, y ha sido adoptada por muchas compañías y proyectos de investigación. A pesar de que se han desarrollado muchas tecnologías de manera paralela que también se concentran en resolver los mismos

problemas a los que va dirigido Jini, todos estos carecen de ciertas características, lo cual los limita en cuanto a efectividad en la implementación de ambientes inteligentes.

CAPITULO VI
VISUALIZACION INTERACTIVA Y PRUEBAS PARA LOS SERVICIOS JINI

Las arquitecturas dinámicas orientadas a servicios, como lo es Jini, tratan de proveer más flexibilidad y robustez al sistema y darles la posibilidad de que puedan funcionar aun con los cambios que puedan haber durante su funcionamiento. Sin embargo, su naturaleza dinámica puede significar ciertos problemas cuando uno trata de entender su funcionamiento, su desarrollo y sobretodo la etapa de pruebas del sistema en donde se debe tratar de localizar los errores o "bugs" en el funcionamiento del sistema.

6.1 Objetivos

Las aplicaciones distribuidas de Jini tienen muchas ventajas, como el incremento de la confiabilidad, mejor rendimiento y mantenimiento, sin embargo, es una tarea más compleja diseñar y construir dichos sistemasdo. Los diseñadores deben de tratar de resolver problemas relacionados con características únicas de los sistemas distribuidos como el incremento de latencia, restricciones de ancho de banda, fluctuaciones, requerimientos de seguridad y el incremento en el riesgo de una falla parcial o total de los nodos en la red.

Los sistemas que dependen totalmente de la implementación de una red informática cada vez se hacen más comunes y populares, con diferentes anchos de banda, algunas pueden ser móviles y tener una variedad muy amplia de dispositivos, es por eso que cada vez más empresas están aplicando esta clase de tecnología, por lo que los analistas y desarrolladores deben de tratar con nuevas complicaciones.

6.2 Principales complicaciones al diseñar e implementar tecnología Jini

Muchos de los problemas a los que un diseñador debe de enfrentarse un sistema dinámico distribuido son resultado de sus características generales.

6.2.1 Complejidad Inherente

En sistemas basados en servicios de descubrimiento dinámico se observa que tiene que basarse en la habilidad de manejar lo que al final será una serie de problemas en el desarrollo del sistema. Debido al uso de software medios o middleware, la abstracción y la asistencia de librerías re-usables, los costos de desarrollo y los objetivos finales son bastante complejos.

6.2.2 Disponibilidad y Estado

Tradicionalmente los sistemas distribuidos están usualmente basados sobre la suposición de que los recursos son conocidos, estables y disponibles. Si los componentes del sistema tienen identidades conocidas y capacidades es mucho más fácil de verificar su estado y disponibilidad durante las pruebas. La transparencia de los métodos usados por el servicio de descubrimiento es una ventaja en el sentido de que se puede hacer que el mantenimiento de sistemas sea una tarea relativamente ligera, pero a su vez, hace que la tarea de hallar errores en las pruebas sea una tarea más difícil.

6.2.3 Cambios de estado en el sistema dinámico

La naturaleza dinámica de los servicios y su interacción significa que a medida que pasa el tiempo de funcionamiento, no hay garantías de que el servicio usado previamente estará disponible nuevamente o que sea el más adecuado desde el punto de vista del cliente.

6.2.4 Dificultad de creación de servicios en aislamiento

Aunque los servicios puedan ser parte de una aplicación general, es potencialmente difícil someterlos a pruebas y detectar sus errores de diseño en aislamiento. Para revisar y hacer pruebas de servicios es necesario desarrollar un código cliente para usar los servicios remotos, y luego hacer pruebas de funcionalidad. Esto significa que es necesario escribir un código adicional para el cliente en forma de unidades de prueba, o desarrollar simultáneamente clientes para estos servicios para probar su correcto funcionamiento. Es

necesario que este proceso se repita en cada paso interactivo del desarrollo. Esto hace que muchos diseñadores se desanimen el adoptar esta tecnología.

Adicionalmente si estos objetos no tienen un rendimiento correcto, es difícil de corregirlos sin usar alguna clase de programa tipo "driver" que pueda ser corrido con el "debugger". Podría ser beneficioso si los servicios podrían ser probados y corregidos al nivel de servicio, usando mecanismos de interacción directa.

6.2.5 Entendimiento de los servicios

En un ambiente donde las arquitecturas orientadas a los servicios son usados, existe la necesidad y la tendencia de usar los servicios existentes. Los desarrolladores usaran los servicios que ya hayan sido escritos por otros, por lo que es importante entender los servicios existentes y recolectar los datos necesarios acerca de ellos. También la habilidad de interactuar con los servicios directamente podría ser de gran utilidad.

6.2.6 Metodologías del desarrollo de interactividad

Existe un gran interés en el desarrollo de metodologías altamente interactivas como Extreme Programming (XP), Cristal y Scrum. Estos también son referidos como procesos de desarrollo rápidos y ligeros. El propósito principal de dichos procesos es la noción de desarrollar sistemas complejos por medio de una interacción fina de prototipos de desarrollo que constantemente evolucionan, son probados y luego rehechos. Es por lo que en estos sistemas el rol de las unidades de prueba y de "debugging" son muy importantes. Estas pruebas o tests deben de correr relativamente rápido, lo cual hace difícil la configuración y el mantenimiento de la infraestructura dinámicamente distribuida, esto también trae complejidad en las tareas y la necesidad de más recursos. Esta es la causa de que muchas personas decidan no usar esta aproximación altamente interactivo.

Para hacer pruebas en esta clase de procesos de un sistema bastante rápido existe la opción del testing exploratorio, el cual es similar a un testing ad hoc, donde el encargado de llevarlo a cabo puede escoger cual de todas las pruebas va a usar y de esta manera explorar el comportamiento del software. Los que proponen este tipo de testing aseguran que tiene una gran efectividad en el diseño y uso de pruebas que estén basados en los resultados de pruebas anteriores. De esta forma un comportamiento inesperado puede ser

explorado y entendido en el mismo momento en que se le encuentre. Este tipo de pruebas no es visto como un método alternativo, es mas bien visto como un complemento a los procesos automáticos de pruebas.

El adoptar esta metodología en el proceso de pruebas para un sistema dinámicamente distribuido permite a los que lo examinan el poder responder a los cambios de la disponibilidad de servicios y otra clase de condiciones de red. Para que todo esto funcione es necesario proveer del mecanismo que permita al desarrollador que este realizando las pruebas el poder monitorear los cambios que se realicen en el sistema y poder invocar él mismo las pruebas especificas para dichos cambios cuando estos sucedan.

6.2.7 La Interacción entre el cliente y el servicio no es visible

Como en Jini el proxy escogido para los servicios es descargado al cliente para satisfacer sus necesidades, este proxy puede usar cualquier protocolo soportado para proveer el servicio que este representa. Si bien esto permite una gran flexibilidad en el tipo de servicios que pueden ser acomodados en un sistema Jini, esto también hace que sea más difícil el monitorear y corregir los servicios individuales y las aplicaciones ya que los protocolos y mecanismos que usan pueden variar según el servicio que el cliente este requiriendo o usando.

6.3 Herramientas requeridas

En los sistemas dinámicos es importante que los desarrolladores puedan reconocer y entender los cambios que este realiza. Es por ello que son importantes las herramientas tipo "browser" que permita a los desarrolladores el investigar e interactuar con el estado actual de un sistema dinámico basado en servicios.

6.3.1 Visualización

La visualización de software es un técnica bastante reconocida en cuanto a la comprensión de los sistemas de software. Algunas acciones empíricas en cuanto a la visualización de la programación pueden ser bastante efectivas, y que uno de los puntos clave es identificar el tipo más apropiado de representación gráfica para un sistema particular.

Para una correcta visualización se debe de considerar los siguientes puntos:

- **Múltiples vistas apropiadas:** El hecho de usar múltiples vistas para visualizar un sistema ayuda bastante en la comprensión del funcionamiento de un sistema. Una de las características comunes entre diferentes arquitecturas de servicio es que ellos proveen un cierto nivel de transparencia en la implementación, escondiendo detalles de la red y otras características del entorno. Un cliente solicita servicios y recibe respuestas acerca de que servicios son los más apropiados según ciertos criterios. La correspondiente vista a nivel de servicio mostrará gráficamente los servicios disponibles. Con la adopción de procesos de desarrollo altamente interactivos , la abstracción visual apropiada del sistema puede cambiar durante las diferentes tareas.
- **Arquitectura extensible para vistas adicionales:** Los servicios pueden presentar una amplia variedad de diferentes servicios y dispositivos, no está claro cual es la más apropiada abstracción para todos los tipos de aplicación. Estas arquitecturas deben ser usadas para implementar computaciones basadas en servicios de alto volumen o representar dispositivos de poco peso y servicios en una infraestructura móvil o integrada. También hay un requerimiento de que una herramienta tipo "browsing" sea extensible para que visitas adicionales puedan ser agregadas con el fin de que puedan soportar de manera apropiada ciertos tipos de sistemas.
- **Actualización dinámica de vistas:** La naturaleza dinámica de este tipo de sistemas hace que las herramientas tengan que actualizar dinámicamente la representación del sistema a medida que este vaya cambiando.
- **Manipulación arreglos de datos potencialmente grandes:** Un problema común en la representación gráfica de datos es que se debe de manipular arreglos de datos que puedan ser potencialmente grandes. Todo sistema que tenga que visualizar una red dinámica de servicios y sus atributos necesitará manejar un gran número de objetos.

6.3.2 Interacción

El contar con muchas técnicas de manipulación directa sobre las entidades gráficas que representan el sistema provee un gran número de ventajas. La interacción directa de

las entidades gráficas provee de medios para inspeccionar su estado y comportamiento.

Un buen ejemplo es los entornos de programación BlueJ, el cual provee un número de medios para que los usuarios interactúen con los programas de Java. Esta herramienta provee de una representación gráfica muy parecida a UML para las clases de Java, mostrando dichas clases y sus respectivas relaciones. Los usuarios interactúan directamente con las representaciones gráficas de las clases para abrir editores, iniciar instancias e inspeccionar sus respectivos estados. También permite a los usuarios experimentar con el código sin necesidad de los overheads usuales por escribir programas tipo driver o clases para pruebas.

A pesar de que BlueJ tiende a enfocarse en proveer visualización e interacción dirigida a los programadores principiantes, también es muy útil en la comprensión de programas y ha dado facilidades para que usuarios más experimentados hagan prototipos y experimentos para probar la funcionalidad de algún código que estén desarrollando.

Esta clase de herramientas interactivos permiten a los usuarios al nivel del objeto y proveen un manera de que los desarrolladores puedan observar y estudiar los servicios existentes y permitirles el poder hacer pruebas de manera exploratoria y basada en el contexto.

CONCLUSIONES

El trabajo realizado ha permitido estudiar un conjunto de herramientas para el estudio y un posible diseño optimo de una red inteligente basada en ontologías.

Se han determinado los posibles entornos en que este tipo de diseño de red inteligente podría ser de gran utilidad, concluyéndose que existen tantas aplicaciones para la vida cotidiana, como el caso de un sistema de almacenamiento de víveres o aplicaciones industriales, como el caso de un sistema inteligente de telefonía IP.

Para realizar un modelo de un sistema de ontologías se cuenta con una serie de herramientas, sin embargo, se escogió usar la herramienta Jini, ya que el lenguaje Java es bastante flexible y dinámico. Para el presente trabajo se usaron modelos de diagrama de flujo para poder explicar el funcionamiento y comportamiento de los sistemas mostrados, teniendo como ejemplo el sistema de Telefonía sobre IP.

Uno de los objetivos en usar este tipo de herramienta consiste en proveer mejores métodos para la investigación y la solución de los posibles problemas en el dominio de un sistema, de esta manera se pueden aplicar técnicas que antes no estaban disponibles debido a la falta de recursos y a un entendimiento en el significado o semántica de la información o de los requerimientos de los usuarios.

En el desarrollo de un sistema telefónico, es necesario usar una arquitectura centralizada y cerrada, también es de esperar que dicha arquitectura fortalezca los servicios de los clientes y usuarios. Se propuso usar Jini para este modelo ya que la naturaleza del ambiente de una red telefónica es bastante dinámico, teniendo a J2EE como una rama de aplicaciones globales empresariales. Es evidente que los servicios de telefonía IP deben de soportar tanto una administración dinámica como un acceso global.

Otra herramienta estudiada en este trabajo es JAIN para las redes integradas, que agrega portabilidad en los servicios además de mejorar la seguridad en el acceso telefónico. La herramienta JAIN se vio como un adicional para este trabajo y de cómo podría ser implementada para mejorar un sistema telefónico inteligente.

En general el tema de usar ontologías está siendo debatido en todo el mundo, con el tiempo se debe de regular y aplicarse las mismas normas o reglas que se aplican a las empresas que usan sistemas afines tradicionales como por ejemplo: la telefonía pública.

Con el transcurso del tiempo es evidente que las necesidades de las personas se harán más puntuales por lo que el uso de un sistema de ontologías se hará mucho más común y se estudiará más a fondo. Actualmente el uso más común que se le está dando es el de la domótica, sin embargo, en este trabajo se decidió sustentar una aplicación más generalizada y más comercial como es el del servicio de telefonía IP.

BIBLIOGRAFÍA

1.- Martin Heidegger. "Ontología. Hermenéutica de la facticidad", Ed. Alianza, Madrid. 1998. Trad. de Jaime Aspiunza.

2.- Jini specification. http://wwws.sun.com/software/jini/specs.

3.- Jini Surrogate Architecture Specification, http://surrogate.jini.org/specs.html.

4.- Kumaran, I. "Jini Technology: an Overview", Prentice Hall PTR, 2001.

5.- Xueqin Huang y John A. Miller. "Building a Web–Based Federated Simulation System with Jini and XML", Univirsity of Georgia, 2004

6.- Perrone, Paul J y Chaganti, Krishna. "J2EE Developer's Handbook". Indianapolis, Indiana: Sam's Publishing. 2003

7.- Bodoff, Stephanie. "The J2EE Tutorial". Boston: Addison-Wesley. 2004

8.- The Davis Project, http://davis.jini.org/index.html#v2_0

9.- Jini Technology - network-centric services, California Software Labs, http://www.cswl.com/whiteppr/tutorials/jini.html.

10.- JavaBeans specification, http://java.sun.com/products/javabeans/docs/spec.html.

11.- Gagan Tandon y Sudipto Ghosh, "Using Subject-Oriented Modeling to Develop Jini Applications", Colorado State University. 2004