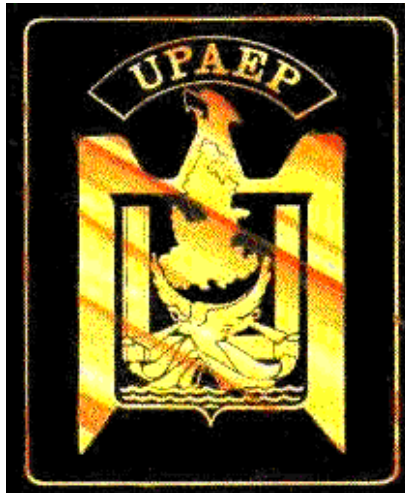


UNIVERSIDAD POPULAR AUTÓNOMA DEL ESTADO DE PUEBLA

Departamento de Ingeniería Industrial y Tecnologías de la Información

Escuela de Ingeniería en Computación



“Acceso a servicios Jini, a través de dispositivos móviles con tecnología J2ME bajo la configuración CLDC y el perfil MIDP”

Tesis profesional presentada por:

José Arturo Tejeda Gómez

Mat. 860053

Como requisito para obtener el título de
Ingeniero en Computación

Puebla, Pue., Mayo de 2003



UPAEP – Secretaría General

Dirección General de Apoyos Académicos

Dirección del Centro de Recursos para el Aprendizaje y la Investigación.

Biblioteca Central - **Karol Wojtyla**

Tesis Digitales Restricciones de uso:

DERECHOS RESERVADOS ©

PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de textos, imágenes, gráficas, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente de donde la obtuvo mencionando el autor o autores involucrados en el documento.

Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

AGRADECIMIENTOS

A Dios... Gracias por darme todo lo que necesito en mi vida, gracias por no abandonarme y seguir junto a mí, gracias por darme la oportunidad de rectificar mi camino, tú sabes que mi vida es tuya.

A mi familia... Deseo agradecer profundamente a Dios que me haya otorgado la oportunidad de compartir su vida conmigo en un hogar maravilloso, el cual siempre recordaré. Sin el apoyo en todo sentido de mis padres y mi padrino, el placer cotidiano de vivir sería simple monotonía. Es difícil imaginar cómo sería mi vida sin recordar su comprensión, su apoyo y su inmenso amor, todo lo que soy se los debo a ustedes.

A mis amigos....Zeus: Eres un amigo único, gracias por apoyarme cuando más lo necesitaba y por compartir tu vida conmigo, fuiste parte fundamental en esta tesis ya que tus sugerencias y contribuciones resultaron de gran utilidad. Luciano: Has estado en diversos momentos de mi vida, te agradezco tu amistad que me has demostrado siempre. Gustavo: Gracias por tu amistad sincera y tu apoyo incondicional. Y en general, a todos mis amigos que han dejado una huella imborrable en mi vida.

Al Dr. Oleg Starostenko Basarab...Gracias por dirigirme cuando todo era un mar de confusión, por su paciencia y por apoyarme en el desarrollo de esta tesis.

A mis maestros...Sin lugar a duda este proyecto de tesis no hubiera sido posible, gracias a la formación que recibí de la escuela de Ingeniería en Computación (U.P.A.E.P.). De manera especial, a los maestros Ricardo López Fabre y Victor Manuel Cortés Galvan, por su apoyo en el desarrollo de este proyecto. Gracias a todos los maestros que contribuyeron en mi desarrollo académico y en mi formación personal.

A mis sinodales...Quiero dar las gracias a las personas que revisaron con paciencia éste trabajo: José Ignacio Castillo Velazquez, Vittorio Zanella Palacios y Hector Simón Vargas.

A Gladys Rivera...Gracias por darle un nuevo sentido a mi vida, gracias por los momentos que pasamos juntos y principalmente por tu cariño y amor.

Por último hago extensivo mi agradecimiento a todas aquellas personas que creyeron en mí, que me devolvieron la alegría cuando todo era difícil y todos aquellos que han puesto de su parte para que la cotidianidad de la vida pueda ser llevadera para mí...

DEDICATORIAS

Con cariño inmenso a Dios, a mis padres y a mi padrino...

SÍNTESIS

La constante evolución de los dispositivos móviles ha generado la necesidad de integrarlos a las redes computacionales, para acceder a la información contenida en ellas e interactuar con los componentes y elementos de software y hardware que las constituyen. El concepto de sistemas distribuidos ha sido absorbido por las redes computacionales con el objetivo de crear redes dinámicas, entre otras cosas.

En este proyecto de tesis, se ha planteado el problema de resolver el acceso e intercambio de información a través de dispositivos móviles (teléfonos celulares), a los servicios provistos por un sistema distribuido, utilizando tecnologías emergentes como lo son: Jini y J2ME (Java 2 Micro Edition). Este planteamiento ha provocado que se analicen los siguientes puntos:

- La tecnología Jini, en el ámbito de los sistemas distribuidos y la implementación de dichos sistemas.
- Las generaciones de dispositivos móviles respecto a la telefonía celular, en específico los que soportan la tecnología J2ME.
- La tecnología J2ME, en el ámbito de los teléfonos celulares y las aplicaciones definidas por la propia tecnología.
- La infraestructura actual de comunicación inalámbrica GPRS (General Packet Radio Service) que da soporte a los teléfonos celulares para la transmisión y recepción de datos.

Con el objetivo de mostrar la interacción de las tecnologías Jini y J2ME, para resolver el problema del acceso de un teléfono celular, que soporte la tecnología J2ME, a servicios residentes en una comunidad Jini, se ha desarrollado un sistema de software basado en aplicaciones definidas por las tecnologías anteriormente mencionadas.

La mayor contribución de este proyecto es proveer una arquitectura de red que sitúe un punto de referencia en el acceso y desarrollo de servicios provistos en un sistema distribuido con el uso de tecnologías Jini y J2ME, así como demostrar con base en la utilización de herramientas de software gratuitas que se puede simular y emular componentes de software para el diseño, implementación y pruebas del propio sistema.

ÍNDICE GENERAL

INTRODUCCIÓN	1
CAPÍTULO 1. El Problema	3
1.1. La problemática actual	3
1.2. Servicios móviles a nivel mundial	6
1.3. Servicios móviles en México	7
1.4. Principales componentes de los servicios móviles	8
CAPÍTULO 2. Especificación del Proyecto	11
2.1. Planteamiento del problema.....	11
2.2. Objetivos	11
2.3. Justificación	12
2.4. Hipótesis	13
2.5. Alcances y limitaciones	14
2.6. Metodología.....	14
CAPÍTULO 3. Los sistemas distribuidos.....	19
3.1. Conceptos básicos y definición	19
3.2. Funciones de un sistema distribuido.....	21
3.3. Aspectos de un sistema distribuido.....	22
3.4. Taxonomía de los sistemas distribuidos	24
3.4.1. Configuraciones Hardware.....	24
3.4.2. Aspectos del Software.....	25
3.5. Implementación de un sistema distribuido	27
3.5.1. Sistema distribuido basado en un sistema operativo distribuido.....	27
3.5.2. Sistema distribuido basado en un sistema operativo de red	28
3.5.3. Sistema distribuido basado en un middleware	29
3.6. Ventajas de los sistemas distribuidos	32
3.7. Desventajas de un sistema distribuido	33

<i>CAPÍTULO 4. La tecnología Jini</i>	34
4.1. Antecedentes	34
4.2. Términos básicos	34
4.3. Jini: Tecnología de integración de sistemas distribuidos	35
4.3.1. Visión	35
4.3.2. Metas	36
4.3.3. Supuestos.....	37
4.4. Componentes de un sistema Jini	37
4.4.1. Infraestructura Jini.....	38
4.4.2. Modelo de programación	38
4.4.3. Servicios	39
4.5. Una arquitectura basada en servicios	39
4.6. Funcionamiento general de Jini	40
4.7. Grupos Jini	43
4.8. Protocolos Jini	45
4.8.1. Protocolos discovery	45
4.8.2. Protocolo join	47
4.8.3. Protocolo lookup	48
4.9. Lookup Service	48
4.9.1. Los roles de un lookup service	49
4.10. Implementación de servicios en Jini	50
4.11. Java RMI, soporte de Jini	51
4.12. Jini Surrogate	55
4.12.1. Requerimientos de la arquitectura surrogate	55
4.12.2. Definiciones	56
4.12.3. Perspectiva General.....	57
4.12.4. Discovery and Surrogate Loading	57
4.12.5. Ejecución surrogate	59
 <i>CAPÍTULO 5. Tecnologías móviles actuales</i>	 60
5.1. Conceptos básicos de telefonía celular	60
5.1.1. La célula	60
5.1.2. Arquitectura básica de un sistema telefónico celular	61
5.2. Historia de la telefonía celular	62
5.3. Generaciones de la telefonía celular	64
5.3.1. Primera generación de la telefonía celular	64

5.3.2. Segunda generación de la telefonía celular	64
5.3.3. La generación 2.5 de la telefonía celular.....	67
5.3.4. Tercera generación de la telefonía celular.....	70
5.4. Situación actual de la tecnología móvil en el mundo.....	72
5.5. Situación actual de la tecnología móvil en México	72
<i>CAPÍTULO 6. La tecnología J2ME</i>	<i>75</i>
6.1. Introducción.....	75
6.2. Objetivos	77
6.3. Arquitectura J2ME	77
6.3.1. Configuraciones	78
6.3.2. Perfiles.....	82
6.3.3. Maquinas virtuales Java	85
6.3.4. Estandarización y compatibilidad de componentes	88
<i>CAPÍTULO 7. El análisis</i>	<i>90</i>
7.1. Jini, tecnología de integración de sistemas distribuidos	90
7.2. Jini y J2ME	95
7.3. La tecnología de comunicación móvil.....	98
7.4. Arquitectura del Sistema	99
7.4.1. Elementos simulados del sistema	99
7.4.2. Mapeo de la arquitectura del sistema	102
7.5. Componentes de software existentes	106
7.5.1. Dispositivo Móvil J2ME	106
7.5.2. Java.....	111
7.5.3. Jini	113
7.5.4. Ambiente de Desarrollo	114
7.6. Construcción de un sistema basado en aplicaciones Jini-J2ME	115
<i>CAPÍTULO 8. El diseño del sistema</i>	<i>123</i>
8.1. Arquitectura del Sistema	123
8.2. Componentes de Software utilizados.....	125
8.2.1. Subsistema Subscriptor	125
8.2.2. Subsistema Infraestructura	128
8.2.3. Subsistema Proveedor de Servicios.....	128
8.2.4. Distribución de los componentes de software.....	128

8.2.5. Localización física de los subsistemas	130
8.2.6. Pila de protocolos utilizadas para la comunicación entre subsistemas	130
8.3. Diseño de los servicios Jini.....	131
8.3.1. Servicio de consulta a la base de datos	132
8.3.2. Servicio de cálculos matemáticos	134
8.3.3. Servicio de transferencia de una imagen.....	136
8.3.4. Funciones del Surrogate host	138
8.3.5. Aplicación MIDlet.....	143
8.4. Diseño de las pruebas a realizar.....	152
8.5. Problemas encontrados en el acoplamiento del sistema	153
 <i>CAPÍTULO 9. Resultados del proyecto</i>	 <i>156</i>
9.1. Resultados de las pruebas del sistema	156
9.2. Resultados del proyecto de tesis.....	158
9.3. Conclusiones	160
9.4. Trabajos futuros.....	161
 <i>GLOSARIO.....</i>	 <i>162</i>
 <i>BIBLIOGRAFÍA.....</i>	 <i>177</i>
 <i>CONTENIDO DEL CD-ROM.....</i>	 <i>186</i>

INTRODUCCIÓN

En años recientes, el número de dispositivos móviles en el mercado ha crecido considerablemente, a razón del uso y demanda por parte de los usuarios, lo que ha incrementado las posibilidades de que los usuarios puedan mantenerse comunicados con diferentes tipos de fuentes de información. El factor principal en el uso y demanda de los dispositivos móviles radica en la comodidad de que una persona lleve consigo un dispositivo que pueda establecer un proceso de comunicación con otros usuarios, así como almacenar y acceder a información requerida por el usuario. Por lo que actualmente, los dispositivos móviles como los teléfonos celulares, computadoras de bolsillo o los asistentes digitales personales (PDA), han incrementado su capacidad de procesamiento y almacenamiento de información, así como la potencialidad para transmitir y recibir dicha información. De manera general, el usuario requiere que el dispositivo móvil pueda conectarse a una red, con el objetivo de acceder, interactuar y utilizar los servicios provistos por otros elementos que conforman la red. Es ahí, donde surge la necesidad de solucionar los problemas que implican la heterogeneidad del conjunto de dispositivos que conforman la red, como lo es la interacción entre dispositivos que no cuentan con los mismos recursos de procesamiento.

El concepto de sistema distribuido surge a partir de la necesidad de tener redes dinámicas, lo que significa que ahora las redes no se limitarán a ser sistemas diseñados para proveer servicios a usuarios geográficamente dispersos, sino que aunado a esto, serán redes que permitan el crecimiento o decrecimiento del sistema según se requiera, en cuanto a recursos y servicios provistos en la red, sin afectar el rendimiento del sistema. Por otro lado, es fundamental tener presente que en un sistema distribuido: la carga de procesamiento y los recursos del sistema, se distribuyen entre los componentes que lo integran. Lo importante de este concepto es que se lleve a cabo de una forma transparente tanto para el administrador como para el usuario. Hoy en día, la utilización de sistemas distribuidos es muy común en áreas: académicas, científicas y empresariales. Por lo que es importante considerar la apertura de los sistemas distribuidos hacia nuevos elementos tecnológicos como lo son los dispositivos móviles, lo que beneficiaría en mayor grado la disponibilidad de los servicios provistos por el sistema distribuido y eliminar la dependencia del uso de una computadora para poder acceder a estos servicios.

El problema que se plantea en este proyecto de tesis es resolver el acceso a través de un dispositivo móvil (teléfono celular) a los servicios provistos por un sistema distribuido con base en el uso de tecnologías emergentes, este problema es orientado hacia los desarrolladores de software de aplicación, con el objetivo de proveer de guías y herramientas que ayuden a entender estas tecnologías, para poder emplearlas como estrategia en el acceso e intercambio de información electrónica.

Las tecnologías emergentes a las que se refiere el anterior párrafo son: Jini y J2ME (Java 2 Micro Edition) desarrolladas por SUN Microsystems y basadas en la plataforma Java. Construir redes que puedan adaptarse a las necesidades de un ambiente de computo dinámico, como lo son los sistemas distribuidos, requieren de una arquitectura innovadora que pueda ser efectiva y eficiente en el manejo y complejidad del sistema. Al mismo tiempo, esa tecnología debe ser fácil de aprender, usar y desarrollar. La tecnología de red Jini esta diseñada para cumplir con estos requerimientos. Por otro lado, la tecnología J2ME se ha enfocado en los dispositivos móviles con un cierto grado de procesamiento, tales como teléfonos celulares, computadoras de bolsillo y PDAs. El objetivo primordial de ésta tecnología, es por una parte dotar de un mayor grado de interconectividad al dispositivos móvil y por otro lado, dar la oportunidad de desarrollar software para éstos dispositivos, dejando a un lado el concepto de software estático (software predeterminado por el fabricante).

En este documento se han analizado de manera general: la situación actual en el mundo y en México respecto a las tecnologías móviles; los dispositivos móviles que soportan la tecnología J2ME, en específico los teléfonos celulares 2.5G; los sistemas distribuidos y la tecnología Jini. Se tomará como base el anterior análisis para resolver el problema del acceso a los sistemas distribuidos por parte de un dispositivo móvil que soporte la tecnología J2ME, utilizando propiamente las tecnologías J2ME y Jini. Es en esta última parte donde surge la necesidad de simular la arquitectura de comunicación de los dispositivos móviles que soportan la tecnología J2ME para poder llevar acabo la integración de las tecnologías Jini y J2ME, en este caso la arquitectura elegida fue GPRS.

Una vez propuesta la arquitectura simulada (dividida en subsistemas y componentes), se integrará en conjunto con una arquitectura Jini para posteriormente analizar el comportamiento del sistema. Las herramientas de software utilizadas para el desarrollo del sistema son evaluadas respecto a sus características de rendimiento, portabilidad, funcionalidad y disponibilidad. Después de que la arquitectura del sistema es implementada, así como los servicios correspondientes, se realizan pruebas para analizar el rendimiento y la interacción de los elementos involucrados en dicha arquitectura. La arquitectura empleada, así como el sistema desarrollado, son de suma importancia, ya que demostrarán la integración de las tecnologías Jini y J2ME, teniendo como resultado el poder ser tomados como base para desarrollar sistemas de mayor complejidad y amplitud, ya sea por agregar nuevos componentes que cumplan con las especificaciones hechas en este proyecto, o por agregar nuevos servicios dentro del sistema Jini. Por lo tanto, la arquitectura empleada como el sistema desarrollado son las aportaciones principales de este proyecto, ya que establecen guías y herramientas (especificadas en este documento) para la implementación de sistemas similares.

CAPÍTULO 1. El Problema

1.1. La problemática actual

Hoy en día, la mayoría de las empresas cuentan con redes computacionales de área local, contenidas dentro de sus instalaciones, en donde se almacena la información y los servicios necesarios para mantener un grado de operatividad dentro de la misma empresa.

Existen frecuentemente problemas comunes en este escenario, ejemplo de ellos se enuncian los siguientes:

- La necesidad del intercambio y acceso a la información, que en los últimos cinco años ha sido realizada a través de computadoras, requiere que los sistemas de computo provean de mecanismos necesarios que aseguren el acceso y consulta de la información de manera eficiente. Aunado a esto, la globalización económica ha provocado que las empresas tengan que compartir información almacenada en medios electrónicos, en otras palabras, ahora la información que en su mayoría esta siendo almacenada en computadoras debe poder compartirse con otras computadoras geográficamente dispersas, pudiéndose convertir así en sistemas distribuidos, lo que hace complicada la administración del propio sistema.
- El surgimiento de los sistemas distribuidos para establecer redes dinámicas, en donde se requiere agregar un nuevo dispositivo y/o un nuevo software al sistema para proveer de algún tipo de servicio a la red, en la mayoría de las veces provocan problemas como la dependencia de la instalación de los controladores de los dispositivos y la dependencia del software, en algunos casos, hacia un sistema operativo. Tomando en cuenta que la instalación de controladores y/o software de aplicación, en la mayoría de los casos, requiere una reinicialización del sistema, ocasionando que el tiempo de operatividad y disponibilidad del sistema se vea afectado.
- Cuando un empleado debe ausentarse por un tiempo determinado de su oficina, para resolver negocios o tareas concernientes a su trabajo, muy posiblemente no pueda acceder a la información y los servicios con los que contaba dentro de las instalaciones de su empresa. El problema fundamental, en este caso, es que el empleado debe contar con algún medio para acceder a la información de la empresa.

La solución más común para éste último punto, es incurrir en gastos para mantener un cierto grado de comunicación entre el empleado y su propia empresa, al tener que comprar (en algunos casos un medio electrónico) una computadora portátil y almacenar ahí toda la información que sea necesaria, haciendo un tanto ineficiente su trabajo, ya que si es indispensable contar con la información actualizada en ese momento, el empleado no podría contar con ella a menos que cuente con algún medio por el cual pueda realizar la transferencia de datos.

Frecuentemente, la manera más fácil en la que una persona puede mantenerse en permanente comunicación con otra, se realiza a través de un teléfono celular (respecto al tipo de información que permite recibir y enviar: voz y datos). Tan sólo en México, se ha incrementado considerablemente el número de usuarios de telefonía celular, como se muestra en la siguiente figura:

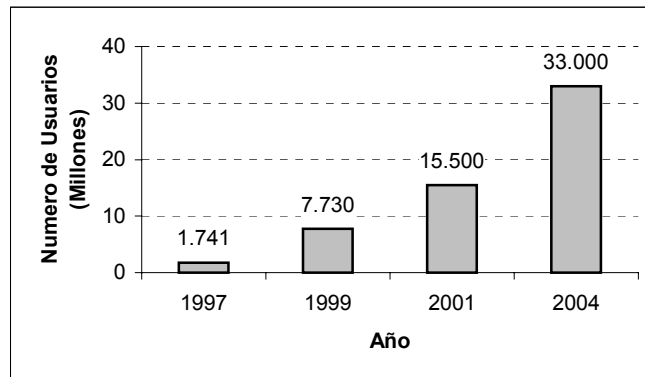


Figura 1. Comportamiento del numero de usuarios de telefonía celular en México [e_CCCM].

El principal factor de que el uso de un teléfono celular sea tan común es el bajo costo (respecto al momento de aparición del dispositivo) en comparación a una PDA, una computadora de bolsillo o una computadora portátil con modem inalámbrico integrado. En otras palabras, un teléfono celular con la última tecnología en su tipo, que apareció en el mercado en el año 2002, tiene un costo menor que una PDA del mismo año, no tomando en cuenta dispositivos telefónicos celulares que tienen integrada una PDA. A pesar de que la telefonía celular fue concebida estrictamente para la voz, actualmente es capaz de brindar otro tipo de servicios como los ofrecidos por las computadoras personales – con ciertas limitaciones –, como lo son: la transmisión y recepción de archivos de texto, audio y video. Las limitaciones de los servicios ofrecidos por un teléfono celular, se deben a que los celulares no tienen los recursos de procesamiento necesarios para ejecutar aplicaciones, que comúnmente se llevan a cabo en una computadora personal.

Un problema fundamental, que hasta hace poco tiempo tenía la tecnología móvil – una vez teniendo el hardware adecuado – era la falta de un software estándar, que permitiera la interconectividad entre dos dispositivos móviles o entre un dispositivo móvil y una computadora personal. La existencia de tecnologías emergentes basadas en Java, ha traído consigo una estandarización de software así como nuevas formas de establecer el acceso e intercambio de información y como consecuencia el desarrollo de una nueva gama de servicios, tal es el caso de Jini (disponible desde el año 1999) y J2ME (disponible desde el año 2000) que posteriormente se detallarán en este documento

El desarrollo tecnológico en las redes de comunicación de datos, tanto móvil como fija, ha provocado la desaparición de la frontera entre los servicios móviles y fijos. Las infraestructuras, definidas por estas tecnologías, sirven como interfaz entre las redes de comunicación fijas y móviles, esto se puede ver con el establecimiento de los servicios de comunicación personales (PCS) que datan del año 1990, en los cuales se incluyen sistemas de comunicación móviles de datos como CSD, HSCSD y GPRS entre otros [EGWC].

Un sistema de comunicación móvil puede definirse cualquier enlace de radiocomunicación entre dos terminales, en localizaciones indeterminadas, de los cuales al menos un elemento está en movimiento, pudiendo el otro ser un terminal fijo. Con infraestructuras de comunicación móvil como GPRS, que le permiten a un dispositivo acceder a redes informáticas de computadoras como Internet, contemplan un nuevo horizonte: “El acceso por parte de los teléfonos celulares a diferentes tipos de servicios a las redes computacionales y su integración a este tipo de redes”.

Como resultado de la aparición de los sistemas de comunicación móviles actuales, la diferencia entre los servicios fijos (los cuales no fueron diseñados para ser accedidos a través de un dispositivo móvil) y los servicios móviles (los cuales fueron diseñados e implementados para ser accedidos a través de un dispositivo móvil), no podrán ser diferenciados por un usuario común, en el momento de ser accedidos. Sin embargo, existe el soporte de una infraestructura que ayuda al dispositivo a poder acceder a los servicios fijos. Es común que un dispositivo móvil pueda acceder a ambos servicios sin que el usuario pueda tener necesariamente la percepción de lo que ocurre, por citar un ejemplo, se pueden acceder indistintamente a sitios HTML o sitios WML, a través de un gateway WAP.

1.2. Servicios móviles a nivel mundial

La constante evolución de las redes móviles o inalámbricas ha provocado el surgimiento de nuevas tecnologías que dan soporte al intercambio y acceso de información y servicios multimedia, a través de Internet. Las redes móviles han incrementado el alcance de su cobertura, de ser regional, para convertirse en redes con cobertura mundial, lo que le facilita al usuario mantener el mismo dispositivo en casi cualquier lugar a donde se dirija sin la necesidad de cambiarlo, manteniendo así la comunicación con otros usuarios, a través del mismo número identificador asignado por su proveedor de servicios original, por lo que la infraestructura de cada proveedor, debe ser lo suficientemente robusta para poder soportar toda una gama de servicios y el alcance de la misma.

Hace por lo menos diez años que los servicios básicos con los que contaba un teléfono celular, dependiendo de su proveedor de servicios, eran [e_TELC]:

- Mensajes escritos de dos vías (entre dos celulares).
- Identificador de llamadas.
- Transferencia de llamada.
- Conferencia tripartita.
- Agenda telefónica.

Actualmente, se han agregado nuevos servicios, muchos de los cuales se acceden a través de Internet, en general podemos enumerar los siguientes [e_TELC] [EGWC]:

- Acceso a funciones determinadas por el dispositivo, a través de la voz.
- Enviar y recibir mensajes multimedia.
- Navegación en Internet.
- Consultar una cuenta de correo electrónico.
- Enviar y recibir correo electrónico, en el mismo dispositivo.
- Instalar aplicaciones Java (J2ME).
- Realizar compras electrónicas (e-business).
- Videoconferencia.
- Reproducción de archivos de audio y video.

Todos los servicios anteriormente mencionados, son proporcionados en mayor parte en Europa, Asia y Estados Unidos de América, a consecuencia de la demanda del mercado, la inversión de los proveedores de servicio y el avance tecnológico desarrollado en esos países, ya que en estas áreas del mundo es donde se concentran el mayor numero de usuarios de telefonía celular como lo muestra la siguiente gráfica:

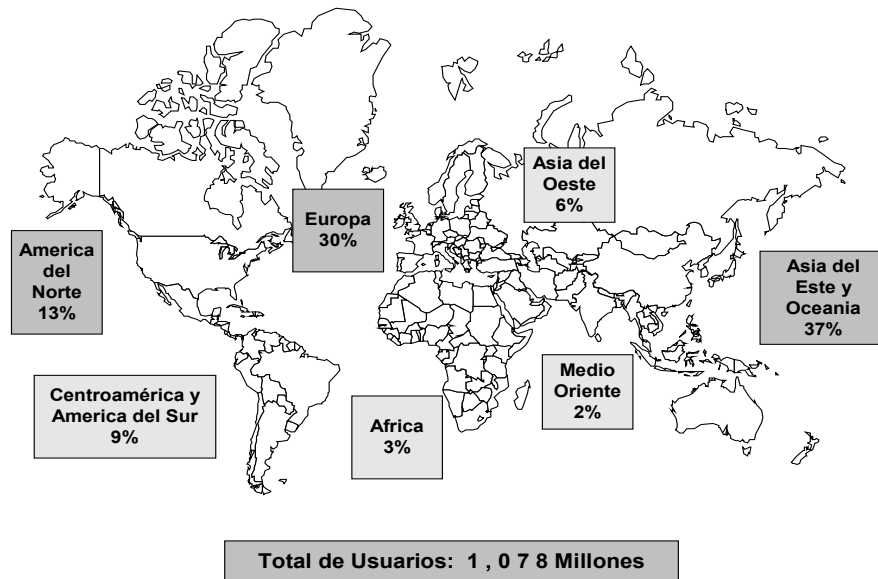


Figura 2. Distribución mundial de usuarios de telefonía celular (Septiembre 2002) [e_3GST].

La figura 2, muestra que los mercados de América del norte (especialmente en Estados Unidos de América), la Comunidad Europea, y Asia del este en conjunto con Oceanía, tienen la mayor concentración de usuarios, por lo que los proveedores de servicios deben contar con una infraestructura vanguardista que satisfaga las necesidades de los usuarios.

1.3. Servicios móviles en México

El rezago de tecnología móvil en México es un punto crítico en el desarrollo y experimentación de los servicios móviles, ya que no es posible competir con países en los cuales se cuenta con la materia prima y el material humano inmerso en este ámbito que cuenta con la experiencia necesaria para desarrollar servicios móviles de última tecnología.

En el presente, México no cuenta ni con un mercado que explote los servicios móviles de última tecnología, mencionados anteriormente, ni con la suficiente oferta de servicios que permita que el costo de acceso se abarate. Aunque en México, aproximadamente se encuentra el 1.86% de todos los usuarios de telefonía celular en el mundo, es importante considerar el dar a conocer la importancia y la disponibilidad de las tecnologías móviles que puedan ser utilizadas para poder desarrollar servicios móviles en beneficio de los propios usuarios.

1.4. Principales componentes de los servicios móviles

Para poder entender de mejor forma la construcción de los servicios móviles, se deben analizar los componentes que la conforman [e_SMOV], como pueden ser:

- Los fabricantes de los equipos de infraestructura de red, dispositivos móviles y plataformas externas.
- Los operadores de redes móviles, los cuales son los que proveen el soporte para el acceso a los servicios móviles y servicios fijos.
- Los proveedores de servicios móviles. En algunos mercados también se encargan de vender el tiempo aire y el uso de la infraestructura para distintos operadores de red. Algunos ofrecen también, servicios de información y servicios periféricos (interactivos) como parte de los paquetes ofertados.
- Los proveedores de servicios móviles de valor agregado. Tratan con los operadores de red y los proveedores de servicios para ofrecerles servicios suplementarios como información de tráfico, noticias o mensajería.
- Los usuarios, como destinatarios finales del servicio.

En la siguiente figura, se muestra la interacción entre los diferentes componentes:

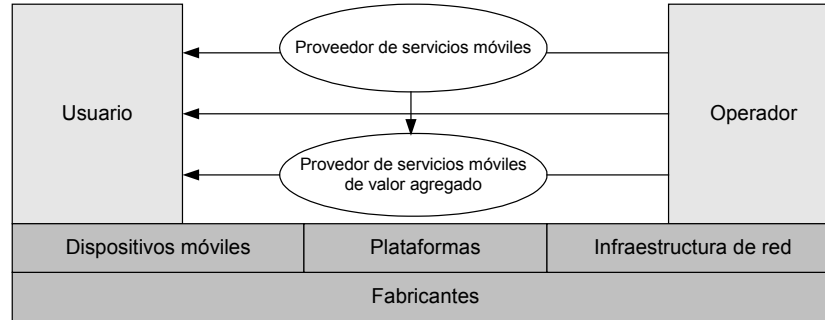


Figura 3. Marco de interacción entre los componentes de servicios móviles.

Se pueden identificar distintos factores que intervienen en el comportamiento del mercado de los servicios móviles:

En el lado de la demanda:

- La movilidad de los usuarios, que provoca una mayor demanda de comunicaciones móviles corporativas.
- La necesidad de acceso a servicios de información.

En el lado de la oferta:

- El desarrollo tecnológico.
- La competencia por la provisión de servicios.
- La diferenciación de los servicios de valor agregado.
- La competencia entre los fabricantes de dispositivos móviles.

Los factores que pueden actuar como inhibidores del desarrollo de servicios móviles son:

- El alto precio de las comunicaciones móviles comparado con la telefonía fija.
- El bajo interés mostrado, hasta el momento, por los proveedores de información.
- La utilidad percibida por el usuario que es todavía escasa.
- La falta de un sistema confiable de tarificación del uso de la infraestructura.

- El capital humano que tenga el conocimiento y experiencia necesarios para desarrollar los servicios móviles.

Sin embargo, es posible que conforme el operador y el proveedor de los servicios móviles vayan madurando, se pueden obtener altos beneficios respecto al costo del acceso a los servicios móviles.

No puede dejarse a un lado, que la forma en la cual se lleva a cabo la interacción entre cada uno de los componentes, esta dado con base en protocolos, lenguajes de programación y la(s) tecnología(s) de soporte.

Las principales tecnologías que actualmente se encuentran en el mercado son:

- CDMA (Code Multiple Division Access).
- TDMA (Time Multiple Division Access).
- GSM (Global System for Mobile Communications) tecnología para dispositivos telefónicos celulares de 2da generación.
- GPRS (GSM Packet Radiates Service) tecnología para dispositivos telefónicos celulares de la generación 2.5.
- EDGE (Enhanced Data GSM Environment) tecnología para dispositivos telefónicos celulares de 3era generación.
- UMTS (Universal Mobile Telecommunications System) tecnología para dispositivos telefónicos celulares de 3era generación.

La posibilidad de que los desarrolladores de servicios móviles conozcan y prueben cada una de estas tecnologías, está en dependencia de la existencia y accesibilidad de las herramientas de hardware y software, así como la propia infraestructura de comunicación, o en su defecto de simuladores que permiten analizar el comportamiento del servicio que se ha construido.

Un problema fundamental para los desarrolladores mexicanos de software con bajo presupuesto, es conseguir las herramientas y adecuarse a las que puedan conseguir – inherentemente en idioma inglés – y adaptarse a la falta de una infraestructura física, en el caso de interesarse en una nueva tecnología móvil con la que no cuente el país.

CAPÍTULO 2. Especificación del Proyecto

2.1. Planteamiento del problema

El problema abordado por este proyecto de tesis, es la integración de un dispositivo móvil a un sistema distribuido utilizando las tecnologías emergentes como lo son: Jini y J2ME. En otras palabras, utilizar dispositivos móviles que soporten la tecnología J2ME para acceder a servicios provistos en un sistema distribuido implementado con base en la tecnología Jini. Un punto importante a tratar, es el acceso a través de un dispositivo móvil, en concreto por un teléfono celular con tecnología J2ME, a los servicios provistos dentro del sistema Jini.

El planteamiento fundamental es implementar una red de área local vista como un sistema distribuido construido con base en la tecnología Jini, en donde se podrán encontrar proveedores de servicios, así como desarrollar las aplicaciones correspondientes en cada tecnología y mostrar la interacción entre los dispositivos móviles que soportan la tecnología J2ME y los proveedores de servicios Jini.

Se debe prever que el rendimiento del sistema no se vea afectado por la instalación del software de soporte para los servicios y la reiniciación del dispositivo asociado, además también se debe prever que los servicios provistos por el sistema cuenten con un cierto grado de portabilidad, esto es que puedan ser implementados en varias plataformas de cómputo, por lo que es importante la búsqueda de herramientas de software gratuitas que puedan proveer la funcionalidad requerida.

2.2. Objetivos

Los objetivos principales del proyecto de tesis son los siguientes:

- Analizar la tecnología Jini, para identificar y conocer el funcionamiento de la integración de los sistemas distribuidos con base en dicha tecnología.
- Identificar las generaciones de dispositivos móviles con la finalidad de conocer las características de los dispositivos que soportan la tecnología J2ME, así como identificar las tecnologías de comunicación móvil que soportan a dichos dispositivos.

- Analizar la tecnología J2ME, para identificar y conocer su funcionamiento en los dispositivos que soportan dicha tecnología.
- Encontrar el grado de factibilidad de interacción entre la tecnología J2ME y la tecnología Jini, con base en las arquitecturas que las soportan.
- Construir un sistema que pueda demostrar la interacción de las tecnologías Jini y J2ME, a través de la identificación, obtención y provisión de los componentes necesarios para su desarrollo, utilizando para su implementación herramientas de software gratuitas que tengan características similares de las herramientas no gratuitas (como es el caso de CodeWarrior [e_CWMT]), con el objetivo de que los estudiantes en el área computacional y todas las afines, puedan tomar como punto de referencia el sistema construido para trabajar en el ámbito de los sistemas distribuidos, la construcción y acceso de servicios en un ambiente distribuido, las redes móviles y los dispositivos móviles, con un bajo presupuesto.
- En el ámbito personal, adquirir experiencia en el campo de las tecnologías de comunicaciones móviles y los sistemas distribuidos.

2.3. Justificación

- En la actualidad, en México, actualmente el material teórico o práctico (software de aplicación) existente es limitado respecto al tema que se presenta y se desarrolla en este proyecto de tesis.
- Utilizar tecnologías emergentes en el desarrollo de aplicaciones para dispositivos móviles (J2ME), así como en el desarrollo de aplicaciones distribuidas (Jini), para el acceso por parte de un dispositivo móvil a los servicios fijos provistos en un sistema distribuido.
- Teniendo en cuenta las tecnologías existentes en comunicaciones móviles de datos como GPRS y las tecnologías emergentes Jini y J2ME, es sumamente importante darlas a conocer a través de este proyecto de tesis, a los estudiantes y profesionales en el área, con el objeto de que puedan introducirse más en el desarrollo de aplicaciones móviles para acceder a servicios provistos en un sistema distribuido, pudiendo aprovechar la infraestructura tecnología en comunicaciones móviles que actualmente existe en México (GPRS), para una futura implementación real.

- Este proyecto de tesis, es una opción para estudiantes o profesionales en el área, que no cuentan con presupuesto o con acceso a una infraestructura real, basándose en la referencia de uso de herramientas de software gratuitas, para diseñar y probar un sistema, en el que un dispositivo móvil pueda acceder a los servicios provistos en un sistema distribuido.

2.4. Hipótesis

El proyecto de tesis tiene como hipótesis los siguientes puntos:

- Con base en el problema planteado en la sección 2.1, la utilización de las tecnologías: Jini y J2ME pueden ayudar a resolver el problema del acceso por parte de un dispositivo móvil a servicios provistos en un sistema distribuido, ya que Jini define una forma de integración de sistemas distribuidos, con base en los servicios proporcionados por sus elementos, lo que facilitaría la implementación de un sistema distribuido y en el caso de J2ME, esta tecnología dota de un cierto grado de interconectividad al dispositivo móvil – en este caso, un teléfono celular – para poder interactuar con otros dispositivos de la misma especie o con otros dispositivos computacionales que cuenten con una plataforma Java. Ya que ambas tecnologías tanto Jini como J2ME, están basadas en Java y ésta última es una plataforma base común en el desarrollo de software de aplicación para múltiples plataformas de computo, se puede inferir que ambas tecnologías pueden interactuar con base en el modelo de programación de Java.
- Existe la posibilidad de simular y/o emular alguna parte de la infraestructura física al trabajar en un ambiente de red, ya que el costo de uso de una infraestructura real, como es la infraestructura de comunicación móvil, puede ser elevado y también existe la posibilidad de que no se cuente con algún elemento o conjunto de elementos físicos, lo que sería un obstáculo en la implementación y pruebas del software desarrollado en este proyecto.
- Cualquier estudiante en el ámbito del área computacional, podrá utilizar la metodología y arquitectura descrita en este proyecto de tesis, como base para poder construir sistemas similares al propuesto.

2.5. Alcances y limitaciones

- Los dispositivos móviles con los cuales se llevará a cabo el proyecto son los teléfonos celulares que soporten la tecnología J2ME, los cuales son contemplados por ésta tecnología en la configuración CLDC y el perfil MIDP, que posteriormente serán definidos en este documento. Ya que el disponer de un teléfono celular de manera real, implica un alto costo en el desarrollo del proyecto, se ha optado por un emulador que lo sustituya sin que esto implique la pérdida de funcionalidad respecto del dispositivo real.
- No se pretende hacer un estudio exhaustivo respecto al marco teórico (sistemas distribuidos, tecnologías Jini y J2ME, entre otros temas) que soporta a este proyecto sino tomar los conceptos necesarios de cada tema que ayuden al entendimiento y desarrollo del propio proyecto de tesis.
- El software desarrollado, no tratará de ser una herramienta robusta, sino que será una forma de presentar y demostrar el acceso a servicios fijos provistos en una red Jini, a través de un solo dispositivo móvil (teléfono celular), no se contempla la inclusión de más dispositivos.

2.6. Metodología

Uno de los puntos importantes en el proyecto, es elegir un modelo de ciclo de vida con el objetivo de definir y agrupar las actividades del proyecto y aunque es necesario puntualizar que, de manera estricta, este proyecto no es un proyecto de desarrollo de software, es necesario seguir un modelo de desarrollo para el proyecto, que permita definir fases en las que se identifiquen actividades, con la finalidad de poder subdividir el trabajo por tiempo y en donde se pueda identificar: las entradas, el proceso y las salidas de cada fase, para poder llevar un mejor control del proyecto.

Para éste proyecto se propuso utilizar el modelo de ciclo de vida compuesto por un modelo de ciclo de vida de cascada simple en conjunto con un modelo en V.

El modelo de ciclo de vida en cascada simple, establece que en cada una de las actividades, se definen entradas y se generan salidas. El modelo de ciclo de vida en V, permite definir fases paralelas a otras, por lo que de esta manera, se pueden realizar dos fases al mismo tiempo, por lo que se lleva a cabo una sinergia entre fases. Este último modelo es utilizado con el objetivo de adecuar las pruebas del software de acuerdo a las especificaciones establecidas.

La siguiente figura muestra de manera gráfica el modelo descrito:

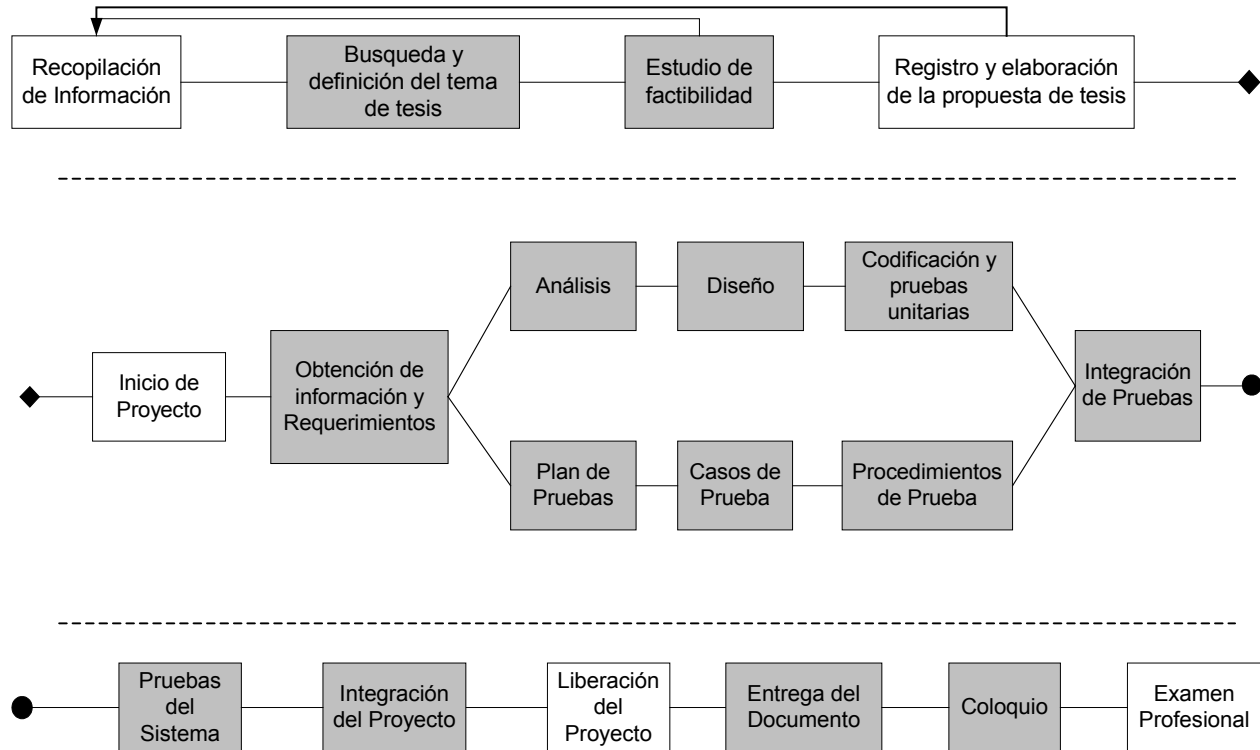


Figura 4. Modelo de ciclo de vida utilizado en el proyecto.

Ahora se procede a detallar las fases establecidas en el ciclo de vida propuesto:

- **Recopilación de información:** en esta fase, se recopiló y se analizó, la información relativa a los temas de soporte de este proyecto de tesis, para tener conocimiento de los posibles trabajos y aplicaciones en los cuales pueda incursionar.
- **Búsqueda y definición de tema de tesis:** en ésta fase se determinó un tema de interés específico para desarrollarlo como proyecto de tesis.
- **Estudio de factibilidad:** una vez que se conoció específicamente el tema en el que se trabajaría, fue necesario determinar si su realización era factible tomando en cuenta cuestiones económicas, tecnológicas y de tiempo. En esta fase se realizó el anteproyecto de tesis.

- **Registro y propuesta de tesis:** en esta fase el tesista registró oficialmente la propuesta del proyecto de tesis ante el Departamento de Ingeniería Industrial y Tecnologías de la Información UPAEP (DIITI), de acuerdo a los estatutos de dicho departamento.
- **Inicio del proyecto:** Una vez que la propuesta del proyecto de tesis fue aceptada, se definieron las actividades correspondientes a las fases subsecuentes de la metodología, así como el posible asesor externo para poder resolver las posibles dudas en el desarrollo del proyecto (en éste caso en particular de no contar con un asesor formal) con el fin de pactar la manera en la que se llevarían a cabo las asesorías.

En esta fase se planearon: la administración de tiempo y recursos dedicados al proyecto, tales como horas a la semana para trabajar, para acudir a asesoría, espacio de trabajo, etc; y la documentación y artefactos de software que se deberían de realizar como los son: [a_PTP] [a_PAP] [a_PCAP] [a_PACS] [a_ERS] [a_PPS].

- **Obtención de información y Requerimientos:** en esta fase se definió y clasificó la información recopilada, necesaria y suficiente para desarrollar el trabajo en lo sucesivo, así como los requerimientos para la realización del proyecto de tesis. La información que se recopiló, se refiere al ámbito de:
 - Sistemas distribuidos.
 - Tecnología Jini.
 - Tecnología J2ME.
 - Tecnologías de comunicación móvil.
 - Referencias de trabajos con Jini y J2ME (véase bibliografía).
- **Análisis:** en esta fase, se analizó cada uno de los requerimientos con la finalidad de obtener sus correspondientes especificaciones, así como determinar los requerimientos funcionales y no funcionales. También se determinó el tipo de herramientas y métodos a utilizar para la fase de diseño.

- **Plan de Pruebas:** esta fase se realizó en paralelo con el análisis, determinando:
 - Los ítems a ser probados.
 - Las características a evaluar.
 - Las características que no se evaluarán.
 - Entregables de las pruebas.
 - Necesidades de ambiente.
 - Responsabilidades.
 - Recursos, riesgos y contingencias.

La especificación del tipo de pruebas que serán realizadas al software son especificadas en esta fase.

- **Diseño:** el objetivo de ésta fase fue plantear la forma en la que serian implementados: la arquitectura, los servicios, la aplicación Jini y la aplicación J2ME para poder conformar el sistema.
- **Casos de Prueba:** esta fase se realizó en paralelo con la fase de diseño, para elegir los casos a probar para cada función del sistema, de tal manera que se pudiera evaluar el desempeño del mismo.
- **Codificación y pruebas unitarias:** en esta fase, se codificó la aplicación Jini y la aplicación J2ME y se llevo a cabo la integración del sistema. En el momento de codificar se realizaron las pruebas unitarias con la finalidad de asegurar que la sintaxis sea la correcta. También se probó el funcionamiento el sistema de con base en los componentes de la arquitectura y el software desarrollado.
- **Procedimientos de prueba:** esta fase se realizó en paralelo con la fase de codificación y pruebas unitarias, su objetivo fue definir el conjunto de instancias y operaciones a utilizar en los casos de pruebas para aplicarlos al sistema, así como de diseñar el formato de los procedimientos de prueba al sistema.
- **Integración de pruebas:** en esta fase, se agruparon y clasificaron todos los procedimientos de prueba, para su posterior ejecución en el sistema o subsistemas generados.
- **Pruebas del sistema:** en esta fase se aplicaron los procedimientos de pruebas con el fin de evaluar la integración del sistema, reportando los resultados, la funcionalidad del sistema y las correcciones que se llevarian acabo.

- **Integración del proyecto:** el objetivo de esta fase fue la integración de toda la documentación generada y el conjunto de herramientas de desarrollo utilizadas en el proyecto, para la edición de la tesis.
- **Liberación del proyecto:** en esta fase, el tesista determina la finalización del proyecto, con base en los resultados obtenidos en la realización del proyecto.
- **Elaboración del documento de tesis:** en esta fase se procedió a elaborar el documento preliminar del proyecto de tesis, para presentarlo al DIITI.
- **Entrega del documento:** una vez elaborado el documento de tesis, se entregó al DIITI, el cual asignó una fecha para el coloquio previo al examen profesional.
- **Coloquio:** una vez que se entregó el documento de tesis, el DIITI designó a los sinodales que tomaron parte en el examen profesional, a los cuales se les presentó el proyecto de tesis para que lo examinaran y aportaran comentarios y/o correcciones.
- **Examen profesional:** en esta etapa, se expone finalmente ante los sinodales el desarrollo del trabajo de tesis, en el cual se interrogará al tesista sobre aspectos clave del trabajo. Al terminar la presentación de la tesis, habrá un fallo para determinar si es aprobada o rechazada.

Para mayor detalle acerca de la información presentada en la metodología del proyecto pueden ser consultados los documentos: [a_PTP] [a_PAP] [a_PCAP] [a_PACS] [a_ERS] [a_PPS], los cuales pueden ser encontrados en el CD-ROM que acompaña a este documento.

CAPÍTULO 3. Los sistemas distribuidos

3.1. Conceptos básicos y definición

La concepto fundamental de un sistema distribuido es un conjunto de computadoras y de sistemas de transmisión de mensajes, bajo un solo punto de vista lógico a través del cual los elementos de cómputo resuelven tareas en forma cooperativa. Se puede afirmar que, el sistema constituye un ente capaz de procesar información debido a dos características esenciales:

- El sistema consiste de una cierta cantidad de computadoras y dispositivos periféricos, cada uno de las cuales pueden tener sus propios recursos de procesamiento y almacenamiento o pueden estar sujetos a otros elementos.
- Todos los elementos del sistema están adecuadamente interconectados, bajo protocolos determinados.

Desde el punto de vista del usuario, un sistema operativo distribuido se comporta como un sistema operativo convencional que se ejecuta en una computadora local; sin embargo, el sistema operativo administra los recursos de un conjunto de computadoras independientes con cierto grado de heterogeneidad e integra una interfaz común hacia el usuario.

El concepto de sistema distribuido surge a partir de la necesidad de tener redes dinámicas. Esto quiere decir que ahora las redes no se limitarán a ser sistemas diseñados para servir a usuarios geográficamente dispersos, sino que aunado a esto, serán redes que permitan el crecimiento o decrecimiento del sistema según se requiera.

Una definición de un sistema distribuido es:

“Un sistema distribuido es aquél al que sus usuarios ven como un sistema operativo centralizado ordinario; sin embargo, se ejecuta en diferentes e independientes CPUs. El concepto clave aquí es la transparencia; en otras palabras, el uso de diversos procesadores deberá ser invisible (transparente) al usuario. Otra forma de expresar esta misma idea es diciendo que el usuario verá al sistema como un uniprocador virtual y no como una colección de máquinas diferentes” [SOPM].

Existen ciertos factores que han propiciado el auge tan elevado de los sistemas distribuidos dentro del procesamiento de las organizaciones modernas en el mundo, algunos de ellos son:

1. Avances en tecnología de cómputo.
2. Desarrollo de las redes locales de alta velocidad (LAN).
3. Desarrollo de redes mundiales (WAN).

Los sistemas distribuidos plantean importantes ventajas en comparación con los sistemas tradicionales centralizados, como lo son el desarrollo de aplicaciones que puedan operar de forma natural en ambiente disperso tales como: bases de datos, sistemas de trabajo cooperativo y juegos.

Otra ventaja importante la constituye una mayor confiabilidad, pues es factible construir un sistema tolerante a fallas si sus componentes comparten la carga de trabajo y, de presentarse alguna falla, dicha carga es asignada a los demás elementos que siguen operando.

Sin embargo, frente a todas las ventajas de los sistemas distribuidos, existen desventajas importantes que deberán ser resueltas de la mejor manera posible antes de considerar un estado maduro de tales sistemas, en lo que respecta al uso difundido de ellos, dentro de las aplicaciones cotidianas de las organizaciones modernas [SOPM]. Las tres desventajas principales de los sistemas distribuidos son:

- El software: se considera una problemática importante dado que para un sistema distribuido es importante tomar en cuenta que éste debe ser de un tipo y capacidades muy especiales, hecho específicamente para administrar y operar sobre un ambiente disperso y heterogéneo.
- Las redes de interconexión: son relevantes porque a través de ellas fluyen los mensajes y paquetes de comunicación entre los diferentes elementos involucrados en la tarea, y de fallar éstas los procesos asociados pueden dañarse o interrumpirse.
- La seguridad: es un elemento a considerar, dado que si cualquier usuario tiene acceso a través de una imagen lógica a todos los elementos y periféricos que integran un sistema distribuido, también puede leer información o datos de otras personas (por lo menos en forma potencial)

3.2. Funciones de un sistema distribuido.

El desarrollo de un sistema distribuido requiere el uso de las siguientes funciones y servicios [d_SODT]:

- **Servicios de Comunicación.** La intención es tener comunicación entre procesos sobre la red de conexión, para lo cual se pueden utilizar:

a) Modelos de interacción:

- Cliente/servidor (2-niveles, 3-niveles o n-niveles)
- Punto a punto: Equilibrio de roles.
- Intermediarios: Proxy, Dispatcher, Caches, etc.
- Comunicación en grupo (Multicast)

b) Tecnologías de comunicación:

- Paso de mensajes: sockets.
 - Llamada a procedimientos remotos (RPC).
 - Invocación de métodos remotos (RMI).
 - Tecnologías de objetos distribuidos: CORBA, DCOM, etc.
- **Administración y asignación de recursos.** Contempla la asignación de recursos a usuarios, toma de decisión de en donde deberán ser ejecutadas las peticiones, creación e instalación de nuevos recursos de red, soporte de replicación para procesos críticos, mecanismos de control de concurrencia y sincronización.
 - **Administración de nombres.** Mecanismos de asignación y mantenimiento de los nombres de los recursos, localización de servidores / usuarios, mantenimiento de directorios, etc.
 - **Servicios de sincronización:** Comprende los conceptos acerca de:
 - Tiempos de ejecución en entornos distribuidos: Sincronización de relojes.
 - Concurrencia y Paralelismo: Exclusión mutua e interbloqueos.

- Algoritmos distribuidos: Elección de líder, coordinación, etc.
- **Funciones de seguridad.** Especificación de los usuarios y sus derechos de acceso, mecanismos de autenticación, políticas de acceso y contra ataques externos.

Desde el punto de vista del usuario, algunos de los requisitos respecto a la funcionalidad de los sistemas distribuidos son:

- Minimización del costo de acceso a los recursos
- Maximización y simplificación de las facilidades de comunicación con otros usuarios o programas.

3.3. Aspectos de un sistema distribuido

- **Transparencia.** Se trata de lograr la imagen de un único sistema, que cualquier usuario piense que la colección de máquinas es un sistema de tiempo compartido de un solo procesador, en otras palabras, el concepto va ligado a la idea de que todo el sistema funcione de forma similar en todos los puntos de la red, independientemente de la posición del usuario. Que permitan trabajar a los usuarios como si se tratara de un equipo único [SOPM].

Existen varios perfiles de transparencia:

1. **Transparencia en el acceso:** Se refiere a la capacidad de poder obtener recursos de la misma manera, sin importar que estos sean remotos o locales.
2. **Transparencia en localización:** El sistema se ve como una entidad y cualquiera de los recursos existentes deben de poder accederse de igual forma sin importar su localización geográfica, en otras palabras, se accede a los recursos sin conocer su ubicación.
3. **Transparencia en migración:** Permite que existan movimientos en los recursos del sistema sin que esto repercuta en las aplicaciones o afecte a los usuarios.

4. **Transparencia en concurrencia y paralelismo:** Tanto las acciones ejecutadas por los usuarios como aplicaciones deben de poder operar al mismo tiempo con los recursos, sin que el trabajo de uno interfiera con el de el otro. En cuanto al paralelismo, éste puede ocurrir sin que el usuario lo note.
 5. **Transparencia en fallos:** Evita que existan perdidas en las tareas de los usuarios, a pesar de que ocurra alguna falla en el hardware o en el software, la ocurrencia de fallos no afecta a los usuarios.
 6. **Transparencia en crecimiento:** Permite que el sistema pueda incrementar o disminuir su tamaño según se requiera, esto debe poder llevarse a cabo sin necesidad de cambiar la estructura del sistema. Es decir, pueden agregarse o quitarse dispositivos.
 7. **Transparencia en replica:** El sistema es libre de fabricar por su cuenta copias adicionales de los archivos y otros recursos sin que lo note el usuario.
 8. **Transparencia en rendimiento:** El sistema debe ofrecer flexibilidad en cuando a la configuración se refiere; esto quiere decir, que debe de poder ser reconfigurado para incrementar su rendimiento y que esto no afecte ni a las aplicaciones, ni a los usuarios.
- **Fiabilidad** (en disponibilidad como en coherencia). Una de las ventajas claras que nos ofrece la idea de sistema distribuido es que el funcionamiento de todo el sistema no debe estar ligado a ciertos elementos de la red, sino que cualquier equipo pueda suplir a otro en caso de que uno se estropee o falle.
 - **Rendimiento.** En particular cuando se ejecuta una aplicación en un sistema distribuido no debe parecer peor que su ejecución en un único procesador. Se pueden utilizar diversas métricas del desempeño, por ejemplo :
 - Tiempo de respuesta.
 - Rendimiento (cantidad de procesos por unidad de tiempo).
 - Uso del sistema.
 - Cantidad consumida de la capacidad de la red.
 - **Escalabilidad.** Facilidad para incrementar el número de componentes y recursos.

- **Flexibilidad** (facilidad para incorporar cambios y extensiones al sistema). El sistema debe estar abierto a los cambios y a las actualizaciones que mejoren el funcionamiento del sistema. Esta necesidad ha provocado una diferenciación entre las dos diferentes arquitecturas del núcleo del sistema operativo:
 - el núcleo monolítico
 - el micronúcleo.

Las diferencias entre ambos son los servicios que ofrece el núcleo del sistema operativo. Mientras el núcleo monolítico ofrece todas las funciones básicas del sistema, integradas en el mismo núcleo, el micronúcleo incorpora solamente las fundamentales, que incluyen únicamente el control de los procesos y la comunicación entre ellos y la memoria. El resto de servicios se cargan dinámicamente a partir de servidores en el nivel de usuario.

- **Seguridad** (privacidad, autenticación, integridad, etc). Los archivos y otros recursos deben ser protegidos contra el uso no autorizado.

3.4. Taxonomía de los sistemas distribuidos

Dado que un sistema distribuido es una colección de hardware distribuido (usualmente de manera geográfica), que da la impresión de ser un ambiente computacional unificado por software, por lo que es necesario clasificar configuraciones de hardware y los aspectos del software.

3.4.1. Configuraciones Hardware

A través de los años, se han propuesto varios esquemas de clasificación para los sistemas computacionales que poseen varios procesadores. La más citada es la que realizó Flynn (1972). Eligió dos características consideradas por él, como esenciales: el flujo de instrucciones y el flujo de datos [d_SDIN]. La clasificación entonces realizada es la siguiente:

- **SISD** - *Single Instruction Single Data*: se refiere a la ejecución de una instrucción e instruyendo a una unidad de procesamiento para que realice la instrucción.

- **SIMD** - *Single Instruction Multiple Data*: se refiere a la ejecución de una instrucción e instruyendo a varias unidades de procesamiento para que realicen la instrucción en paralelo, cada una a sus propios datos. Estas máquinas son útiles cuando se requiere realizar el mismo cálculo sobre varios conjuntos de datos diferentes.
- **MISD** - *Multiple Instruction Single Data*: se refiere a la ejecución en paralelo de múltiples instrucciones sobre un mismo conjunto de datos.
- **MIMD** - *Multiple Instruction Multiple Data*. se refiere a la ejecución en paralelo de múltiples instrucciones sobre conjuntos de datos diferentes. Todos los sistemas distribuidos pertenecen a ésta última categoría (MIMD), por lo que realiza una subdivisión principalmente en dos grupos: aquellos que tienen memoria compartida, a los que llama **multiprocesadores** y, aquellos que no tienen memoria compartida (memoria distribuida), los que denomina **multicomputadoras**.

Por otra parte, además de distinguir el tipo de procesamiento de los sistemas computacionales, es posible distinguir entre sistemas fuertemente acoplados y débilmente acoplados. El acoplamiento es un concepto abstracto que nos indica el grado de interdependencia entre módulos de procesamiento. En los sistemas fuertemente acoplados, el retraso que se experimenta al enviar un mensaje desde una máquina a la otra es bajo, con una alta tasa de transmisión de datos; en cambio, en los sistemas débilmente acoplados el retraso suele ser mayor y poseen una tasa de transmisión más baja. Los sistemas fuertemente acoplados tienden a utilizarse más como sistemas paralelos, en contraste con los débilmente acoplados que suelen utilizarse como sistemas distribuidos.

3.4.2. Aspectos del Software

Aunque el hardware es importante, la imagen que presenta un sistema computacional a los usuarios queda, en gran medida, determinada por el software. Análogamente a la clasificación hecha sobre el hardware, el software también podría clasificarse como débil o fuertemente acoplado, aún cuando esta división no es clara cuando se trata del software.

El software débilmente acoplado permite que las máquinas y los usuarios de un sistema distribuido sean independientes entre sí en lo fundamental, pero que interactúen en cierto grado, cuando se requiera.

A continuación se analizan las principales combinaciones de hardware y software [SOPM] [d_SODT] [d_SDIN]:

- **Sistemas Operativos de Red** (Software débilmente acoplado, hardware débilmente acoplado). En este modelo, cada usuario tiene una estación de trabajo para su uso exclusivo, con su propio sistema operativo y usualmente todos los trabajos se ejecutan en forma local. Eventualmente, un usuario pudiera requerir ejecutar programas que estén en una estación remota. Para ello el usuario debe realizar una operación de conexión remota mediante algún comando, convirtiendo su estación de trabajo en un terminal de la máquina remota. la transferencia de archivos entre máquinas se realiza en forma explícita.

El sistema operativo a utilizar en este tipo de ambiente debe controlar las estaciones de trabajo individualmente, a los servidores de archivos y también debe encargarse de la comunicación entre ellos. Es posible que todas las máquinas posean el mismo sistema operativo, pero esto no es necesario.

- **Sistemas Realmente Distribuidos** (software fuertemente acoplado, hardware débilmente acoplado). Su objetivo es crear la ilusión de que la red de computadores es un sistema de tiempo compartido, en lugar de una colección de diferentes máquinas.
- **Sistemas de Multiprocesador con memoria compartida** (Software fuertemente acoplado, Hardware fuertemente acoplado) Su característica clave es la existencia de una cola de ejecución contenida en la memoria compartida. En estos sistemas, es esencial que el planificador se ejecute como región crítica, con el fin de evitar que dos procesadores elijan al mismo proceso para su ejecución inmediata.

La siguiente tabla muestra un comparativo de los tres tipos de sistemas analizados.

Descripción	Sistema de Red	Sistema Distribuido	Sistema Multiprocesador
Se ve como un uniprocador virtual	No	Sí	Sí
Todos tiene que ejecutar el mismo SO	No	Sí	Sí
Copias existentes del SO	n	n	1
La comunicación se logra a través de...	Archivos compartidos	Mensajes	Memoria compartida
Se requiere acuerdo en los protocolos de red	Sí	Sí	No
Existe cola de ejecución	No	No	Sí
Existe semántica bien definida para los archivos compartidos	No, por lo general	Sí	Sí

Tabla 1. Comparación de los sistemas respecto al acoplamiento de software y de hardware.

3.5. Implementación de un sistema distribuido

La implementación de un sistema distribuido puede estar basado principalmente en [d_SODT]:

1. Un Sistema Operativo Distribuido.
2. Un Sistema Operativo de Red.
3. Una plataforma de servicios de distribución (*Middleware*).

3.5.1. Sistema distribuido basado en un sistema operativo distribuido

Se comporta como un sistema “único”, que se basa en la distribución tanto de servicios como de aplicaciones, en todas las maquinas que conforman el sistema. Cada maquina ejecuta la misma copia del sistema operativo. Todos los computadores deben

ejecutar el mismo sistema operativo, en este caso, el sistema distribuido hereda las capacidades y características del sistema operativo.

Los sistemas operativos distribuidos abarcan servicios de red, logrando integrar recursos en una sola máquina virtual que el usuario accede en forma transparente. Es decir, ahora el usuario ya no necesita saber la ubicación de los recursos, sino que los conoce por nombre y los usa como si todos ellos fuesen locales a su lugar de trabajo habitual.

La siguiente figura muestra de manera grafica, la arquitectura de estos sistemas.

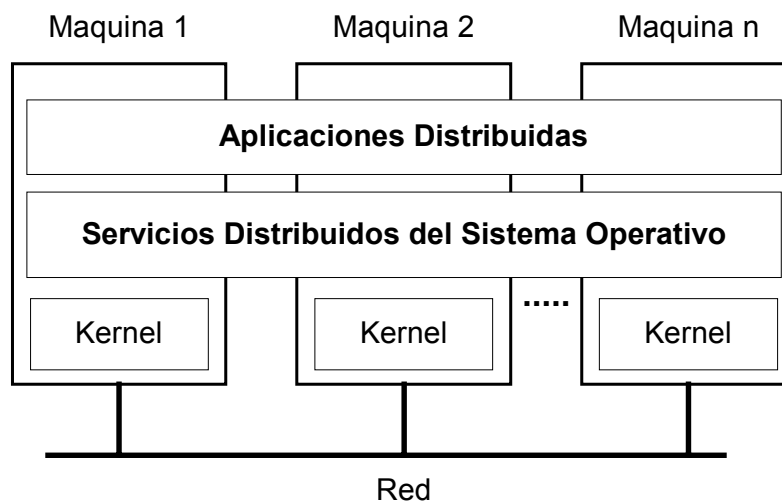


Figura 5. Sistema distribuido basado en un sistema operativo distribuido.

3.5.2. Sistema distribuido basado en un sistema operativo de red

Este sistema, el usuario ve un conjunto de máquinas independientes, en las cuales puede haber solamente aplicaciones distribuidas en todo el sistema, sin embargo no existe de manera estricta el concepto de transparencia, porque debe saber de manera explícita, en que máquina se encuentra la aplicación, de otra manera no puede accederla, también cabe señalar que cada maquina ejecuta una copia del sistema operativo, lo contrario a los sistemas operativos distribuidos.

Es un tanto difícil implementar aplicaciones distribuidas, ya que el sistema operativo en el cual se basan, no cuenta de manera natural con la comunicación por paso de mensajes.

El acceso a los servicios y recursos se lleva a cabo de manera remota. Este tipo de sistemas toman importancia, ya que al poseer como base un sistema operativo de red, nos permite verlo como parte de un sistema abierto, en otras palabras, garantiza la comunicación con otros sistemas similares o con un grado de heterogeneidad, el cual no esta directamente relacionado con el modelo que lo sustenta.

La siguiente figura muestra de manera grafica, la arquitectura de estos sistemas.

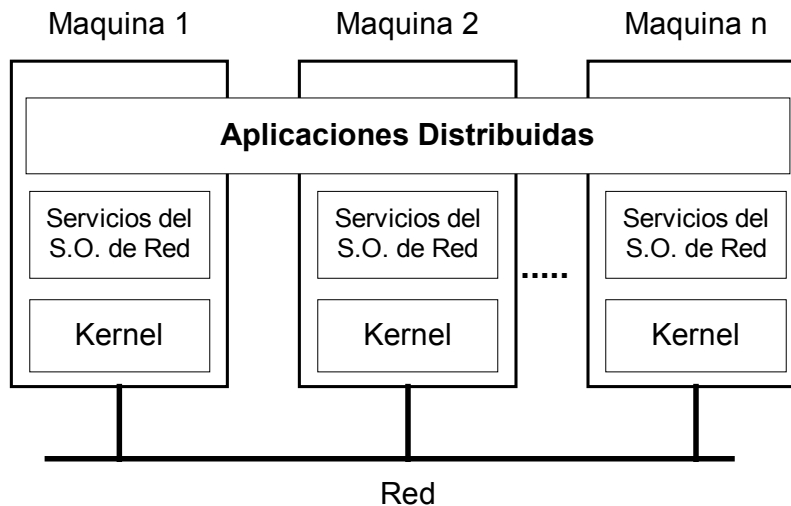


Figura 6. Sistema Distribuido basado en un sistema operativo de Red

3.5.3. Sistema distribuido basado en un middleware

El concepto de *middleware* lo podemos definir como el conjunto de componentes de software que forman una capa entre el sistema operativo y la aplicación facilitando la conexión e interacción entre aplicaciones; estos componentes facilitan el desarrollo, despliegue y gestión de aplicaciones que deben interactuar. La función principal de estos componentes es enmascarar la heterogeneidad de los sistemas distribuidos y proporcionar un conjunto estándar de interfaces y servicios.

El concepto de *middleware* no es un concepto nuevo. Los primeros monitores de teleproceso de los grandes sistemas basados en tecnología cliente/servidor ya se basaban en él, pero es con el nacimiento de las tecnologías basadas en el concepto de sistemas abiertos que el concepto de *middleware* adquiere importancia.

En estos sistemas basados en un *middleware* los servicios y protocolos se encuentran estandarizados (por lo que se puede ver como un sistemas abierto) y

también ofrecen servicios no incluidos en el sistema operativo, además de ser independientes del Hardware y del sistema operativo subyacente.

La siguiente figura muestra de manera grafica, la arquitectura de estos sistemas.

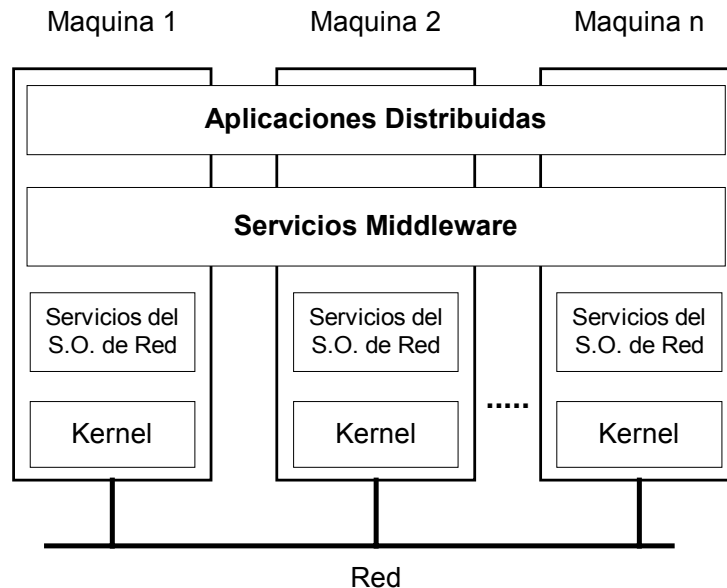


Figura 7. Sistema distribuido basado en un middleware

Las principales categorías del middleware podríamos definir las de la siguiente forma:

- **Monitores de proceso de transacciones distribuidos** (DTPM's Distributed Transaction Processing Monitors). Herederos de la tecnología mainframe, son ampliamente demandados para intercomunicar distintos sistemas en distintos entornos.
- **Llamadas a procedimientos remotos** (RPC's Remote procedure Call) Diseñado como servicios síncronos para permitir gestión remota de redes.
- **Middleware orientado a mensajes** (MOM Messaging Oriented Middleware) Diseñado para servicios de mensajes con tecnología asíncrona.
- **Middleware para tecnologías orientadas a objetos** (ORB Objects Request Broker) Objetos piden servicios de objetos que se encuentran en la red. El estándar más conocido de esta tecnología es CORBA (Common Object Request Broker Architecture).

- **Middleware de acceso a Bases de Datos** (Data Base Access Middleware). Para acceso estándar a bases de datos. Permite desarrollar sistemas independizándolo de la base de datos que lo soporte. En la actualidad representa el 50% del mercado del middleware.

A continuación se presenta, tablas comparativas de las diferentes implementaciones.

Sistema	Descripción	Objetivo
Sistema Operativo Distribuido	Sistema fuertemente acoplado para multiprocesadores y multicomputadoras homogéneas.	Hace transparente el uso y manejo de recursos del sistema.
Sistema Operativo de Red	Sistema débilmente acoplado para multicomputadoras heterogéneas (LAN y WAN)	Ofrece servicios locales a clientes remotos
Middleware	Capa adicional introducida en un sistema operativo de red para implementar servicios de propósito general.	Provee transparencia en la distribución de servicios

Tabla 2. Comparativo de las bases de implementación de los sistemas distribuidos

Elemento	Sistemas distribuidos		Sistema operativo de red	Sistemas basados en middleware
	Multiprocesadores	Multicomputadoras		
Grado de Transparencia	Muy Alto	Alto	Bajo	Alto
Todos los nodos con el mismo sistema operativo	Si	Si	No	No
Numero de copias del sistema operativo	1	N	N	N
Base de comunicación	Memoria compartida	Mensajes	Archivos	Depende del modelo especifico (ORB, RMI,etc)
Manejo de recursos	Global y central	Global y distribuido	Por nodo	Por nodo
Escalabilidad	Ninguna	Moderada	Total	Puede variar
Es un sistema Abierto.	No	No	Si	Si

Tabla 3. Comparativo de los diferentes tipos de sistemas distribuidos.

3.6. Ventajas de los sistemas distribuidos

Por su estructura y sus características los sistemas distribuidos ofrecen grandes ventajas [SOPM], como los son:

- El costo y el rendimiento de los sistemas distribuidos son reducidos, debido a que cada dispositivo del sistema se desenvuelve como una entidad y es programado por separado y no dependiendo de otros dispositivos. No se requieren costosos sistemas de redes que requieran de una configuración exhaustiva.
- Otra ventaja importante es la modularidad, ya que un sistema distribuido deja de ser centralizado respecto a sus recursos de software y hardware y cada una de las entidades que lo componen tiene integrado su propio sistema de control.

- Además, un sistema distribuido es expandible y escalable. Por la misma razón que cada entidad es programada de tal forma que sea independiente en el sistema, por lo que las entidades pueden ser añadidas o removidas de un sistema, a esto se refiere la escalabilidad.

3.7. Desventajas de un sistema distribuido

Así como los sistemas distribuidos exhiben grandes ventajas, también se pueden identificar algunas desventajas [SOPM], algunas de ellas tan serias que han frenado la producción comercial de sistemas operativos distribuidos en la actualidad.

- El problema más importante en la creación de sistemas distribuidos es el software: los problemas de compartir recursos es tan complejo que los mecanismos de solución generan mucha sobrecarga al sistema haciéndolo ineficiente, con el paso de mensajes, por lo que el software de manejo, es complejo.
- Otro problema es la Seguridad. El posibilitar el acceso distribuido a los datos puede convertirse en un gran problema de seguridad, facilitando también el acceso a datos privados que no deseen ser compartidos. esto implica proveer al sistema de mecanismos tendientes a evitar intrusiones no deseadas a los datos.
- Lo anteriormente descrito en este capítulo servirá para identificar el tipo de sistema distribuido en el cual se basa la tecnología Jini.

CAPÍTULO 4. La tecnología Jini

4.1. Antecedentes

En 1995, el vicepresidente y cofundador de Sun Microsystems, Bill Joy, se dio cuenta gracias a su equipo de trabajo, que Java tenía limitaciones cuando se implementaban aplicaciones en un ambiente distribuido de red [d_JNTA][JAV1].

Jini, se inició como un proyecto en los laboratorios de Sun Microsystems, donde Jim Waldo realizaba un proyecto de investigación conjuntamente con Ken Arnold (inventor de CORBA), Ann Wollrath y Peter Jones, acerca de sistemas distribuidos, con la firme necesidad de crear una nueva tecnología que hiciera más fácil su manejo y explotación.

Como resultado de este trabajo:

- Ann Wollrath y Ken Arnold, crearon y desarrollaron la tecnología Java RMI, la cual se ha convertido en una de las herramientas más poderosas de Java.
- Ken Arnold, creó y desarrolló JavaSpaces, el cual desde sus inicios comenzó a ser un modelo de transacciones para Jini.

La primera versión de Jini fue liberada en 1998 y fue oficialmente lanzada en el mercado, el 25 de enero de 1999.

4.2. Términos básicos

A continuación se define la terminología base (acerca de Jini) que se utilizará en este documento:

- Federación: Consiste en un grupo de dispositivos, programas de software (cualquier programa que es compilado con Java bajo el concepto bytecode) o servicios, que forman un único y dinámico sistema distribuido.
- Recurso: Los dispositivos y programas dentro de una federación son llamados recursos.

- Servicio: Un recurso provee de un servicio. Es un concepto lógico que consiste en la definición del comportamiento de un recurso cuando éste es solicitado con un fin determinado.
- Proxy: bajo el concepto de RMI, es un objeto lógico programado, el cual contiene información necesaria y suficiente (propiedades y atributos) del objeto referenciado, que permite invocar y ejecutar en una computadora diferente, los métodos del objeto invocado.

4.3. Jini: Tecnología de integración de sistemas distribuidos

La tecnología Jini que esta basada en Java, y fue creada para facilitar la integración de sistemas distribuidos, mas adelante se verá de que manera lo facilita. La siguiente tabla muestra que beneficios hereda a partir de que esta basada en Java [d_JNTA]:

Característica	Beneficio
Maquina Virtual de Java	Red Homogénea
Código de objeto Portable	Independencia de arquitectura
Descarga de código	Ambiente dinámico

Tabla 4. Características y beneficios de Jini.

Algunos investigadores en el tema afirman que Jini son siglas que significa *Java Intelligent Network Infrastructure*, es decir, Infraestructura de Red Inteligente Java, aunque realmente Sun Microsystems oficialmente no ha confirmado ni publicado que sea correcta la semántica propuesta de las siglas. Desde un punto de vista en particular, es posible conceptuar la tecnología Jini a través de las siglas propuestas, ya que definen de una manera general el funcionamiento de la tecnología.

4.3.1. Visión

Jini, vista como una tecnología dinámica distribuida, tiene la siguiente visión [JITO]:

- Proveer una infraestructura para conectar cualquier dispositivo a cualquier hora y en cualquier parte de la red. La visión de Jini es proveer una infraestructura que pueda ayudar a diferentes usuarios de la red a descubrir, agregarse y participar en cualquier comunidad de red de forma espontánea.

- Proveer una infraestructura en donde se puedan activar dispositivos bajo el concepto “plug and play”. La meta de Jini es hacer que cualquier dispositivo que pueda proveer de algún servicio, pueda agregarse a la red sin una instalación y/o configuración previa. La visión es tener nulidad en instalación y configuración. Hoy en día, los dispositivos trabajan en un escenario “plug and play”, pero realmente, aún contando con controladores y configuración apropiados, pueden no funcionar.
- Soportar una arquitectura basada en servicios por niveles de abstracción de hardware y software. La visión de Jini es proveer una arquitectura basada en servicios de red en lugar de estar basada en computadoras o dispositivos de red. La arquitectura simplifica el dominio natural del cómputo de red, tratando a todos los dispositivos de la red como servicios. La ventaja de la abstracción, de esta manera, es permitir a la infraestructura estar diseñada para contener un solo tipo de entidad, el servicio. Todos los protocolos que utiliza Jini, pueden ser definidos con respecto al tipo de servicio. La abstracción también permite hacer transparente la implementación del proveedor de servicio hacia el solicitante del servicio.
- Proveer una arquitectura que maneje fallos parciales en la red. Una arquitectura distribuida no esta completa, hasta que provee un mecanismo de manejo de fallos parciales. La visión de Jini es proveer una infraestructura y un modelo de programación asociado que pueda manejar fallos parciales y que permita el establecimiento de una auto-recuperación de los servicios de red.

4.3.2. Metas

Un sistema Jini es un sistema distribuido basado en la idea de federaciones de grupos de usuarios y de recursos requeridos por estos usuarios. La meta principal de Jini es que la red sea flexible, que cuente con una manera fácil de administración con la cual los recursos sean fácilmente encontrados por los usuarios y/o clientes. El enfoque de este sistema es convertir a una red en una entidad dinámica, y por dinámica se debe entender la habilidad de agregar y eliminar servicios sin afectar el rendimiento del sistema.

Las metas principales son [JIN2]:

- Permitir a usuarios compartir servicios y recursos dentro de una red.
- Proveer a los usuarios un fácil acceso a recursos en cualquier parte de la red, independientemente del lugar en donde se encuentren los usuarios.

- Simplificar la tarea de construir, mantener y modificar la red con respecto a los dispositivos que forman parte de ella, así como del software y los usuarios.

El sistema Jini extiende el ambiente de aplicación de Java de una sola maquina virtual a una red de maquinas. El ambiente de aplicación Java provee una plataforma para el computo distribuido ya que tanto el código como los datos pueden ser trasladados de maquina a maquina.

La arquitectura Jini explota esta característica del ambiente de aplicación Java para simplificar la construcción de un sistema distribuido. La arquitectura Jini agrega mecanismos que permiten la comunicación entre todos los componentes del sistema distribuido, extendiendo la habilidad de trasladar objetos en toda la red del sistema.

4.3.3. Supuestos

La tecnología Jini esta basada en los siguientes supuestos [JITO]:

- La existencia de una razonable latencia de red. Esto asegura que la latencia de red no afecta el rendimiento del sistema Jini, ya que Jini depende de la característica de Java de código-móvil.
- Cada dispositivo activado por Jini, en otras palabras, que se encuentra dentro de una federación de Jini, cuenta con memoria y con cierto nivel de procesamiento. Para dispositivos que no cuentan con memoria o capacidad de procesamiento, puede existir un *host* o *proxy*, que contenga capacidad de procesamiento y memoria para proveerlos. Esto es un supuesto fundamental, ya que asegura que todos los elementos de la red tengan un mínimo de capacidad de procesamiento, memoria y habilidad para comunicarse entre sí [JIN3].
- Cada dispositivo debe estar provisto de una Maquina Virtual de Java (JVM). La disponibilidad de una JVM hace más fácil la activación mediante los mecanismos ofrecidos por Java, de cualquier dispositivo.
- Los componentes del servicio son implementados usando el lenguaje Java.

4.4. Componentes de un sistema Jini

Los componentes de un sistema Jini son [JITO] [JIN2]:

- Una **infraestructura**, la cual es el conjunto de componentes que permiten construir un sistema Jini federado, en el cual los servicios son entidades dentro de la federación.
- Un **modelo de programación**, es un conjunto de interfaces que permiten la construcción de servicios confiables.
- Los **servicios** soportados por los miembros del sistema.

4.4.1. Infraestructura Jini

La infraestructura Jini es el componente central de la arquitectura y su meta consiste en proveer mecanismos para que los dispositivos, servicios y usuarios puedan acceder a los servicios. La infraestructura a su vez tiene los siguientes elementos:

- **Protocolos discovery, join y lookup**, los cuales son protocolos de servicio que permiten a los propios servicios, descubrir, anunciarse a otros miembros de la federación para poder ser accedidos por algún miembro. Estos protocolos definen la forma en la que un servicio se vuelve parte de un sistema Jini.
- **Sistema de seguridad distribuido**, integrado dentro de RMI, que extiende el modelo de seguridad de java al sistema distribuido. Define como las entradas son identificadas y establece derechos sobre los servicios.
- **Lookup service**, el cual sirve como repositorio de los servicios, al contener los proxies de los servicios. Las entradas del lookup service son objetos implementados en el lenguaje de programación Java, éstos objetos pueden ser descargados como parte de la operación del lookup service, y actúa como un proxy local del servicio que se encuentra registrado en el lookup service. El lookup service refleja los miembros que actualmente se encuentran en la federación y actúa como un centro de reunión de los miembros de la federación, donde se ofrecen y se encuentran servicios

4.4.2. Modelo de programación

El modelo de programación esta basado en Java y tiene la habilidad de poder programar el traslado del código entre elementos de la red, el cual puede basarse en RMI, además de estar soportado por la infraestructura. Los servicios constituyen un conjunto de interfaces, las cuales definen protocolos de comunicación que son utilizados por los mismos servicios y la infraestructura para comunicarse entre ellos,

como lo puede ser RMI. Estas interfaces, conforman la extensión distribuida del lenguaje de programación Java que constituye el modelo de programación de Jini, estas interfaces son [JITO] [d_JNTA]:

- **Interfaz Leasing**, define la forma de asignar y liberar recursos usando la renovación por tiempo.
- **Interfaz de Notificación de eventos**, la cual es una extensión del modelo de eventos usado por los componentes de JavaBeans, para un ambiente distribuido el cual permite la comunicación basada en eventos entre los servicios. Un objeto puede permitir a otro objeto registrar el conjunto de eventos que son de su interés y que éste reciba la notificación de la ocurrencia del evento.
- **Interfaz de Transacción**, la cual permite al sistema manejar transacciones orientadas a objetos, introduciendo un protocolo muy simple, para coordinarlos cambios de estado. La interfaz no define los mecanismos de transacción pero provee reglas para el manejo de los mismos, por lo que da libertad de elegir mecanismos de transacción.

4.4.3. Servicios

La infraestructura y el modelo de programación nos permiten desarrollar servicios para que sean ofrecidos dentro de la federación. Los servicios hacen uso de la infraestructura para descubrirse, llamarse y comunicarse unos con otros.

Los servicios tienen una interfaz que define las operaciones que pueden ser requeridas por otro servicio. El tipo de servicio determina la interfaz que será implementada, y esto también definirá el conjunto de métodos que pueden ser accedidos por el servicio que lo solicite.

4.5. Una arquitectura basada en servicios

La parte fundamental de un sistema Jini es el conjunto de protocolos Discovery, Join, y el lookup, que a continuación de describen [JITO] [d_JNTA] [JITB]:

- Discovery específicamente es utilizado cuando un cliente o un servicio esta buscando a un lookup service, el servicio lo hace para registrarse y el cliente para solicitarlo.

- Join, es utilizado cuando un servicio ha localizado a un lookup service, en donde desea agregarse.
- Lookup es utilizado cuando un cliente o usuario necesita invocar un servicio por medio de su interfaz (proxy) que se encuentra registrado en el lookup service.

La figura 8 muestra como se conforma la arquitectura Jini, se puede apreciar la independencia de plataforma, en la que se puede correr una aplicación Jini.

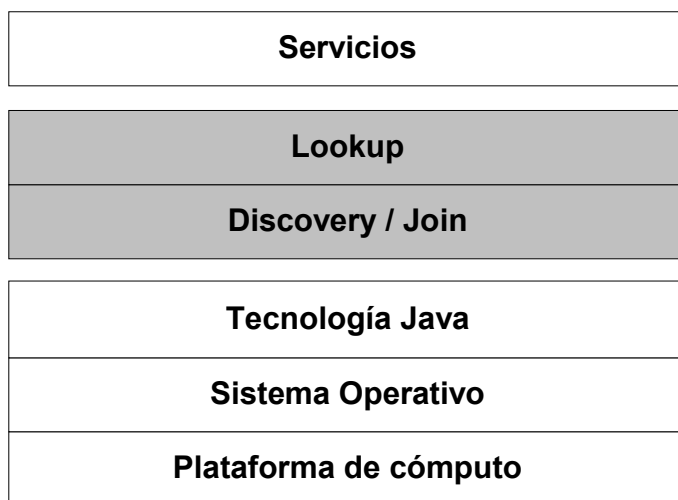


Figura 8. Arquitectura Jini.

Las capas de Discovery / Join y Lookup, son las capas que representan a la tecnología Jini, en la capa de Tecnología Java, cuenta con una JVM y contiene la implementación de RMI [JIN3], que es la que se encarga de la comunicación en el acceso a los servicios.

4.6. Funcionamiento general de Jini

A continuación se describirán los pasos fundamentales en el funcionamiento de Jini [JIN2] [PFJN] [d_JNTA]

Paso1. Descubrimiento de el lookup service por parte del proveedor del servicio.

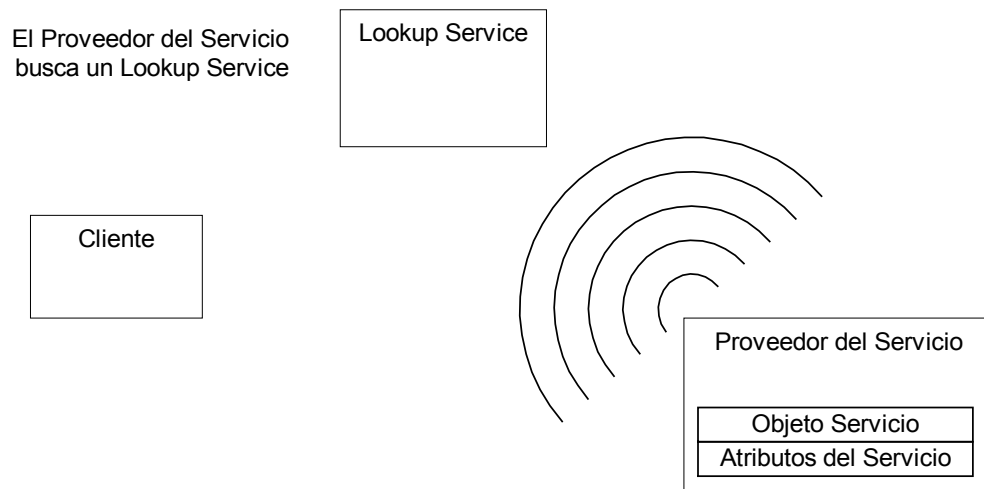


Figura 9. Primer paso del funcionamiento de Jini.

Se hace uso del protocolo discovery para que un proveedor del servicio pueda encontrar un Lookup Service.

Paso 2. El proveedor del servicio registra un proxy en el lookup service.

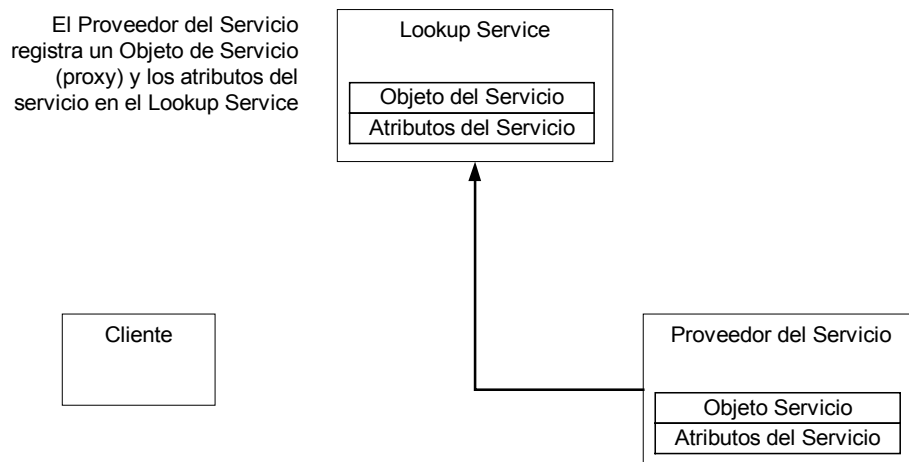


Figura 10. Segundo paso del funcionamiento de Jini.

Haciendo uso del protocolo join, el proveedor del servicio registra su proxy, en el lookup service.

Paso 3. Un cliente descubre al lookup service.

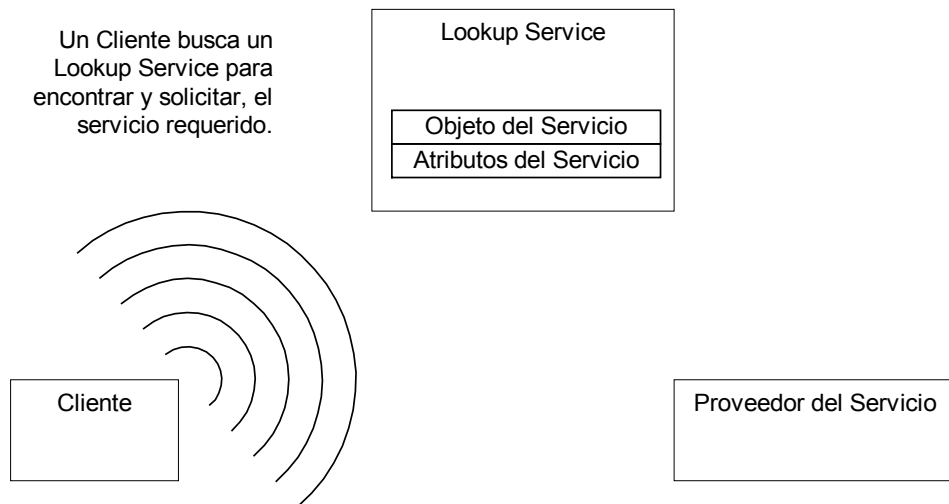


Figura 11. Tercer paso del funcionamiento de Jini.

Un cliente hace uso del protocolo discovery para descubrir al lookup service.

Paso 4. Un cliente requiere del proxy del servicio para acceder al servicio.

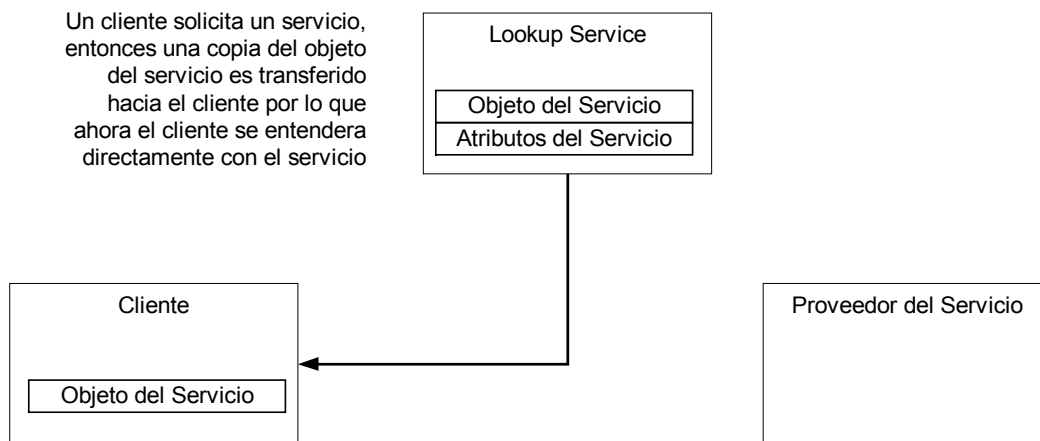


Figura 12. Cuarto paso del funcionamiento de Jini.

El cliente hace uso del protocolo lookup para poder acceder al proxy del servicio y referenciar la copia del proxy.

Paso 5. El cliente hace uso del servicio.

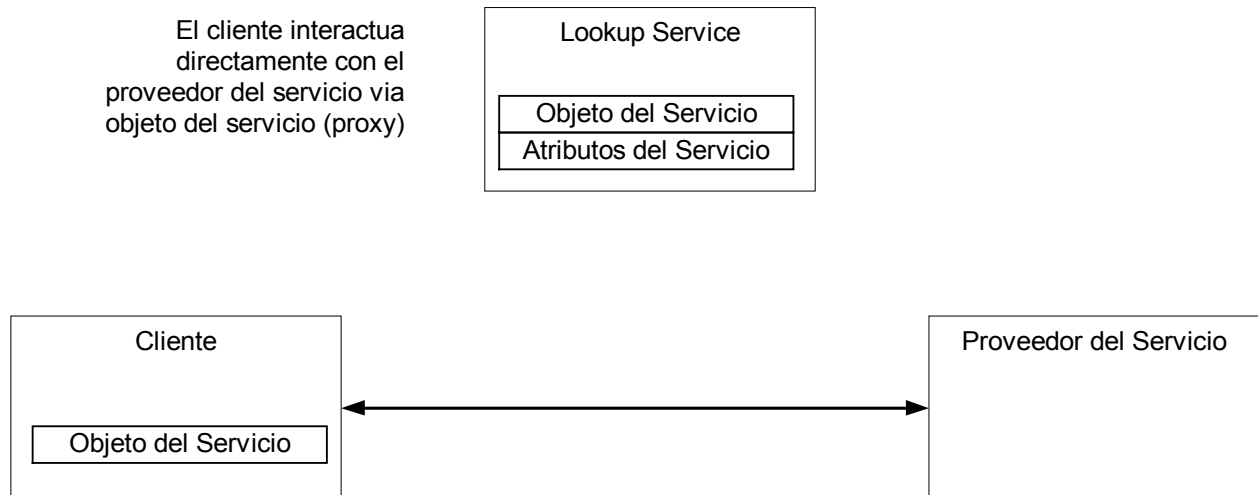


Figura 13. Quinto paso del funcionamiento de Jini.

El cliente puede acceder al servicio, a través del proxy del servicio, por lo que el proxy debe estar sincronizado con el del proveedor del servicio.

4.7. Grupos Jini

Con el objetivo de restringir el acceso a dichos servicios e imponer restricciones, pueden existir enormes colecciones de servicios que pueden estar divididos por grupos. El concepto de grupos o comunidades, divididos físicamente dentro de la red Jini, es muy diferente al concepto de federación. En un grupo o comunidad, los servicios y clientes se vuelven miembros de un grupo específico dentro de la red, un servicio o cliente puede pertenecer a múltiples grupos y un *lookup service* dentro de una red Jini puede dar servicio a uno o más grupos definidos. En cambio una federación es un conjunto de comunidades o grupos Jini que pueden interactuar entre ellas. Las federaciones pueden ser completamente independiente de cualquier grupo o de cualquier protocolo discovery [PFJN].

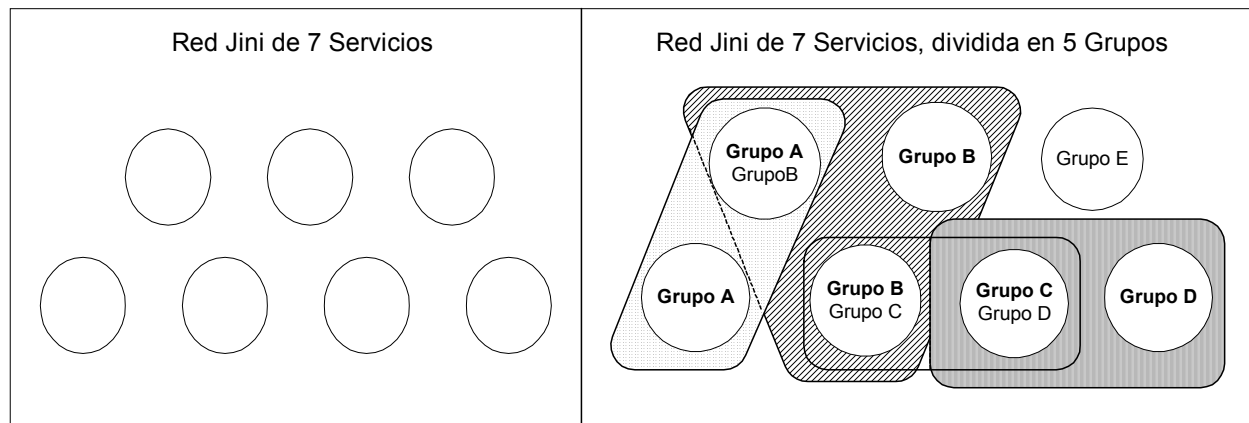


Figura 14. Ejemplo de división por grupos en una red Jini.

La lista de grupos asociada a un lookup service, en otras palabras, los servicios y/o los clientes asociados al grupo pueden no ser estáticos, es decir, pueden ser cambiados a otro, por periodos de tiempo, de forma automática o por un administrador del sistema. En la figura 14, se muestra una topología lógica de los servicios divididos por grupos.

Aunque las limitaciones de extensión de red no están preescritas, los desarrolladores de la tecnología Jini han utilizado federaciones robustas y en diferentes áreas geográficas en la implementación de Jini, ya que Jini tiene la habilidad de manejar de forma remota la interacción entre ellas, un ejemplo de esta interacción de federaciones es ilustrada en la figura 15.

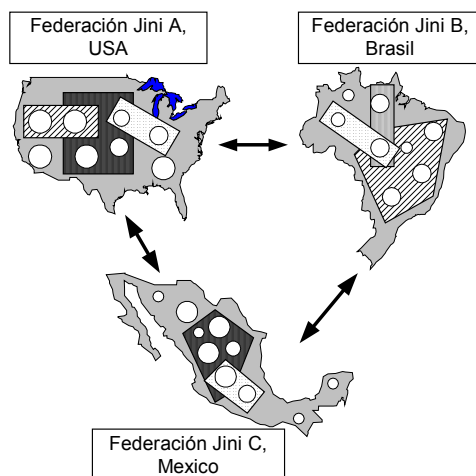


Figura 15. Conjunto de Federaciones Remotas.

4.8. Protocolos Jini

4.8.1. Protocolos discovery

Por medio de este protocolo es como un cliente encuentra o descubre a un lookup service para posteriormente registrarse y como un cliente encuentra o descubre a un lookup service cuando desea acceder a algún servicio.

A diferencia de los sistemas convencionales cliente / servidor, los clientes y servicios de un sistema Jini están débilmente acoplados a través de los lookup services.

Ambos, tanto los clientes como los servicios, necesitan localizar a un lookup service para incorporarse dentro de la red Jini. Los servicios se publican a ellos mismos, registrando sus proxies en el lookup service [PFJN].

En las figuras 9 y 10 se ilustran los posibles casos de utilización del protocolo discovery.

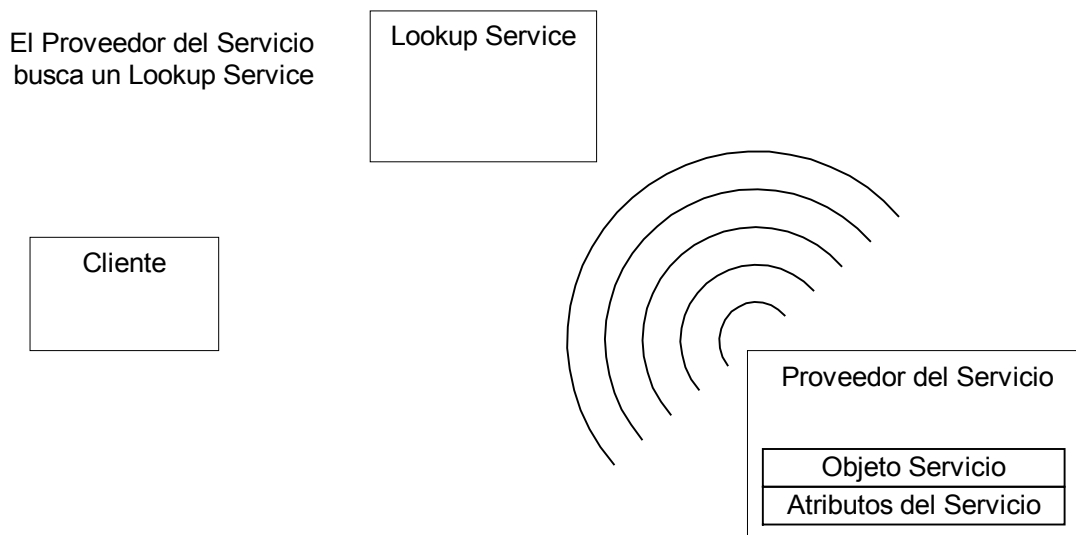


Figura 16. Protocolo Discovery, el proveedor del servicio busca un lookup service (caso 1).

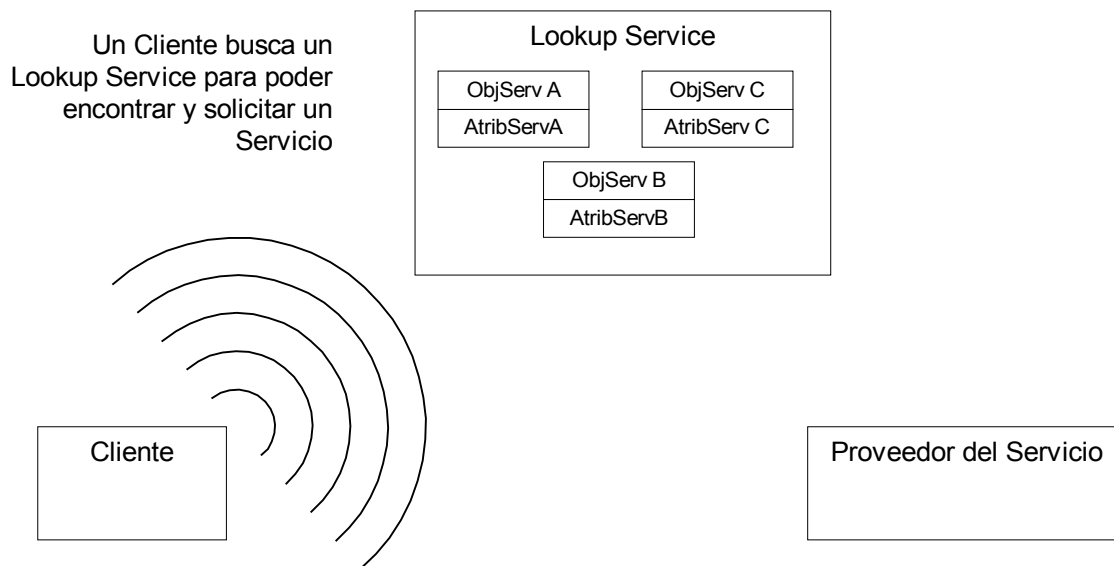


Figura 17. Protocolos Discovery, un cliente busca al lookup service (caso 2).

Existen 2 métodos discovery:

- **Multicast Discovery**, generalmente usado para descubrir y contactar cualquier servicio que este cercano, a través de grupos definidos, basado en la forma de transmisión de datos en TCP/IP, Multicast.
- **Unicast Discovery**, es cual utiliza un URL estático, en el cual se especifica un servicio que ya esta registrado, para ser contactado, aún cuando éste se encuentre alejado (en otra subred), basado en la forma de transmisión de datos en TCP/IP, Unicast.

Cuando el servicio se integra a una nueva comunidad Jini, realiza una operación de Discovery. La operación realizada puede ser multicast o unicast, cuando el servicio hace un Discovery de tipo unicast se registra con un solo lookup service predefinido por el mismo servicio . Por el contrario, si el servicio realiza un Discovery multicast, entonces el servicio tendrá la opción de registrarse con todos los lookup service que respondan su petición.

Estos métodos recaen dentro de los siguientes 3 protocolos:

- Unicast Discovery Protocol
- Multicast Request Protocol
- Multicast Announcement Protocol

Unicast Discovery Protocol.

Este protocolo se utiliza cuando los servicios y los clientes ya conocen la localización del lookup service, que desean o requieren. En este caso, el lookup service puede ser accedido de manera remota, a diferencia de los 2 siguientes protocolos, éste protocolo está basado completamente en una conexión TCP.

Multicast Request Protocol.

Este protocolo es usado por los servicios y clientes cuando se inician en la red Jini para solicitar la respuesta de los lookup services cercanos, el término cercanos se refiere a un radio multicast definido de acuerdo al protocolo multicast IP.

Multicast Announcement Protocol

Este protocolo permite a un lookup service anunciar su presencia cuando acaba de ser dado de alta en la red a los clientes y servicios.

4.8.2. Protocolo join

Este protocolo a diferencia del protocolo discovery, no es de bajo-nivel. Se refiere a un conjunto de requerimientos que deben ser seguidos por un servicio cuando se agrega o deja una federación Jini. Su función es proveer a los servicios de elementos que aseguren su buen funcionamiento aún cuando lleguen a ocurrir eventos como fallas en la energía eléctrica, bloqueos en el sistema o reinicializaciones, así como prever y garantizar la consistencia y estabilidad dentro del sistema Jini. Un sistema Jini es presentado del mismo modo a los servicios que a los clientes, como un sistema de “larga duración”, desde que los lookup services y los servicios residen en diferentes máquinas y redes, es inevitable que ambas puedan sufrir algún tipo de daño, por lo que se debe proveer mecanismos por los que se pueda mantener al sistema en un estado de “larga duración”.

Los principales elementos de información para que un servicio pueda guardar un cierto estado de persistencia, son:

- Un **identificador del servicio (ID)**, el cual se le asigna al servicio cuando éste se registra por primera vez en el lookup service.

- Un **conjunto de atributos**, los cuales pueden borrarse o actualizarse según las necesidades del servicio y esta operación se realiza en todos los lookup services en los que el servicio esté registrado.
- Un **conjunto de grupos** a los que el servicio pertenece según su características o diseño.
- Un **conjunto de lookup services** donde el servicio se quiere registrar, aunque este elemento (lookup service) será referenciado en el servicio siempre y cuando éste no se esté registrando por primera vez.

4.8.3. Protocolo lookup

Este protocolo se encarga de las reglas y transacciones para poder buscar el servicio requerido por el cliente en el lookup service y así poder acceder al proxy del servicio para poder interactuar con el proveedor del servicio.

4.9. Lookup Service

Un lookup service, es la parte fundamental de Jini. De hecho, sin los lookup service que se encuentran dentro de la comunidad Jini, no existiría una red dinámica Jini. El sistema que utiliza Jini, consiste en un conjunto de servicios que se registran en un lookup service, cuya función es mantener información dinámica sobre los servicios disponibles en la red; los objetos proxy que permiten conocer su funcionamiento y los atributos que lo definen. En una comunidad Jini podemos encontrar uno o varios lookup services, cuya tarea principal es la administración de la red local [PFJN].

De manera general un lookup service, tiene las siguientes responsabilidades:

- Asistente que sirve para iniciar servicios dentro de la comunidad Jini.
- Localizador de servicios.
- Mantener el conjunto de objetos proxy, registrados en él.
- Mantener al sistema de forma activa y funcional.
- Asignar automáticamente un identificador único (ID) a cada servicio registrado.

4.9.1. Los roles de un lookup service

El siguiente diagrama representa la interacción entre un lookup service, un cliente y un servicio, dentro de una red Jini [PFJN].

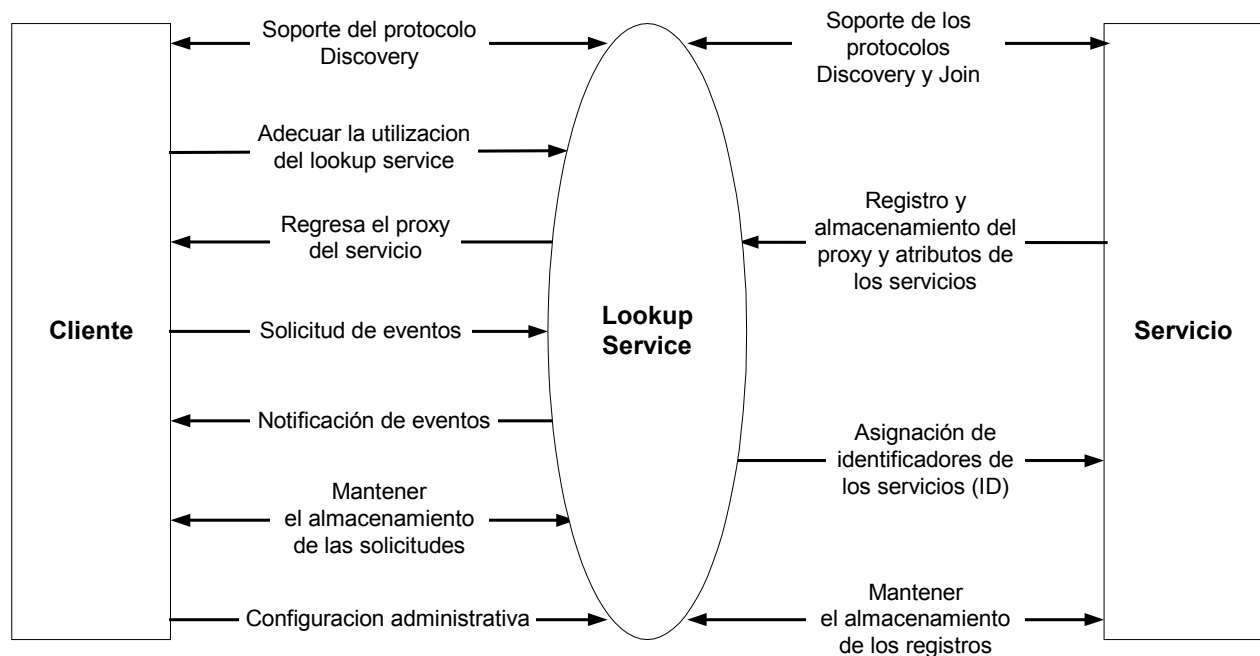


Figura 18. Interacción entre un cliente, un lookup service y un servicio.

- **Soporte del protocolo discovery:** El cliente encuentra un lookup service usando unicast o multicast discovery. El lookup service debe soportar ambos protocolos (unicast y multicast).
- **Adecuar la utilización del lookup service:** El cliente localiza un servicio usando el lookup service. El cliente debe proveer de una plantilla de búsqueda, la cual contiene los criterios para la localización del servicio.
- **Regresa el proxy del servicio:** El lookup service envía el objeto proxy del servicio, cuando un cliente lo solicite.
- **Solicitud de eventos:** El cliente le avisa al lookup service, que esta interesado en ser notificado, cuando ocurran cambios dentro del lookup service (v. gr. cuando un servicio específico, se agregue en la federación). El lookup service almacenará la solicitud.

- **Notificación de eventos:** El lookup service notificara al cliente vía remota cuando el evento, en el cual el cliente esta interesado, ocurra.
- **Mantener el almacenamiento de las solicitudes:** Todas las solicitudes son almacenadas un cierto tiempo. El lookup service concede un tiempo específico para el almacenamiento de solicitudes (leasing). El cliente debe renovar el tiempo antes de que expire el límite especificado.
- **Configuración administrativa:** El cliente puede controlar y configurar ciertos aspectos de un lookup service (v.gr. la frecuencia de transmisión de los paquetes multicast).
- **Soporte de los protocolos Discovery y Join:** El servicio localiza un lookup service utilizando el protocolo Discovery y se agrega a él, utilizando Join. El servicio interactúa con el lookup service de manera constante a través del protocolo Join, como ya se vio anteriormente.
- **Registro y almacenamiento del proxy y atributos de los servicios:** Un servicio registra su propio proxy y atributos (si los atributos son requeridos) en el lookup service, el cual los almacenará de forma persistente.
- **Asignación de identificadores de los servicios (ID):** Cada instancia a un servicio en el universo de la red Jini tienen un único identificador (ID), generalmente esto ocurre cuando un servicio se inicia dentro del lookup service.
- **Mantener el almacenamiento de los registros:** El registro de cualquier proxy es almacenado durante cierto tiempo (leasing). Cualquier servicio que falle en renovar su registro de su proxy, será removido del almacenamiento persistente.

4.10. Implementación de servicios en Jini

Existen por lo menos cinco formas principales de implementar servicios en Jini, basadas en el proxy del servicio [PFJN]:

- Un proxy se ejecuta totalmente en el cliente.
- Un proxy que utiliza RMI para comunicarse con el servicio, independientemente del cliente.
- Un proxy que utiliza RMI para comunicar al cliente con el servicio.

- Un proxy que utiliza un protocolo específico para comunicarse con el servicio (v.gr. comunicación basada en sockets).
- Un proxy que hereda ciertas propiedades del sistema donde se encuentra el servicio (v.gr. un servicio CORBA).

De igual manera, existen patrones de implementación de servicios, como parte del diseño, hay que concentrarse en la distribución de trabajo entre el proxy y la lógica del servicio. De acuerdo a la forma en la que implementemos el proxy tendremos los siguientes puntos:

- Un proxy se ejecuta totalmente en el cliente: la funcionalidad del servicio es llevado localmente por la maquina virtual del cliente.
- Un proxy que utiliza RMI para comunicarse con el servicio, independientemente del cliente : la funcionalidad del servicio es realizado de manera remota en la maquina virtual, donde reside el servicio, el proxy en realidad es un RMI stub hacia la interfaz del servicio.
- Un proxy que utiliza RMI para comunicar al cliente con el servicio: la funcionalidad del servicio es dividida entre la maquina virtual del cliente y la del servicio, esta forma es llamada 'smart proxy', donde la comunicación entre el cliente y el servicio esta basada en RMI.
- Un proxy que utiliza un protocolo específico para comunicarse con el servicio (v.gr. comunicación basada en sockets), la funcionalidad del servicio es dividida entre la maquina virtual del cliente y la del servicio, pero ésta vez, el proxy no utiliza RMI para comunicarse con el servicio, en lugar de ello utiliza su propio protocolo usando sockets.
- Un proxy que hereda ciertas propiedades del sistema donde se encuentra el servicio (v.gr. un servicio CORBA), la funcionalidad del servicio es realizada con base en CORBA.

4.11. Java RMI, soporte de Jini.

Una de las ventajas que tiene Jini es la habilidad poder implementar el acceso a los servicios a través de RMI. Un proveedor de servicio puede implementar su proxy con base en RMI, por lo que el cliente debe invocar el método de implementación del

servicio. Es importante conocer el funcionamiento de RMI, por lo que ahora se procederá a describir brevemente [JRMI].

La Invocación Remota de Métodos (RMI) es una arquitectura desarrollada con base en Java, para desarrollar aplicaciones con objetos distribuidos, en otras palabras, un objeto que se esté ejecutando en un maquina virtual Java puede invocar o llamar los métodos de otro objeto que este residente en otra maquina virtual diferente.

RMI puede verse como la evolución en la forma de comunicación en el computo distribuido, entre un cliente y un servidor. Como muestra la figura 19, la evolución del computo distribuido respecto a la forma de comunicación entre un cliente y un servidor, en su ultima evolución es RMI.

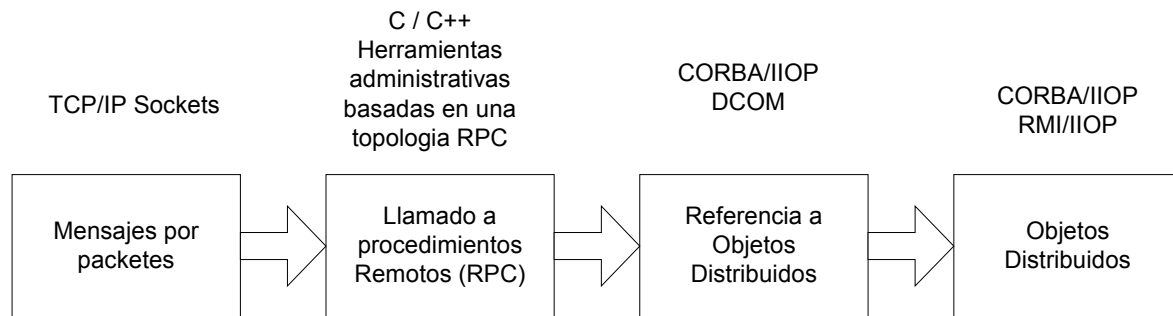


Figura 19. Evolución de la forma de comunicación del computo distribuido.

Los principales objetivos de RMI son:

1. Permitir invocación remota de métodos en objetos que residen en diferentes Máquinas Virtuales.
2. Integrar el Modelo de Objetos Distribuidos al lenguaje Java de modo natural y preservando en lo posible la semántica de objetos en Java.
3. Permitir la distinción entre objetos locales y remotos.
4. Preservar la seguridad de tipos dada por el ambiente de ejecución Java.
5. Permitir diferentes semánticas en las referencias a objetos remotos: no persistentes, persistentes, de activación lenta.
6. Mantener la seguridad del ambiente dada por los manejadores de seguridad y los cargadores de clases.

7. Facilitar el desarrollo de aplicaciones distribuidas

Arquitectura RMI

RMI se puede analizar en términos de un modelo de capas. El programador solo se ocupa de la capa de Aplicación; las demás capas son responsabilidad del sistema RMI [PFJN].

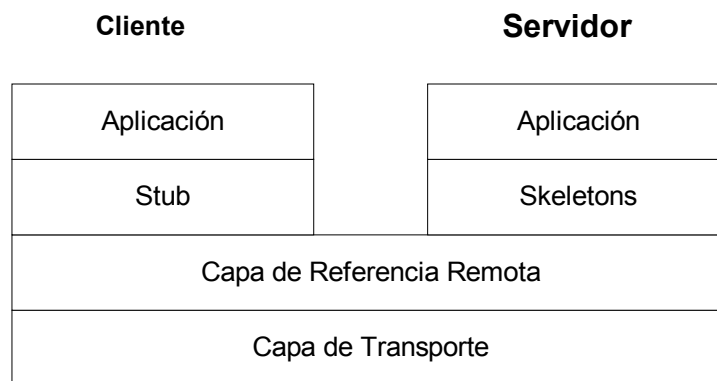


Figura 20. Arquitectura RMI, por capas.

La función de las distintas capas se explica a continuación:

- **Capa de Aplicación:** En esta capa se encuentran los objetos que se implementan en la aplicación. En general en este nivel se distinguen dos tipos de elementos: los clientes, que son objetos que invocan métodos o hacen peticiones a otros objetos remotos y los servidores, que son objetos que reciben peticiones de objetos remotos.
- **Capa de Stubs y Skeletons:** . En esta capa se encuentran los objetos que actúan como representantes locales de objetos remotos. Se encargan del embalaje y desembalaje (marshalling / unmarshalling) de las invocaciones, argumentos y resultados de métodos remotos. El embalaje y desembalaje es el proceso por el cual debe pasar toda información para que ésta sea utilizable en ambientes heterogéneos, es decir, en este caso este proceso ayuda a que el paso de parámetros de un procedimiento o método, sea independiente de plataforma (windows NT, Posix), en otras palabras, si se ejecuta un programa en una sola computadora se tiene la seguridad que la representación de datos esta conforme a esa plataforma, sea: SunSolaris, Linux, Windows..etc, sin embargo,

si se intentan pasar parametros a un procedimiento en un servidor Linux de una computadora utilizando Windows NT debe implementarse este procedimiento.

- **Capa de Referencias Remotas:** . En esta capa se realiza la interpretación de las referencias a objetos remotos, las referencias locales a talones o esqueletos se resuelven a sus contrapartes remotas y estos datos, junto con los paquetes "embalados" (marshalled) que contienen las invocaciones o los resultados de invocaciones se pasan a la Capa de Transporte.
- **Capa de Transporte:** En esta capa se encuentra el protocolo de comunicación, se encarga de transportar los mensajes que intercambian los distintos objetos. RMI utiliza por omisión el protocolo TCP/IP. Por encima de TCP/IP, RMI puede utilizar IIOP (Internet Inter-ORB Protocol).

Funcionamiento en una aplicación Genérica:

Servidor:

- Crea objeto remoto
- Crea referencia al objeto remoto
- Espera a que un cliente invoque un método en el objeto remoto

Cliente:

- Obtiene referencia a un objeto remoto en el servidor
- Invoca un método remoto

Componentes de un sistema RMI

Un sistema RMI se compone de varias partes:

- Definiciones de Interfaces para servicios remotos.
- Implementación de los servicios remotos.
- Archivos Stub y Skeleton: los cuales son creados de manera automática por Java, cuando se compilan.

- Un servidor donde se encuentran los servicios remotos.
- Un servicio de nombres para que los clientes puedan encontrar los servicios remotos.
- Un proveedor de clases.
- Un programa cliente que requiere de servicios remotos.

4.12. Jini Surrogate

Para que un dispositivo pueda ser parte de una red Jini, debe satisfacer varios requerimientos, como se menciona en la sección 4.3.3, respecto a los recursos de procesamiento y la ausencia de una máquina virtual de Java.

En ciertos dispositivos, por una razón u otra, no pueden satisfacer alguno de los requerimientos de la arquitectura Jini, como lo puede ser la falta de habilidad para almacenar código en memoria, ya sea porque sean limitados en cuanto a los recursos con los que cuentan o porque tienen algunas deficiencias con respecto a la conectividad de red, por lo que no pueden formar parte de una red Jini.

La arquitectura *Jini Surrogate* (Arquitectura Sustituta para Acceso Jini) aborda este problema y define medios por los cuales estos dispositivos, con ayuda de una tercera parte, pueden participar dentro de una red Jini [e_JSUR].

4.12.1. Requerimientos de la arquitectura surrogate

Los siguientes requerimientos para la arquitectura *surrogate* pueden ser clasificados de la siguiente manera:

- **Independencia del Tipo de Dispositivo:** La arquitectura *surrogate* debe tener la habilidad de dar soporte a una amplia gama de componentes de hardware y software con diferentes capacidades computacionales.
- **Independencia del Tipo de Red:** La arquitectura *surrogate* debe poder alojar o contener una amplia gama de tecnologías, con lo que el nivel de conectividad podrá incrementarse. La independencia de la red incluye el soporte para el uso de diferentes protocolos simultáneos en la misma capa de transporte.
- **Preservar la característica *Plug-and-Play*:** La arquitectura *surrogate* debe preservar como característica el modelo *Plug-and-Play* de la arquitectura Jini, en

otras palabras, no tener la necesidad de la instalación de controladores, ya que en la arquitectura Jini se incluye el concepto de descubrimiento (*discovery*), almacenamiento de código y arrendamiento de recursos distribuidos.

4.12.2. Definiciones

Para este tema, los siguientes términos son definidos:

- **Dispositivo:** se refiere a componentes de software o de hardware que no están directamente participando en la red Jini.
- **Surrogate:** es un objeto que representa a un dispositivo. La implementación de un objeto puede ser una colección de componentes de software Java y otros recursos.
- **Maquina host con recursos de procesamiento:** es un sistema que permite bajar o almacenar código, en donde puede ejecutarse como un sustituto de la entidad original. Esta maquina host es parte de la red Jini y debe ser accesible a la entidad que esta siendo “sustituida”.
- **Surrogate host:** es una estructura (*framework*) que reside en el Maquina host con recursos de procesamiento y provee del ambiente de aplicación de Java para poder ejecutar los componentes de la arquitectura surrogate.
- **Servidor de exportación de recursos:** es un componente o una colección de componentes, que trabajaban con el surrogate host o que pueden ser parte del mismo. El servidor de exportación de recursos provee los medios por los cuales los recursos surrogate, son exportados y son puestos a disponibilidad para ser accedidos y almacenados por entidades remotas. La exportación de recursos, particularmente archivos .class, son necesarios para la mayoría de las operaciones en la arquitectura Jini.
- **Interconexión:** es la conexión lógica y física entre el surrogate host y el dispositivo, puede que haya mas de un dispositivo conectado, esto lo define la conexión física. Para que exista la conexión física y lógica, éstas deben ser compatibles con la arquitectura Jini, respecto de los protocolos utilizados por ésta arquitectura.
- **Protocolo de Interconexión:** incluye los mecanismos específicos de interconexión para el *discovery*, recuperación de información del *surrogate* y un ciclo de vida.

4.12.3. Perspectiva General

La primera suposición de la arquitectura *surrogate* es que existe una maquina host con recursos de procesamiento, que esta interconectada al dispositivo y a la red Jini.

Por definición, un maquina host con recursos de procesamiento es una entidad que tiene recursos computacionales para ejecutar el código escrito en Java, que es vista como sustituto del dispositivo que no cuenta con esos recursos, por lo que éste puede proveer de tantos recursos como sea posible (el programa lo pida) al dispositivo.

La siguiente figura describe, en términos generales, los componentes de la arquitectura *surrogate*, las relaciones entre estos componentes y el mecanismo de ejecución.

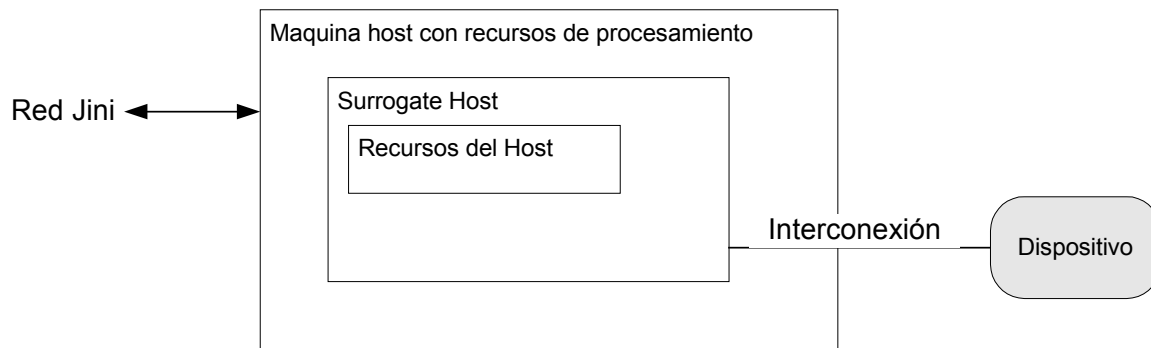


Figura 21 Componentes de la arquitectura Surrogate.

En la figura 21, se observa que la Interconexión del dispositivo con la maquina host con recursos de procesamiento y la Red Jini, son vistas como 2 caminos diferentes y separados. Pero en la práctica, sin embargo, pueden ser el mismo camino. También se observa en esta figura, a un *Surrogate host* dentro de la maquina host. El *surrogate host* monitorea la conexión con el dispositivo y es posible que un solo *surrogate host* puede monitorear mas de una interconexión (más de 1 dispositivo).

4.12.4. Discovery and Surrogate Loading

El *surrogate host* es responsable de implementar el protocolo de interconexión para establecer una conexión especifica con el dispositivo. La primera parte del

protocolo de interconexión es el *discovery*¹. *Discovery*, en este contexto, es el protocolo que es usado para que dispositivo y el *surrogate* puedan encontrarse. Una de los objetivos funcionales del protocolo *discovery*, es la interconexión y puede ser muy diferente de cualquier otra interconexión *discovery*, ya que depende de las capacidades de conexión del dispositivo y del dispositivo mismo.

Un escenario probable para el dispositivo sería: emitir una petición para un *surrogate host* a través de una interconexión. El *surrogate host* respondería dejando que el dispositivo tenga conocimiento que hay un *surrogate host* disponible. Si el dispositivo o interconexión, no soporta el protocolo *discovery*, entonces puede que sea necesario que el *surrogate host* detecte la llegada del dispositivo en la interconexión.

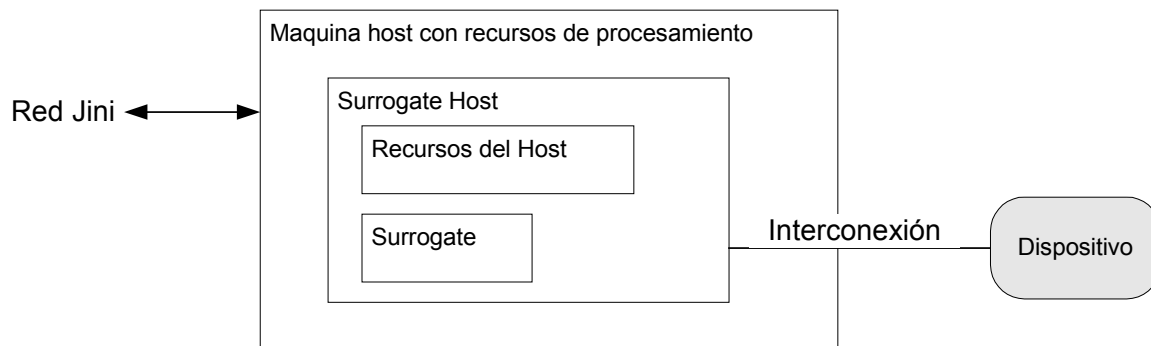


Figura 22. Discovery y Surrogate establecidos

Una vez que el *discovery* ha sido realizado, el objeto generado por el dispositivo (*device's surrogate*) debe ser guardado. Dependiendo de la interconexión y de las capacidades del dispositivo, esta operación puede ser:

- **Push**, el dispositivo introduce el surrogate al *surrogate host*.
- **Pull**, el *surrogate host* debe extraer, o acceder y almacenar el *surrogate* del dispositivo. También es posible que el *surrogate* se encuentre en otro lugar diferente del dispositivo.

¹ No se debe confundir el termino **discovery**, en este contexto, con el termino discovery del Jini lookup discovery protocol. Cuando este documento se refiera al **Jini lookup discovery protocol** se utilizará el termino **Jini discovery** o **Jini discovery protocol**.

4.12.5. Ejecución surrogate

El *surrogate* es activado por el *surrogate host*, y puede hacer uso de los recursos provistos por el *surrogate host*. Ejemplos de estos recursos pueden ser un servicio HTTP o el uso de clases de la arquitectura Jini. Una vez que el *surrogate* ha sido activado, puede llevar acabo cualquier tarea necesaria en nombre del dispositivo, incluyendo el acceso a la red Jini. El *surrogate* puede incluso comunicarse de forma *duplex* hacia el dispositivo, usando cualquier protocolo apropiado.

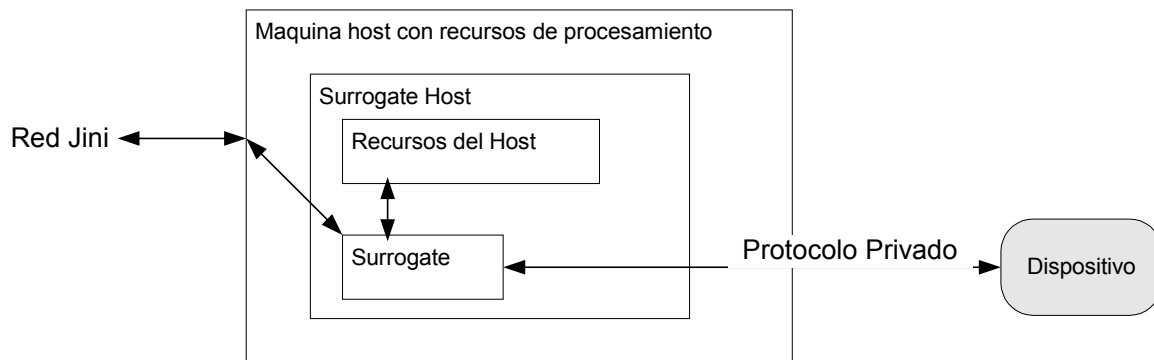


Figura 23. Ejecución surrogate.

Ciclo de vida de la ejecución surrogate

Una vez que el *surrogate* es llamado y activado, cualquier *surrogate* o *surrogate host* puede monitorear el dispositivo por un determinado periodo de tiempo. El ciclo de vida del surrogate implica que existe un camino disponible de comunicación entre un *surrogate* y el dispositivo, por lo que ambas entidades deben estar activas. La perdida del camino de comunicación puede ser el resultado de las condiciones del *surrogate host*, el *surrogate*, la interconexión o el dispositivo.

Si el dispositivo no es accesible, por un fallo en la interconexión o en el dispositivo, o porque el dispositivo ha sido desconectado o apagado, el *surrogate* que lo representa debe ser desactivado, por lo que los recursos asignados al dispositivo, deben ser liberados. También un *surrogate* debe liberar cualquier recurso en espera, como por ejemplo los registros hechos en el surrogate. El dispositivo debe poder finalizar la interacción con el surrogate, ya sea porque pierda contacto con el *surrogate*, esto puede deberse a una conexión fallida, un fallo en el *surrogate* o en el *surrogate host*. En cualquiera de estos eventos, el dispositivo debe descubrir otro *surrogate* mediante el protocolo *discovery* o de otra manera prepararse a encontrar un nuevo *surrogate*.

CAPÍTULO 5. Tecnologías móviles actuales

5.1. Conceptos básicos de telefonía celular

5.1.1. La célula

Antes de poder hablar un poco mas a detalle a cerca de las tecnologías móviles directamente relacionadas con la telefonía celular, deben entenderse los componentes básicos del sistema.

El concepto inicial de la telefonía celular es la célula, la cual se puede definir como el área cubierta por la señal de una estación base, teóricamente el área de cobertura de una estación base en una región plana y sin obstáculos tiene la forma de un círculo, pero el área de cobertura asume un perímetro irregular, determinado por la superficie de la región [EGWC].

Para facilitar los análisis matemáticos se adaptó como representación de la célula como un hexágono, que es un polígono regular con perfil próximo al de un círculo, como se muestra en la figura 24. También difiere de la teoría el límite entre células. En la práctica las áreas de cobertura de las células tienen los límites sobrepuestos o traslapados, lo que posibilita la continuidad de la cobertura conocido como *Handoff*, el cual se define como la transición del enlace de comunicación de una estación base a otra cercana, en la figura 24, se ilustra este concepto.

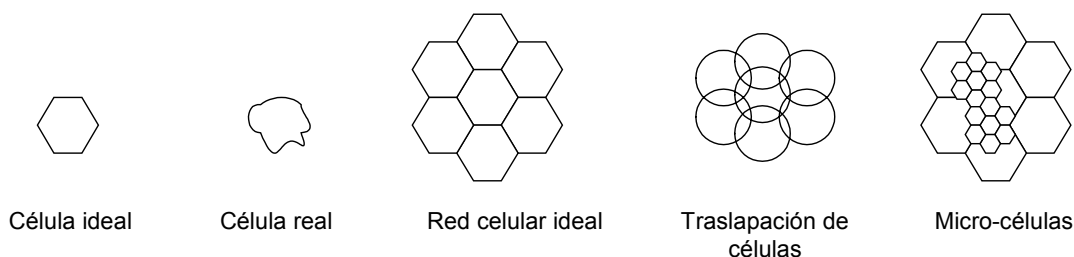


Figura 24. Tipos y características de las células en la telefonía móvil.

En la figura 24, también se ilustra el concepto del uso de micro-células, las cuales son usadas, cuando existen una gran cantidad de usuarios en una área pequeña, por ejemplo: un edificio o una calle.

5.1.2. Arquitectura básica de un sistema telefónico celular

Para proporcionar un servicios de comunicación móvil dentro de una región particularmente geográfica, se debe emplear una red integrada de estaciones base (BTS), también conocido como *transceiver*, para proporcionar la suficiente cobertura radial a todos los usuarios móviles encontrados en esa zona. Las estaciones base, a su vez, deben estar conectadas a un controlador de estaciones base (BSC), los cuales a su vez a un eje central llamado centro de conmutación móvil (MSC). El MSC proporciona conectividad entre la red pública de telefonía conmutada (PSTN) y las numerosas estaciones base, y por último, entre todos los abonados móviles de un sistema. La PSTN forma la red de telecomunicaciones global que interconecta los centros de conmutación de telefonía convencional (terrestres), llamados oficinas centrales, con los MSCs de todo el mundo [d_SWAC] [TERC].

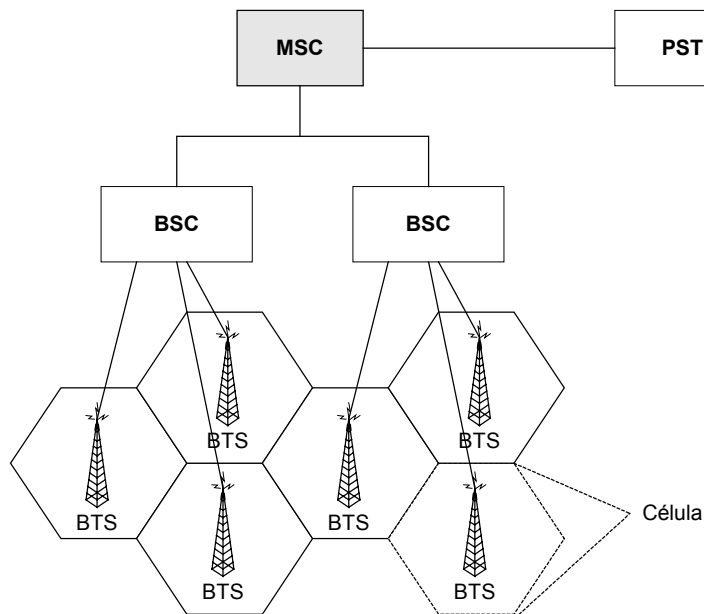


Figura 25. Arquitectura básica de un sistema telefónico celular.

5.2. Historia de la telefonía celular

El primer dispositivo de comunicación inalámbrico apareció en la década de 1920, con el primer radioteléfono, en Michigan en los Estados Unidos de América [e_WRCH]. De manera comercial el inicio de la historia de la telefonía celular es marcado por la empresa AT&T que introdujo el primer servicio telefónico móvil en los Estados Unidos el 17 de junio de 1946 en San Luis, Missouri. El sistema operaba con 6 canales en la banda de 150 MHz con un espacio entre canales de 60 KHz y una antena muy potente. Este sistema se utilizó para interconectar usuarios móviles, que usualmente eran autos, con la red telefónica pública, permitiendo así, llamadas entre estaciones fijas y usuarios móviles. Estos sistemas telefónicos móviles se basaban en una transmisión en el espectro de la Frecuencia Modulada (FM) [e_HTCE].

Sin embargo, después de haber utilizados teléfonos móviles que trabajaban en FM, no fue hasta principios de los 70s, que se empezó con la tecnología basada en células, dando lugar así al primer sistema de telefonía celular.

Martin Cooper fue el pionero en esta tecnología, a él se le considera como "el padre de la telefonía celular" al introducir el primer radioteléfono basado en células en 1973 en los Estados Unidos mientras trabajaba para Motorola; pero no fue hasta 1979 en que aparece el primer sistema comercial en Tokio Japón por la compañía NTT (Nippon Telegraph & Telephone). Apartir de ese momento empezó el auge de la telefonía celular como una alternativa más confiable y de mayor alcance como una alternativa a la telefonía convencional inalámbrica. Como respuesta a la nueva invención, algunos países del mundo crearon sus propios sistemas de comunicación de telefonía celular como lo fueron: Estados Unidos, Japón, Países Nórdicos e Inglaterra.

En 1978, en Estados Unidos, comenzó a operar AMPS (Advanced Mobile Phone System), el cual era un sistema celular analógico. Posteriormente, a todos los sistemas analógicos de telefonía celular constituirían la primera generación de telefonía celular (1G). El desarrollo de AMPS fue muy rápido, un sistema comenzó a operar en mayo de 1978 en Arabia Saudita, en Tokio en diciembre de 1979 y en México en 1981.

En Europa, el sistema celular Telefonía Móvil Nórdico o Nordic Mobile Telephone System (NMT450) inició operaciones en Dinamarca, Suecia, Finlandia y Noruega en el rango de 450 MHz. En 1985 la Gran Bretaña empezó a usar TACS en la banda de 900 MHz. Más tarde, Alemania Occidental implementó C-Netz; Francia, el Radiocom 2000; e Italia el RTMS (Radio Telephone Mobile System). Todos ellos ayudaron a que hubiera nueve sistemas incompatibles, a diferencia de los Estados Unidos que no tenían este problema. Desde aquí se pensó en un plan para crear un sistema digital único para Europa.

A finales de los 80's el interés emergió hacia los sistemas celulares de tipo digital, donde ambos, la voz y el control fueran digitales, lo que sentaría las bases para lo que posteriormente fuera llamada la segunda generación de la telefonía celular (2G).

En 1989 surge GSM (Global System for Mobile Communications) primero conocido como Grupo Especial Móvil y luego como Sistema Global para Comunicaciones Móviles. Lo más destacado de este sistema es que unifica los sistemas europeos, además que rápidamente GSM trajo consigo una nueva gama de servicios, tales como: SMS (Short Message Service).

En 1990, el sistema celular en Estados Unidos, agregó una nueva característica, el tráfico de la voz se convirtió en digital. Esto triplicó la capacidad con el muestreo, digitalización y multi-canalización de las conversaciones.

Para 1991, el servicio celular digital comenzó a emerger reduciendo el costo de las comunicaciones inalámbricas y mejorando la capacidad de manejar llamadas de los sistemas celulares analógicos. Con ello, dio origen a la aparición de D-AMPS (Digital AMPS) el cual es una evolución de AMPS en una versión digital, este sistema fue diseñado para ser compatible con AMPS, lo que a su vez dio origen a la dualidad de los teléfonos celulares, los cuales podían funcionar en un ambiente digital como analógico. En Japón, sucedió algo parecido, evolucionando de su sistema analógico JTACS (Japan Total Access Communications System) al sistema digital PDC (Personal Digital Cellular).

La digitalización de los sistemas de telefonía celular tendría por resultado aprovechar el mismo espectro en cuanto al numero de subscriptores, surgiendo métodos de multiplexación de señales, tales como: TDMA (Time Division Multiple Access) y CDMA (Code Division Multiple Access), entre otros; prueba de ello es que sistemas como: GSM, D-AMPS y PDC están basados en TDMA, y IS-95a en CDMA, posteriormente IS-95a fue nombrado como CDMAOne [EGWC].

La generación 2.5 (2.5G), nació con la adición de tecnologías que permitieran la transferencia de datos a través del protocolo WAP (Wireless Application Protocol), para acceso a Internet, las más importantes son: HSCSD y GPRS.

El inicio de la Tercera generación (3G) consistió en superar las limitaciones técnicas de las tecnologías precedentes. La tercera generación es tipificada por la convergencia de la voz y datos con acceso inalámbrico a Internet con tasas de transferencia superiores a las tecnologías de la anterior generación, así como un mayor ancho de banda y el uso de aplicaciones multimedia a gran escala, entre otras.

5.3. Generaciones de la telefonía celular

5.3.1. Primera generación de la telefonía celular

La primera generación de telefonía celular (1G), inicio aproximadamente en 1983 y la conforman los sistemas analógicos, los cuales eran utilizados para la comunicación por voz, la gran deficiencia de estos sistemas era la vulnerabilidad respecto a la interferencia que sufrían. Otra sería desventaja es que eran sumamente inseguros, ya que alguien que quisiera intervenir alguna llamada podría hacerlo, a través de un sintonizador de radio. Las tecnologías más populares en esta generación fueron: AMPS en Estados Unidos, JTACS en Japón y TACS en el Reino Unido, las cuales operaban en bandas de frecuencia entre 400Mhz y 970Mhz [e_HTCE] [EGWC].

5.3.2. Segunda generación de la telefonía celular

La segunda generación de telefonía celular (2G), se inicio aproximadamente hasta 1992 y a diferencia de la primera se caracterizó por ser digital. La generación 2G utiliza protocolos de codificación más sofisticados y son los sistemas de telefonía celular usados inclusive en la actualidad.

Las tecnologías principales en esta generación son: GSM, IS-95a y PDC. Sin embargo, la más predominante es GSM, ya que para finales del año 2000 GSM había cubierto toda Europa y también se había instalado en Asia, Australia, Estados Unidos y parte de Latinoamérica.

Los protocolos empleados en los sistemas 2G soportan velocidades de transmisión de información más altas para voz pero limitados en transferencia de datos. Se pueden ofrecer servicios auxiliares tales como: *roaming*, redireccionamiento de llamadas, envío de mensajes a múltiples usuarios, recepción de fax y SMS, entre otros.

Además que la mayoría de los protocolos de segunda generación ofrecen diferentes niveles de encriptación para una mayor seguridad. A todos los servicios desarrollados en esta generación son conocidas también como PCS (Personal Communications Services), aunque este concepto también es usado para los sistemas 2.5G [e_HTCE] [EGWC].

5.3.2.1. Arquitectura de sistemas 2G GSM

La arquitectura GSM consta de múltiple nodos, como se puede apreciar en la figura 26, el funcionamiento general se describirá en los siguientes párrafos [TERC] [e_GSMA] [e_GSMR].

Se tiene una estación móvil (MS) que está formada por el equipo móvil (ME), es decir una terminal GSM y por el Modulo de Identificación del Subscriptor (SIM), que es una pequeña tarjeta dotada de memoria y microprocesador (smartcard), que permite identificar al abonado independientemente del terminal usado; y por tanto la posibilidad de continuar recibiendo y efectuando llamadas y utilizar todos los servicios subscriptos insertando la tarjeta SIM también en un terminal que no sea la propia.

El MS transmite los datos necesarios a la estación base (BTS) más cercano, comúnmente llamada célula; en frecuencias determinadas por GSM, las cuales se muestran en la tabla 5.

Tipo GSM	Frecuencia de envío (MHz)	Frecuencia de recepción (MHz)
GSM 450	450.4 – 457.6 o 478.0 – 486.0	460.4 – 467.6 488.4 – 496.0
GSM 900	880.0 – 915.0	925.0 – 960.0
GSM 1800	1710 – 1785	1805 – 1880
GSM 1900	1850 – 1910	1930 – 1990

Tabla 5. Frecuencias GSM [EGWC].

El BTS es el encargado de alojar todos los receptores transmisores que sirven una celda y que se interesan por recibir y enviar información al canal radio, abasteciendo una interfaz física entre la MS y el BSC. le hace llegar la información a un controlador de estaciones base (BSC), el cual puede soportar varias BTSs.

El BSC es el encargado de gobernar los recursos radio para una o más BTS, controlando la conexión entre las BTS y las MSC (centrales de conmutación que proporcionan la conexión a la red física y a otras redes), y además gestionando los canales radio y la señal, entre otras cosas. Cabe señalar que el subsistema de estaciones base (BSS), lo conforman un BTS y un BSC

A su vez, un BSC se conecta con un centro de conmutación móvil (MSC), el cual es el encargado de establecer, gestionar y abrir conexiones, así como de enrutar las llamadas a la célula correcta, el MSC proporciona la interfaz con red pública de

conmutación telefónica (PSTN), además de las funcionalidades que serán detalladas a continuación..

El registro de ubicación de origen (HLR), se lleva a cabo cuando un usuario suscribe un nuevo abono a la red GSM, todas las informaciones para su identificación se memorizan en la HLR. Y tiene la función de comunicar al VLR, que posteriormente se describirá, algunos datos relativos a los abonados, en el momento en que estos se desplazan desde un región a otra.

El registro de ubicación del visitante (VLR) es una base de datos que memoriza de modo temporal los datos de todos los abonados que se encuentran en un área geográfica bajo su control. Estos datos se piden al HLR perteneciente al abonado. En general para simplificar las señalizaciones requeridas y la estructura del sistema, los fabricantes implementan el VLR y el MSC juntos, de modo que el área geográfica controlada por el MSC corresponde a la controlada por el VLR.

El centro de autenticación o validación (AC) es una función del sistema que se ocupa de verificar si el servicio ha sido solicitado por un abonado legítimo, proporcionando ya sea los códigos para la autenticación como la clave contenida en la SIM, para proteger tanto al abonado como al operador de red, de intrusiones del sistema por parte de terceros y evitar clonaciones (dos teléfonos celulares con una misma línea telefónica).

El registro de identidad de equipo (EIR) es una base de datos en la que se verifica si un ME está autorizado o no para acceder al sistema

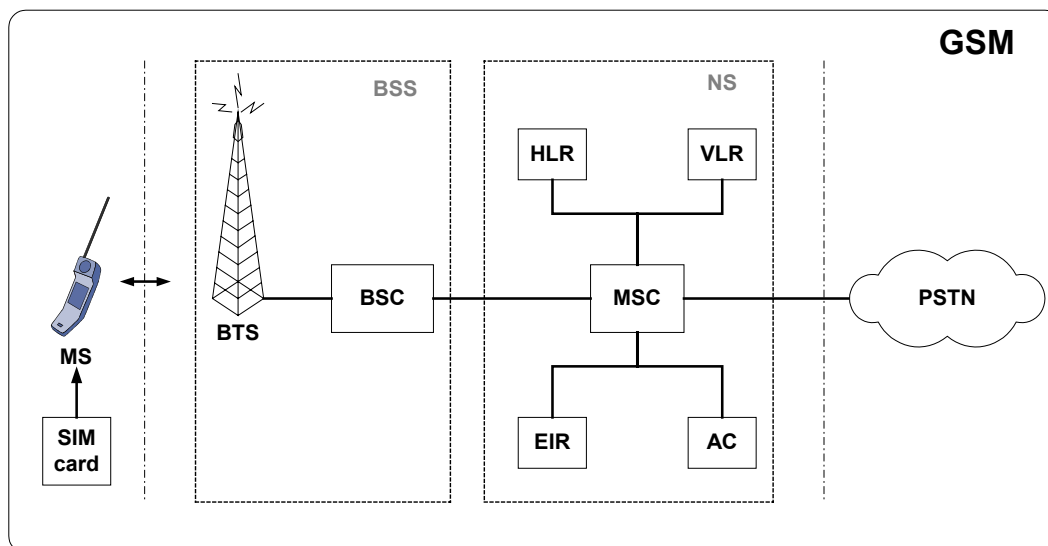


Figura 26. Arquitectura GSM.

5.3.3. La generación 2.5 de la telefonía celular

La generación 2.5G, ofrece características extendidas para ofrecer capacidades adicionales como transferencias de datos y navegación, a través de Internet, con base en el protocolo WAP. Las tecnologías desarrolladas para la transferencia de datos son: HSCSD (High Speed Circuit Switched Data) y GPRS (General Packet Radio System) que permiten una mayor tasa de transferencia de datos, tanto HSCSD como GPRS son considerados como una evolución de GSM, siendo el más importante actualmente GPRS [EGWC].

Es importante destacar, que en esta generación surge el protocolo WAP, estandarizado a partir de 1998, permitiendo a los teléfonos celulares navegar a través de Internet de una manera más eficiente. El elemento de importante de la arquitectura WAP es el Gateway WAP, el cual es el encargado de las transformaciones del protocolo HTTP a WAP. Así como HTTP define el lenguaje HTML, WAP define el lenguaje WML. Con el uso de WAP se hace mas eficiente el acceso de Internet, por parte de los teléfonos celulares.

Para fines de este proyecto se detallará la arquitectura GPRS.

5.3.3.1. Arquitectura de sistemas 2.5G GPRS

De todos los escenarios de acceso móvil a Internet es GPRS, dentro de la generación 2.5G. GPRS fue diseñado bajo el concepto de empaquetamiento de datos, como opción para la transferencia de datos para la red de circuitos conmutados (CSD) de GSM [e_GSMW].

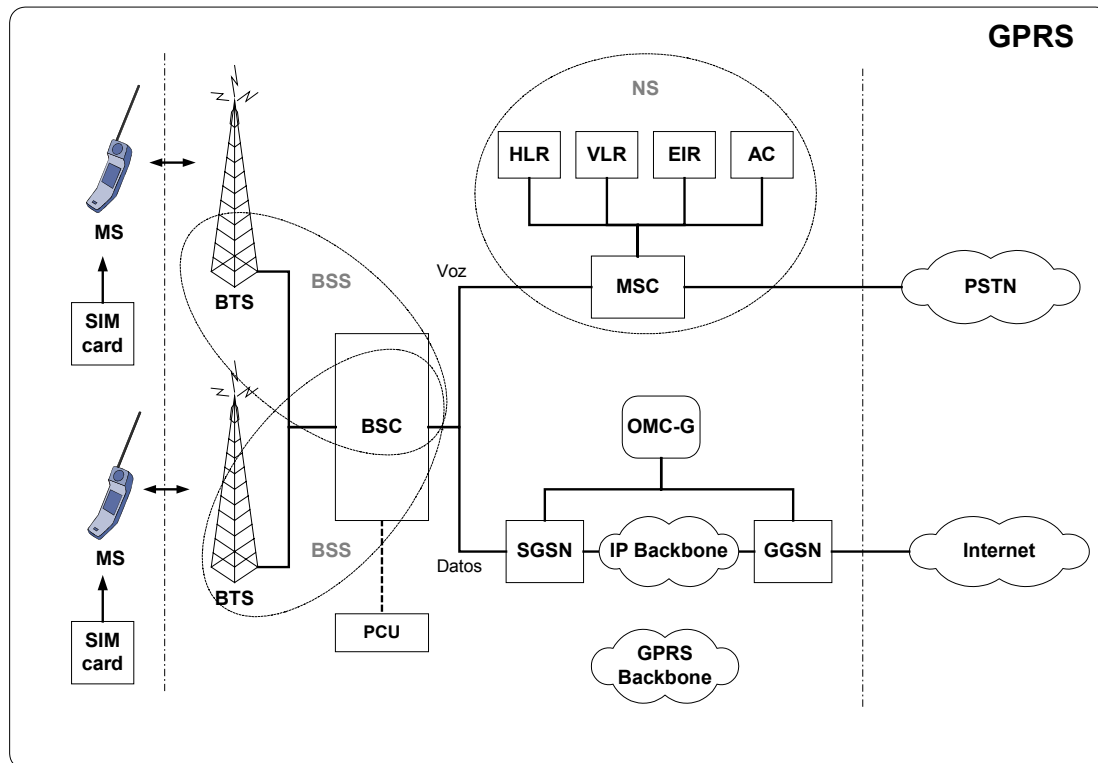


Figura 27. Arquitectura GPRS.

En la figura 27, muestra que la arquitectura GPRS, puede verse como la integración de la arquitectura GSM y elementos para el manejo y transferencia de datos, por lo que la funcionalidad de GPRS no afecta los servicios de GSM existentes. El funcionamiento de la arquitectura GPRS se describe en los siguientes párrafos [e_GARC] [e_GARP] [d_GPIP] [EGWC].

Básicamente el proceso que sigue del MS al BSC en GPRS es el mismo mencionado en la sección 5.3.2.1 de la arquitectura GSM, pero a diferencia de GSM, el BSC es encargado de separar los diferentes flujos de información: voz y datos, así como direccionarlos al respectivo componente, por lo que ahora se ha agregado el componente llamado unidad de control de paquetes (PCU), encargado de separar los tipos de datos.

Para el caso de GPRS, los datos son direccionados al nodo de soporte de servicio GPRS (SGSN), el cual tiene una función similar al MSC en una red GSM, el SGSN es una interfaz importante, ya que es donde la red de radiofrecuencia se “ateriza”, el SGSN enrutar el paquete de datos del MS, entre otras cosas y el nodo de soporte gateway GPRS (GGSN), el cual es una interfaz con otras redes GPRS o hacia redes basadas en el protocolos X.25 o TCP/IP; además de tener la capacidad de

enrutar los datos provenientes del MS hacia su destino y viceversa, por lo que se puede decir que su funcionamiento es parecido al de un gateway, un firewall y un ruteador.

Hay que señalar también que, las funciones del VLR, HLR, ERL y AC, se siguen llevando a cabo en GPRS, los cuales son accedidos a través del sistema de señalización SS7.

El núcleo de la red GPRS, consta de dos nodos principales: el SGSN y el GGSN, los cuales son conocidos en conjunto como nodos GSN (GPRS Support Nodes). La conexión entre diferentes nodos GSN y otros componentes de la red se le conoce como **GPRS Backbone**, dentro del GPRS backbone, son utilizados los protocolos TCP/IP, lo que permite direccionar y procesar paquetes y conexiones de este tipo, lo que se conoce como **IP Backbone**.

A continuación, en la figura 28, se muestra un mapeo de la pila de protocolos utilizados en GPRS respecto al modelo ISO/OSI [e_GARP].

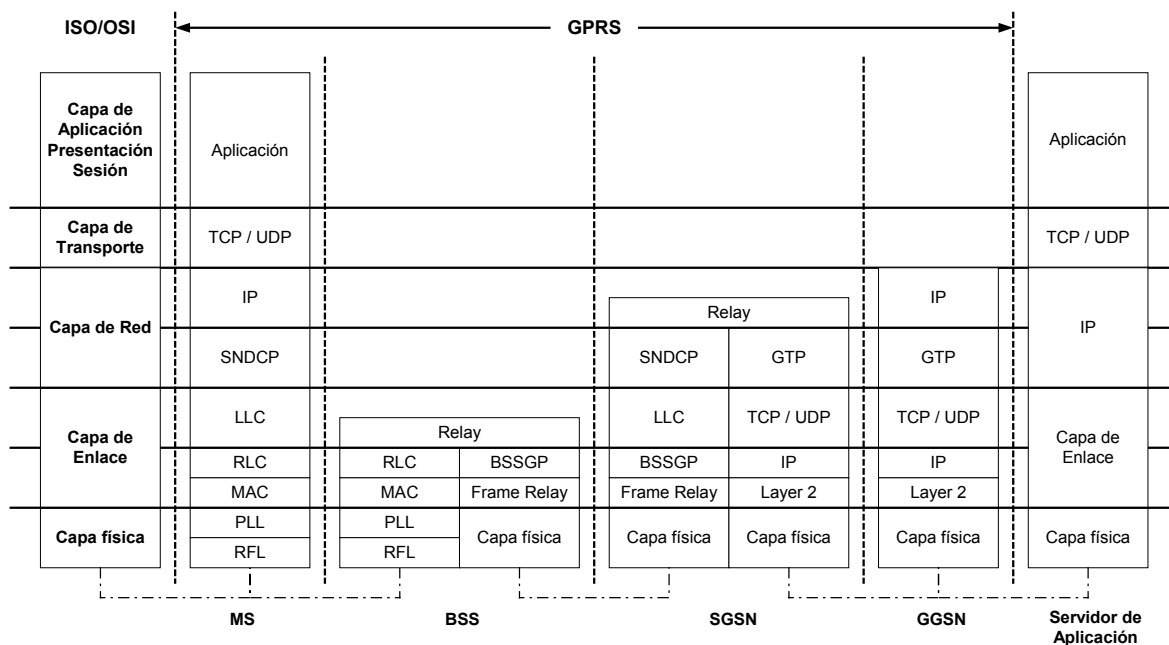


Figura 28. Protocolos GPRS [e_GARP].

Con el establecimiento de GPRS, trajo consigo la creación y acceso a nuevos servicios, como lo son [e_GSMW]:

- **Chat:** Se puede participar en grupos de chat existentes en Internet (distribución de un mensaje a un grupo de personas).
- **Información textual y visual:**
 - Envío y recepción de mensajes que tengan incluidos archivos de audio y/o imagen.
 - Envío y recepción de archivos de audio e imagen sobre la misma red GPRS.
 - Envío y recepción de archivos de imágenes animadas.
- **Documentos compartidos:** dos o mas personas puedan trabajar sobre el mismo documento al mismo tiempo.
- **Correo electrónico corporativo:** acceso a una cuenta de correo electrónico residente en una LAN corporativa.
- **Correo electrónico en Internet:** acceso a una cuenta de correo electrónico residente en un servidor en Internet.
- **Acceso remoto LAN:** acceso a información contenida en una red corporativa, a través de diferentes protocolos de comunicación como: FTP, TELNET, HTTP o Java (J2ME).

5.3.4. Tercera generación de la telefonía celular

El propósito de la Tercera generación (3G) consiste en superar las limitaciones técnicas de las tecnologías precedentes. Los sistemas 3G ofertan un acceso permanente a internet, al igual que GPRS, pero con mayor ancho de banda, videoconferencia y una mayor calidad de la voz en las llamadas telefónicas con calidad muy parecida a un CD de audio, sólo por nombrar algunos.

Entre las tecnologías contendientes de la tercera generación se encuentran: UMTS (Universal Mobile Telephone Service), W-CDMA (Wideband-CDMA, CDMA2000, EDGE (Enhanced Data rates for GSM Evolution), 3GSM (Third generation services GSM), entre otras. El impulso de los estándares para la convergencia de servicios de los sistemas 3G está siendo apoyado por la Unión Internacional de Telecomunicaciones (ITU), esta organización desarrollo lo que hoy se le conoce como Telecomunicaciones Móviles Internacionales para el año 2000 (IMT-2000), ya que fue planeada desde al año 1992; para definir las características, habilidades y

funcionamiento de los sistemas 3G. IMT-2000 engloba a los sistemas: W-CDMA, CDMA2000 y EDGE [EGWC] [e_EWC].

A razón de los desarrollos de las nuevas tecnologías, algunas han tratado ser compatibles con otras, es el caso de UMTS que es una versión de W-CDMA para tener compatibilidad con GSM [EGWC], así como 3GSM que define una gama de servicios GSM que pueden se accedidos a través de las interfaces de W-CDMA [e_GSMW].

En la siguiente tabla comparativa, se muestran las principales características de las tres generación de telefonía celular.

Características	Generaciones de telefonía celular			
	1G	2G	2.5G	3G
Tipo de transmisión	Análoga	Digital	Digital	Digital
Tipo de multiplexación	FDMA	TDMA CDMA	TDMA CDMA	TDMA, W-CDMA
Principales funciones de transmisión	Voz	Voz y Datos	Voz y Datos	Voz y Datos
Principales sistemas	AMPS	GSM D-AMPS CDMA	HSCSD GPRS	W-CDMA UMTS CDMA2000
Tasas de transferencia de datos (Kbps)	Hasta 9.6	9.6 – 14.4	14.4 – 115.2	144 – 2000
Espectro (Mhz)	450 – 900	450 -- 1990	890 – 1990	1885 -- 2200
Acceso a Internet	No	No	Si	Si

Tabla 6. Comparativa de las tres generaciones de telefonía celular.

5.4. Situación actual de la tecnología móvil en el mundo

El mercado de la telefonía móvil se encuentra en un momento de crecimiento espectacular. Por otro lado, a medida que aumenta el número de clientes, decrece progresivamente el consumo por línea, debido a la incorporación de clientes más preocupados por el coste del servicio. Al mismo tiempo, este crecimiento en la red móvil provoca un flujo positivo sobre la red fija al producir un incremento del tráfico de la red fija hacia la móvil.

Los sistemas analógicos de telefonía han disminuido el número de abonados, en contraparte, el número de abonados a sistemas globales de telefonía celular lo han incrementado, tal es el caso de GSM, por lo pronto en Latinoamérica creció un 37%, del año 2001 al año 2002 [e_WCCM] [e_3GST].

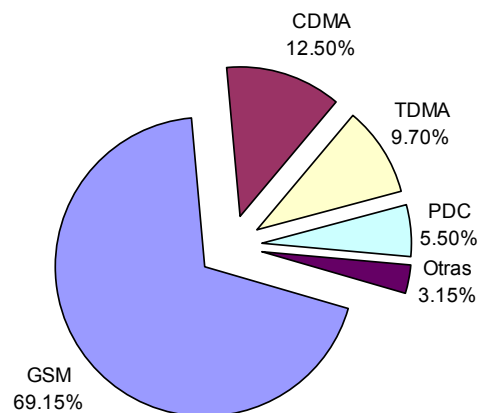


Figura 29. Distribución mundial de los sistemas de telefonía celular (Septiembre 2002) [e_3GST].

En países como Canadá, Inglaterra, Australia y Japón, no solo han globalizados sus sistemas, sino que han implantado UMTS, con gran éxito, aunque otros tantos países ya se cuenta con esta tecnología de manera experimental. Muchos de los proveedores de servicios de telecomunicaciones llamados *carriers*, han evolucionado sus sistemas 2G a sistemas 2.5G antes de entrar masivamente a 3G, tal es el caso de Latinoamérica [e_CNIW].

5.5. Situación actual de la tecnología móvil en México

En México, hasta el año de 1984 cuando la secretaría de comunicaciones y transportes convocó la introducción de la telefonía celular en nuestro país en las nueve diferentes regiones en que fue dividido la empresa "Radiomóvil Dipsa S.A. de C.V." obtiene la concesión para explotar la red de servicio radiotelefónico móvil en el área metropolitana de la Ciudad de México, operando en las bandas radiofónicas de 450-470 y 470-512 MHz.. En 1989, nace una nueva empresa de telefonía celular, Iusacell, que actualmente es denominada como "Iusacell PCS, S.A. de C.V.", siendo la primera compañía de telefonía celular en ofrecer el servicio en la Ciudad de México. Fue hasta 1990, que surge la marca Telcel y fue entonces cuando la misma, ofrece los servicios de telefonía celular en la ciudad de Tijuana B.C. A partir de 1990 Telcel y Iusacell expanden los servicios de telefonía celular en el Distrito Federal y su zona metropolitana y paulatinamente ofrecen el servicio a escala nacional[e_HTCE].

Actualmente, existen varios proveedores de telefonía celular y de servicios inalámbricos, ejemplo de ellos son: IUSACELL, TELCEL, UNEFON, PEGASO y Telefónica Movistar [e_COFE].

Los principales servicios ofrecidos por estas compañías son:

IUSACELL [e_IUSA]

- Identificador de llamadas
- Acceso a Internet
- Mensajes escritos de 2 vías
- Acceso a una cuenta de correo electrónico
- RED EXPRESS 3G: servicio de transmisión de datos de alta velocidad con anchos de banda de hasta 144 Kbps

TELCEL[e_TELC] [e_TGSM]

- Identificador de llamadas
- Acceso a Internet
- Mensajes escritos de 2vías
- Recepción de mensajes escritos a través de Internet
- Llamada en espera
- Portal Datum
- Multiconferencia
- Buzón de voz
- CDPD
- GSM
- HSCSD
- GPRS

UNEFON [e_UNEF]

- Identificador de llamadas
- Recepción de mensajes escritos a través de Internet
- Mensajes escritos de 2vías
- Buzón de voz
- Red privada móvil

- Transferencia de llamadas
- Acceso a Internet
- Mensajes de 2 vías
- Recepción de mensajes escritos a través de Internet

PEGASO [e_PEGA]

- Conferencia tripartita
- Establecer un límite de llamadas
- Llamada en espera

Telefónica MoviStar [e_TMST]

- Buzón de voz
- Recepción de mensajes escritos a través de Internet
- Identificador de Llamadas

A continuación se presentan las tecnologías utilizadas por cada proveedor de servicios:

Proveedor de servicios	Sistema	Frecuencia (Mhz)
IUSACELL	AMPS, CDMA	800 y 900
TELCEL	AMPS, TDMA, GSM, HSCSD, GPRS	800 y 1900
UNEFON	CDMA	1900
PEGASO	CDMA	1900
Telefónica MoviStar	CDMA GSM	800 1900

Tabla 7. Redes celulares de México [e_CNIW].

CAPÍTULO 6. La tecnología J2ME

6.1. Introducción

Como resultado de una explosión tecnológica, Java es una de las mas importantes, actualmente es la plataforma de soporte para otras tecnologías, por lo que en la actualidad es poco realista concebir Java como un simple lenguaje de programación; por el contrario, resulta más conveniente considerarlo como un conjunto de tecnologías diseminadas en todos los ámbitos y plataformas de computo, con dos elementos en común:

- El código fuente en lenguaje Java es compilado a código intermedio interpretado por una JVM.
- Todas las tecnologías comparten un conjunto más o menos amplio de APIs básicas del lenguaje.

En el mundo de los dispositivos electrónicos, como teléfonos móviles, PDAs o dispositivos embebidos, la situación es caótica, debido a que las plataformas en consideración son fuertemente disimilares en cuanto a capacidad. Han surgido en el pasado numerosas iniciativas, con frecuencia incompatibles entre sí, para introducir Java en el desarrollo de aplicaciones para estos dispositivos.

En los últimos tiempos, Sun Microsystems ha reconocido esta diversidad de ámbitos tecnológicos en los que Java se desenvuelve, y los ha agrupado en tres grandes entornos de aplicación, o *ediciones*, por seguir la terminología de la empresa [e_JATR]:

- J2EE (Java 2 Platform, Enterprise Edition), orientada al desarrollo de servicios web y aplicaciones distribuidas mediante EJB (Enterprise Java Beans) y tecnologías relacionadas, con APIs para la gestión de transacciones, persistencia de objetos, servicios de nombres, XML, autenticación, etc.
- J2SE (Java 2 Platform, Standard Edition), que abarca las APIs orientadas a la programación de aplicaciones de usuario final: interfaz gráfica de usuario, multimedia y conectividad. Esta edición es la que en cierta forma recoge la iniciativa original del lenguaje Java.

- J2ME (Java 2 Platform, Micro Edition), enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes.

Aunque también se podría hablar de Java Card, la cual es una plataforma dirigida hacia las tarjetas con tecnología inteligente, las cuales tienen integrada un chip, en donde reside un ambiente Java reducido, incluida una maquina virtual de Java llamada Card VM.

La siguiente figura ilustra gráficamente los diferentes ámbitos tecnológicos para los entornos de aplicación.



Figura 30. Entornos de aplicación Java.

Sólo de forma muy simplista puede considerarse que J2ME y J2EE son versiones reducida y ampliada, respectivamente de J2SE, en realidad, aunque cada una de las ediciones enfoca ámbitos de aplicación radicalmente distintos e independientes entre sí.

Se considera que para el año 2006 el 40% de las PDAs y el 68% de los teléfonos móviles (celulares) soportarán la tecnología Java [d_YGIM].

6.2. Objetivos

J2ME esta dirigido hacia “micro-dispositivos” limitados en capacidad de procesamiento, memoria y despliegue de datos, muchos de ellos no tienen alguna opción de almacenar e instalar software, mas allá del que fue configurado en el proceso de manufactura, tales como: PDAs y los teléfonos celulares. Con la introducción de J2ME, los “micro-dispositivos” dejaron de ser “estáticos” por naturaleza, por lo que ahora podremos acceder a características inherentes del lenguaje y plataforma Java.

Java 2 Micro Edition, fue introducido a principios del año 2000 con el objetivo de que los micro-dispositivos puedan utilizar y explotar las propiedades de la plataforma Java y que estaban fuera del alcance de las plataformas J2SE y J2EE, por lo que el concepto y uso de una única plataforma Java no sería del todo adecuada.

6.3. Arquitectura J2ME

Al contrario que en otras tecnologías orientadas a las computadoras personales, en J2ME la gama de dispositivos considerados, varía enormemente en cuanto a capacidad de procesamiento, memoria y despliegue gráfico de información. Ante la imposibilidad de establecer una arquitectura común que se adecue a esta variedad de entornos hardware, J2ME define una serie de componentes a partir de los cuales se construye una implantación concreta para un dispositivo determinado [CJMT], estos componentes se agrupan en los siguientes tipos:

- Máquinas virtuales Java
- Configuraciones
- Perfiles
- Paquetes opcionales

Un entorno de ejecución de J2ME se compone entonces, de una selección de máquina virtual, una configuración y un perfil. En la figura 31, se muestran las relaciones entre los distintos componentes de J2ME.

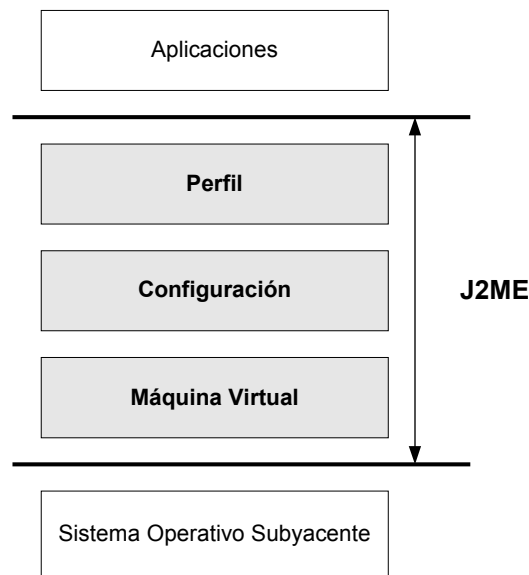


Figura 31. Arquitectura J2ME.

Antes de definir cualquier tipo de componente, es importante iniciar con las configuraciones definidas por J2ME, ya que es la base para el desarrollo de los demás componentes.

6.3.1. Configuraciones

Para dar soporte a la amplia gama de productos que están dentro del alcance de J2ME, se introdujo el concepto de configuración.

Una Configuración define una plataforma Java para un rango de dispositivos, a su vez, la configuración esta directamente relacionada con una Máquina Virtual Java (JVM) y define las características del lenguaje Java y las librerías de la JVM para una configuración en particular, en otras palabras también podemos decir que una configuración es un conjunto de APIs básicas de Java que definen un entorno generalizado de ejecución [CJMT].

Una configuración no está orientada a ningún campo específico de aplicación, sino que más bien trata de agrupar los dispositivos en cuanto a sus capacidades computacionales y entorno genérico de operación, por lo que la configuración esta ligada a la capacidad de procesamiento, memoria, despliegue de información, conectividad de red (tipo de red a la cual el dispositivo puede ser conectado).

Existen dos tipos de configuraciones, con respecto a las características de los dispositivos [J2NT]:

- **Connected Device Configuration (CDC)**
- **Connected Limited Device Configuration (CLCD)**

6.3.1.1. Connected Device Configuration

La configuración J2ME conocida como Connected Device Configuration (CDC) está dirigida hacia dispositivos con un cierto grado de capacidad computacional, con las siguientes características [CJMT]:

- 512 kilobytes (mínimo) de memoria ROM para correr Java
- 256 kilobytes (mínimo) de memoria RAM para el *runtime* de Java
- Capacidad de conectividad de red, posibilidad de persistencia
- Conectividad en red con un ancho de banda grande

Los dispositivos típicos a los que la CDC se orienta son:

- Decodificadores de Televisión Digital
- Teléfonos con navegación de Internet
- Sistemas de navegación para coches (telemática)

En cuanto al tipo de maquina virtual que utiliza la configuración CDC es, en términos generales sobre una JVM estándar; no obstante, las implementaciones de referencia, desarrolladas para los sistemas Linux y VxWorks, utilizan la Máquina Virtual CVM, la cual sera descrita brevemente en la sección 6.3.3. Cabe señalar que, la configuración CDC está basada en J2SE v1.3, e incluye varios paquetes Java de dicha edición [J2NT].

6.3.1.2. Connected Limited Device Configuration

La configuración J2ME conocida como Connected Limited Device Configuration (CLDC) está orientada a dispositivos con un bajo grado de capacidad computacional, de acuerdo a los siguientes características [CJMT]:

- 128 kilobytes (mínimo) de memoria ROM para correr Java
- 32 kilobytes (mínimo) de memoria RAM para el *runtime* de Java
- Interfaz de usuario restringida
- Capacidad de conectividad de red, generalmente para redes móviles
- Para un ancho de banda pequeño

Entre los dispositivos a los que la CLDC se orienta tenemos:

- PDAs de bajas prestaciones (tipo Palm)
- Teléfonos móviles GSM y 2.5G (GPRS)
- Pagers
- Terminales de transacciones electrónicas

Con el objeto de adaptarse a las restrictivas capacidades de este tipo de dispositivos, la CLDC no opera sobre una JVM estándar, sino en la Máquina Virtual KVM.

En cuanto a la APIs incluida, parte de ella proviene de la edición estándar de Java (J2SE v.1.3.) basadas en las librerías como: `java.lang`, `java.util` y `java.io`, además de incluir clases propias de CLDC como `javax.microedition.io`. Todas las APIs de la CLDC están también contenidas en la CDC [CJMT].

La relación entre todas las APIs entre J2SE, CDC, CLDC se muestra en la siguiente figura:

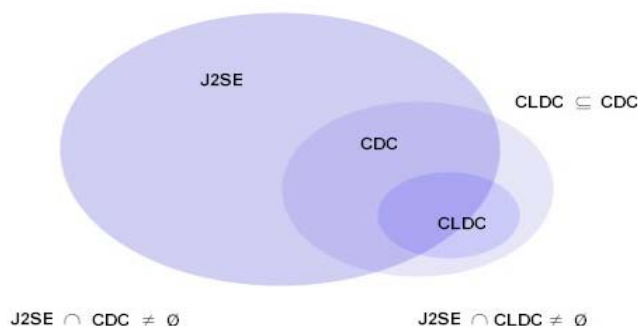


Figura 32. Relación entre las APIs de J2SE, CDC y CLDC.

El ámbito de CLDC conjuntamente con KVM, cubre los siguientes puntos:

- Máquina virtual y soporte al lenguaje Java
- Modelo de seguridad
- Entrada/Salida
- Soporte a conexiones de red.
- Internacionalización

El ámbito de CLDC conjuntamente con KVM, no cubre los siguientes puntos:

- Instalación y gestión del ciclo de vida de las aplicaciones Java.
- Interfaz de usuario.
- Gestión de eventos.
- Modelo de aplicación a alto nivel.
- Soporte a almacenamiento persistente.

Las diferencias de CLDC/KVM respecto a J2SE, son las siguientes:

- No soporta datos punto flotante (float).
- No soporta finalización.
- Limitaciones en el manejo de errores.
- No soporta Java Native Interface (JNI).
- No soporta reflexión de clases.
- No soporta cargadores de clase.
- No soporta grupos de hilos.
- Verificación de código en dos fases: estándar y preverificación.

Las dos versiones mas recientes de CLDC, son las versiones 1.0.3 y 1.0.4, cada versión ha traído consigo evoluciones importantes en la definición de la configuración CLDC. Para fines de este proyecto se detalla en dichas versiones.

Las principales características de la versión 1.0.3., respecto a las anteriores, son:

- Optimizaciones de rendimiento respecto a: sincronización de operaciones, compactación de código, manejo de cadenas.
- Aumento de librerías para eventos de sistema y librerías para conectividad de red, para una mejor adaptación con la referencia de implementación del perfil MIDP versión 1.0.
- Aumento del nivel de la interfaz de ejecución paso a paso y del colector de basura.
- Alto rendimiento en la implementación del JAM (Java Application Manager).
- Tamaño de memoria de la KVM ajustable por línea de comando.

Las principales características de la versión 1.0.4., respecto a la anterior, son:

- Soporte de la Interface Nativa K
- Mayor grado de portabilidad en la implementación del verificador runtime.
- Preverificador compatible con la versión 1.4 del J2SE.
- Adaptación y compatibilidad con el perfil MIDP en su versión 2.0.

Sun Microsystems proporciona una implementación de CLDC de referencia para las plataformas: Windows y Linux [e_CLDC].

Para cada uno de los perfiles CDC y CLDC, existen APIs como paquetes opcionales que tienen por objetivo extender la plataforma J2ME [e_J2ME].

6.3.2. Perfiles

Para poder orientar las capacidades de los dispositivos y proveer de mayor flexibilidad a los cambios tecnológicos, se introdujo el concepto de perfil a la plataforma J2ME.

Un perfil es una extensión de una configuración, ya que provee de librerías específicas para desarrollar aplicaciones para un tipo determinado de dispositivo, en otras palabras, es un conjunto de APIs orientadas a un ámbito de aplicación determinado, para una configuración J2ME dada. Los perfiles identifican un grupo de dispositivos por la funcionalidad que proveen (por ejemplo, teléfonos móviles, electrodomésticos) y el tipo de aplicaciones que típicamente correrán sobre ellos [CJMT].

La librería para implementar una interfaz gráfica es generalmente una componente muy importante de la definición de un perfil. Las interfaces gráficas difieren mucho entre distintos perfiles (pantallas textuales en teléfonos móviles, interfaces basadas en *touchscreen* para PDAs, etc.), y en algunos perfiles pueden no existir (sistemas embebidos, por ejemplo).

Un perfil siempre se define a partir de una configuración dada. Así, cabe pensar en un perfil como un conjunto de APIs que dotan a una configuración J2ME de funcionalidad específica para una familia de dispositivos concreta y para un tipo de aplicaciones determinado. Algunos perfiles son [J2NT]:

Para CDC:

- **Foundation Profile:** Perfil básico para dispositivos CDC, es la base para otros perfiles.
- **Personal Profile:** Agrega la funcionalidad de interfaz gráfica de usuario al Foundation Profile..
- **RMI Profile:** Agrega las librerías de J2SE respecto a la Invocación Remota de Métodos (RMI) al Foundation Profile.

Para CLDC:

- **Personal Digital Assistant Profile (PDAP)**, perfil desarrollado para las PDAs, es muy similar a MIDP, solo que esta dirigido a PDAs que tienen una mejor pantalla y mayor memoria.
- **Mobile Information Device Profile (MIDP)**, es un perfil desarrollado para las capacidades limitadas de dispositivos móviles tales como: teléfonos celulares, *paggers* interactivos y PDAs.

Para fines de éste proyecto de tesis, se procederá a detallar el perfil MIDP.

6.3.2.1. Mobile Information Device Profile

El MIDP esta basado en CLDC/KVM, heredando consigo las características respecto a los dispositivos inmersos en la configuración CLDC, pero éste perfil se orienta específicamente a dispositivos con las siguientes características [CJMT]:

- Reducida potencia computacional y memoria, teniendo como mínimo: 128KB ROM, 32 KB RAM y al menos 8KB para almacenar información persistente de las aplicaciones.
- Conectividad limitada (como la provista por las redes celulares actuales, entre 9600 bps y 14000bps, como CDPD y GPRS).
- Capacidades gráficas muy reducidas (una pantalla al menos de 96×54 pixeles monocromática o a color).
- Entrada de datos alfanuméricos reducida.
- Soporte para conectividad móvil.

Los tipos de dispositivos que mejor se adaptan entonces al MIDP son teléfonos celulares de la generación 2.5G y 3G, mencionados en el anterior capítulo.

El MIDP proporciona un API que soporta la siguiente funcionalidad:

- Una forma limitada de almacenamiento de datos
- Conectividad basada en HTTP 1.1.
- Timers
- Presentación de imágenes de baja resolución
- Entrada de datos de usuario
- Un modelo rudimentario de ciclo de vida de las aplicaciones: MIDlet

El API para este perfil contiene paquetes tales como: `javax.microedition.rms` (persistencia), `javax.microedition.midlet` (ciclo de vida), y `javax.microedition.lcdui` (interfaz de usuario).

Sun Microsystems proporciona una implementación de referencia basada en un emulador de MIDP para las plataformas: Windows y Linux, además de un entorno completo de desarrollo para este mismo sistema operativo [e_MIDP].

Actualmente existen dos versiones del perfil MIDP: 1.0 y 2.0.

La versión MIDP 1.0, básicamente fue descrita en esta sección, aunque tiene una restricción importante respecto a la conectividad en red. Dentro del API, de esta versión se define una referencia de implementación para sockets: TCP y UDP, que pueden ser probados en el entorno de aplicación [d_MIR1], aunque de manera real, MIDP no implementa las clases de implementación para sockets, por el motivo de que no todos los dispositivos móviles soportaban este tipo de conexión [e_MIDS].

La versión MIDP 2.0, agrega las siguientes funcionalidades a la versión anterior [d_MIR2]:

- Reproducción de archivos de audio: WAV y MIDI, de bajo nivel.
- Seguridad a las suites MIDlet para protegerse de MIDlets no confiables.
- Un API de juegos, que permite un mejor desarrollo de los juegos.
- Componentes y características adicionales LCDUI.
- Protocolos de red, en esta versión son implementados protocolos como: UDP, TCP, SSL, entre otros.
- Push, habilidad para iniciar un MIDlet como respuesta a un mensaje enviados a través de PAP (Push Access Protocol).

6.3.3. Maquinas virtuales Java

En la plataforma Java, el motor que da soporte a cualquier aplicación Java es la Máquina Virtual de Java (JVM). Una máquina virtual de Java (JVM, Java Virtual Machine) es un programa encargado de traducir código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma correspondiente, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente.

Las implementaciones tradicionales de JVMs son en general muy pesadas en cuanto a memoria ocupada (footprint) y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos, que en algunos casos se apartan de la especificación estándar relajando algunas exigencias con el fin de obtener implementaciones más ligeras. En la figura 33, podemos observar la generación de un archivo ejecutable java y su manera de ejecutarse[J2PR] [J2PG].

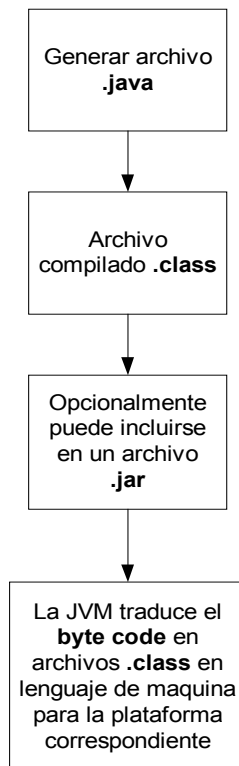


Figura 33. Ciclo para la ejecución de una aplicación Java

La JVM es también responsable de proveer seguridad, asignar y liberar memoria y manejar los hilos de ejecución.

Para la configuración CLCD, se ha desarrollado una maquina virtual llamada Maquina Virtual K (KVM, K Virtual Machine) y para la configuración CDC, se desarrolló la Maquina Virtual C (CVM, C Virtual Machine).

6.3.3.1. KVM

La Máquina Virtual K KVM (la *K* viene de *kilobyte*, haciendo referencia a la baja ocupación en memoria) se desarrolló en el programa de investigación *Spotless* de Sun Microsystems, dedicado a la implementación de un sistema Java para el Organizador Palm. Se trata de una implementación de referencia para una JVM reducida especialmente orientada a dispositivos con bajas capacidades computacionales y memoria limitada. En concreto, la KVM presenta las siguientes limitaciones (entre otras) con respecto a la especificación estándar de la JVM [CJMT]:

- No hay soporte para tipos de punto flotante (float y double).
- No hay soporte para métodos nativos (Java Native Interface, JNI).
- No se soporta la reflexión de clases.
- No se soporta la finalización de objetos.
- Tratamiento limitado de excepciones.

Además, el sistema de seguridad de la JVM ha sido simplificado y modificado para ajustarse a limitaciones computacionales y de memoria más estrictas.

Es claro que la KVM no es la tradicional JVM, y tiene las siguientes características:

- Requiere de 40 a 80 kilobytes de memoria.
- Requiere de 20 a 40 kilobytes de memoria dinámica (heap).
- Puede correr en procesadores de 16 bits con una frecuencia de 25 MHz.

La KVM está escrita en lenguaje C y se encuentra disponible para las siguientes plataformas:

- Palm OS
- Windows
- Solaris

Hay que señalar que para las computadoras de bolsillo (Pocket PC), Java provee un runtime llamado PersonalJava.

6.3.3.2. CVM

La Máquina Virtual C (CVM) es una implementación de referencia escrita en lenguaje C de la máquina virtual de Java (JVM) estándar, orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2 MB o más de memoria RAM. Según Sun Microsystems, la CVM está diseñada de forma que su portabilidad a distintos sistemas operativos y/o procesadores sea relativamente fácil, lo cual debe ayudar a extender las configuraciones J2ME basadas en una JVM estándar (no reducida). La ocupación en memoria (*footprint*) de la CVM está en torno a los 256 KB [J2NT].

En la actualidad, CVM tan sólo se encuentra disponible para el sistema operativo Linux sobre procesadores Intel x86.

La CVM ha sido tomada como JVM de referencia en la definición de la CDC

6.3.4. Estandarización y compatibilidad de componentes

La intencionada indefinición en cuanto a especificación y selección de componentes de la arquitectura J2ME podría dar lugar a la proliferación de numerosos entornos de ejecución incompatibles y de no mediar alguna clase de esfuerzo coordinado de estandarización en el seno de la industria de los dispositivos electrónicos.

Con el fin de involucrar a la industria software y hardware en la evolución y el soporte de la tecnología Java, Sun ha creado el programa Java Community Process (JCP), un foro de estandarización con aproximadamente 300 participantes entre particulares, instituciones académicas y empresas de software, electrónica, telefonía móvil, comunicaciones, etc. El trabajo del programa JCP se articula en torno a la producción y aprobación de documentos estándar conocidos como JSRs (Java Specification Requests) [e_JACP].

Uno de los campos de actividad del programa JCP es la especificación de componentes de J2ME, con el objetivo de llegar a un consenso amplio en torno a un número razonablemente pequeño de entornos de ejecución de J2ME agrupados por mercados de aplicación definidos, como el de las PDAs o el de la telefonía móvil. De los esfuerzos del programa JCP, han surgido ya componentes estandarizados de J2ME, con diferentes grados de aceptación e implantación en la industria. Ante la multiplicidad de entornos de ejecución que presenta J2ME, el antiguo lema “Write Once, Run Anywhere”, es decir, “Construye una vez, ejecuta en donde quiera”, por su significado

en ingles, asociado al lenguaje Java no parece muy aplicable en el mundo de los dispositivos electrónicos. Esto difícilmente puede verse como una debilidad de la plataforma J2ME, teniendo en cuenta la enorme variedad de dispositivos y ámbitos de aplicación que esta tecnología abarca.

CAPÍTULO 7. El análisis

7.1. Jini, tecnología de integración de sistemas distribuidos

Con base en lo que se definió en el capítulo 4, un sistema Jini puede verse como un sistema distribuido basado en federaciones de grupos de usuarios, recursos y servicios, a partir de este concepto se debe analizar en que tipo de sistemas distribuidos, es posible implementar Jini.

El análisis parte de la arquitectura de Jini, en donde se puede apreciar que Jini provee un conjunto de componentes de software y un conjunto de protocolos, que forman una capa entre el sistema operativo y la aplicaciones, facilitando la conexión e interacción entre elementos del sistema distribuido (clientes y proveedores de servicios), a este concepto (capas intermedias) se le conoce como middleware, entonces un sistema Jini se puede ver como un sistema distribuido basado en un middleware. En la figura 34, se muestra de manera grafica, un sistema distribuido basado en Jini:

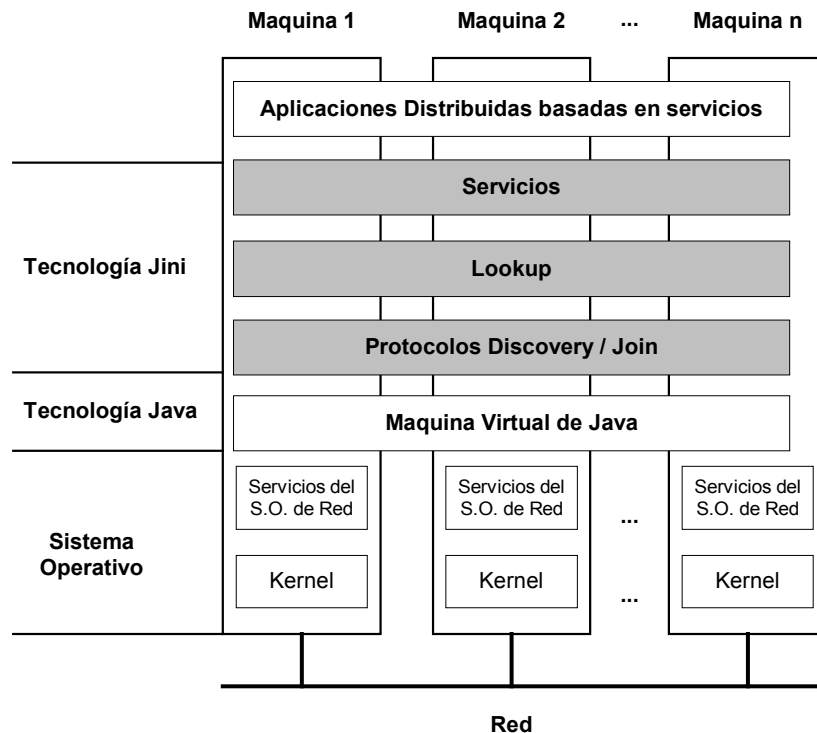


Figura 34. Sistema distribuido basado en Jini.

El middleware de Jini, esta basado en el paradigma orientado a objetos, ya que al estar basado en Java y éste ser desarrollado con base en éste paradigma, el middleware hereda esta característica.

Jini, también es considerada como una tecnología para la integración de elementos de un sistemas distribuido, por lo tanto debe cumplir de manera general con las siguientes aspectos de un sistema distribuido:

Respecto a la Transparencia:

1. **Transparencia en el acceso:** Esconde el acceso del usuario a los servicios, mediante un el uso de una interfaz de usuario.
2. **Transparencia en localización:** Esconde los detalles de localización de los servicios disponibles, a través del lookup service.
3. **Transparencia en migración:** Permite que los servicios residan en ubicaciones diferentes, sin modificar el nombre del proxy asociado al servicio, esto es, solo se tiene que registrar al proxy desde la nueva ubicación del proveedor de servicio. el cual es registrado en el lookup service.
4. **Transparencia en concurrencia:** Permite que cierto numero de usuarios puedan acceder al mismo tiempo a los servicios disponibles, sin que esto interfiera en el trabajo de cada uno, para ello se hacen instancias del servicio.

Respecto a la Flexibilidad:

De manera general, Jini es soportado por sistemas operativos de núcleo monolítico.

Respecto a la Fiabilidad:

1. **Tolerancia a fallos:** En este aspecto, Jini, provee un manejador de transacciones y de persistencia, el cual se puede soportar fallos en el acceso e implementación del servicio.
2. **Seguridad:** En Jini, cada servicio puede implementarse niveles de seguridad con respecto a lectura y escritura de datos, acceso a servicios, persistencia de los servicios y transacciones. Aunque el sistema operativo, también provee de herramientas de seguridad, como lo pueden ser los permisos de escritura y lectura de archivos.

3. **Administración del sistema:** Jini ofrece flexibilidad en cuanto a la configuración se refiere; esto quiere decir que puede ser reconfigurado para agregar y eliminar el número de servicios disponibles y poder modificarlos, sin que esto afecte a otras aplicaciones y a los usuarios.

Respecto a la escalabilidad:

Permite que la federación Jini pueda incrementar o disminuir su tamaño según se requiera, con respecto al número de grupos de servicios, el número de servicios y el número de entidades pertenecientes, sin la necesidad de cambiar la estructura del sistema. En Jini, existe un componente centralizado como lo es el lookup service, este elemento es la “puerta de entrada al sistema”, ya que permite de manera general al cliente identificar los servicios disponibles que se encuentran en el sistema y poder comunicarse con el proveedor del servicio. De esta manera el grado de escalabilidad depende del diseño de la federación Jini con respecto a los lookup services utilizados dentro de la federación.

El desarrollo de un sistema distribuido requiere el uso de funciones específicas, como lo son:

1. **Servicios de Comunicación.** La intención es tener comunicación entre los elementos de la red Jini, para lo cual se pueden utilizar:
 - a. **Modelos de interacción:** en este caso Jini, hace uso del modelo cliente/servidor a múltiples niveles (es el caso de la arquitectura Jini Surrogate), la comunicación en grupo al implementar el protocolo multicast y el uso del lookup service como intermediario entre el proveedor de servicio y el cliente.
 - b. **Tecnologías de comunicación:** Jini, se basa en la familia de protocolos TCP/IP para llevar a cabo la comunicación entre sus entidades. En la siguiente figura, se muestra un mapeo respecto a los protocolos de las capas de la arquitectura Jini, TCP/IP y ISO/OSI.

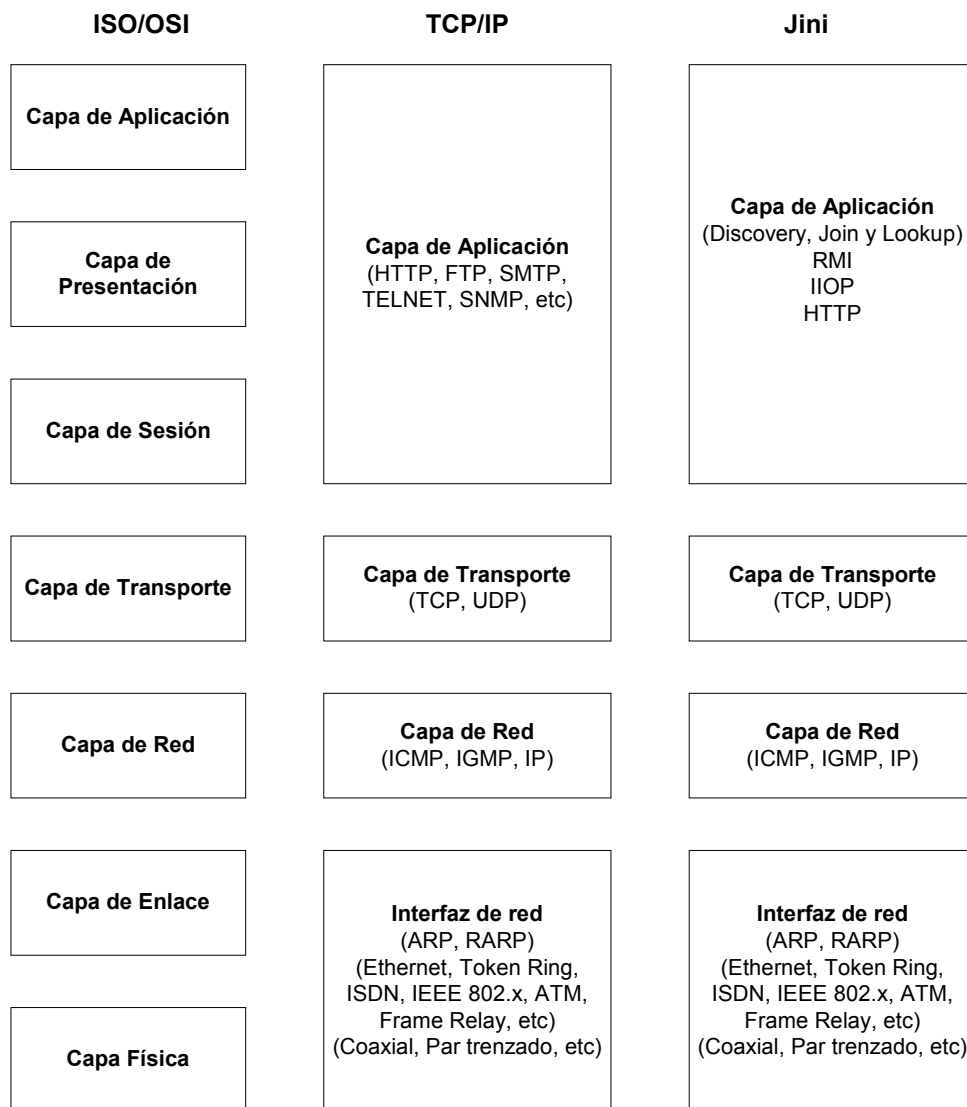


Figura 35. Mapeo de la arquitecturas: Jini, TCP/IP y ISO/OSI

En la figura 35, se puede observar que Jini, tiene las misma capa de interfaz de red que la arquitectura TCP/IP, lo cual hace que la definición de los protocolos para un algún tipo de conexión física, sea indiferente. En la capa de red, utiliza el protocolo IP para direccionar hacia la entidad la información correspondiente, aunque aquí pueden implementarse dos maneras de transmisión: unicast y multicast . En la capa de transporte, se implementan conexiones en la modalidad: TCP o UDP. En la capa de aplicación, son utilizados los protocolos Discovery, Join y Lookup, los cuales estan soportados sobre RMI, IIOP y HTTP. Lo que demuestra un soporte para

implementación de servicios basados en sockets, invocación a métodos remotos (RMI) y CORBA.

2. **Administración y asignación de recursos.** La administración esta determinada por el agrupamiento de recursos y usuarios llamadas federaciones Jini, en cuanto a la asignación de recursos, esta dada por lookup service y el proveedor de servicios, el lookup service tiene la tarea de asignarle al cliente el proveedor de servicio asociado a su solicitud, a su vez el proveedor de servicio debe contar con los suficientes recursos de procesamiento para poder llevar a cabo la interacción con el cliente y la implementación del servicio.
3. **Administración de nombres.** Son resueltos a través del sistema operativo.
4. **Servicios de sincronización:** Son resueltos a través del sistema operativo.
5. **Funciones de seguridad.** Son resueltos por el sistema operativo (permisos de usuario) y por las políticas de seguridad de Java.

Respecto a la clasificación de un sistema distribuido con base al acoplamiento tanto del software y del hardware, el sistema distribuido en el que se puede implementar Jini, es débilmente acoplado en ambos casos, uno de los motivos es que la arquitectura de Jini esta soportada en un sistema operativo de red (véase capítulo 3).

Existen otras tecnologías como la propuesta por Salutation Consortium, Salutation, o la propuesta por Microsoft, UPnP (Universal Plug and Play), para la integración de sistemas distribuidos. Aunque existen varias alternativas, el objetivo de este proyecto es la tecnología Jini, por las siguientes razones:

- La compatibilidad con J2ME, ya que ambas tecnologías están basadas en Java y comparten algunos paquetes del API (java.lang, java.util y java.io), que permiten de una manera más fácil, implementar la interacción y conectividad, entre ambas plataformas.
- Existencia una herramienta disponible de desarrollo (SDK) con licencia gratuita, provista por el mismo desarrollador de la tecnología Jini (Sun Microsystems), para el desarrollo de aplicaciones Jini.
- El uso de la tecnología Jini, para desarrollar servicios fijos, provoca el análisis del comportamiento del dispositivo móvil J2ME en el acceso a dicho tipo de servicios.

En la tabla 1, se hace una comparativa entre Jini, Salutation y UPnP.

Tecnologías	Ventajas	Desventajas
Jini	• Independencia de Plataforma. • Independencia del sistema operativo de red. • No requiere controladores del fabricante del dispositivo para poder ser integrado a la red Jini.	• Alta dependencia de Java • Implementación sobre redes IP. • Implementación de un controlador Java, para el dispositivo.
Salutation	• Independencia del lenguaje • Puede operar en cualquier tipo de red. • No requiere controladores del fabricante del dispositivo para poder ser integrado al sistema	• Elaboración de interfaces para los dispositivo integrado a la red.
UPnP	• Uso de protocolos estandares TCP/IP	• Alta dependencia de plataforma. • Alta dependencia del sistema operativo. • Implementación sobre redes IP

Tabla 8. Ventajas y desventajas de Jini, Salutation y UPnP [d_SJUP].

7.2. Jini y J2ME

Como ya se vio en el capítulo 2, las especificaciones de J2ME, clasifican a los dispositivos móviles que integran esta tecnología en configuraciones y perfiles.

En este proyecto se ha escogido la configuración CLDC v.1.0.4 y el perfil MIDP v.2.0. (véase capítulo 6), por los siguientes motivos:

- La configuración CLDC y el perfil MIDP fueron definidos para dispositivos móviles, y en específico para teléfonos celulares 2.5G, que actualmente ya cuentan con la suficiente capacidad de procesamiento para ejecutar aplicaciones J2ME: MIDlet (definidas en el perfil MIDP).

- Existen las herramientas disponibles de software (SDKs) con licencia libre, provistas por el mismo desarrollador de la tecnología J2ME (Sun Microsystems), para desarrollar aplicaciones MIDlet.

Una vez, que se ha delimitado el tipo de dispositivos con los cuales se enfocará el desarrollo de este proyecto, como lo son los teléfonos celulares 2.5G, se analizará la manera en la que este tipo de dispositivos pueden acceder a los servicios de una red Jini.

La integración de los dispositivos móviles J2ME a un sistema Jini, puede ser establecida tomando como base la arquitectura Jini Surrogate (descrita en el capítulo 4), como se puede apreciar en la figura 36.

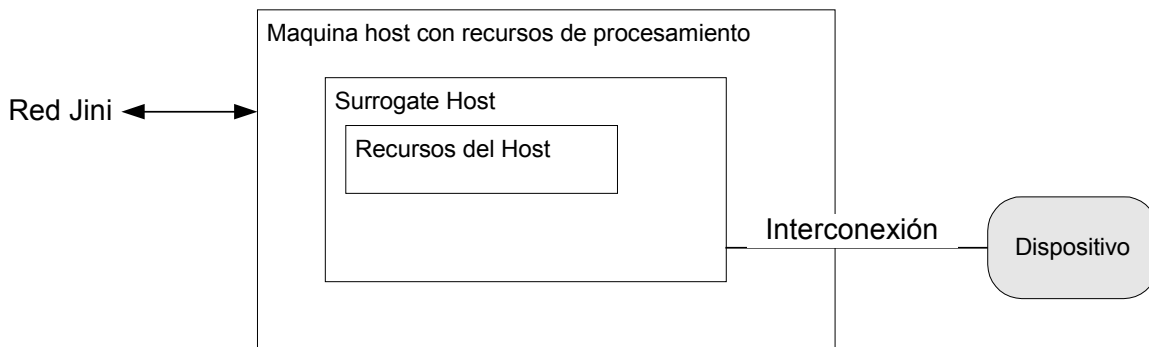


Figura 36. Arquitectura Jini Surrogate.

En ésta arquitectura se definen los componentes, las reglas y protocolos, a través de los cuales un dispositivo, que no necesariamente debe ser móvil – es decir que simplemente no cumpla con los supuestos y requerimientos de Jini (mencionados en el capítulo 4) – puede integrarse al sistema o red Jini y poder acceder a los servicios disponibles que ahí se encuentran.

La idea principal de la arquitectura Surrogate, es establecer una maquina host que pueda proveer de recursos al dispositivo móvil, basándose en los requerimientos de Jini, como el de que todos los elementos de la red Jini cuenten con una JVM estándar, en el caso del dispositivo móvil, la maquina virtual de Java con la que cuenta no es una JVM sino es la KVM, la diferencia fundamental es la capacidad que tiene cada una respecto a la interpretación de instrucciones o comandos (véase capítulo 6).

Una de las partes importantes en la arquitectura Jini Surrogate es establecer una interconexión entre el dispositivo y la maquina host. Por lo que hay que identificar el tipo

de protocolos de interconexión que ofrece el perfil MIDP, a través de los cuales se puede realizar un intercambio de información con la maquina host, la cual permitirá al dispositivo móvil acceder a los servicios provistos dentro de la red Jini. Los protocolos implementados en el perfil MIDP, se enuncian a continuación [CJMT]:

- Conexión a través del protocolo HTTP.
- Conexión a través de los protocolos: TCP y UDP (sockets).

Es importante señalar que en el momento de analizar las especificaciones de las herramientas de software para el desarrollo de este proyecto, no había sido liberada la versión 2.0 de MIDP, ni la versión 1.0.4. de CLDC. Aunque, MIDP v.1.0., no implementaba realmente la conexión por sockets en los dispositivos que cuentan con la tecnología J2ME, ya que en ese período de tiempo no todos los dispositivos móviles J2ME soportaban ese tipo de conexión, existían las referencias de implementación en dicha versión de MIDP, que podrían ser habilitadas para analizar el comportamiento del dispositivo respecto a la implementación de sockets en la herramienta de desarrollo de aplicaciones J2ME, incluida en la especificación del perfil MIDP. En la versión 2.0 de MIDP, oficialmente se puede implementar la conexión por socket de manera real, es decir que cualquier dispositivo móvil J2ME que soporte la versión 2.0. inherentemente soporta el tipo de conexión por sockets [d_MIR1] [d_MIR2]. De hecho, no existe una diferencia significativa entre ambas versiones, respecto al uso de sockets, ya que en las dos versiones se pueden deshabilitar o habilitar la implementación de sockets.

La razón más importante para pensar que el dispositivo móvil J2ME puede acceder a los servicios disponibles en la red Jini, se basa en que la arquitectura Jini esta soportada sobre los protocolos TCP/IP, en otras palabras esta basada sobre una arquitectura TCP/IP, por lo que es correcto pensar que cualquier elemento de la red Jini (en este caso particular, el surrogate host), puede interactuar con un dispositivo móvil J2ME definido en el perfil MIDP, a través de los protocolos: HTTP, TCP o UDP (véase figura 35).

La análisis comparativo entre el uso de los protocolos: HTTP, TCP y UDP para realizar una interconexión, se basa en un primer plano en que el protocolo utilizado permita llevar un seguimiento del estado de la conexión entre el servidor y el cliente, en este caso entre la maquina host y el dispositivo móvil. Otro criterio, es que el protocolo utilizado asegure que los paquetes de datos enviados entre el cliente y el servidor sean recibidos. Por lo que la mejor opción es utilizar el protocolo TCP. El tipo de interconexión definida con base en el protocolo TCP son los sockets, sabiendo que la implementación de sockets TCP, puede ser habilitada en la primera versión de MIDP de manera experimental, por lo cual no será un obstáculo ya que posteriormente, en la versión 2.0 de MIDP la implementación de sockets TCP, es soportada de manera general [d_MIR1].

7.3. La tecnología de comunicación móvil

El siguiente paso, es conocer la tecnología de comunicación móvil que debe ser elegida para que un dispositivo móvil J2ME basado en el perfil MIDP, pueda interactuar con el surrogate host.

En el capítulo 5, se describieron las generaciones de telefonía celular, siendo en la generación 2.5G, en donde los dispositivos móviles cuentan con los recursos de procesamiento necesarios para conectarse y poder navegar en Internet, realizar transmisión y recepción de datos, así como desplegar imágenes en varios formatos (PNG, WBMP, GIF) y soportar aplicación J2ME. Basándose en el hecho de que un dispositivo móvil J2ME (teléfono celular 2.5G) puede acceder a Internet y una red Jini esta basada en una arquitectura TCP/IP y puede estar contenida dentro de Internet, el medio por el cual puede interactuar dicho dispositivo con la red Jini es a través de Internet.

Entre las tecnologías que soportan la transmisión y recepción de datos, en la generación 2.5G, están: HSCSD y GPRS; ambas ya disponibles en México [e_TGSM].

Sin embargo, las diferencias más importantes son las siguientes:

- GPRS provee de un mayor ancho de banda que HSCSD. GPRS tiene un ancho de banda de 115.2Kbps y HSCSD de 57.6Kbps [EGWC]. Es importante contar con un ancho de banda razonable para que la descarga de las aplicaciones Java no sea lenta o dificulte el intercambio de información entre el dispositivo móvil y el surrogate host.
- En HSCSD, se emplea el canal de voz para la transmisión y recepción de datos (tipo modem), y en GPRS, se utilizan canales alternos (lo que permite realizar una llamada al mismo tiempo que se puede estar conectado a Internet). Se requiere que el usuario no se vea afectado al estar utilizando una aplicación Java y no poder utilizar su dispositivo móvil para recibir llamadas.
- Dentro de los servicios ofrecidos por GPRS, están: la descarga de aplicaciones Java, acceso a una LAN y transferencia de archivos en formato de imagen, entre otros [e_GSMW].

Todas estas razones son las que tomaron en cuenta para elegir a GPRS como la tecnología de comunicación móvil para la transferencia y recepción de datos, aunque UMTS también fue considerada, pero por la razón que en México, no se encuentra disponible para una futura implementación real, no fue utilizada.

7.4. Arquitectura del Sistema

Una vez analizado, el soporte teórico del proyecto, se propondrá la arquitectura del sistema que será desarrollado para mostrar la interacción Jini-J2ME, así como la forma de acceso a los servicios Jini en un ambiente distribuido en una red de área local, a través de un dispositivo móvil J2ME basado en el perfil MIDP.

7.4.1. Elementos simulados del sistema

Con la finalidad de encontrar los elementos que pueden ser simulados, se ha tomado como base la arquitectura GPRS, la cual se ha dividido en tres subsistemas, como lo muestra la figura 37.

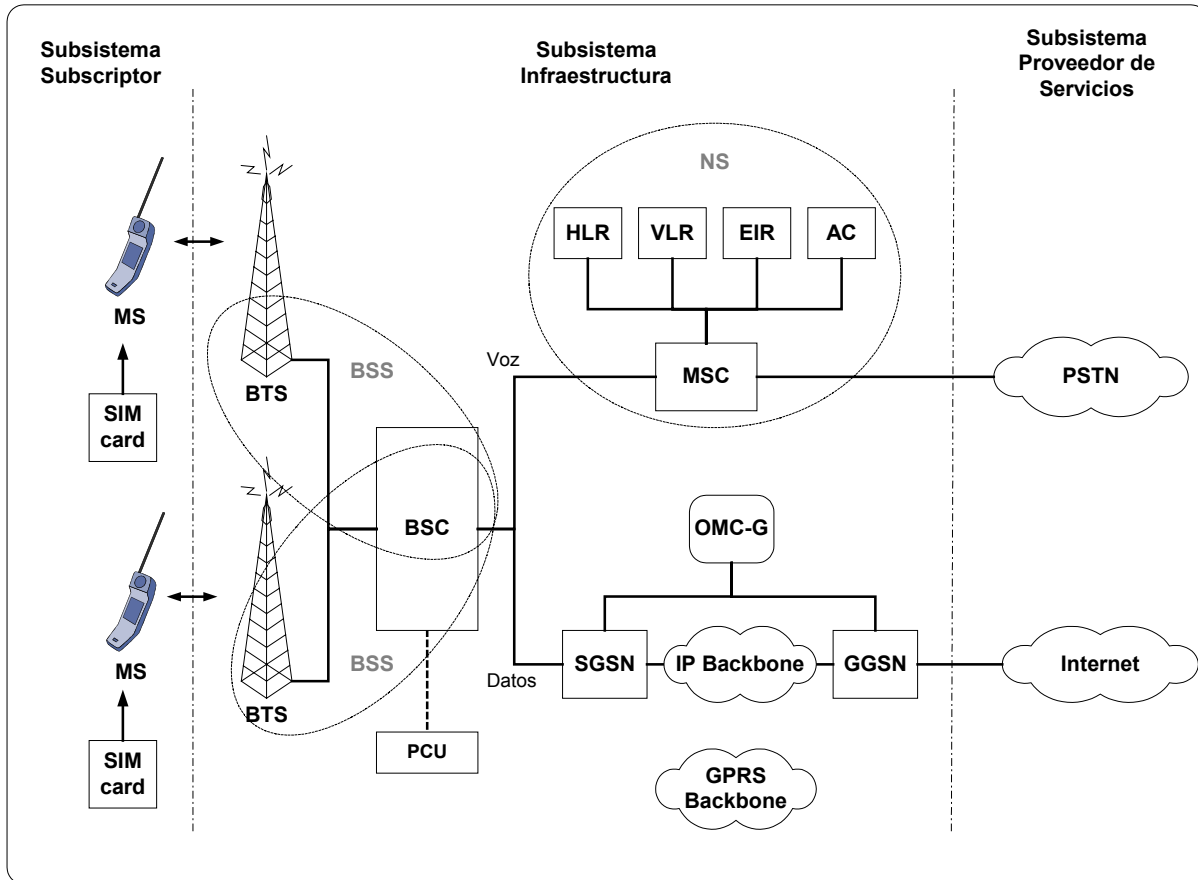


Figura 37. Arquitectura GPRS.

El primer elemento, se encuentra dentro del subsistema subscriptor, este elemento es el dispositivo o estación móvil (MS). El dispositivo móvil (MS) será conceptualizado como el dispositivo móvil J2ME para fines de este proyecto, el cual será emulado, a través de un SDK.

El segundo elemento es el subsistema infraestructura el cual será simulado. Para llegar a la factibilidad de simular este subsistema se analizaron los protocolos de las capas de la arquitectura GPRS, mostrados en el capítulo 5 e identificados en la figura 38.

El tercer elemento es el subsistema proveedor de servicios, el cual consiste en los proveedores de servicios que se encuentran en Internet, entre ellos se puede encontrar una red Jini.

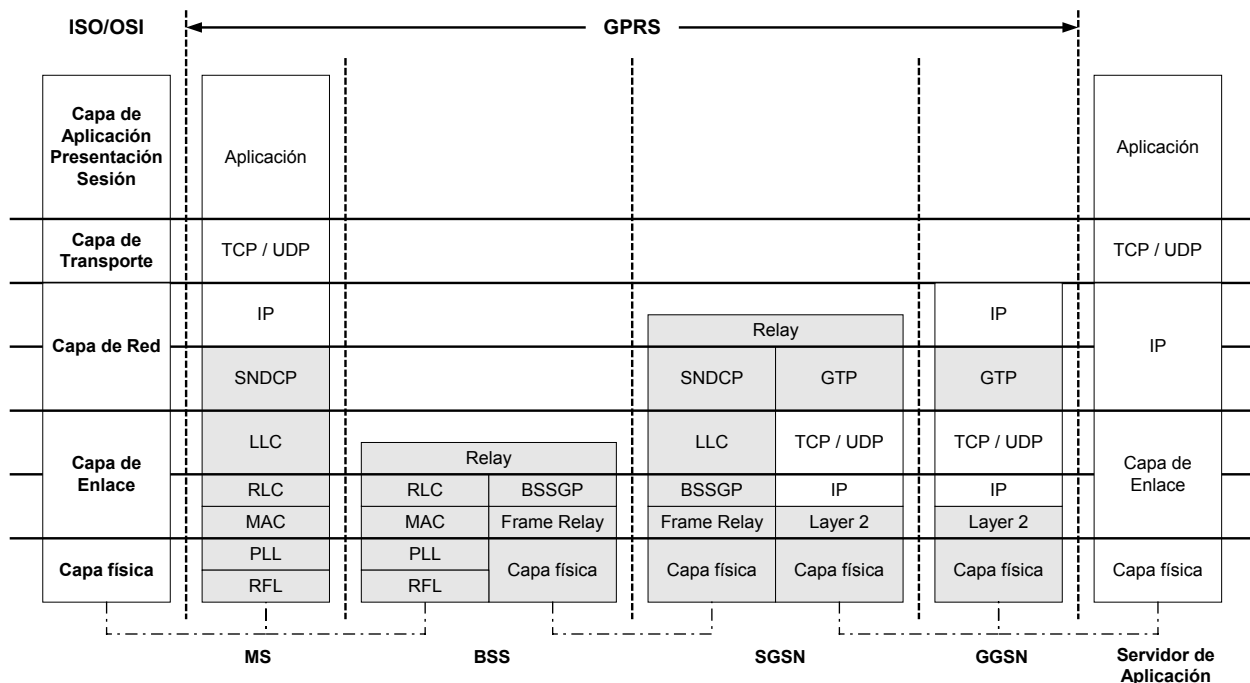


Figura 38. Capas de protocolos de la arquitectura GPRS [e_GARP].

En la figura 38, se muestra que parte del soporte de la arquitectura GPRS, esta basada en los protocolos TCP/IP; se asume que el dispositivo móvil (MS), tiene la capacidad de soportar en la capa de red, el protocolo IP y en la capa de transporte los protocolos TCP o UDP, utilizados por alguna aplicación ya sea por un navegador o por alguna aplicación J2ME en este caso.

Para el caso del navegador de Internet basado en WAP, se debe agregar un elemento después del GGSN y antes del servidor de aplicación, [d_BDWA], pero para fines de este proyecto solo se analiza el caso de la comunicación, con base en los protocolos TCP/IP.

Por último, se deben identificar las funcionalidades de cada uno de los subsistemas y los componentes que están incluidos. El cliente móvil tiene como función almacenar la aplicación J2ME (MIDlet) y poder ejecutarla, así como proveer una interfaz al usuario para poder acceder a los servicios provistos por el el servidor de aplicación. Los elementos BSS, SGSN y GGSN, en su conjunto tienen como función recibir los datos del dispositivo móvil, enrutarlos y dirigirlos hacia otras redes TCP/IP. El servidor de aplicación tiene como función proveer servicios Jini a los clientes, los cuales deben ser previamente identificados, este análisis se ilustra en la figura 39.



Figura 39. Análisis del funcionamiento GPRS con base en TCP/IP.

Una vez, conociendo que de manera general las funciones que se realizan en la arquitectura GPRS con base en los protocolos TCP/IP, la simulación del bloque intermedio de la figura 39, puede ser simulado por un proveedor de servicios de Internet (ISP); solo se contempla utilizar al ISP en las funciones de enrutamiento y resolución de nombres, ya que las funciones de tarificación, señalización y autenticación dentro de la arquitectura GPRS, no son parte de los objetivos de este proyecto de tesis.

7.4.2. Mapeo de la arquitectura del sistema

Ahora, es importante conocer el tipo de protocolos que resuelven la comunicación entre cada subsistema (véase figura 37).

Subsistemas	Protocolos utilizados
Subscriber ↔ Infraestructura	TCP/IP
Infraestructura ↔ Proveedor de servicios	TCP/IP

Tabla 9. Protocolos y lenguajes utilizados en la arquitectura del sistema.

Para la realizar posteriormente la simulación de la arquitectura del sistema real, es necesario reemplazar cada subsistema descrito en la figura 37, por su correspondiente componente. En la siguiente figura, se muestra el mapeo de los subsistemas a los componentes.

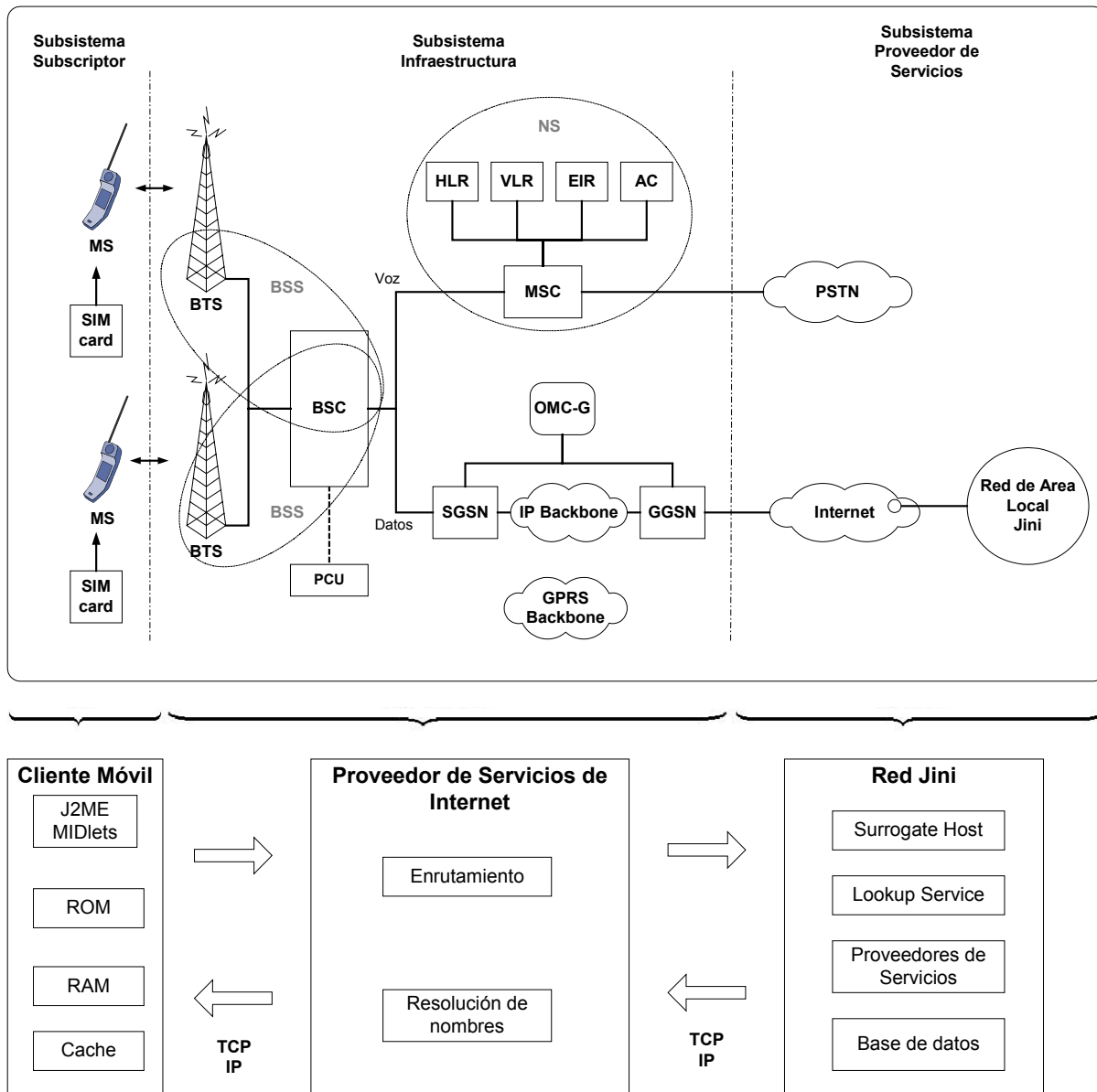


Figura 40. Mapeo de la Arquitectura GPRS a componentes.

Una vez que la arquitectura GPRS ha sido mapeada a componentes, puede ser simulada. Por lo que el siguiente paso es adquirir las herramientas de desarrollo tanto de hardware como de software, para poder realizar la simulación. Un ejemplo de una posible simulación se muestra en la siguiente figura:

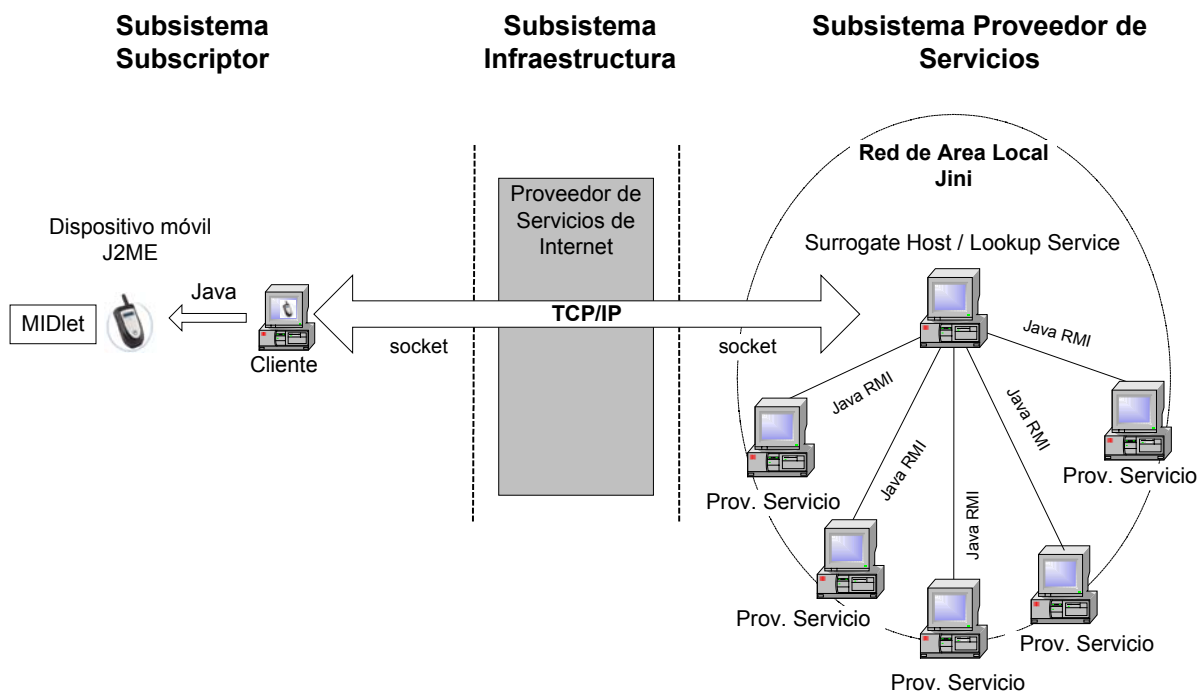


Figura 41. Arquitectura para un sistema simulado Jini-J2ME.

El funcionamiento de cada componente utilizado en la simulación serán descritos a continuación:

Subsistema Subscriptor

De manera general, el cliente es un dispositivo móvil J2ME emulado, a través de un SDK, y en cual se encuentra una aplicación MIDlet, instalada. Por medio de la aplicación MIDlet, el cliente solicita una conexión al surrogate host, por medio de un socket TCP, el surrogate host forma parte de la red de área local, la cual cuenta son servicios Jini. Es por el surrogate host que el dispositivo móvil J2ME, puede acceder a la red Jini e interactuar con los proveedores de los servicios. Toda la información, producto de la implementación del servicio será enviada al dispositivo móvil J2ME, a través del socket TCP, para que el usuario pueda visualizarla, interactuar con base en ella o bien, utilizarla de manera general.

Subsistema Infraestructura

Este subsistema consiste del proveedor de servicios de Internet, el cual tiene las siguientes funciones:

- Permitir la conexión física y lógica, esto es que el cliente pueda conectarse con el proveedor de servicios, para que este pueda proveerlo de los servicios de Internet.
- Enrutamiento, el cual es un método que tiene por objetivo que la información enviada por el dispositivo móvil, llegue hasta el surrogate host y viceversa, así como resolver el camino que debe seguir la información (circuito virtual) e inherentemente de conmutar la información.
- Resolución de nombres, el cual se refiere a mapear el nombre de alto nivel de la maquina a su dirección IP y viceversa, cuando el cliente haga referencia al nombre o dirección IP de la maquina surrogate host.

De forma implícita el proveedor de servicios de Internet, tiene la función de facilitar el acceso del usuario a los servicios de Internet, en cualquier momento.

Subsistema Proveedor de Servicios

La principal función de éste subsistema es integrar los proveedores de una red de área local con base en Jini, dando como resultado una red Jini; así como de proveer de un surrogate host para que el dispositivo móvil J2ME, pueda acceder a los servicios Jini.

Para que el dispositivo móvil, pueda acceder a los servicios Jini, el surrogate host debe tener las siguientes funciones:

- Permitir una conexión vía IP, a través de sockets TCP, con base en una arquitectura TCP/IP.
- Registrar al cliente, a través de un nombre de usuario y una clave de acceso o contraseña.
- Recibir las solicitudes de acceso al servicio Jini, especificado por el cliente.
- Activar la función surrogate, para poder acceder a los servicio Jini.
- Arrendar los recursos de procesamiento al dispositivo móvil, para poder interactuar con los proveedores de servicios Jini.
- Enviar los resultados, producto de la ejecución del servicio Jini.

Los proveedores de servicios Jini, de acuerdo a las especificaciones de Jini, tienen la responsabilidad de proveer un proxy -- en el cual se definen las características, atributos e implementación del servicio -- al lookup service y de registrarlo en el mismo, para que cualquier cliente que desee acceder al servicio, lo pueda hacer a través del proxy.

7.5. Componentes de software existentes

La objetivo fundamental es encontrar SDKs que cuenten con licencia libre y que los requerimientos de software y hardware, no tengan un coste elevado.

7.5.1. Dispositivo Móvil J2ME

Motorola iDEN SDK versión 1.0.0

Las principales características de éste kit de desarrollo [e_MISD], son:

- Cumple con las especificaciones de J2ME, para la configuración CLDC versión 1.0.3 y el perfil MIDP 1.0, pero agrega el soporte de sockets (TCP y UDP).
- Soporta los protocolos: HTTP, HTTPS, UDP, TCP, SSL.
- Permite instalar MIDlets.
- Contiene compilador y verificador para crear archivos jad y jar
- Contiene 3 emuladores de los siguiente modelos de teléfonos motorola: i95cl, i90c, i85s.
- Permite configurar características de la red para simular la latencia de la red y el throughput de la red.
- Permite configurar la memoria disponible para la aplicación.
- Permite monitorear el uso de memoria por parte de la aplicación en ejecución.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 3.

Requerimiento	Especificaciones
Procesador	Pentium 500 MHz
Memoria RAM	128 MB RAM
Espacio en disco duro	12 MB
Sistema Operativo	Microsoft Windows NT Workstation v.4.0 (Service Pack 3), Microsoft Windows 98 o Microsoft Windows2000
Otros	J2SE SDK 1.2.2, J2SE SDK 1.3.1, J2SE SDK 1.4.0 (recomendado)

Tabla 10. Requerimientos de sistema, iDEN SDK versión 1.0.0

Principales desventajas:

- No contiene una lista de bugs o problemas que puedan presentarse en el desarrollo de aplicaciones en este kit de desarrollo.
- No contiene un editor para construir los MIDlets.

Sun Microsystems, J2ME Wireless Toolkit version 1.0.4_01

Las principales características de éste kit de desarrollo [e_SMWT], son:

- Cumple con las especificaciones de J2ME, para la configuración CLDC versión 1.0.3 y el perfil MIDP 1.0.
- Soporta los protocolos: HTTP, TCP, UDP.
- Permite instalar MIDlets.
- Provee de un compilador y un verificador para la creación archivos jad y jar.
- Contiene 3 emuladores definidos por el kit de desarrollo.
- Contiene una lista de bugs o problemas, concernientes al desarrollo de aplicaciones en este kit de desarrollo.
- Soporte para el protocolo Over-the-Air [d_OTAM].
- Permite configurar características de la red para simular la latencia de la red y el throughput de la red.
- Permite configurar la memoria disponible para la aplicación.
- Permite monitorear el uso de memoria por parte de la aplicación en ejecución.
- Permite acelerar el procesamiento de la maquina virtual de java.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 5.

Requerimiento	Especificaciones
Procesador	Pentium 166 MHz
Memoria RAM	64 MB
Espacio en disco duro	30 MB
Sistema Operativo	Microsoft Windows NT v. 4.0. o Microsoft Windows 2000
Otros	J2SE SDK 1.3.0.

Tabla 11. Requerimientos de sistema, J2ME Wireless Toolkit version 1.0.4.

Principal desventaja:

- No contiene un editor para construir los MIDlets.

Sun Microsystems, J2ME Wireless Toolkit version 2.0 beta 2

Las principales características de éste kit de desarrollo [e_SMWT], son:

- Cumple con las especificaciones de J2ME, para la configuración CLDC versión 1.0.4 y el perfil MIDP 2.0.
- Soporta los protocolos: HTTP, HTTPS, UDP, TCP, SSL.
- Permite instalar MIDlets.
- Provee de un compilador y un verificador para la creación archivos jad y jar.
- Contiene 4 emuladores definidos por el kit de desarrollo.
- Contiene una lista de bugs o problemas, concernientes al desarrollo de aplicaciones en este kit de desarrollo.
- Soporte para el protocolo Over-the-Air [d_OTAM].
- Permite configurar características de la red para simular la latencia de la red y el throughput de la red.
- Permite configurar la memoria disponible para la aplicación.
- Permite monitorear el uso de memoria por parte de la aplicación en ejecución.
- Permite acelerar el procesamiento de la maquina virtual de java.
- Provee de un API (MMAPI, Mobile Media API) para la implementación y el manejo multimedia (audio y video).
- Provee de un API (WMA, Wireless Messaging API) para la implementación y el manejo de mensajería SMS y CBS.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 4.

Requerimiento	Especificaciones
Procesador	Pentium166 MHz
Memoria RAM	64 MB RAM
Espacio en disco duro	50 MB
Sistema Operativo	Microsoft Windows XP o Windows 2000
Otros	J2SE SDK 1.4.0.

Tabla 12. Requerimientos de sistema, J2ME Wireless Toolkit version 2.0 beta 1.

Principal desventaja:

- No contiene un editor para construir los MIDlets.

Palm OS Simulator version 5.0

Las principales características de éste kit de desarrollo [e_PALM], son:

- Permite la instalación de un ambiente Java para J2ME, con la configuración CLDC versión 1.0.3 y el perfil MIDP 1.0.
- Soporta los protocolos: HTTP, HTTPS, TCP, UDP, SSL.
- Permite instalar MIDlets (aunque no directamente).
- Permite configurar la memoria disponible para la aplicación.
- Permite la implementación de sockets (TCP y UDP).
- Permite instalar nuevos paquetes de software que incremente su funcionalidad.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 6.

Requerimiento	Especificaciones
Procesador	Intel x86
Memoria RAM	No especificado
Espacio en disco duro	No especificado por el desarrollador Aproximadamente 23 MB
Sistema Operativo	Microsoft Windows NT 4, Microsoft Windows 2000, Microsoft Windows XP, Microsoft Windows 98 Microsoft Windows ME..
Otros	Sun microsystems, midp4palm-1_0

Tabla 13. Requerimientos de sistema, Palm OS Emulator versión 3.4.

Principales desventajas:

- No contiene un editor para construir los MIDlets.
- No contiene ni compilador ni verificador para crear archivos jad y jar.
- Se necesita instalar la maquina virtual de java, definida por J2ME, para poder ejecutar los MIDlets.
- Se necesita una herramienta para convertir archivos del formato jad al formato prc.
- El emulador tiene un pobre funcionamiento, es decir, no ejecuta adecuadamente los MIDlets, algunas veces falla.
- No contiene una lista de bugs o problemas, concernientes a la ejecución de aplicaciones J2ME (MIDlets) en este emulador.

MobileSet versión 3.01

Las principales características de este paquete de desarrollo [e_BJMS], son:

- Provee herramientas para el desarrollo de aplicaciones J2ME.
- Hereda las características de JBuilder, al ser un paquete adicional.
- Provee de plantillas para la construcción de aplicaciones J2ME.
- Contiene una lista de problemas conocidos en el uso de este Kit de desarrollo.
- Provee las archivos de clases y librerías para poder compilar aplicaciones J2ME.
- Provee soporte para los emuladores de los siguientes SDKs:
 - Sun Java 2 Platform, Micro Edition (J2ME) Wireless Toolkit en las siguientes versiones: 1.0.3._01 y 1.0.4._01.
 - Nokia Developer's Suite for Java 2 Platform, Micro Edition (J2ME)
 - Siemens Mobility Toolkit (SMTK)
 - Sprint PCS Wireless Toolkit for the Java Platform
- Contiene el Sun Microsystems, J2ME Wireless Toolkit version 1.0.4, para ejecutar las aplicaciones MIDlet.
- Permite correr paso a paso las aplicaciones MIDlet.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 7.

Requerimiento	Especificaciones
Procesador	Pentium II 233 MHz o compatible
Memoria RAM	128 MB (mínimo recomendado) 256 MB (recomendado)
Espacio en disco duro	20MB
Sistema Operativo	Microsoft Windows 2000 Service Pack 2, Microsoft Windows XP Microsoft Windows NT 4.0 Service Pack 6
Otros	JBuilder v.7.0

Tabla 14. Requerimientos de sistema, JBuilder MobileSet versión 3.01.

Principales desventajas:

- Los emuladores a los cuales provee soporte, no cuentan con soporte para sockets.
- La compilación y ejecución de las aplicaciones, así como la construcción de las aplicaciones, dependen de tener JBuilder versión 7.0, instalado.

7.5.2. Java

Java 2 Standard Edition Software Development Kit versión 1.4.0_003

Las principales características de éste kit de desarrollo [e_SMJ2], son:

- Provee de comandos para la construcción de aplicaciones Java.
- Provee de un ambiente runtime para el uso de este kit de desarrollo.
- Provee de una maquina virtual de Java.
- Provee de librerías de clases para la construcción de programas, escritos en el lenguaje de programación Java.
- Provee de librerías de clases para la implementación de CORBA.
- Contiene una lista de bugs o problemas, concernientes al desarrollo de aplicaciones en este kit de desarrollo.
- También disponible para plataformas como Solaris y Linux.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 8.

Requerimiento	Especificaciones
Procesador	Pentium 166MHz
Memoria RAM	32MB
Espacio en disco duro	70MB
Sistema Operativo	Microsoft Windows 95, Microsoft Windows 98, Microsoft Windows NT 4.0 Service Pack 5, Microsoft XP, Microsoft Windows 2000 Professional, Microsoft Windows 2000 Server o Microsoft Windows 2000 Advanced Server

Tabla 15. Requerimiento de sistema, J2SE Software Development Kit versión 1.4.0_003

Principales desventajas:

- No contiene un editor de texto para el desarrollo de aplicaciones Java.
- Programas compilados con versiones anteriores pueden no ser compatibles con la presente versión.

Java Media Framework versión 2.1.1b

Las principales características de éste kit de desarrollo [e_SMJM], son:

- Provee de librerías de clases para la construcción de programas, escritos en el lenguaje de programación Java, para la implementación de aplicaciones Java multimedia.
- Contiene una lista de bugs o problemas, concernientes al desarrollo de aplicaciones en este kit de desarrollo.
- Provee soporte para formatos: MPEG, AVI y MOV, entre otros.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 8.

Requerimiento	Especificaciones
Procesador	Pentium 166MHz, PowerPC 160 MHz, o UltraSparc 166 MHz
Memoria RAM	32MB
Espacio en disco duro	No especificado por el fabricante Aproximadamente 5 MB
Tarjeta de video	8, 16, 24 or 32 bits
Sistema Operativo	Microsoft Windows 95, Microsoft Windows 98, Microsoft Windows NT Solaris
Otros	J2SE SDK v.1.1.6 o superior
Opcional	Tarjeta de sonido compatible con SoundBlaster

Tabla 16. Requerimiento de sistema, Java Media Framework versión 2.1.1b

Principales desventajas:

- No existe una versión liberada formalmente por el desarrollador para el sistema operativo Linux.

7.5.3. Jini

Jini Technology Starter Kit version 1.2.1_001

Las principales características de éste kit de desarrollo [e_SMJI], son:

- Contiene herramientas y utilidades que ayudan a desarrollar, ejecutar, analizar y documentar programas basados en las especificaciones de la tecnología Jini.
- Provee de librerías de clases para la construcción de programas, escritos en el lenguaje de programación Java.
- Contiene las especificaciones para el desarrollo de aplicaciones y servicios basados en la tecnología Jini.
- Versión probada bajo el J2SE SDK 1.4.0.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 9.

Requerimiento	Especificaciones
Procesador	Intel/x86
Memoria RAM	No especificado
Espacio en disco duro	No especificado
Sistema Operativo	Microsoft Windows XP Professional, Microsoft Windows 2000 Professional, Microsoft Windows NT 4.0 Service Pack 6, Solaris y Linux.
Otros	J2SE SDK 1.4.0

Tabla 17. Requerimientos de sistema, Jini Technology Starter Kit version 1.2.1_001.

Principales desventajas:

- No contiene un editor para el desarrollo de aplicaciones Jini.

7.5.4. Ambiente de Desarrollo

JBuilder versión 7.0

Las principales características de este paquete de desarrollo [e_BJBL], son:

- Provee herramientas para el desarrollo de Bases de Datos en Java (JDataStore).
- Provee herramientas para aplicaciones gráficas en Java.
- Permite la configuración de perfil de usuario.
- Provee de un compilador y depurador de aplicaciones Java.
- Provee de un editor de código Java.
- Provee de plantillas para la construcción de aplicaciones Java.
- Contiene la versión de J2SE SDK 1.3.1.
- Permite definir otra versión de J2SE SDK superior a la versión 1.3.1.
- Permite definir librerías de clases, para la compilación de código.
- Existe una versión en español.

Los requerimientos mínimos de sistema de éste SDK, es muestran en la tabla 9.

Requerimiento	Especificaciones
Procesador	Pentium II 233MHz
Memoria RAM	128 MB (mínimo recomendado) 256 MB (recomendado)
Espacio en disco duro	750MB
Sistema Operativo	Microsoft Windows 2000 Service Pack 2, Microsoft Windows XP Microsoft Windows NT 4.0 Service Pack 6

Tabla 18. Requerimientos de sistema, JBuilder versión 7.0.

Principales desventajas:

- Para poder acceder a los archivos de ayuda, tiene que instalarse los archivos correspondientes desde la pagina de Internet de Borland.

7.6. Construcción de un sistema basado en aplicaciones Jini-J2ME

Para poder probar la funcionalidad y rendimiento de la arquitectura del sistema, así como cumplir con el objetivo principal del proyecto de tesis, de construir un sistema que pueda demostrar la interacción de las tecnologías Jini y J2ME, con base en herramientas de software gratuitas, se desarrolló un producto de software con esos objetivos.

El producto de software desarrollado, consta principalmente de dos aplicaciones, una aplicación Jini y una aplicación J2ME (MIDlet).

- La aplicación Jini tiene por objetivo, implementar un sistema Jini, basado en servicios, incluyendo a un surrogate host que pueda interactuar posteriormente con un dispositivo móvil J2ME y que le permita a éste último el acceso a los servicios Jini.
- La aplicación J2ME (MIDlet) tiene por objetivo construir la interfaz por la cual el usuario interactúa con el dispositivo móvil, para poder acceder a los servicios Jini.

Al producto de software en su conjunto se le ha puesto el nombre de Sistema Demostrativo de Acceso a Servicios Jini (SDASJ).

Los requerimientos más importantes para el desarrollo de este software demostrativo de aplicación son los siguientes:

1. La aplicación Jini deberá implementar por lo menos 3 servicios basados en la tecnología Jini, residentes en una red de área local.
2. La aplicación MIDlet, deberá implementar una interfaz de interacción con el usuario, de fácil manejo.
3. La arquitectura definida, deberá de soportar servicios basados en Jini, permitiendo la integración del dispositivo móvil J2ME emulado.
4. El acceso a los servicios deberá funcionar sobre un sistema 2.5G.

Para mayor detalle acerca de la descripción, funcionalidad, requerimientos y restricciones del producto, se debe consultar el documento [a_ERS].

Ahora se procederá a analizar los procesos y el flujo de información del sistema con base en la descripción y requerimientos, descritos en el documento [a_ERS]

Como se mencionaba al inicio de esta sección el SDASJ, esta compuesto por dos aplicaciones, una aplicación Jini y una aplicación J2ME. En la figura 42, se muestra en un plano contextual, los flujos de información entre las entidades y las aplicaciones J2ME y Jini.

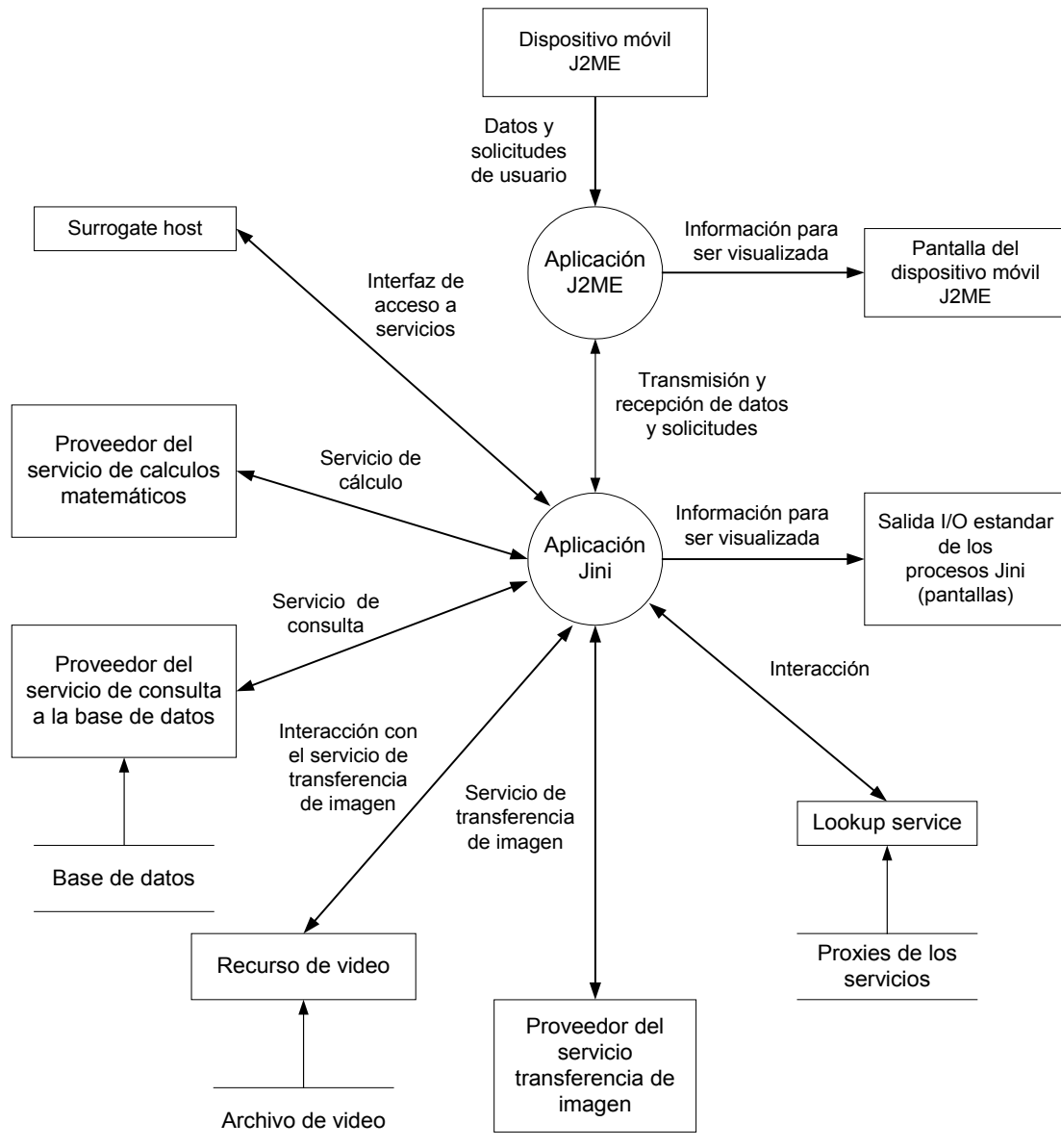


Figura 42. DFD nivel 0 SDASJ.

La aplicación J2ME, permite al usuario interactuar con el dispositivo móvil J2ME, a través del teclado del dispositivo y mostrará los resultados de la interacción en la pantalla del dispositivo, por lo que la aplicación J2ME servirá como la interfaz entre el dispositivo y el usuario, y es a través de la aplicación J2ME que el usuario podrá acceder a los servicios disponibles en la red Jini.

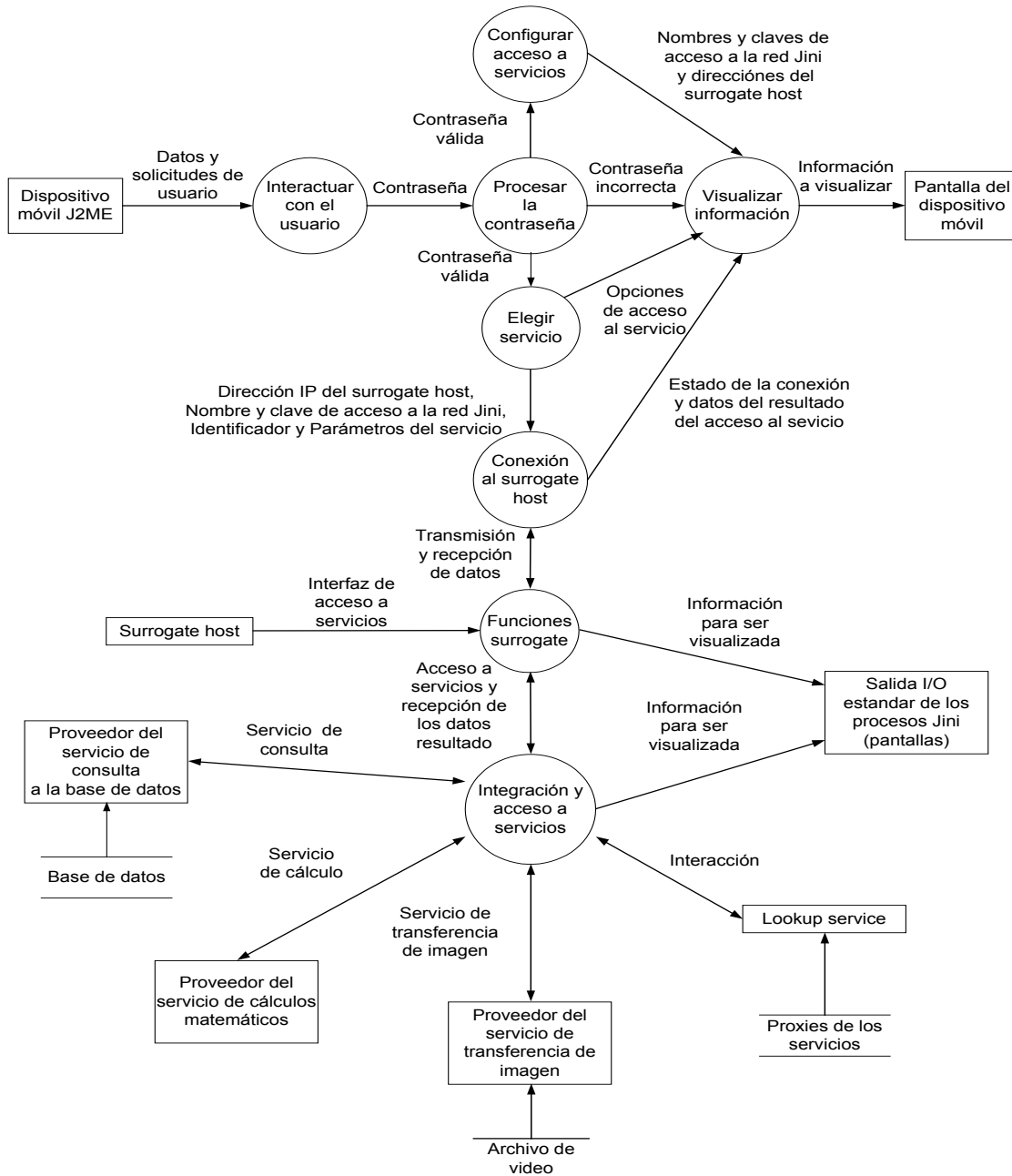


Figura 43. DFD nivel 1 SDASJ.

La aplicación Jini, permitirá integrar a todos servicios, residentes en sus propios proveedores, a través de lookup service, el cual contendrá todos los proxies de los servicios, implementando procesos Jini, a través de los protocolos establecidos por dicha tecnología (discovery y join), para poder registrarlos y ponerlos a disposición a los clientes, dentro de la red Jini. Todos los resultados de los procesos Jini realizados son mostrados en las pantallas de los propios procesos.

Antes de que el dispositivo móvil J2ME, pueda acceder a los servicios, disponibles en la red Jini, los proveedores de servicios, deben registrar los proxies de sus servicios, en el lookup service (utilizando para ello procesos Jini), para que a través del proxy del servicio asociado, cualquier cliente dentro de la red Jini, pueda acceder al propio servicio.

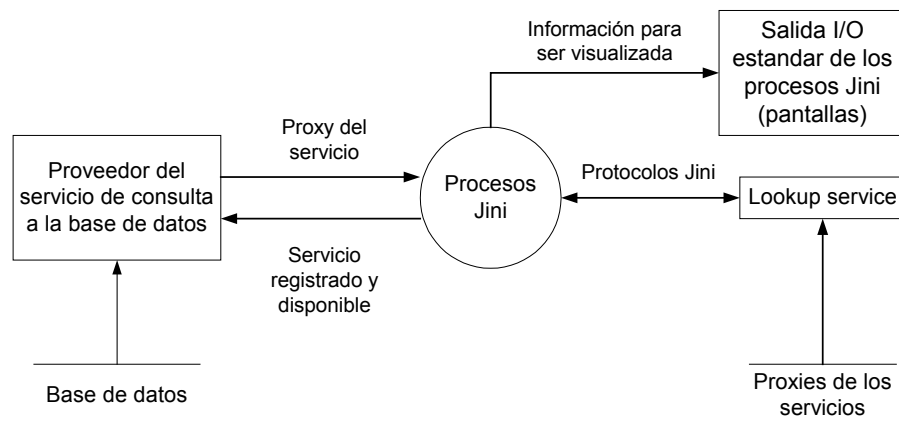


Figura 44. DFD nivel 2 registro del servicio de consulta a la base de datos SDASJ.

En la figura 44, se muestra que el proveedor de servicio de consulta a la base de datos, registra el proxy del servicio en el lookup service, a través de los procesos Jini (descubrimiento del lookup service y registro de proxy), se hace notar, que los proxies deben estar almacenados en el lookup service y que en el proveedor de servicio de consulta a la base de datos se encuentra la base de datos BDPS.

Para llevar a cabo el registro del servicio de transferencia, se inicia el recurso de video, una vez iniciado, se registra el proxy del servicio en el lookup service, a través de los procesos Jini (descubrimiento del lookup service y registro de proxy), como se muestra en la figura 45.

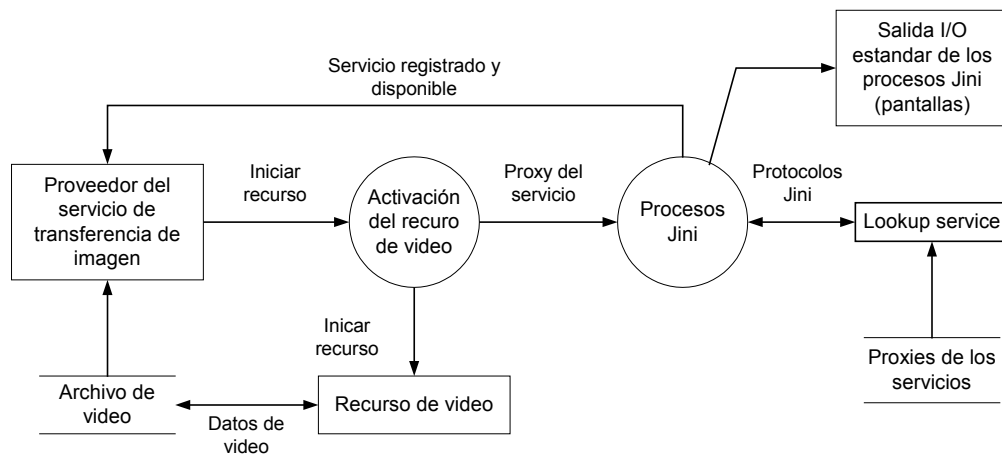


Figura 45 DFD nivel 2 registro del servicio transferencia de imagen SDASJ.

En la figura 46, se muestra que el proveedor de servicio de cálculos matemáticos, registra el proxy del servicio en el lookup service, a través de los procesos Jini (descubrimiento del lookup service y registro de proxy).

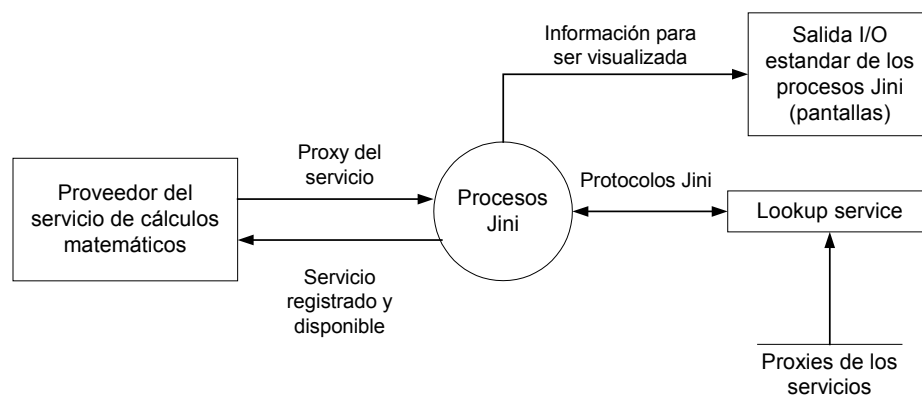


Figura 46. DFD nivel 2 registro del servicio de cálculos matemáticos.

Una vez que se lleva acabo el registro de los proxies de los servicios, es cuando el surrogate host, se inicia en el sistema. El surrogate host tiene dos funciones básicas:

- **Función de registro:** el surrogate host debe registrar a cualquier cliente (en este caso, el dispositivo móvil J2ME) que intente acceder a los servicios de la red Jini, en otras palabras, el cliente debe identificarse a través de un nombre de usuario y una contraseña. La función de registro es accedida, del cliente al surrogate por una conexión socket, vía IP, en un puerto especificado.

- Función surrogate: en ésta función, tiene como requisito la función de registro, una vez que se haya registrado el usuario y su identificación haya sido correcta se activa ésta función, que tiene por objeto, acceder a los servicios en nombre del cliente que ha sido registrado e identificado. La función de registro es accedida, del cliente al surrogate por una conexión socket, vía IP, en un puerto especificado, diferente al puerto especificado en la función de registro.

Después de que los proveedores de servicios, registraron sus proxies y que el surrogate host, ha sido iniciado en el sistema, es cuando el usuario del dispositivo móvil, puede acceder a los servicios de la red Jini.

Una vez que la aplicación J2ME es ejecutada en el dispositivo móvil J2ME, el usuario se debe identificar, a través de un nombre de usuario y una contraseña para poder acceder a la funcionalidad de la aplicación (véase figura 43)., una vez que haya introducido estos datos, el usuario puede configurar el acceso al surrogate host, introduciendo la dirección IP del surrogate host (especificando puertos de conexión por socket para poder acceder a las funciones del surrogate host), el cual realizará el acceso a los servicios contenidos en la red Jini, como se muestra en la figura 47.

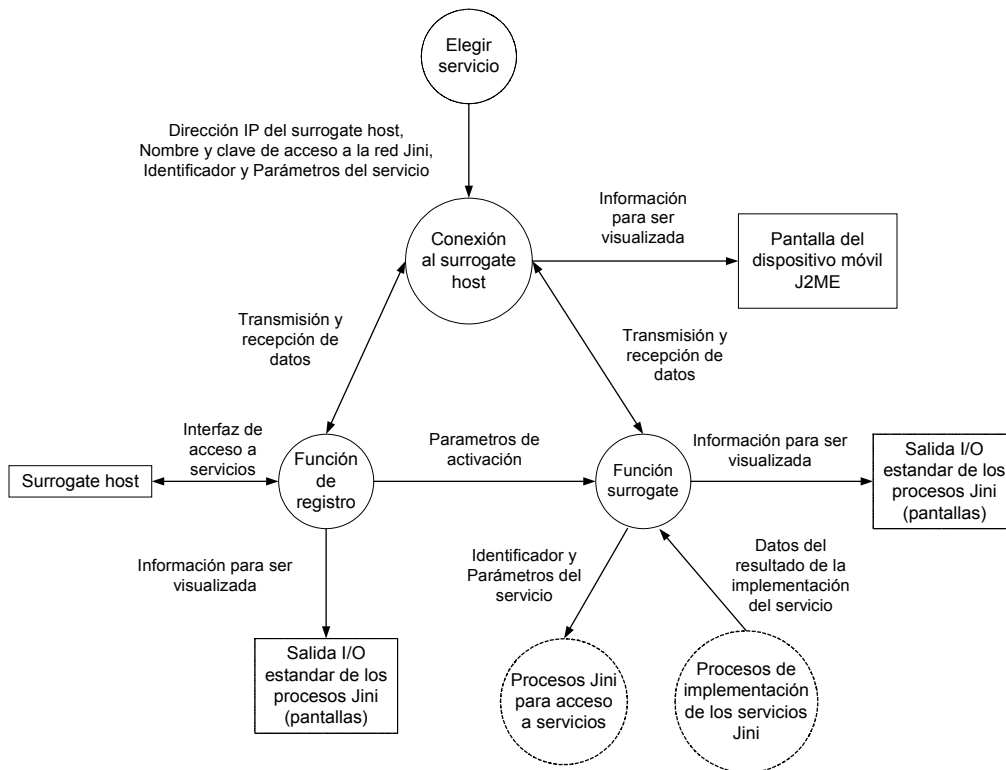


Figura 47. DFD nivel 2, interacción entre el dispositivo móvil J2ME y el surrogate host.

Las opciones de los servicios disponibles en ésta aplicación son:

- Consulta a la base de datos (asociado al servicio Jini: consulta a la base de datos).
- Transferir Imagen (asociado al servicio Jini: transferencia de imagen).
- Cálculos matemáticos (asociado al servicio Jini: cálculos matemáticos)

Cuando el usuario elija una opción, la aplicación J2ME, hará que el dispositivo móvil se comuniqué con el surrogate host (a través del IP del surrogate host), el cual activará la función de registro, para identificar que el nombre de usuario y la contraseña sean correctas para poder activar la función surrogate. Para poder interactuar con los proveedores de servicios, la aplicación J2ME, enviará a través del dispositivo móvil, los parámetros del servicio a acceder, por lo que una vez que se identifiquen los parámetros y el acceso (véase figura 47), el surrogate host accederá al servicio, comunicándose con el proveedor de servicio especificado, a través del proxy del servicio asociado.

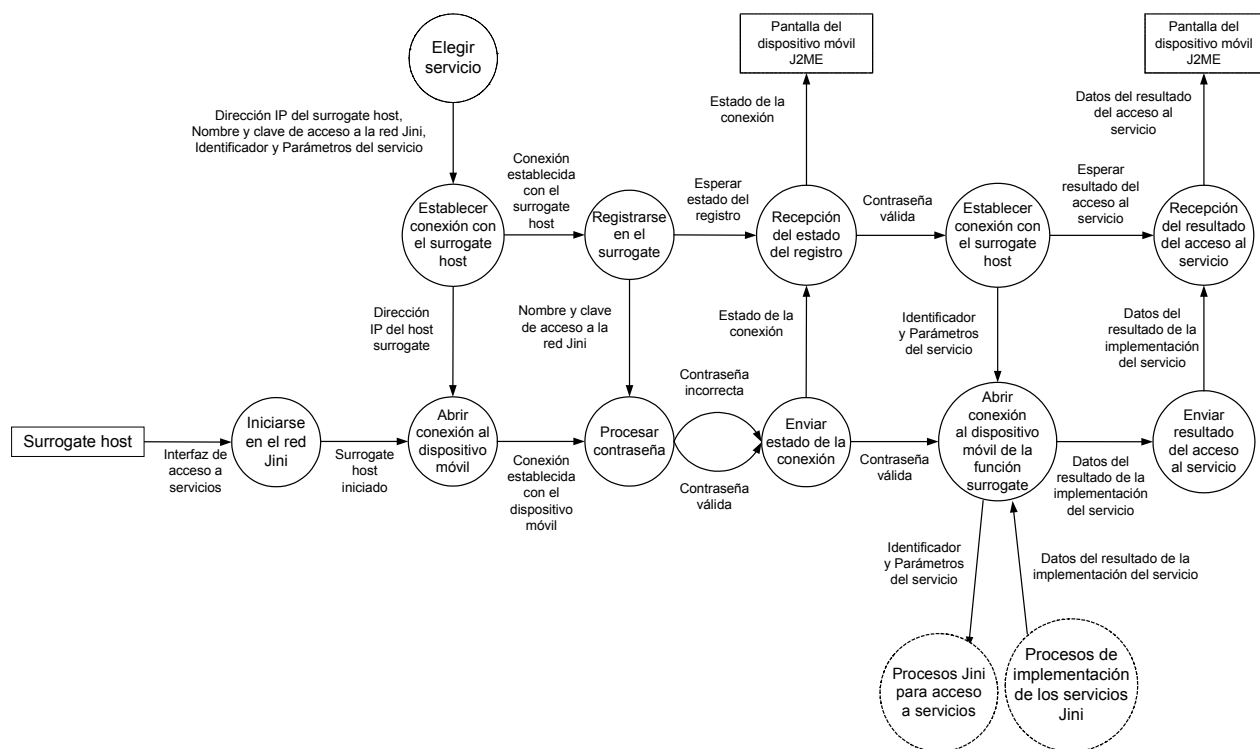


Figura 48. DFD nivel 3, entre el dispositivo móvil J2ME y el surrogate host.

En la figura 48, se muestra el proceso de acceso a los servicios Jini, de manera general una vez, que el surrogate cuenta con el identificador del servicio al que se va a

acceder y los propios parámetros del servicios, accede al lookup service, para encontrar el servicio que se desea acceder, el lookup service le comunica la disponibilidad del servicio, respecto si existe el proxy registrado del servicio al que se desea acceder. La comunicación entre el surrogate host y el lookup service, se lleva acabo a través de los protocolos Jini. Una vez, que el proxy es encontrado, el surrogate hace referencia al servicio a través del proxy del servicio y de esta manera se comunica con el proveedor de servicio, el cual implementa el servicio y regresa el resultado el surrogate host, y una vez que el surrogate host recibe el resultado de la implementación, envía al dispositivo móvil los datos correspondientes, para que sean visualizados a través de la aplicación J2ME, en la pantalla del dispositivo.

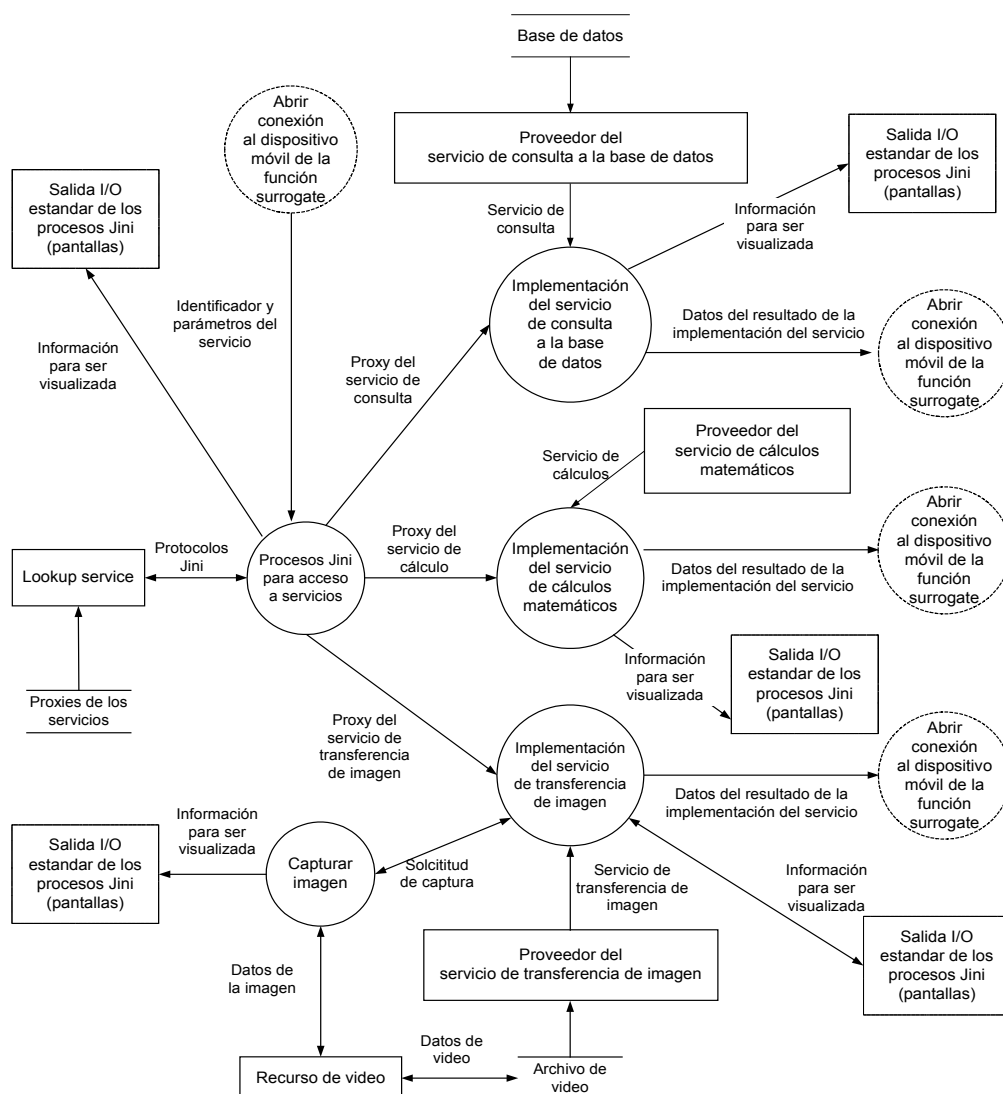


Figura 49. DFD nivel 3, acceso del surrogate host a los servicios Jini.

CAPÍTULO 8. El diseño del sistema

8.1. Arquitectura del Sistema

Con base en el capítulo anterior, se diseñó la arquitectura que fue utilizada para demostrar el acceso a servicios Jini a través de un dispositivo móvil J2ME.

A continuación en la figura 50, se muestra la configuración de la arquitectura que se utilizó para la implementación del sistema.

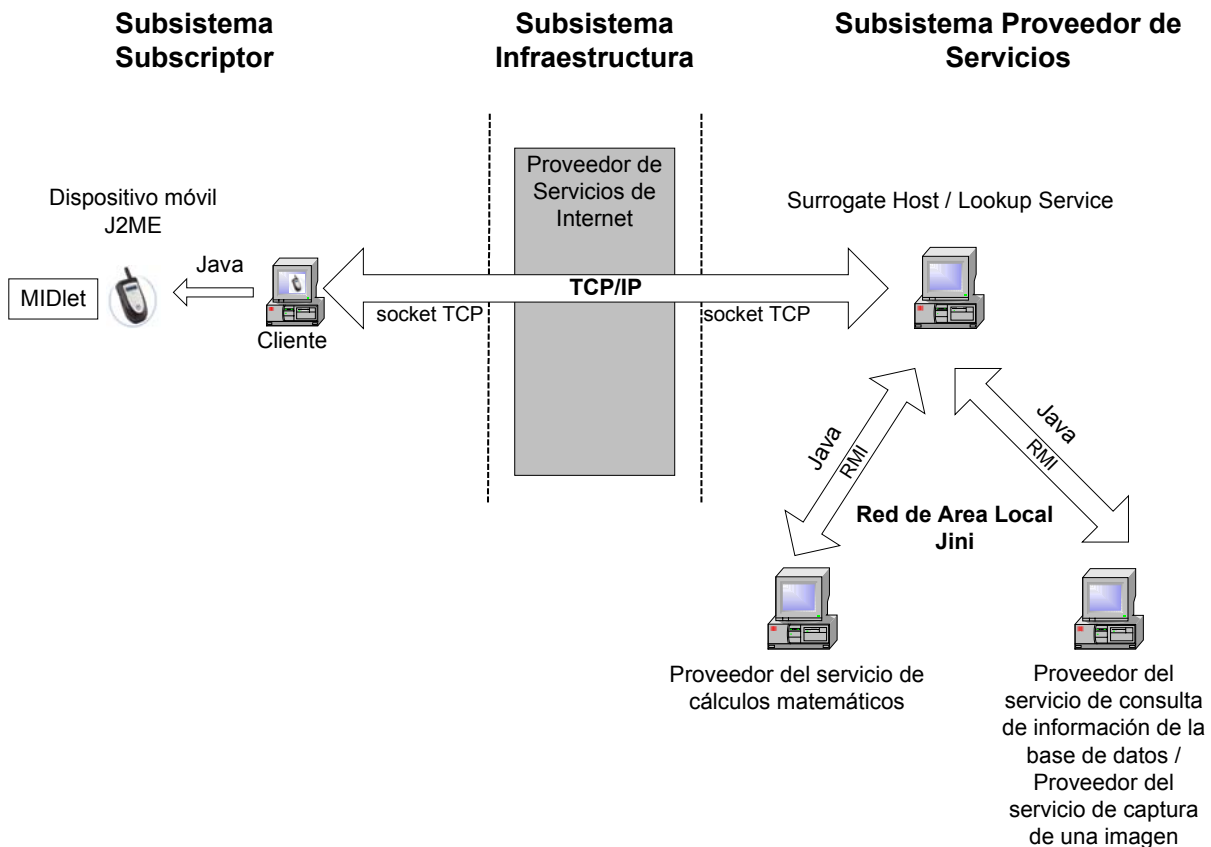


Figura 50. Arquitectura del Sistema

Subsistema subscriptor.

Se encarga de emular el dispositivo móvil J2ME físico, con el cual el cliente accede a los servicios Jini contenidos dentro de una red de área local. El usuario deberá interactuar con el dispositivo J2ME a través de la aplicación MIDlet con la finalidad de poder acceder a los servicios Jini que estén disponibles en la red de área local.

La computadora cliente se conecta a través Internet, por sockets TCP/IP a un surrogate host, el cual le provee de recursos de procesamiento al dispositivo móvil J2ME y de esta manera el surrogate host le dará el acceso a los servicios Jini de la red de área local.

Subsistema de la infraestructura.

Se encarga de simular la parte de la infraestructura de la arquitectura GPRS necesaria para comunicar al cliente (que en este caso es el dispositivo móvil J2ME) con el surrogate host a través de sockets TCP, vía IP.

Subsistema proveedor de servicios.

Este subsistema consta de 3 elementos:

- **Surrogate host / lookup service**, como surrogate host, tiene la capacidad de recibir peticiones del cliente, integrar al cliente a la red Jini: haciendo las solicitudes del servicio, invocar al servicio “en nombre del cliente”; así como enviar los resultados de la solicitud hecha, hacia el cliente. Como lookup service, tiene la capacidad de almacenar los proxies de los servicios, enviar el proxy del servicio al cliente que solicite el servicio determinado, entre otras cosas. La comunicación entre el surrogate host y el emulador del dispositivo móvil J2ME se llevará acabo a través de sockets TCP/IP. La comunicación entre el surrogate host y el lookup service se realiza a través de los protocolos definidos por Jini.
- **Proveedor del servicio de consulta de la base de datos / proveedor del servicio de captura de una imagen**, como proveedor del servicio de consulta a la base de datos, reside la base de datos BDPS y propiamente el servicio de consulta. Como proveedor del servicio de captura de una imagen, reside el recurso de video en ejecución y propiamente el servicio de captura, por lo que también tiene la capacidad de procesar y satisfacer las solicitudes hechas por parte del cliente, a través del surrogate host. La comunicación que se realice

entre los proveedores de servicios con el surrogate host y con el lookup service será a través de RMI.

- **Proveedor del servicio de cálculos matemáticos**, en éste elemento reside propiamente el servicio de cálculo, por lo que tiene la capacidad de procesar y satisfacer las solicitudes hechas por parte del cliente, a través del surrogate host. La comunicación que se realice entre los proveedores de servicios con el surrogate host y con el lookup service será a través de RMI.

La interacción entre cada uno de los componentes, así como el flujo de información fue descrita en el capítulo anterior.

8.2. Componentes de Software utilizados

Una vez que la arquitectura del sistema ha sido establecida, es necesario elegir las herramientas de software que permitan posteriormente implementar cada uno de los componentes.

8.2.1. Subsistema Subscriptor

De todas las alternativas mencionadas anteriormente, se eligió utilizar Mobilset v.3.1. para la codificación e integración del software, por las siguientes razones:

- Cumple con las especificaciones de J2ME, para la configuración CLDC versión 1.0.3 y el perfil MIDP 1.0.
- Soporta los protocolos: UDP, TCP, para la implementación de sockets.
- Permite instalar y ejecutar MIDlets.
- Permite ejecutar paso a paso los MIDlets.
- Contiene compilador y verificador para crear archivos jad y jar.
- Contiene un emulador del dispositivo móvil J2ME.
- Pueden agregarse nuevos emuladores.
- Los datos pueden ser introducidos desde el teclado.
- Tiene licencia gratuita.

Para la ejecución de las pruebas se eligió utilizar J2ME Wireless toolkit v.1.0.4_01, por las siguientes razones:

- Cumple con las especificaciones de J2ME, para la configuración CLDC versión 1.0.3 y el perfil MIDP 1.0.
- Soporta los protocolos: UDP, TCP, para la implementación de sockets (aunque de manera experimental).
- Permite instalar y ejecutar MIDlets.
- Contiene compilador y verificador para crear archivos jad y jar.
- Pueden agregarse nuevos emuladores.
- Los datos pueden ser introducidos desde el teclado.
- Tiene licencia gratuita.
- Otras versiones posteriores, como la versión 2.0. beta 2, de la misma herramienta de desarrollo no son una versiones finales y pueden contener errores que afecten el desempeño de la aplicación MIDlet.

Es importante señalar que fue agregado un emulador SonyEricsson modelo P800 [e_SEEM], ya que dicho modelo provee de una pantalla con un mayor rango de visibilidad que los provistos por la herramienta de desarrollo J2ME Wireless toolkit v.1.0.4_01.

El objetivo para que se hayan escogido dos herramientas de software diferentes tiene las siguientes razones:

- Para la etapa de codificación e integración del software, es necesario ejecutar paso a paso (debug) la aplicación y analizar el comportamiento del emulador, por lo que Mobileset, a través de JBuilder, muestra gráficamente el comportamiento de las variables y permite controlar la ejecución de la aplicación MIDlet.
- Para la etapa de pruebas, no es necesario ejecutar paso a paso (debug), sino que el objetivo es analizar el comportamiento de todo el sistema, por lo que no tiene caso instalar otro software necesario para el funcionamiento del emulador, como es el caso de Mobileset, que necesita JBuilder v. 7.0.

La tabla 1, muestra la comparativa de las principales herramientas para el desarrollo de aplicaciones J2ME.

Características	Herramienta de desarrollo				
	iDEN SDK v. 1.0.0	J2ME Wireless Toolkit v.1.0.4	J2ME Wireless Toolkit v.2.0. beta 2	Palm OS Simulator v. 5.0	MobileSet v. 3.01
Desarrollador	Motorola	Sun Microsystems	Sun Microsystems	Palm OS	Borland
Procesador necesario	Pentium 500 Mhz	Pentium 166 MHz	Pentium 166 MHz	Intel x86	Pentium II 233 MHz
Memoria RAM necesaria	128 MB	64 MB	64 MB	No especificado	128 MB
Espacio necesario en disco duro	12 MB	30 MB	50 MB	23 MB	20 MB
Sistemas Operativos	Microsoft Windows: NT, 98, 2000	Microsoft Windows: NT, 2000	Microsoft Windows: 2000, XP	Microsoft Windows: NT, 2000, XP, 98, ME	Microsoft Windows: NT, 2000, XP
Numero de Emuladores disponibles	3	6	4	1	6
Versión compatible de J2SE SDK	1.2.2, 1.3.1, 1.4.0	1.3.0.	1.4.0	N/A	1.3.1, 1.4.0.
Versión CLDC	1.0.3	1.0.3	1.0.4	1.0.3	1.0.3
Versión MIDP	1.0	1.0	2.0	1.0	1.0
Protocolos para el soporte de sockets	TCP, UDP, SSL	--	TCP, UDP, SSL	--	TCP, UDP, SSL
Otro software necesario	--	--	--	--	JBuilder v.7.0.

Tabla 19. Comparativa de características de los más importantes J2ME SDKs.

8.2.2. Subsistema Infraestructura

El proveedor de servicios de Internet elegido es Prodigy Internet de Telmex, ya que cuenta con una tasa de transferencia de 56Kbps y provee al cliente con una dirección IP ruteable.

8.2.3. Subsistema Proveedor de Servicios

Existe solo una herramienta existente para la implementación de la tecnología Jini, por lo que inherentemente fue Jini Technology Starter Kit version 1.2.1_001, ya que hasta el momento de realización del proyecto era la versión más actual.

Para la construcción de los servicios Jini, se utilizó JBuilder versión 7.0. En este caso, JBuilder fue elegido, aún habiendo otras herramientas como Sun One Studio de Sun Microsystems, ya que JBuilder tiene una versión en español y el manejo de la herramienta es más sencilla y fácil de utilizar a comparación de otras herramienta, en opinión del tesista.

Todas las aplicaciones Java construidas en este proyecto, fueron desarrolladas con base en la herramienta de desarrollo Java 2 Standard Edition versión 1.4.0_003, en dependencia de la herramienta de desarrollo para Jini, ya que ésta herramienta fue construida y probada sobre la versión 1.4.0_003 del J2SE SDK, por lo que el uso de otras versiones de J2SE SDK, no garantizan la compatibilidad entre las herramientas de desarrollo del software.

8.2.4. Distribución de los componentes de software

Subsistema Subscriptor

- **Dispositivo Móvil Emulado**
 - Java 2 Standard Edition SDK versión 1.4.0_003
 - Sun Microsystems, J2ME Wireless Toolkit version 2.0 beta 2.
 - Borland, Mobileset v.3.01.
 - Borland, JBuilder v.7.0.
 - Software instalado en una PC Pentium II, 450 Mhz con 384MB en RAM

- Sistema Operativo: Microsoft Windows 2000 Professional.
- Dirección IP: variable, dependiendo de la dirección asignada por el proveedor de servicios de Internet.
- Nombre de la computadora: artég

Subsistema Proveedor de Servicios

- **Surrogate host / lookup service.**
 - Java 2 Standard Edition SDK versión 1.4.0_003
 - Jini Technology Starter Kit versión 1.2.1_001
 - Instalados en una PC Pentium III a 800Mhz con 128 MB en RAM
 - Sistema Operativo: Linux Slackware 7.1
 - Dirección IP: 207.249.165.80
 - Nombre de la computadora: computacion.upaep.mx
- **Proveedor del servicio de consulta de la base de datos / proveedor del servicio de captura de una imagen**
 - Java 2 Standard Edition SDK versión 1.4.0_003
 - Jini Technology Starter Kit versión 1.2.1_001
 - Java Media Framework versión 2.1.1. para Linux.
 - Instalados en una PC Pentium III a 650Mhz con 64 MB en RAM
 - Sistema Operativo: Linux Slackware 7.1
 - Dirección IP: 200.33.13.43
 - Nombre de la computadora: windu.ico.upaep.mx
 - Base de datos BDPS.
 - Archivos de video.
- **Proveedor del servicio de cálculos matemáticos**

- Java 2 Standard Edition SDK versión 1.4.0_003
- Jini Technology Starter Kit versión 1.2.1_001
- Instalados en una PC Pentium a 133Mhz con 64MB en RAM
- Sistema Operativo: Linux Slackware 7.1
- Dirección IP: 200.33.13.44
- Nombre de la computadora: quigon.ico.upaep.mx

8.2.5. Localización física de los subsistemas

En la siguiente figura, se muestra la localización de los subsistemas.

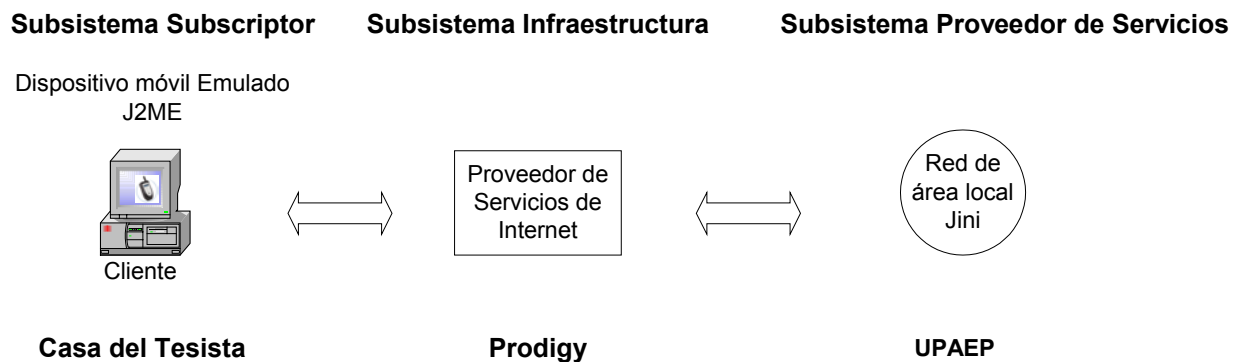


Figura 51. Localización física de los subsistemas

Como se presenta en la figura 51, el subsistema subscriptor se localiza en la casa del tesista, el cual se comunicara con la red de área local Jini, la cual se encuentra localizada en UPAEP, a través del proveedor de servicios de Internet: Prodigy.

8.2.6. Pila de protocolos utilizadas para la comunicación entre subsistemas

De manera general, la pila de protocolos de comunicación del sistema, define la manera en la que los subsistemas se comunican.

En el sistema subscriptor, el dispositivo móvil emulado J2ME, utiliza un IP activo de la maquina donde reside, el cual es adquirido del IP asignado por el proveedor de

servicios de Internet. Cuando el dispositivo móvil J2ME, desea comunicarse con la red de área local, lo lleva acabo a través de un IP activo de salida, asignado por el proveedor de servicios de Internet, el cual es diferente del IP activo asociado a la maquina donde reside el dispositivo móvil emulado.

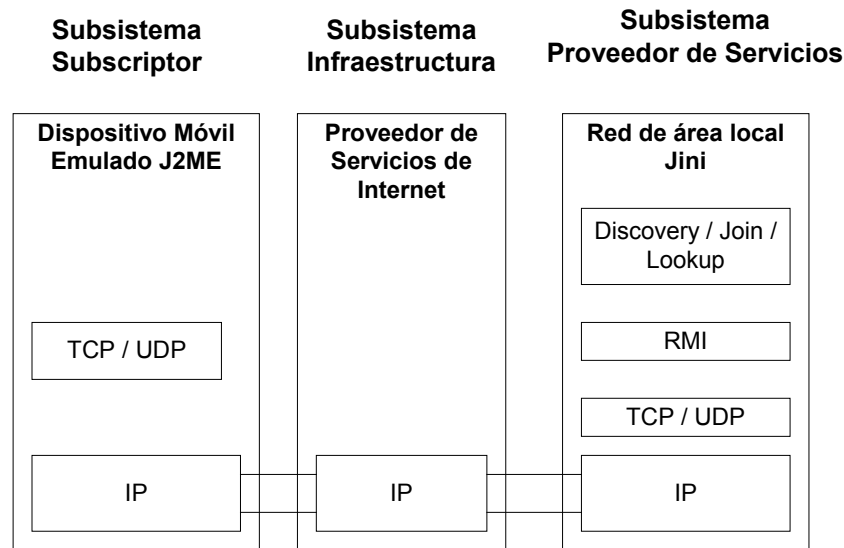


Figura 52. Pila de protocolos de comunicación del sistema.

8.3. Diseño de los servicios Jini

El conjunto de servicios Jini que se describirán en las siguientes secciones se han agrupado en el grupoJini llamado “Tesis”.

Es importante dejar establecido que existen diferentes formas de diseñar e implementar los servicios Jini, como se mencionó en el capítulo 4, para fines de este proyecto se ha tomado en cuenta que cada proveedor de servicio, implemente su propio servicio con sus recursos de procesamiento (en su misma máquina virtual Java), ya que de otra forma el cliente que solicite los servicios deberá procesar la implementación del servicio solicitado, con sus propios recursos; la forma en la que se puede evitar que el cliente (en este caso el surrogate host) procese la implementación del servicio es diseñar e implementar el servicio haciendo uso de RMI.

Otro punto importante considerar, es el método por el cual se descubrirá al surrogate, el proceso de descubrimiento del lookup service será llevado acabo por el método Unicast, ya que el uso de Multicast requiere de infraestructura con la que no se

cuenta (véase capítulo 4), además de no ser indispensable para el objetivo de este proyecto.

En las siguientes secciones, el diseño del sistema estará representado en diagramas UML y los cuales son diagramas de clases basadas en Java, ya que todas las tecnologías manejadas en este proyecto están basadas en Java. Para mayor detalle acerca de la descripción, funcionalidad, requerimientos y restricciones del producto, se debe consultar el documento [a_ERS].

8.3.1. Servicio de consulta a la base de datos

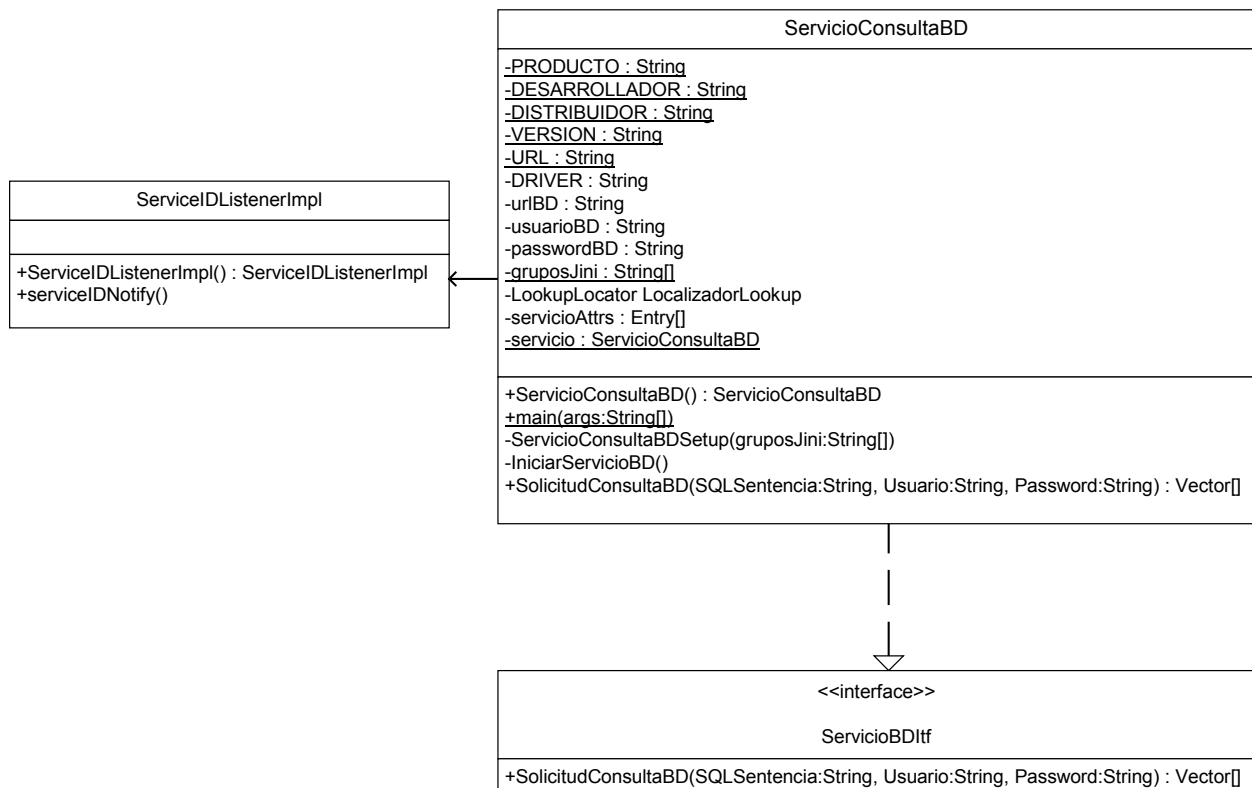


Figura 53. Diagrama de clases del servicio de consulta.

La clase ServicioConsultaBD tiene por objetivo llevar acabo los procesos de Jini de descubrimiento del lookup service, registrar el servicio de consulta a la base de datos, esperar la solicitud de un cliente, así como de llevar acabo la implementación del propio servicio, una vez que ha sido solicitado por un cliente.

A continuación se definen los atributos de la clase ServicioConsultaBD:

- PRODUCTO: Nombre del servicio.
- DESARROLLADOR: Nombre del desarrollador
- DISTRIBUIDOR: Nombre del distribuidor del producto
- VERSIÓN: Versión del producto
- URL: Dirección IP o nombre del lookup service.
- DRIVER: Nombre del driver para acceder a la base de datos BDPS.
- urlBD: Ruta en donde se encuentra la base de datos BDPS.
- usuarioBD: Nombre de usuario para acceso a la base de datos BDPS.
- passwordBD: Contraseña que permite acceso a la base de datos BDPS (asociado al usuarioBD).
- gruposJini: Nombre del grupo al que el servicio se registrará.
- LocalizadorLookup: Localizador del lookup service por vía remota.
- servicioAttrs: Contiene los atributos del servicio.
- servicio: Instancia de la clase ServicioConsultaBD.

Ahora se definirán los métodos de la clase ServicioConsultaBD:

- ServicioConsultaBD(): Constructor de la clase
- main(args): Método para instanciar la clase.
- ServicioConsultaBDSetup(gruposJini): Método por el cual se llevarán acabo los procesos Jini, tales como el descubrimiento del lookup service y el registro del mismo servicio, tiene como parámetro gruposJini, para poder realizar los procesos que anteriormente se mencionaron.
- IniciarServicioBD(): Método que inicia el servicio para esperar solicitudes de algún cliente.
- SolicitudConsultaBD(SQLSentencia, Usuario, Password): Método que tiene por objetivo la implementación del servicio, en donde SQLSentencia, es la sentencia en lenguaje SQL, que servirá para realizar la consulta a la base de datos BDPS,

la implementación del servicio tiene como resultado un un vector de datos con base en la solicitud SQL introducida.

La clase ServicioConsultaBD implementa el método de la clase ServicioBDItf, la cual define la solicitud de consulta a la base de datos. La clase ServiceIDListenerImpl es una clase miembro de la clase ServicioConsultaBD, utilizada para recibir del lookup service el identificador del servicio, producto del registro del servicio de consulta a la base de datos.

8.3.2. Servicio de cálculos matemáticos

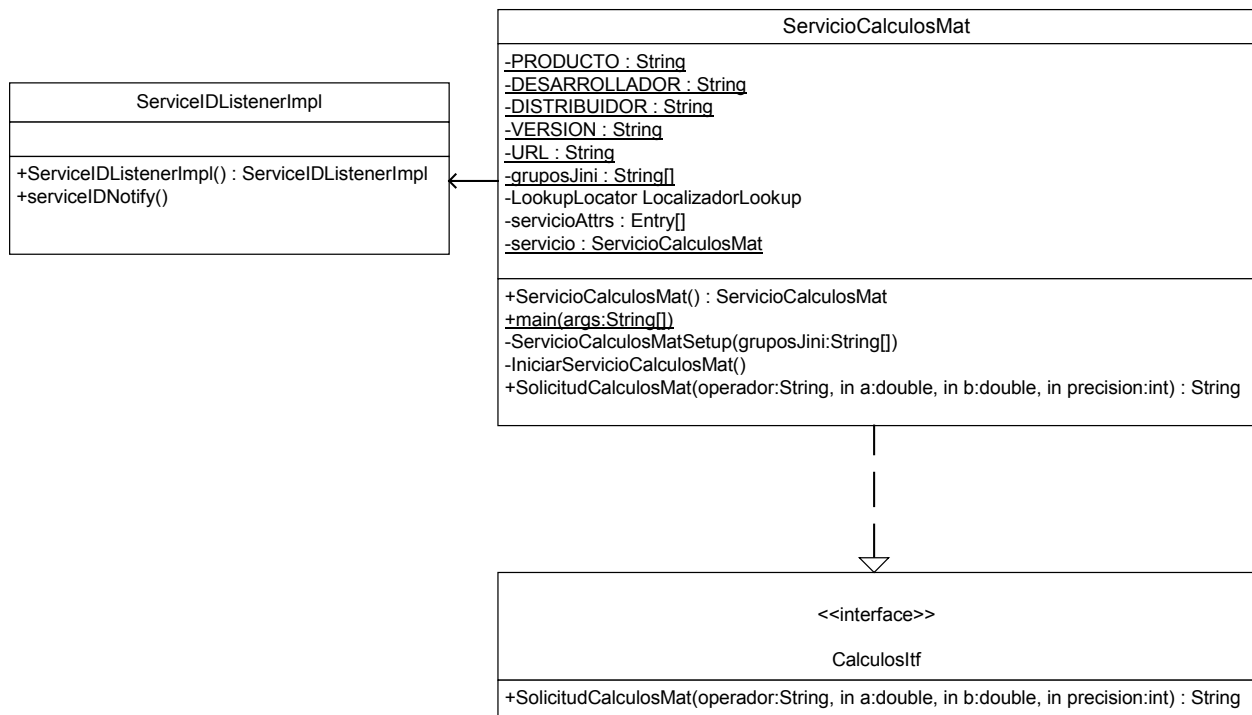


Figura 54. Diagrama de clases del servicio de cálculo.

La clase ServicioCalculosMat tiene por objetivo llevar a cabo los procesos de Jini de descubrimiento del lookup service, registrar el servicio de cálculos matemáticos, esperar la solicitud de un cliente, así como de llevar a cabo la implementación del propio servicio, una vez que ha sido solicitado por un cliente.

A continuación se definen los atributos de la clase ServicioCalculosMat:

- PRODUCTO: Nombre del servicio.
- DESARROLLADOR: Nombre del desarrollador
- DISTRIBUIDOR: Nombre del distribuidor del producto
- VERSIÓN: Versión del producto
- URL: Dirección IP o nombre del lookup service.
- gruposJini: Nombre del grupo al que el servicio se registrará.
- LocalizadorLookup: Localizador del lookup service por vía remota.
- servicioAttrs: Contiene los atributos del servicio.
- servicio: Instancia de la clase ServicioCalculosMat.

Ahora se definirán los métodos de la clase ServicioCalculosMat:

- ServicioCalculosMat(): Constructor de la clase
- main(args): Método para instanciar la clase.
- ServicioCalculosMatSetup (gruposJini): Método por el cual se llevarán acabo los procesos Jini, tales como el descubrimiento del lookup service y el registro del mismo servicio, tiene como parámetro gruposJini para poder realizar los procesos que anteriormente se mencionaron.
- IniciarServicioCalculosMat(): Método que inicia el servicio para esperar solicitudes de algún cliente.
- SolicitudCalculosMat(operador, a, b, precision): Método que tiene por objetivo la implementación del servicio, en donde operador, es un carácter que especifica el tipo de operación a realizar, los números reales están representados por los parámetros a y b, por último, se encuentra el parámetro precision, que se refiere al número de decimales con los que se deberá realizar la operación.

La clase ServicioCalculosMat implementa el método de la clase CalculosItf, la cual define la solicitud de cálculos matemáticos. La clase ServiceIDListenerImpl es una clase miembro de la clase ServicioCalculosMat, utilizada para recibir del lookup service el identificador del servicio, producto del registro del servicio de cálculos matemáticos.

8.3.3. Servicio de transferencia de una imagen

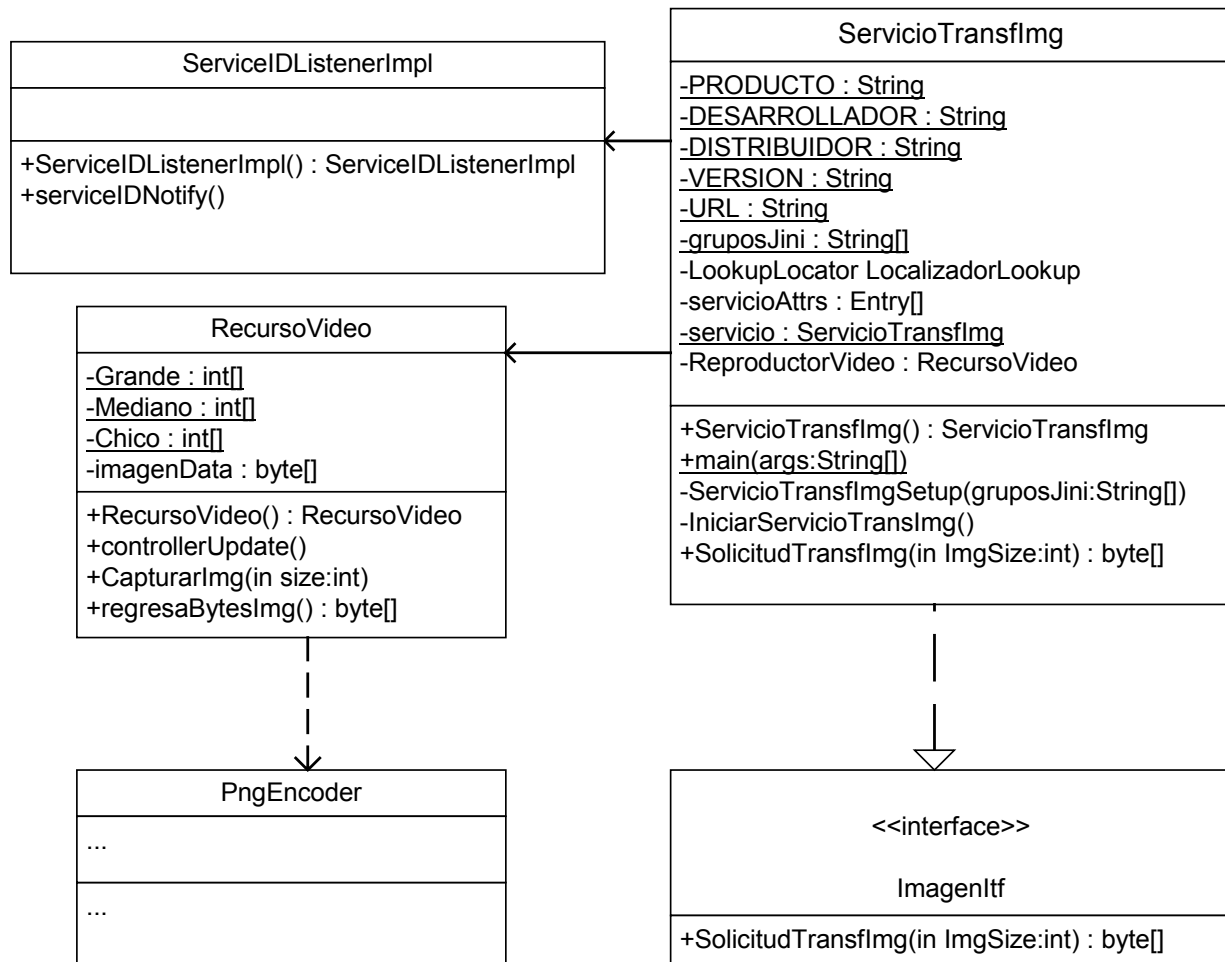


Figura 55. Diagrama de clases del servicio de transferencia

La clase ServicioTransfImg tiene por objetivo llevar acabo los procesos de Jini de descubrimiento del lookup service, registrar el servicio de transferencia de una imagen, esperar la solicitud de un cliente, así como de llevar acabo la implementación del propio servicio, una vez que ha sido solicitado por un cliente.

A continuación se definen los atributos de la clase:

- PRODUCTO: Nombre del servicio.
- DESARROLLADOR: Nombre del desarrollador

- DISTRIBUIDOR: Nombre del distribuidor del producto
- VERSIÓN: Versión del producto
- URL: Dirección IP o nombre del lookup service.
- gruposJini: Nombre del grupo al que el servicio se registrará.
- LocalizadorLookup: Localizador del lookup service por vía remota.
- servicioAttrs: Contiene los atributos del servicio.
- servicio: Instancia de la clase ServicioTransfImg.
- ReproductorVideo: instancia de la clase Recurso de video, la cual su objetivo es iniciar al recurso de video y capturar la imagen que posteriormente será transferida.

Ahora se definirán los métodos de la clase ServicioTransfImg:

- ServicioTransfImg(): Constructor de la clase
- main(args): Método para instanciar la clase.
- ServicioTransfImgSetup(gruposJini): Método por el cual se lleva acabo los procesos Jini, tales como el descubrimiento del lookup service y el registro del mismo servicio, tiene como parámetro gruposJini para poder realizar los procesos que anteriormente se mencionaron.
- IniciarServicioTransfImg(): Método que inicia el servicio para esperar solicitudes de algún cliente.
- SolicitudTransfImg(ImgSize): Método que tiene por objetivo la implementación del servicio, en donde ImgSize, es un valor entero que especifica el tamaño de la imagen que será capturada.

La clase ServicioTransfImg implementa el método de la clase ImagenItf, la cual define la solicitud de cálculos matemáticos. La clase ServiceIDListenerImpl es una clase miembro de la clase ServicioTransfImg, utilizada para recibir del lookup service el identificador del servicio, producto del registro del servicio de transferencia de una imagen.

La clase RecursoVideo es la encargada de iniciar el recurso de video, que en este caso, empieza a reproducir de manera cíclica un archivo de video, por los motivos mencionados en el documento [a_ERS]. Cuando un cliente solicita el servicio de transferencia de una imagen, se accede al método CapturarImg(size) de la clase RecursoVideo, el cual captura la imagen con un tamaño determinado por el parámetro

size, dándole el formato PNG a la imagen, utilizando el método correspondiente de la clase PngEncoder. La clase PngEncoder no fue diseñada por el tesista, sino que fue creada por un desarrollador independiente [e_PNGE], avalado por el propio desarrollador del formato PNG [e_PNGD] [e_PNGS].

El formato PNG fue elegido por ser el formato soportado y reconocido por la herramienta de desarrollo J2ME Wireless toolkit v.1.0.4_01 [e_SMWT].

8.3.4. Funciones del Surrogate host

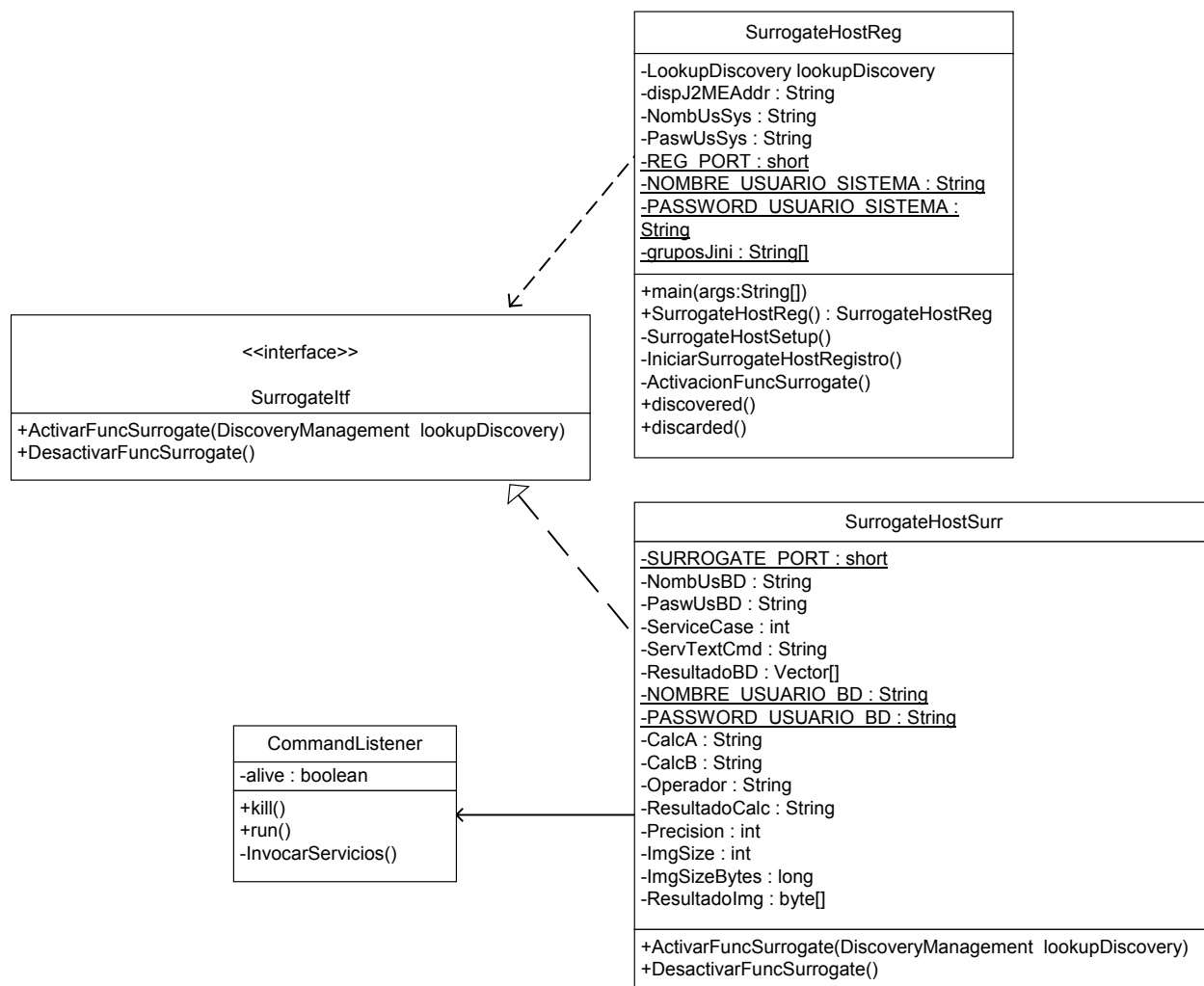


Figura 56. Diagrama de clases del surrogate host.

La clase SurrogateHostReg tiene por objetivo llevar a cabo el proceso de Jini de descubrimiento del lookup service, abrir un socket TCP/IP para la comunicación con el dispositivo móvil J2ME, esperar la conexión del dispositivo, así como de identificar y registrar al usuario, además de activar la clase SurrogateHostSurr, una vez que la identificación y registro del usuario se realizaron.

A continuación se definen los atributos de la clase SurrogateHostReg:

- lookupDiscovery: atributo que descubre al lookup service con base en el grupo Jini especificado.
- dispJ2MEAddr: Dirección IP del dispositivo.

- NombUsSys: Nombre de usuario para acceso a la red Jini.
- PaswUsSys: Contraseña que permite el acceso a la red Jini, asociado al NombUsSys.
- REG_PORT: Número de puerto TCP, para permitir la conexión del dispositivo móvil.
- NOMBRE_USUARIO_SISTEMA: Nombre de usuario establecido para acceder a los servicios de la red Jini.
- PASSWORD_USUARIO_SISTEMA: Contraseña que permite acceder a los servicios de la red Jini, asociada al NOMBRE_USUARIO_SISTEMA,.
- gruposJini: Nombre del grupo de servicios Jini, al cual el surrogate host deberá buscar el servicio solicitado y accederlo.

Ahora se definirán los métodos de la clase SurrogateHostReg:

- main(args): Método para instanciar la clase.
- SurrogateHostReg(): Constructor de la clase.
- SurrogateHostSetup(): Método por el cual se lleva a cabo el proceso Jini de descubrimiento del lookup service, para posterior acceso a los servicios Jini.
- IniciarSurrogateHostRegistro(): Método que inicia la función de registro del surrogate host, abriendo un puerto TCP/IP y esperando a que un cliente (en este caso el dispositivo móvil) establezca una conexión, también tiene la función de registrar al cliente, identificándolo y verificando los datos de acceso (nombre de usuario y contraseña de acceso a la red Jini).
- ActivacionFuncSurrogate(): Método que tiene por objetivo la activación de la función surrogate.
- discovered(): Método que es invocado cuando es encontrado un lookup service perteneciente al grupo Jini especificado.
- discarded(): Método que es invocado cuando no es posible encontrar un lookup service perteneciente al grupo Jini especificado.

La clase SurrogateHostSurr tiene por objetivo abrir un socket TCP/IP, para la comunicación con el dispositivo móvil J2ME, el cual le enviará solicitudes de acceso a los servicios Jini al surrogate host, por lo que esta clase también se encarga de recibir

las solicitudes del dispositivo, verificar el servicio la disponibilidad del servicio solicitado y accederlo, para posteriormente enviar al dispositivo el resultado de la implementación del servicio.

La clase SurrogateHostSurr implementa los métodos de la clase SurrogateItf, la cual define los métodos de activación y desactivación de la función surrogate. La clase CommandListener es una clase miembro de la clase SurrogateHostSurr utilizada para implementar la apertura del socket TCP/IP y monitorear cualquier tipo de fallo respecto al envío y recepción de datos y solicitudes del dispositivo móvil, así como desactivar la función surrogate en el caso de encontrar los fallos mencionados.

A continuación se definen los atributos de la clase SurrogateHostSurr:

- SURROGATE_PORT: Número de puerto TCP, para permitir la conexión del dispositivo.
- NombUsBD: Nombre de usuario para el acceso a la base de datos BDPS (parámetro de implementación del servicio de consulta a la base de datos).
- PaswUsBD: Contraseña o clave de acceso a la base de datos BDPS asociada al NombUsBD (parámetro de implementación del servicio de consulta a la base de datos).
- ServiceCase: Identificador del servicio solicitado.
- ServTextCmd: Sentencia SQL para realizar la consulta a la base de datos BDPS (parámetro de implementación del servicio de consulta a la base de datos).
- ResultadoBD: Vector que contiene los datos resultado de la implementación del servicio de consulta a la base de datos BDPS.
- NOMBRE_USUARIO_BD: Nombre de usuario establecido para el acceso a la base de datos BDPS.
- PASSWORD_USUARIO_BD: Contraseña o clave de acceso (previamente establecida) a la red Jini, asociada al NOMBRE_USUARIO_SISTEMA.
- CalcA: Número real para realizar el calculo matemático (parámetro de implementación del servicio de cálculos matemáticos).
- CalcB: Número real para realizar el calculo matemático (parámetro de implementación del servicio de cálculos matemáticos).

- **Operador:** Identifica la operación a realizar con los numeros CalcA y CalcB (parámetro de implementación del servicio de cálculos matemáticos).
- **ResultadoCalc:** Número real resultado de la implementación del servicio de cálculos matemáticos.
- **Precision:** Número de decimales asociado al formato de salida del número resultado de la implementación del servicio de cálculos matemáticos.
- **ImgSize:** Número asociado al tamaño de imagen a capturar del recurso de video (parámetro de implementación del servicio de transferencia de una imagen).
- **ImgSizeBytes:** Numero de bytes de la imagen capturada resultado de la implementación del servicio de transferencia de una imagen.
- **ResultadoImg:** Vector de bytes que conforman la codificación en formato PNG de la imagen capturada resultado de la implementación del servicio de transferencia de una imagen.

Ahora se definirán los métodos de la clase SurrogateHostSurr:

- **ActivarFuncSurrogate(lookupDiscovery):** Método por el cual se activa la función surrogate, tiene como parámetro al lookupDiscovery, el cual contiene la información del lookup service asociado a un grupo Jini, este parámetro es utilizado anteriormente en la clase SurrogateHostReg. A través, de esta función es que el surrogate conocer donde se encuentra el lookup service, para poder encontrar los proxies de los servicios y así, poder acceder a ellos.
- **DesactivarFuncSurrogate():** Método que es invocado cuando ocurre alguna falla respecto al envío y recepción de datos y solicitudes del dispositivo móvil,

Es importante señalar que existe un API para la implementación de la función surrogate [e_JSUR], pero la principal desventaja por la que no se utilizó, es la dificultad de entender la total funcionalidad que provee, además de no contar con ejemplos prácticos que faciliten la utilización de su API. Por lo que la función surrogate fue diseñada con base en los conceptos básicos de esta API.

8.3.5. Aplicación MIDlet

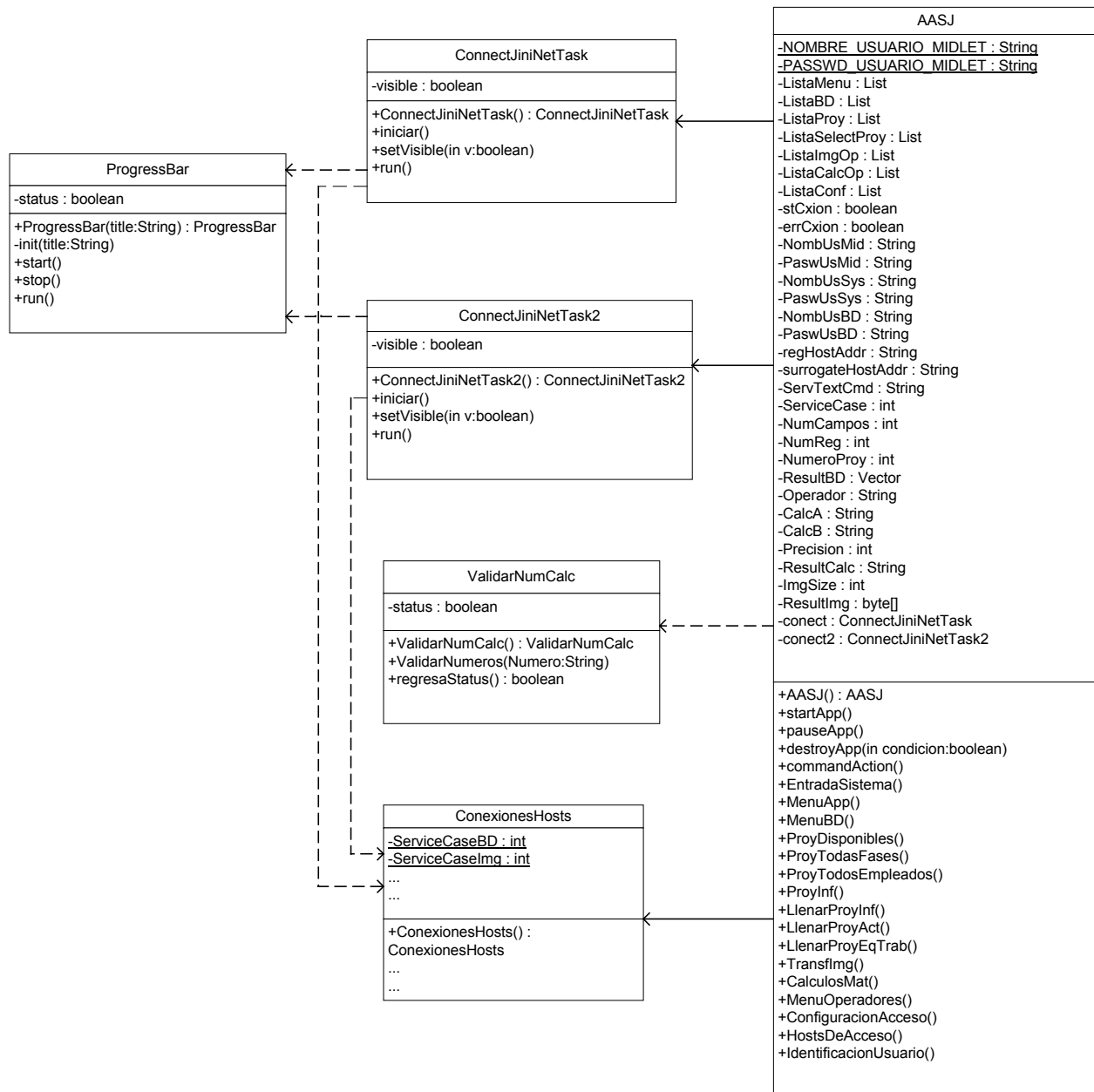


Figura 57. Diagrama de clases (a) de la aplicación MIDlet.

La clase AASJ tiene por objetivo procesar los datos y solicitudes del usuario, además de permitirle acceder a los servicios Jini a través del dispositivo móvil J2ME, estableciendo una conexión por medio de sockets TCP/IP al surrogate host para el

envío y recepción de datos (uno para la función de registro y el otro para la función surrogate).

Para poder acceder a los servicios Jini, el usuario debe introducir parámetros de identificación y de configuración de la propia aplicación MIDlet, tales como:

- Nombre de usuario y contraseña para el acceso a las funciones de la aplicación MIDLET
- Nombre de usuario y contraseña para el acceso a la red Jini.
- Nombre de usuario y contraseña para el acceso a la base de datos BDPS.
- Dirección IP + Puerto TCP, de la ubicación del surrogate host en su función de registro.
- Dirección IP + Puerto TCP, de la ubicación del surrogate host en su función surrogate.

La clase ASJJ es encargada de procesar los datos y solicitudes de usuario, conformando los parámetros de los servicios, que serán enviados al surrogate host para el acceso correspondiente al servicio solicitado por el usuario del dispositivo móvil, así como de visualizar la información recibida por el surrogate host, este tipo de información puede ser: el estado de la conexión y el resultado de la implementación del servicio.

A continuación se definen los atributos de la clase ASJJ:

- NOMBRE_USUARIO_MIDLET: Nombre de usuario establecido para acceder a las funciones de la aplicación MIDlet.
- PASSWD_USUARIO_MIDLET: Clave de acceso (establecida previamente) a las funciones de la aplicación MIDlet y asociada al NOMBRE_USUARIO_MIDLET.
- ListaMenu: Contiene opciones del menú general de la aplicación MIDlet.
- ListaBD: Contiene opciones del para la consulta a la base de datos BDPS.
- ListaProy: Contiene la lista de proyectos disponibles para ser consultados de la base de datos.

- ListaSelectProy: Contiene opciones de consulta para un proyecto disponible.
- ListaImgOp: Contiene opciones de tamaño para la transferencia de una imagen.
- ListaCalcOp: Contiene opciones de cálculos matemáticos para dos números reales.
- ListaConf: Contiene opciones de configuración de la aplicación MIDlet, así como de configuraciones de identificación de usuario y de especificación de la dirección IP del surrogate host.
- stCxion: Refiere el status de la conexión con el surrogate host.
- errCxion: Refiere el status de acceso a la base de datos.
- NombUsMid: Nombre de usuario de la aplicación MIDlet.
- PaswUsMid: Contraseña de acceso a la aplicación MIDlet.
- NombUsSys: Nombre de usuario de la red Jini.
- PaswUsSys: Contraseña de acceso a la red Jini.
- NombUsBD: Nombre de usuario de la base de datos BDPS.
- PaswUsBD: Contraseña de acceso a la base de datos BDPS.
- regHostAddr: Dirección IP + Puerto TCP del surrogate host (función de registro).
- surrogateHostAddr: Dirección IP + Puerto TCP del surrogate host (función surrogate).
- ServTextCmd: Sentencia SQL para la consulta a la base de datos.
- ServiceCase: Identificador de servicio Jini.
- NumCampos: Numero de campos que contiene la tabla resultado de la consulta a la base de datos.
- NumReg: Numero de registros que contiene la tabla resultado de la consulta a la base de datos.

- NumeroProy: Número de proyectos disponibles en la base de datos.
- ResultBD: Vector que contiene todos los datos resultado de la consulta a la base de datos.
- Operador: Carácter que identifica el tipo de operación o de cálculo que debe ser realizado a los números reales.
- CalcA: Número real.
- CalcB: Número real.
- Precision: Número de decimales con los que se debe realizar la operación matemática de los dos números reales.
- ResultCalc: Resultado del cálculo de la operación especificada de dos números reales.
- ImgSize: Tamaño de imagen solicitada para ser transferida hacia el dispositivo móvil.
- ResultImg: Vector de bytes que contiene a la imagen en formato PNG.
- connect: Manejador de la conexión y de la barra de progreso de la conexión.
- conect2: Manejador de la conexión y de la barra de progreso de la conexión, para recibir información de la base de datos.

Ahora se definirán los métodos de la clase ASJJ:

- AASJ(): Constructor de la clase.
- startApp(): Método que es invocado cuando inicia la aplicación
- pauseApp(): Método que es invocado cuando se lleva a cabo un estado de pausa en la aplicación.
- destroyApp(condicion): Método que es invocado cuando termina la aplicación.

- `commandAction()`: Método que es invocado cuando el usuario selecciona un comando.
- `EntradaSistema()`: Método por el cual se presenta la información de entrada a la aplicación.
- `MenuApp()`: Método por el cual se presenta las opciones primarias de la aplicación.
- `MenuBD()`: Método por el cual se presenta las opciones de consulta a la base de datos.
- `ProyDisponibles()`: Método por el cual se presenta la lista de proyectos de la base de datos.
- `ProyTodasFases()`: Método por el cual se presenta la información de las fases definidas para cada proyecto.
- `ProyTodosEmpleados()`: Método por el cual se presenta la información de los empleados registrados en la base de datos.
- `ProyInf()`: Método por el cual se presenta opciones de la información de un proyecto en particular.
- `LlenarProyInf()`: Método por el cual se presenta la información de un proyecto en particular.
- `LlenarProyAct()`: Método por el cual se presenta el plan de trabajo de un proyecto en particular.
- `LlenarProyEqTrab()`: Método por el cual se presenta la información del equipo de trabajo de un proyecto en particular.
- `TransfImg()`: Método por el cual se presenta las opciones de tamaño de la imagen que se desea transferir al dispositivo móvil.
- `CalculosMat()`: Método por el cual se introducen los números reales que posteriormente serán transformados en una operación matemática.
- `MenuOperadores()`: Método por el cual se presentan las opciones para realizar la operación matemática a los dos números reales.

- ConfiguracionAcceso(): Método por el cual se presentan las opciones de configuración para el acceso a los servicios Jini.
- HostsDeAcceso(): Método por el cual se configura la dirección IP o nombre del surrogate host.
- IdentificacionUsuario(): Método por el cual se configuran los nombres de usuario y contraseñas de acceso a la red Jini y a la base de datos.

Las clases ConnectJiniNetTask y ConnectJiniTask2 son clases miembros de la clase SurrogateHostSurr, ambas clases son utilizadas para instanciar las clases ProgressBar y ConexionesHosts.

La clase ConexionesHosts es utilizada para enviar datos al surrogate host, así como recibir los datos del mismo, implementa la apertura del socket TCP/IP y monitorear cualquier tipo de fallo respecto al envío y recepción de datos del surrogate host (véase figura 58), a continuación se definen de manera general los atributos de ésta clase:

- ServiceCaseBD, ServiceCaseImg y ServiceCaseCalc, son los identificadores de los servicios Jini: consulta a la base de datos, transferencia de una imagen y cálculos matemáticos respectivamente.
- stCxion y errCxion, son utilizados para verificar el estado de la conexión con el surrogate.
- NumCampos y NumReg son utilizados para recibir la información del servicio de consulta a la base de datos. Los demás atributos: ResultadoBD, ResultadoImg y ResultadoCalc son utilizados para recibir los resultados del acceso a los servicios Jini: consulta a la base de datos, transferencia de una imagen y cálculos matemáticos respectivamente.

La diferencia fundamental entre las clases ConnectJiniNetTask y ConnectJiniTask2, es que la primera se utiliza para acceder a los servicios de transferencia de una imagen y cálculos matemáticos, así como acceder al servicio de consulta a la base de datos para realizar la consulta primaria respecto a los proyectos disponibles, fases de proyectos y todos los empleados registrados; la segunda es utilizada para acceder al servicio de consulta a la base de datos, para realizar la consulta de toda la información de un proyecto en particular (equipo de trabajo, actividades del proyecto, descripción general, etc).

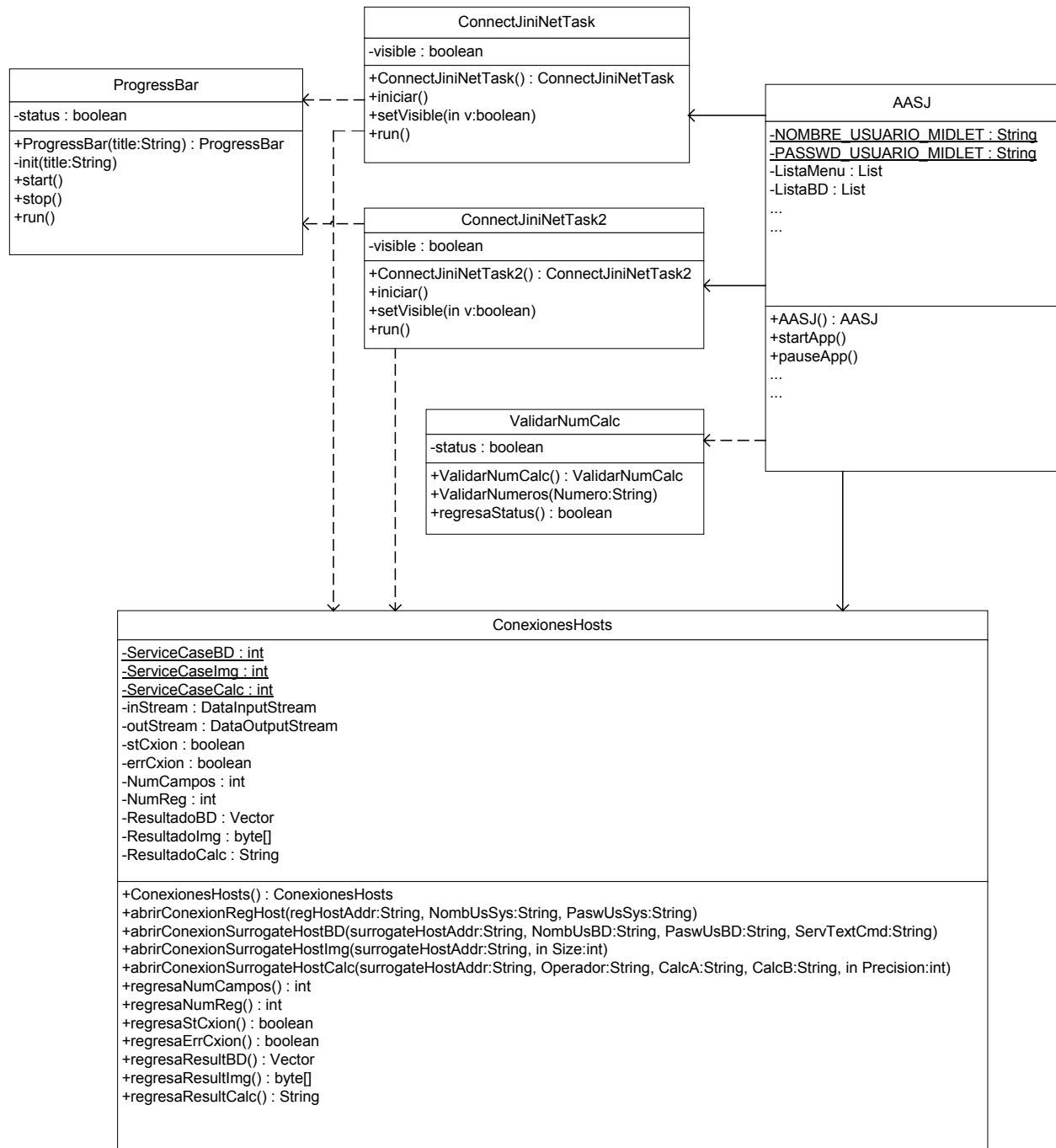


Figura 58. Diagrama de clases (b) de la aplicación MIDlet

Ahora se definirán los métodos de la clase ConexionesHosts:

- ConexionesHosts() : Constructor de la clase.
- abrirConexionRegHost(regHostAddr, NombUsSys, PaswUsSys): Método que tiene por objetivo establecer una comunicación con el surrogate host a través de un socket TCP/IP y enviar una petición de registro. Los parámetros de éste método son:
 - regHostAddr: Dirección IP + Puerto TCP del surrogate host en la función de registro.
 - NombUsSys: Nombre de usuario a la red Jini.
 - PaswUsSys: Contraseña de acceso a la red Jini.
- abrirConexionSurrogateHostBD(surrogateHostAddr, NombUsBD, PaswUsBD, ServTextCmd): Método que tiene por objetivo establecer una comunicación con el surrogate host a través de un socket TCP/IP, para enviar la solicitud de acceso al servicio de consulta a la base de datos, con base en los siguientes parámetros:
 - surrogateHostAddr: Dirección IP + Puerto TCP del surrogate host en la función surrogate.
 - NombUsBD: Nombre de usuario a la base de datos BDPS.
 - PaswUsBD: Contraseña de acceso a la base de datos BDPS.
 - ServTextCmd: Sentencia SQL de consulta a la base de datos BDPS.
- abrirConexionSurrogateHostImg(surrogateHostAddr, Size)
 - surrogateHostAddr: Dirección IP + Puerto TCP del surrogate host en la función surrogate.
 - Size: Tamaño de la imagen que será capturada y transferida al dispositivo móvil.
- abrirConexionSurrogateHostCalc(surrogateHostAddr, Operador, CalcA, CalcB, Precision)
 - surrogateHostAddr: Dirección IP + Puerto TCP del surrogate host en la función surrogate.
 - Operador: El operador que esta asociado al cálculo matemático de los dos números reales.
 - CalcA: Número real.
 - CalcB: Número real.
 - Precision: Número de decimales que tendrá como formato el resultado del cálculo matemático de los dos números reales.

- `regresaNumCampos()`: Método que obtiene el resultado del numero de campos de la tabla resultado de la consulta a la base de datos.
- `regresaNumReg()`: Método que obtiene el resultado del numero de registros de la tabla resultado de la consulta a la base de datos.
- `regresaStCxion()`: Método que obtiene el estado de la conexión con el surrogate.
- `regresaErrCxion()`: Método que obtiene el estado de la conexión respecto al acceso a la base de datos
- `regresaResultBD()`: Método que obtiene el resultado (tabla de datos) del acceso al servicio de consulta a la base de datos.
- `regresaResultImg()`: Método que obtiene el resultado (imagen) del acceso al servicio de transferencia de una imagen.
- `regresaResultCalc()`: Método que obtiene el resultado (número real) del acceso al servicio de cálculos matemáticos.

La clase `ProgressBar` es utilizada por las clases `ConnectJiniNetTask` y `ConnectJiniTask2`, para hacer visible en la pantalla del dispositivo móvil una barra de progreso mientras se lleva acabo la conexión con el surrogate host y las transacciones correspondientes.

La clase `ValidarNumCalc` es utilizada por clase `ASJJ`, para validar los números reales que posteriormente serán enviados al servicio de cálculos matemáticos, junto con un operador asociado y el número de decimales, para realizar el correspondiente cálculo.

Para mayor detalle de la codificación de estas clases, puede ser consultado el material suplementario del CD-ROM que acompaña a éste documento de tesis (véase apéndice A), así como la guía para la implementación del sistema.

8.4. Diseño de las pruebas a realizar

La estrategia para el diseño de las pruebas del sistema se describe gráficamente en la siguiente figura:

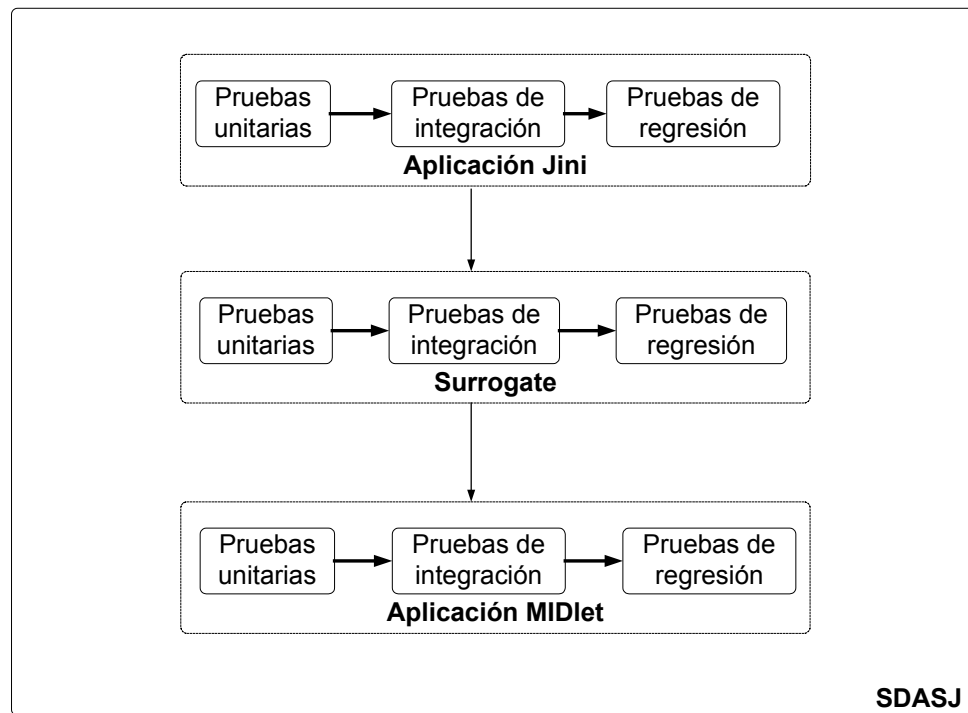


Figura 59. Estrategia de pruebas de los módulos del sistema.

Todos los componentes de los subsistemas subscriptor y proveedor de servicios serán probados con base en los requerimientos especificados en el documento [a_ERS]. Los tipos de pruebas realizadas son: unitarias, de regresión y de integración. Para las pruebas de interacción del sistema (véase figura 60) serán utilizadas pruebas de integración, pruebas de regresión y pruebas basadas en escenarios, para mayor detalle del tipo de pruebas realizadas puede ser consultado el documento [a_PPS].

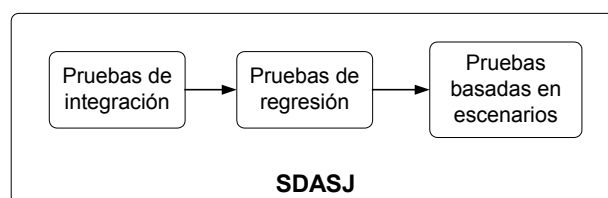


Figura 60. Estrategia de pruebas para el sistema.

Para cada uno de los requerimientos a probar, se han diseñado casos de prueba que contienen un conjunto de instancias, cada instancia a su vez, define entradas de sistema y espera correspondientes salidas, cada uno de los procesos llevado a cabo están especificados en el documento [a_ERS].

El criterio de aceptación del sistema esta basado en encontrar entre un 3% a un 7% de errores del total de instancias, ya que es sumamente difícil obtener un sistema libre de errores, por lo que estadísticamente puede ser aceptable un sistema que tiene un porcentaje superior al 90% de fiabilidad.

Los elementos a probar en el proyecto son los siguientes:

- Subsistema Subscriptor.
 - Emulador del dispositivo móvil J2ME.
 - Aplicación MIDlet
- Subsistema Proveedor de Servicios
 - Aplicación Jini
 - Servicios Jini.
 - Surrogate host.

8.5. Problemas encontrados en el acoplamiento del sistema

- La arquitectura propuesta en la sección 8.1 de este capítulo, no pudo ser implementada debido a que la herramienta de software Java Media Framework no existía una versión oficial del propio desarrollador [e_JMFL] para el sistema operativo Linux, y la versión encontrada de otro desarrollador no esta disponible en el momento del desarrollo del proyecto [e_JMFB]. Por otro lado el sistema operativo que tienen instalados todas las computadoras pertenecientes a UPAEP, es Linux y no puede ser cambiado por otro por razones propias, por lo que la arquitectura finalmente utilizada para la implementación y las pruebas del sistema es mostrada en la figura 61. La solución llevada acabo por el tesista fue trasladar el proveedor de servicio de transferencia de una imagen a su propia computadora (ya que ésta cuenta con un sistema operativo Windows) e instalar la versión 2.1.1b de JMF para Windows, además de almacenar en la misma computadora el archivo de video utilizado por el servicio Jini de transferencia de una imagen.

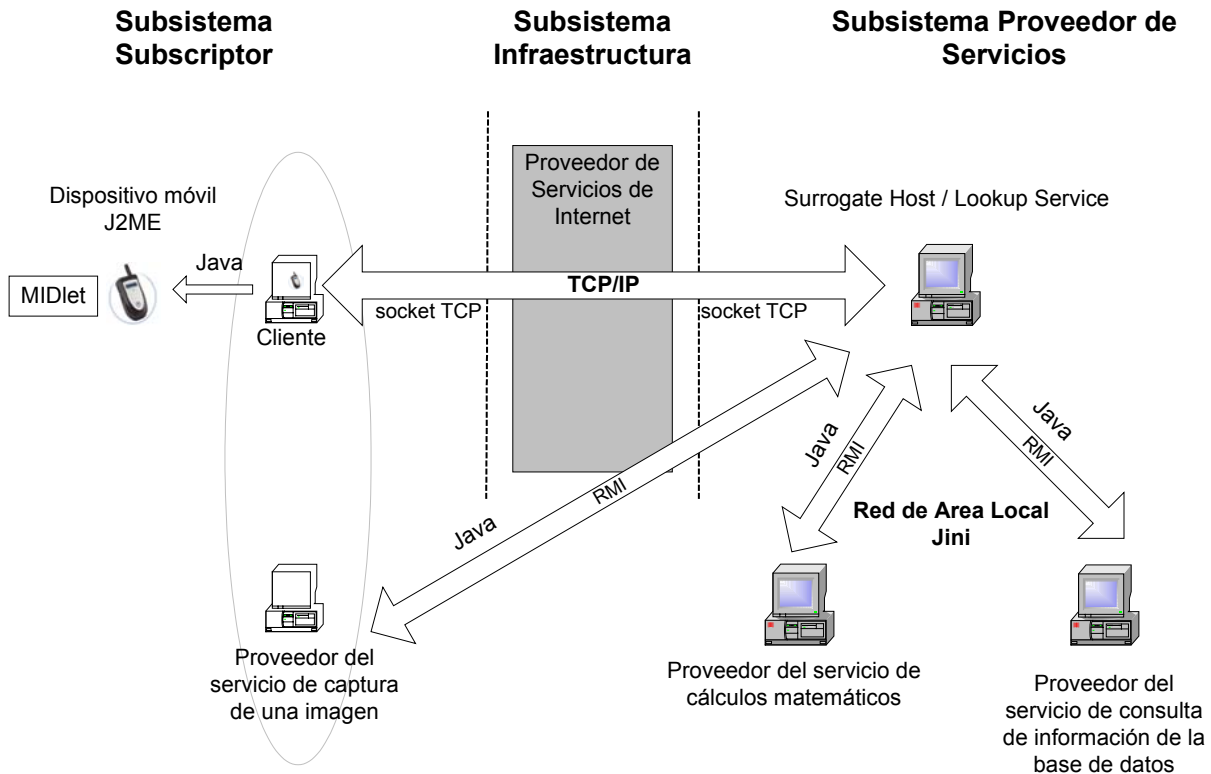


Figura 61. Arquitectura final del sistema.

- Al llevar a cabo la implementación de la arquitectura mencionada en el punto anterior, se había pensado en otro proveedor de servicios de internet: Megared, pero surgió el problema de que éste proveedor [e_MEGR], de acuerdo a la arquitectura de red con la que cuenta, la dirección IP asignada a la computadora donde reside el proveedor del servicio Jini de transferencia de una imagen, es una dirección del tipo red privada, por lo el acceso a través de RMI, no es posible, ya que es necesario una dirección IP real. La solución propuesta por el tesista fue emplear a otro proveedor de servicio de Internet que pudiera proveer de una dirección IP real a la computadora donde residía dicho proveedor del servicio Jini, el cual fue Prodigy Internet de Telmex [e_PROT]. Por lo que tanto el emulador del dispositivo móvil J2ME y el proveedor del servicio Jini, mencionado anteriormente, residen en la misma computadora (véase figura 61).
- Para que el servicio Jini de consulta a la base de datos BDPS, la cual fue construida con base en JDatastore v.6.0., necesita de las librerías encontradas en el archivo jdsserver.jar, encontrado en el directorio de las librerías utilizadas

por JBuilder. La solución propuesta por el tesista fue copiar y almacenar el archivo `jdserver.jar` en un directorio local de la computadora donde reside la base de datos BDPS.

La conformación de los proxies de los servicios Jini solo deben contener el esqueleto (stub) RMI (`Servicios_Stub.class`) y el archivo de la interfaz del servicio (`ServicioXlrf.class`), por lo que no es necesario incluir todas los archivos `.class` que intervienen en la implementación del servicio.

CAPÍTULO 9. Resultados del proyecto

9.1. Resultados de las pruebas del sistema

Se realizaron dos ejecuciones de prueba para el sistema desarrollado. En ambas ejecuciones, primeramente se realizaron las pruebas a la aplicación J2ME y después se realizaron las pruebas del sistema en conjunto.

La primera ejecución presentó errores y no cumplió con el criterio de aceptación, a continuación se muestran los resultados de la primera ejecución:

- Treinta y seis requerimientos probados, cada requerimientos con su propio caso de prueba y cada caso de prueba con su propio procedimiento de prueba.
- Los casos de pruebas tuvieron en promedio 12 instancias de prueba.
- Cada instancia fue ejecutada 2 veces.
- 432 instancias probadas.
- 48 errores fueron encontrados.
- El porcentaje de errores fue 11.1%.
- Tiempo que duró la sesión de pruebas: 7hrs 15mins.
- Las principales errores fueron los siguientes:
 - Los proxies de los servicios no habían sido configurados correctamente.
 - No fue posible el acceso al servicio Jini de transferencia de una imagen.
 - La validación de los números reales (utilizados como parámetros para el servicio de cálculos matemáticos) no funcionaba correctamente.
 - El tiempo de acceso definido en la implementación del socket de comunicación con el surrogate host era pequeño, por lo que no permitía la comunicación del dispositivo móvil J2ME interactuar con él, si el usuario deseaba seguir accediendo a los servicios Jini.

Una vez que los errores fueron corregidos, se procedió a realizar una segunda ejecución de pruebas, obteniendo los siguientes resultados:

- Treinta y seis requerimientos probados, cada requerimientos con su propio caso de prueba y cada caso de prueba con su propio procedimiento de prueba.
- Los casos de pruebas tuvieron en promedio 12 instancias de prueba.
- Cada instancia fue ejecutada 2 veces.
- 432 instancias probadas.
- 15 errores fueron encontrados.
- El porcentaje de errores fue 3.5%.
- Tiempo que duró la sesión de pruebas: 8hrs 31mins.
- Las principales errores fueron los siguientes:
 - Validación de números reales (utilizados como parámetros para el servicio de cálculos matemáticos)
 - Cualquiera de los dos números es cero no se le permite elegir alguna opción de cálculo.
 - Si se introduce uno de los números con un valor igual a “-.0”, no permite elegir alguna opción de cálculo.
 - Si no se introduce algún número entero para definir el número de decimales que tendrá el formato del resultado del cálculo, no permite elegir alguna opción de cálculo.

Como resultado de la segunda ejecución de pruebas el porcentaje de aceptación del sistema se incremento del 88.9% al 96.5%, como consecuencia de la corrección de errores. Por lo que el porcentaje del 96.5% de aceptación del sistema fue suficiente para no realizar alguna otra ejecución de pruebas, de acuerdo a la especificación de los criterios de aceptación del sistema, detallada en el documento [a_PPS].

9.2. Resultados del proyecto de tesis

- El acceso a servicios Jini a través de dispositivos J2ME, bajo la configuración CLDC y el perfil MIDP, es demostrado con base en el desarrollo de este proyecto de tesis y descrito en este documento.
- El problema de la integración de un dispositivo móvil a un sistema distribuido es resuelto con el uso de las tecnologías Jini y J2ME, basándose principalmente en el hecho de que la tecnología Jini además de proveer su propia arquitectura, también cuenta con una arquitectura Jini Surrogate, en la cual se define la forma en la que un dispositivo que no cuenta con los recursos necesarios para integrarse a la red Jini puede acceder a los servicios provistos por ésta. Entre este tipo de dispositivos son contemplados los dispositivos móviles J2ME.
- La arquitectura Jini Surrogate provee de un API, sin embargo el funcionamiento de las interfaces definidas en ésta API son difícil de entender, además de no proveer de documentación en donde se muestren posibles ejemplos de implementación, por lo que se identificó la funcionalidad del API y se diseñó e implementó una interfaz de programación similar, simplificando así, su entendimiento e implementación.
- Con base en el objetivo de este proyecto de utilizar teléfonos celulares como un medio de intercambio de información vía datos y que cuenten con una cierta capacidad de procesamiento, se identificaron los teléfonos celulares que soportan la tecnología J2ME, este tipo de dispositivos son clasificados en la tecnología J2ME, en la configuración CLDC y el perfil MIDP. Respecto a las generaciones de telefonía celular, éstos son contemplados a partir de la generación 2.5G.
- Se identificó que en la primera versión de MIDP estaba deshabilitada la opción para el soporte de protocolos TCP y UDP, ya que no todos los dispositivos podían soportar estos protocolos, pero que podía ser habilitada para analizar el comportamiento de estos dispositivos al implementar el uso de los protocolos mencionados.
- Los dispositivos móviles J2ME, como lo son los teléfonos celulares 2.5G, no cuentan con grandes capacidades de visualización de datos e imágenes así como capacidades de procesamiento, sin embargo el soporte de J2ME aumenta el grado de usabilidad del dispositivo, permitiendo la instalación de aplicaciones J2ME (MIDlet) cada una con diferentes funcionalidades. El uso de la tecnología J2ME facilita el desarrollo de aplicaciones para dispositivos móviles pero por las propias limitantes físicas del dispositivo, la interfaz de acceso hombre-máquina

seguirá siendo limitada respecto a las que se pueden desarrollar para una computadora personal.

- Existe un alto grado de factibilidad de la interacción entre las tecnologías Jini y J2ME, ya que ambas tecnologías están basadas en Java y comparten interfaces de comunicación similares basadas en los protocolos TCP/IP, por lo que se puede decir que un sistema Jini-J2ME, tiene la característica de tener también un alto grado de homogeneidad en cuanto al software utilizado para su implementación, facilitando así el desarrollo de aplicaciones en ambas tecnologías y su interacción entre ambas.
- El sistema de software construido en este proyecto para mostrar el acceso a servicios Jini a través de dispositivos J2ME, esta basado en una arquitectura “híbrida” ya que por un lado se ha mapeado la arquitectura GPRS a componentes, identificando previamente la funcionalidad requerida, respecto al soporte de los protocolos TCP/IP, para poder simularla. Por otro lado, se utilizó una arquitectura Jini en conjunto con la arquitectura Jini Surrogate, logrando así poder integrarlas en una sola arquitectura, con base en el hecho de que todas estas arquitecturas están soportadas sobre una arquitectura TCP/IP. La emulación del dispositivo real, la simulación de una arquitectura GPRS, así como el uso de herramientas de desarrollo de software gratuitas reduce significativamente el costo total del desarrollo del proyecto.
- Se proveen guías para el diseño e implementación de aplicaciones Jini y J2ME, con base en las aplicaciones desarrolladas en este proyecto.
- Los servicios Jini desarrollados para probar la arquitectura propuesta, son sólo un ejemplo de la posible potencialidad en el desarrollo de una nueva gama de servicios basados en la plataforma Java.
- El empleo de un sistema Jini-J2ME elimina la total dependencia de una computadora personal para el acceso e intercambio de información, haciendo uso de dispositivos móviles J2ME y de una red Jini.

9.3. Conclusiones

Con base en el desarrollo de este proyecto y los resultados obtenidos, se puede concluir lo siguiente:

- Este proyecto de tesis sitúa un punto de partida no solo para estudiantes y personas interesadas en áreas afines, sino que también puede estar dirigido a desarrolladores de servicios inalámbricos, para que pueden desarrollar sistemas similares al sistema propuesto en este proyecto.
- El sistema desarrollado en este proyecto tuvo por objetivo mostrar de manera experimental el acceso de un dispositivo móvil J2ME (teléfono celular) a los servicios provistos dentro de una red Jini, por lo que la implementación real del sistema puede verse afectada principalmente por: la infraestructura de comunicación móvil de datos utilizada por el proveedor de servicios de telefonía celular(v.gr. tiempo de acceso), el numero de dispositivos móviles J2ME que deseen acceder a los servicios Jini, la maquina virtual de Java (KVM) instalada en el dispositivo real, las características del dispositivo real respecto al hardware (v.gr. capacidad y recursos de procesamiento) y los propios servicios de la red Jini (diseño e implementación).
- El resultado de la simplificación del API de la arquitectura Jini Surrogate, no es una API equivalente a la propuesta por su desarrollador. Sin embargo, pueden ser agregadas o adaptadas nuevas funcionalidades al API Jini Surrogate propuesta en esta proyecto como los son: las funciones de seguridad (encriptación de paquetes), etc..
- El numero de computadoras utilizadas en este proyecto puede reducirse a un mínimo de una sola, basándose en el hecho del soporte genérico del sistema operativo a los protocolos TCP/IP, aunque de esto puede afectar el análisis del rendimiento respecto a la implementación real del sistema e incrementaría la necesidad de contar con una mayor cantidad recursos en esta computadora.
- Finalmente, este proyecto de tesis ha descrito guías para el análisis, diseño, implementación del acceso a servicios Jini a través de dispositivos móviles J2ME (teléfonos celulares).

Las principales aportaciones de proyecto de tesis son:

- Una arquitectura de red que permite el mapeo de una arquitectura de comunicación móvil de datos 2.5G como lo es GPRS, basada en el

análisis de las funcionalidades de dicha arquitectura, así como la integración con la arquitectura Jini.

- Proveer los componentes compatibles necesarios que cumplan con la funcionalidad identificada dentro de los subsistemas de la arquitectura mapeada, para asegurar un funcionamiento adecuado del sistema. Cualquier componente dentro de la arquitectura propuesta puede ser fácilmente remplazado por otro, siempre y cuando cumpla con las especificaciones, ya sea de J2ME, Jini o GPRS, así como de la pila de protocolos soportados por estas tecnologías.
- Se provee de un análisis comparativo de los principales herramientas de desarrollo de software con licencia gratuita, lo que permite identificar la funcionalidad de cada una poder ser elegida dependiendo de las necesidades del proyecto.

9.4. Trabajos futuros

La continua evolución de la tecnología y de las herramientas de desarrollo de software abren nuevas perspectivas en la interacción de las tecnologías Jini-J2ME.

- Servicios multimedia y otros tipos de servicios basados en la tecnología Java, pueden ser agregados en la federación Jini, para posteriormente ser accedidos a través de dispositivos móviles J2ME con las nuevas versiones de CLDC 1.0.4 y MIDP 2.0. Estos servicios también pueden implementar de manera robusta las capacidades de la tecnología Jini, como lo es el manejo de transacciones.
- El incremento o decremento del número de dispositivos móviles J2ME activos en el sistema puede ayudar a identificar de una manera mas concreta el comportamiento del sistema con un mayor o menor tráfico de datos en el acceso a los servicios Jini.
- La aparición de nuevas tecnologías y arquitecturas en el ámbito de las redes de telecomunicaciones móviles como UMTS, pueden ser simuladas con base en el mapeo a componentes de software que puedan soportar los protocolos base de dichas tecnologías, para poder desarrollar y probar nuevos servicios asociados a la correspondiente tecnología.

GLOSARIO

A

Ancho de banda

Cantidad de información que puede enviarse a través de una conexión. Usualmente, se mide en bits por segundo.

API

Application Programming Interface. Interfaz para la programación de aplicaciones, consiste en un conjunto de clases que sirven como referencia para implementar aplicaciones.

Aplicación

Programa diseñado para ayudar en la ejecución de una tarea específica.

AMPS

Advance Mobile Phone Service. Es un sistema estándar para la comunicación analógica entre dispositivos móviles, generalmente teléfonos celulares.

ASJJ

Acrónimo del título del Proyecto de Tesis: “Acceso a servicios Jini, a través de dispositivos móviles con tecnología J2ME bajo la configuración CLDC y el perfil MIDP”

B

BDPS

Base de Datos de Proyectos de Software.

Bps

Bits por segundo. Unidad de medida de la capacidad de transmisión por una línea de telecomunicación.

BTS

Base Transceiver Station. Estación base transmisor/receptor, también tiene propiedades de un transductor.

BSC

Base Station Controller. Controlador de estaciones base.

BSS

Base Station System. Sistema de estación base.

Bytecode

Código binario producto de la compilación de un programa escrito en el lenguaje de programación Java.

C**C++**

Es un lenguaje de programación orientado a objetos que puede ser visto como la evolución del lenguaje C.

C

Es un lenguaje de programación estructurado y procedimental que ha sido ampliamente por sistemas operativos y aplicaciones.

CDC

Connected Device Configuration es una configuración J2ME conocida dirigida hacia dispositivos con un cierto grado de capacidad computacional.

CDMA

Code-Division Multiple Access. Acceso múltiple por división de código. Es una forma de multiplexación que permite que varias señales de comunicación ocupen un mismo canal de datos, optimizando la disponibilidad del ancho de banda usado.

CDMA2000

También conocida como IMT-CDMA o IS-136, es una versión de CDMA desarrollada por la ITU, la cual puede soportar comunicaciones de datos con un ancho de banda entre los 144 kbps y los 2Mbps.

CDPD

Cellular Digital Packet Data. Es una especificación para el soporte de acceso inalámbrico a Internet y a otras redes públicas, trabaja sobre un ancho de banda promedio de 19.2Kbps.

CLDC

Connected Limited Device Configuration. Configuración para la Conexión de Dispositivos Limitados, esta es una configuración definida por la plataforma J2ME, en el cual se define capacidades de procesamiento y recursos en el dispositivo.

Cliente/Servidor

Es un modelo que describe la interacción entre dos elementos computacionales (programas). Se definen dos tipos de elementos: cliente el cual solicita un servicio a otro programa y el servidor es el que se encarga de satisfacer la petición hecha por un cliente.

Controlador Java

Interfaz de software programada con Java para interactuar con un dispositivo.

CORBA

Common Object Request Broker Architecture. Es una arquitectura y especificación para la creación, implementación y manejo de objetos distribuidos en una red. Permite a diferentes tipos de programas desarrollados por diferentes fabricantes poderse comunicar entre ellos. El concepto esencial de CORBA es el ORB (Object Request Broker) el cual da soporte a clientes para la comunicación con los servidores. CORBA es desarrollados por OMG.

CPU

Central Processing Unit. Es un termino utilizado para nombrar al procesador o microprocesador de una computadora.

CSD

Circuit-Switched Data es un sistema para la transmisión de datos de manera inalámbrica, desarrollado para un ancho de banda entre los 9Kbps y los 14.4 Kbps, compatible con GSM.

CVM

C Virtual Machine. Maquina virtual C, definida en la tecnología J2ME.

D**D-AMPS**

Digital-Advanced Mobile Phone Service, es una versión digital de AMPS, D-AMPS agrega el tipo de multiplexación TDMA a AMPS, por lo aumenta la capacidad tres veces de una canal AMPS.

DCOM

Distributed Component Object Model, similar a CORBA en términos de proveer un conjunto de servicios distribuidos a clientes, pero esta versión fue desarrollada por Microsoft. DCOM esta basado en COM, el cual provee de un conjunto de interfaces que permiten a clientes y servidores comunicarse dentro de una mismo computadora.

Dispositivo Móvil J2ME

Dispositivo móvil que soporta la tecnología J2ME.

Duplex

Se refiere a la comunicación entre dos elementos es recíproca, es decir, que ambos elementos pueden enviar y recibir datos al mismo tiempo.

E**E-business**

Comercio electrónico, se refiere a realizar compras a través de sitios de Internet.

EDGE

Enhanced Data GSM Environment, es una versión evolucionada de GPRS, y esta diseñada para un ancho de banda superior a los 384 Kbps, además de permitir transmisiones de datos multimedia y aplicaciones que requieren un ancho de banda considerable.

EIR

Equipment Identity Register. Registro de identificación de equipo.

EJB

Enterprise Java Beans, es una arquitectura que define un conjunto de componentes de software, desarrollados en el lenguaje de programación Java y que residen en los servidores de una red que utilizan el modelo cliente/servidor. EJB esta construido con base en la tecnología JavaBeans.

F**Federación Jini**

Es un conjunto de comunidades o grupos de usuarios y servicios Jini.

Firewall Sistema que se coloca entre una red local e Internet. La regla básica es asegurar que todas las comunicaciones entre dicha red e Internet se realicen conforme a las políticas de seguridad de la organización que lo instala. Además, estos sistemas suelen incorporar elementos de privacidad, autenticación, etc.

FTP File Transfer Protocol. Protocolo de Transferencia de Archivos. Uno de los protocolos de transferencia de archivos definido por la arquitectura TCP/IP.

G

Gateway

Pasarela. Puerta de Acceso. Dispositivo que permite conectar entre si, dos redes normalmente de distinto protocolo o un host a una red

GGSN

Gateway GPRS Support Node. Nodo de soporte pasarela GPRS.

GPRS

General Packet Radio Service. Servicio General de Radio Paquetes Es un estándar para las comunicaciones inalámbricas, basado en paquetes de datos para la comunicación inalámbrica, el cual ha sido desarrollado para anchos de banda entre los 56 kbps y los 114 kbps, este tipo de tecnología permite a los usuarios conectarse a Internet de manera continua. GPRS esta basado en GSM.

GPS

Global Positioning System, consiste en una conjunto de 24 satélites que orbitan el planeta tierra, por los cuales se puede identificar una la ubicación geográfica de cualquier persona u objeto que contenga un aparato compatible con este sistema.

Grupo Jini

Conjunto de usuarios y servicios Jini.

GSM

Global Systems for Mobile Communications, abreviación en ingles de Sistemas Globales de Comunicaciones Móviles, es un sistema digital para teléfono móviles que utiliza una variación del tipo de multiplexación TDMA, es el sistema con mayor grado de utilización por los usuarios de telefonía celular en el mundo.

H

HLR

Home Location Register.

HotSpot

Software de soporte para la tecnología Java.

Host

Anfitrión. Computador conectado a Internet. Computador en general

HSCSD

High-Speed Circuit-Switched Data es un sistema para la transmisión de datos de manera inalámbrica, desarrollado para un ancho de banda superior a los 38 Kbps, compatible con GSM.

HTML

Hyper Text Markup Language. Lenguaje utilizado para crear los documentos de hipertexto que se emplean en la Internet. Los documentos HTML son simples archivos de texto que contienen instrucciones (llamadas tags) entendibles por el Navegador (Browser).

HTTP

Hypertext Transport Protocol. Protocolo de Transferencia de Hipertexto. Es este el protocolo de Internet usado para transferir archivos desarrollados en el lenguaje HTML.

I

IEEE

Institute of Electrical and Electronics Engineers. Instituto de Ingenieros Electrónicos y Eléctricos.

IP

Internet Protocol. Protocolo de Internet.

ITU

International Telecommunications Union. Unión Internacional de Telecomunicaciones.

J

JavaSpaces

Tecnologías desarrollada por Java basada en espacios, los cuales contienen comportamientos definidos de un sistema.

J2EE

Java 2 Enterprise Edition. Java 2 Edición Empresarial. Plataforma y tecnología orientada al desarrollo de servicios de Internet y aplicaciones distribuidas mediante EJB (Enterprise Java Beans) y tecnologías relacionadas, con APIs para la gestión de transacciones, persistencia de objetos, servicios de nombres, XML, autenticación, etc.

J2SE

Java 2 Standard Edition. Java 2 Edición Estándar, es una plataforma Java, dirigido a computadoras con capacidades y recursos autónomos de procesamiento, para el desarrollo de aplicaciones Java, esta plataforma provee de una maquina virtual de Java y de un API.

J2ME Java 2 Micro Edition. Java 2 Micro Edición, es una plataforma Java, dirigido hacia los dispositivos móviles limitados en capacidad de procesamiento, memoria y despliegue de datos, esta plataforma los provee de una maquina virtual de Java reducida y de un API.

Java

Es una tecnología que esta definida para varias plataformas de cómputo y provee de un lenguaje de programación de alto nivel desarrollado por SUN Microsystems.

JavaBeans

Es una interfaz de programación orientada a objetos desarrollada por Sun Microsystems que permite la reutilización de aplicaciones o programas llamados componentes independiente de la plataforma del sistema operativo.

Jini

Es una tecnología extendida de la plataforma Java, que facilita la integración de los sistemas distribuidos, que sirve para desarrollar y utilizar servicios en una red computacional con base en un API.

Jini surrogate

Arquitectura por la cual un dispositivo que no cuenta con recursos de procesamiento necesarios para acceder a los servicios Jini, logra accederlos a través de un surrogate host.

JMF

Java Media Framework, es un API de referencia para la implementación de aplicaciones multimedia en Java.

JVM

Java Virtual Machine, es una implementación de la especificación de la maquina virtual de Java, la cual es un intérprete del bytecode de Java para el procesador de una computadora, por lo que Java fue diseñado para poder ejecutarse en cualquier plataforma de procesadores.

K**Kbps**

Kilobits por segundo. Unidad de medida de la capacidad de transmisión por una línea de telecomunicación. Cada kilobit está formado por mil bits.

KHz

KiloHertz. Medida de frecuencia.

KVM

K Virtual Machine. Maquina virtual K definida por la tecnología J2ME.

L**LAN**

Local Area Network. Red de área local.

Linux Versión de libre distribución del sistema operativo UNIX; fue desarrollada por Linus Torvald

Log

Archivo de registro de resultados.

Lookup service

Es un elemento de la infraestructura Jini, el cual sirve como repositorio de los servicios, al contener los proxies de los servicios. Las entradas del lookup service son objetos implementados en el lenguaje de programación Java, éstos objetos pueden ser descargados como parte de la operación del lookup service, y actúa como un proxy

local del servicio que se encuentra registrado en el lookup service. El lookup service refleja los miembros que actualmente se encuentran en la federación y actúa como un centro de reunión de los miembros de la federación, donde se ofrecen y se encuentran servicios.

M

Mbps

Megabits por segundo. Unidad de medida de la capacidad de transmisión por una línea de telecomunicación. Cada megabit está formado por un millón de bits.

MIDlet

Aplicación definida por J2ME para ser ejecutada en dispositivos móviles que cuentan con esta tecnología.

MHz

MegaHertz. Unidad de frecuencia.

Middleware

En términos generales es una capa formada por componentes de software que permiten interactuar a capas anteriores y posteriores.

MIDP Mobil Information Device Profile. Abreviación en inglés de Perfil de Dispositivos Móviles de Información, éste perfil es definido por la plataforma J2ME, define características referentes a interfaz gráfica, conectividad, entre otras cosas.

Modem

Es un modulador/demodulador que actúa como un convertidor de señales digitales a analógicas y viceversa.

MS

Mobile Station. Estación móvil.

MSC

Mobile Switching Center. Centro móvil de conmutación.

Multicast

Es una forma de transmisión definida por la arquitectura TCP/IP, los datos sólo se envían una vez desde el servidor, sin importar el número de receptores, y estos datos sólo se envían a aquellas partes de la red donde haya usuarios interesados en

recibirlos. Por tanto la red no está sobrecargada con un mismo envío. La Multidifusión IP puede utilizarse en una red física o a través de una red de redes, en cuyo caso los ruteadores Multicast especiales envían los datagramas de Multidifusión. Un ruteador de Multicast envía los paquetes IP a otras redes físicas conforme sea necesario además modifica el campo Tiempo de Vida de los datagramas de Multidifusión para limitar los entre ruteadores

O

OMG

Object Management Group. Grupo de Manejo de Objetos.

P

Palm

Dispositivos electrónicos con un cierto grado de procesamiento.

Pager

Es un dispositivo pequeño para la comunicación inalámbrica que puede recibir (en algunos casos también puede transmitir) mensajes reducidos de texto.

PC

Personal Computer, Computadora personal.

PCS

Personal Communications Services, a todo el conjunto de tecnologías para la comunicación móvil inventadas a finales de 1990 son nombradas con este nombre, tales como: GSM, HSCSD, GPRS, etc.

PDA

Personal Digital Assistant, es un término utilizado para referirse a dispositivos pequeños que cuentan con una cierta capacidad para el procesamiento de datos.

PDC

Personal Digital Celular, sistema de comunicación móvil desarrollado en Japón.

Plug and Play o Plug and Work

Concepto utilizado para hacer referencia cuando un nuevo dispositivo se conecta a una red computacional o a una computadora sin necesidad de reiniciar el sistema.

PNG

Portable Network Graphics, es un formato de compresión de imágenes.

Proxy

Es un objeto lógico programado, el cual contiene información necesaria y suficiente (propiedades y atributos) del objeto referenciado, que permite invocar y ejecutar en una computadora diferente, los métodos del objeto invocado.

Proxies

Plural de proxy (véase proxy).

PSI

Proveedor de Servicios de Internet.

PSTN

Public Switched Telephone Network, es un conjunto de redes telefónicas publicas orientadas a la transmisión de datos de voz.

R**RAM**

Random Access Memory. Memoria de acceso aleatorio de tipo volátil o temporal.

Red Jini

Red computacional de área local, vista como un sistema distribuido, que integra servicios, basada en la tecnología Jini.

Roaming

Es un servicio por el cual se tiene acceso a los servicios provistos por el proveedor sin importar la ubicación geográfica del cliente.

ROM

Read-Only Memory, memoria de solo lectura.

RPC

Remote Procedure Call es un protocolo por el cual, un programa puede solicitar un servicio de otro programa localizado en otra computadora dentro de una red sin tener que entender los protocolos de red. RPC esta basado en el model cliente/servidor.

RMI

Remote Method Invocation. Es una arquitectura desarrollada con base en Java, para desarrollar aplicaciones con objetos distribuidos.

RTMS

Radio Telephone Mobile System. Sistema móvil radio teléfono, desarrollado en Italia.

Runtime

Es referido al estado de un programa en ejecución.

S**SDASJ**

Sistema Demostrativo de Acceso a Servicios Jini.

SDK

Software Development Kit. Herramienta para el desarrollo de software.

Servicio fijo

Es un atributo del servicio, que se refiere a un servicio el cual no fue diseñado para ser accedido de un dispositivo móvil.

Servicio móvil

Es un atributo del servicio, que se refiere a un servicio el cual fue diseñado para ser accedido de un dispositivo móvil.

SGSN

Serving GPRS Support Node. Nodo de soporte para servicios GPRS.

SIM

Subscribe Identity Module. Modulo de identificación del subscriber.

Sistema Jini

Conjunto de elementos definidos en la arquitectura Jini.

SMS

Short Message Service, es un servicio para el enviar mensajes que contienen entre 160 y 224 caracteres.

Socket

Es un punto de comunicación por el cual un proceso puede emitir o recibir información, de manera local o remota.

SQL

Structured Query Language. Abreviación en Ingles de Lenguaje Estructurado de Solicitud.

SSL

Secure Socket Layer es un protocolo utilizado para el manejo de seguridad (encriptación) en la transmisión de datos por sockets.

Surrogate host

Computadora que provee de recursos de procesamiento necesarios para acceder a servicios Jini a dispositivos que no cuentan con dichos recursos.

T**Transceiver**

Es una combinación de transmisor/receptor/transductor, el cual recibe datos por ondas electromagnéticas y envía datos a través de medios como cables o fibra óptica.

TCP

Transmission Control Protocol. Abreviación en inglés de Protocolo de Control de Transmisión, es un protocolo definido dentro de la pila de protocolos TCP/IP.

TCP/IP

Transmisor Control Protocol/Internet Protocol. Pila definida de protocolos para establecer una comunicación entre dos computadoras. TCP/IP: (). Conjunto de protocolos que definen a la Internet

TDMA

Time Division Multiple Access, Acceso múltiple por división de tiempo es una tecnología basada en la multiplexación de señales por división de tiempo, utilizada por teléfonos celulares digitales.

U**UDP**

User Datagram Protocol, es un protocolo de computación que ofrece un intercambio de información con base en datagramas, definido dentro de la arquitectura TCP/IP.

Unicast

Forma de transmisión definida por la arquitectura TCP/IP, (el más habitual en Internet) envía los datos sólo a aquellas partes de la red donde haya usuarios/as interesados en recibirlos. En este sentido es más eficiente que el broadcast. Sin embargo, el emisor tiene que enviar una copia para cada receptor, sobrecargando la red con copias de los mismos datos.

UMTS

Universal Mobile Telephone Service, es una versión de W-CDMA para compatible con GSM para la transferencia de datos, con un ancho de banda superior a los 144 Kbps.

UPAEP

Universidad Popular Autónoma del Estado de Puebla.

V**VLR**

Visitor Location Register. Registro de ubicación de visitantes.

VxWorks

Sistema operativo de tiempo real.

W

W-CDMA

Wideband-CDMA, es una versión de CDMA desarrollado con un mayor ancho de banda superior a los 144 Kbps.

WAN

Wide Area Network. Red de área mundial.

WAP

Wireless Application Protocol, es un protocolo estandarizado para la comunicación de dispositivos móviles o inalámbricos como los teléfonos celulares, el cual es utilizado en términos generales para el acceso a Internet.

Windows

Sistema operativo desarrollado por la empresa Microsoft para PCs.

Windows NT

Sistema operativo de red desarrollado por la empresa Microsoft para PCs.

WML

Wireless Markup Language, es un lenguaje que permite desarrollar paginas de Internet especialmente para dispositivos móviles que soportan el protocolo WAP, como lo son los teléfonos celulares y las PDAs.

BIBLIOGRAFÍA

Libros

- [ADOO] Grady Booch, Análisis y Diseño Orientado a Objetos, 2da edición, Addison Wesley, México, 1996. ISBN:968-444-352-8.
- [CJMT] John W. Muchow, Core J2ME Technology & MIDP, Prentice Hall, USA, 2001. ISBN: 0130669113
- [CRJN] W. Keith Edwards, Core Jini, USA, Prentice Hall, 1999. ISBN 0-13-014469-X
- [DJAJ] Hinkmond Wong, Developing Jini Applications Using J2ME Technology, Addison-Wesley, USA, 2002. ISBN: 0201702444
- [EGWC] Andy Dornan, The Essential Guide to Wireless Communications Applications, Prentice Hall, USA, 2001. ISBN:0-13-031716-0
- [J2NT] Kim Topley, J2ME in a Nutshell, O'Reilly, USA, 2002. ISBN:0-596-00253-X
- [J2PG] Francisco Javier Cevallos, Java 2 Curso de Programación, Alfaomega, México, 2000. ISBN:970-15-0604-9
- [J2PR] H. M. Deitel y P.J. Deitel, Como Programar en Java, Prentice Hall, México, 1998. ISBN:970-17-0044-9
- [JAV1] Java, Prentice Hall, España, 1997. ISBN: 84-89660-59-X
- [JITO] S. Ilango Kumaran, Jini Technology: An overview, Prentice Hall, USA, 2002. ISBN: 0-13-033385-9
- [MLGT] Manual de Lineamientos Generales de Titulación, Departamento de Sistemas, Universidad Popular Autónoma del Estado de Puebla, 2002
- [MIDP] Enrique Ortiz, Mobile Information Device Profile for J2ME: Professional Developers Guide, Wiley, USA, 2001. ISBN: 0471034657
- [PFJN] Sing li, Professional Jini, Wrox Press, USA, 2000. ISBN: 1861003552

- [QSPM] Robert Futrell y otros, Quality Software Project Management, Software Quality Institute Series, Prentice Hall, USA, 2002. ISBN: 0-13-091297-2
- [SOPM] Andrew S. Tanenbaum, Sistemas Operativos Modernos, Prentice Hall, México, 1993. ISBN:968-880-323-5
- [TERC] Uyless Black, Tecnologías Emergentes para redes de computadoras, Edición en español, Prentice Hall, 1999. ISBN: 970-17-0268-9
- [UMLM] James Rumbaugh y otros, El Lenguaje Unificado de Modelado, Manual de Referencia, Addison Wesley, España, 2000. ISBN:84-7829-037-0
- [UMLP] Craig Larman, UML y Patrones, Prentice Hall, MÉXICO, 1999. ISBN:970-17-0261-1

Documentos en Línea *

- [e_3GST] 3G Americas, estadísticas acerca de las tecnologías móviles
<http://www.3gamericas.org/Spanish/Statistics/>
- [e_BJBL] Borland, JBuilder
<http://www.borland.com/jbuilder/index.html>
- [e_BJMS] Borland, MobileSet
<http://www.borland.com/jbuilder/mobileset/index.html>
- [e_CWMT] Metrowerks, Code Warrior Wireless Studio 7
http://www.metrowerks.com/MW/Develop/Wireless/Wireless_Studio/default.htm
- [e_CCCM] Celulares en México
<http://www.yucatan.com.mx/especiales/celular/celularesenmexico.asp>
- [e_CLDC] Connected Limited Device Configuration (CLDC), especificaciones
<http://java.sun.com/products/cldc>
- [e_CNIW] Cellular networks in the world
<http://www.cellular-news.com/coverage>

* Ultima visita: Febrero 16, 2003

- [e_COFE] Comisión Federal de Telecomunicaciones
<http://www.cft.gob.mx/>
- [e_EWC] Wireless Communications Evolution
<http://www.itu.int/osg/spu/ni/3G/technology/>
- [e_GARC] GPRS Architecture
<http://www.motorola.com/networkoperators/GPRS-Architecture.htm>
- [e_GARP] GPRS Architecture and Protocol Stack
<http://www.dcs.gla.ac.uk/~lewis/teaching/Tik-111.htm>
- [e_GSMA] GSM Architecture
<http://www.motorola.com/networkoperators/GSM-Architecture.htm>
- [e_GSMR] GSM Arquitectura y funcionamiento
<http://telcel-gsm.com/pages/gsmCostaRica.htm>
- [e_GSMW] GSM World
<http://www.gsmworld.com/technology/gsm.shtml>
- [e_HTCE] Historia de la telefonía celular
http://telecom.fi-b.unam.mx/Telefonia/Telefonia_Celular2.htm
- [e_IEEE] Institute of Electrical and Electronics Engineers – Standard Association.
<http://standards.ieee.org/>
- [e_IUSA] IUSACELL PCS
<http://www.iusacell.com.mx>
- [e_ITU] International Telecommunication Union, estándares.
<http://www.itu.int/>
- [e_ISOS] ISO/OSI and TCP/IP Network Models
http://www.uwsg.iu.edu/usail/network/nfs/network_layers.html
- [e_J2ME] Java 2 Micro Edition Technologies
<http://java.sun.com/j2me>
- [e_JACP] Java Community Process
<http://www.jcp.org/>

[e_JATR]	Java Technology Resource http://java.sun.com
[e_JDST]	JDataStore SQL reference http://info.borland.com/techpubs/jbuilder/jbuilder6/datastore/sql_ref.html
[e_JINI]	Jini Community Resource http://www.jini.org
[e_JMFB]	Java Media Framework Linux version http://www.blackdown.org/java-linux/jdk1.2-status/jmf-status.html
[e_JMFL]	Java Media Framework on Linux http://java.sun.com/products/java-media/jmf/2.1.1/jmf-linux.html
[e_JNWP]	Jini for Mobile Devices Paper Research http://www.cs.rit.edu/~anhinga/whitepapers/JiniMEWhitePaper/printable.html
[e_JRMI]	Java RMI Specification http://java.sun.com/j2se/1.4/docs/guide/rmi/spec/rmiTOC.html
[e_JSUR]	Jini Surrogate Project http://surrogate.jini.org/
[e_MEGR]	Megared, Proveedor de servicios de Internet http://www.pub.megared.com.mx
[e_MIDP]	Mobile Information Device Profile (MIDP), especificaciones y referencias http://java.sun.com/products/midp
[e_MIDS]	MIDP Sockets Implementation http://wireless.java.sun.com/midp/articles/socketRMI/
[e_MISD]	Motorola IDEN SDK http://idenphones.motorola.com/iden/developer/developer_home.jsp
[e_NK3G]	Comunicaciones Móviles de 3era Generación http://www.nokia.es/3g/what_is.html
[e_PALM]	PALM Emulator http://www.palmos.com/dev/tools/emulator/

- [e_PEGA] PEGASO PCS
<http://www.pegasopcs.com.mx>
- [e_PNGD] Portable Network Graphics (PNG) Creator Site
<http://www.w3.org/Graphics/PNG/>
- [e_PNGE] Portable Network Graphics (PNG) Encoder
<http://catcode.com/pngencoder/>
- [e_PNGS] Portable Network Graphics (PNG) Developer Site
<http://www.libpng.org/pub/png/png.html>
- [e_PROT] Prodigy Telmex, Proveedor de servicios de Internet
<http://www.prodigy.com.mx>
- [e_SEEM] Sonyericsson device emulators
http://www.ericsson.com/mobilityworld/sub/open/technologies/java/tools/java_sonyericsson_device_emulators
- [e_SMJ2] Sun Microsystems, Java 2 Standard Edition SDK
<http://java.sun.com/j2se/>
- [e_SMJI] Sun Microsystems, Jini Technology Starter Kit
<http://developer.java.sun.com/developer/products/jini/>
- [e_SMJM] Sun Microsystems, Java Media Framework (JMF) SDK
<http://java.sun.com/products/java-media/jmf/>
- [e_SMOV] Servicios móviles de valor agregado
<http://www.tid.es/presencia/publicaciones/comsid/esp/articulos/vol812/smoviles/smoviles.html>
- [e_SMWT] Sun Microsystems, J2ME Wireless Toolkit
<http://java.sun.com/products/j2mewtoolkit/>
- [e_TELC] Telcel
<http://www.telcel.com>
- [e_TGSM] Telcel GSM
<http://www.telcel.com/gsm/>
- [e_TMSM] Telefonía celular en México
http://www.latamnews.com/telefiniacelularmexico_050702.html

- [e_TMST] Telefónica MoviStar
<http://www.telefonicamoviles.com.mx>
- [e_UMLT] Tutorial de UML
<http://www.dcc.uchile.cl/~psalinas/uml/>
- [e_UMTS] UMTS Forum
<http://www.ums-forum.org/>
- [e_UNEF] UNEFON
<http://www.unefon.com.mx>
- [e_UTUT] UML Tutorial
http://www.sparxsystems.com.au/UML_Tutorial.htm
- [e_WCCM] World Cellular Coverage maps for all networks: GSM, TDMA, CDMA, UMTS.
<http://www.gsmcoverage.co.uk/>
- [e_WSUN] Wireless Sun
<http://wireless.java.sun.com>
- [e_WRCH] Wireless Communication History
<http://www.rpi.edu/~shaha/honsem/history.htm>

Documentos electrónicos²

- [d_AJTE] Steve Hashman, Steven Knudsen, The Application of Jini Technology to Enhance the Delivery of Mobile Services, PsiNaptic Inc. 2001. (Documento electrónico PDF).
- [d_BDWA] Bill Day, Developing Wireless Applications using the Java 2 Platform, Micro Edition, Technology Evangelist, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_BDWD] Bill Day, Wireless Development for Open Networks, Chief Wireless Technology Evangelist, Sun Microsystems, Inc. 2002. (Documento electrónico PDF).
- [d_BDWJ] Bill Day Developing Wireless Applications Using Java Technology, Sun Microsystems, Inc. 200. (Documento electrónico PDF).

² Los documentos electrónicos pueden ser consultados en el CD-ROM que acompaña a este documento.

- [d_BJCT] Bill Day, Jini Connection Technology, Architecture Overview, Technology Evangelist, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_CLD3] Connected Limited Device Configuration (CLDC) v.1.0.3., referencia de implementación, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_CLD4] Connected Limited Device Configuration (CLDC) v.1.0.4., referencia de implementación, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_CPCJ] Sean Landis, Cell Phone Computing with Jini Network Technology, Distinguished Member of the Technical Staff, Motorola Labs for Java One Conference, Sun Microsystems, Inc. 2002. (Documento electrónico PDF).
- [d_GPIP] Internet, IP Networks and GPRS, Internet Engineering Task Force Organization. 2001. (Documento electrónico PDF).
- [d_HDJA] Hinkmond Wong, Developing Java 2 Platform, Micro Edition Applications Using Jini Technology, Staff Engineer, Sun Microsystems, Inc. 2000. (Documento electrónico PDF).
- [d_JIN1] Jini v.1.2.1_001, Jini Technology Core Platform Specification, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_JIN2] Jini v.1.2.1_001, Jini Architecture Specifications, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_JIN3] Jini v.1.2.1_001, Jini Device Architecture Specification, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_JIN4] Jini v.1.2.1_001, A Collection of Jini Technology Helper Utilities and Services Specification, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_JINT] Jini Introduction, Fakultet for informasjonsteknologi, matematikk oglekroteknikk e Institutt for telematikk, 2002. (Documento electrónico PDF).
- [d_JNTA] Jini Network Techonology, Advanced Technology Group, Digital India, 200, White Paper, 2001. (Documento electrónico PDF).

- [d_MIR1] Mobile Information Device Profile (MIDP) v.1.0.3., Reference Implementation Porting Guide, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_MIR2] Mobile Information Device Profile (MIDP) v.2.0., Reference Implementation, Sun Microsystems, Inc. 2002. (Documento electrónico PDF).
- [d_MPWD] Jeffery Hackert, Mark Karmelich and Jon Strabala, MIDP Programming for Wireless Devices, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_OTAM] Over The Air User Initiated Provisioning Recommended Practice for the Mobile Information Device Profile, Version 1.0, 2001. (Documento electrónico PDF).
- [d_PJAR] Pradip Lamsal, J2ME architecture and Related embedded Technologies, Paper, 2002. (Documento electrónico PDF).
- [d_PJPM] Peeter Paal, Java 2 Platform Micro Edition, Helsinki University of Technology Telecommunications Software and Multimedia Laboratory, 2000. (Documento electrónico PDF).
- [d_SDIN] Universidad Carlos III de Madrid, Introducción a los sistemas distribuidos, (Documento electrónico PDF).
- [d_SJSA] Sang Shin, Jini Surrogate Architecture, Technology Evangelist, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_SJUP] Salutation Architectures and the newly defined service discovery protocols from Microsoft and Sun, (Documento electrónico PDF).
- [d_SODT] Fernando Pérez, Sistemas operativos distribuidos, (Documento electrónico PDF).
- [d_SWAC] Teresa A. Dahlberg, Survivable Wireless Access, (Documento electrónico PDF).
- [d_WJPS] David Roger Nordstedt, The Wireless Jini Projection Service, High Honors Report, Computer and Information Science and Engineering, Department of CISE, University of Florida, Gainesville, FL. Fall 2000. (Documento electrónico PDF).

- [d_WTSS] Wireless Technology Summit, Sun Microsystems, Inc. 2001. (Documento electrónico PDF).
- [d_YGIM] Yael Gavish, Implementing MIDP Communication Using WAP as Bearer, Member of Technical Staff, Sun Microsystems, Inc. 2002. (Documento electrónico PDF).

Artefactos de software³

- [a_ERS] Especificación de Requerimientos del Software (Ver apéndice E)
- [a_PACS] Plan de Aseguramiento de la Calidad del Software (Ver apéndice D)
- [a_PAP] Plan de Administración del Proyecto (Ver apéndice B)
- [a_PCAP] Plan de Configuración de la Administración del Proyecto (Ver apéndice C)
- [a_PPS] Plan de Pruebas del Software (Ver apéndice F)
- [a_PTP] Plan de Trabajo del Proyecto (Ver apéndice A)

³ Los artefactos pueden ser consultados en el CD-ROM que acompaña a este documento y en cual están referidos como apéndices.

CONTENIDO DEL CD-ROM

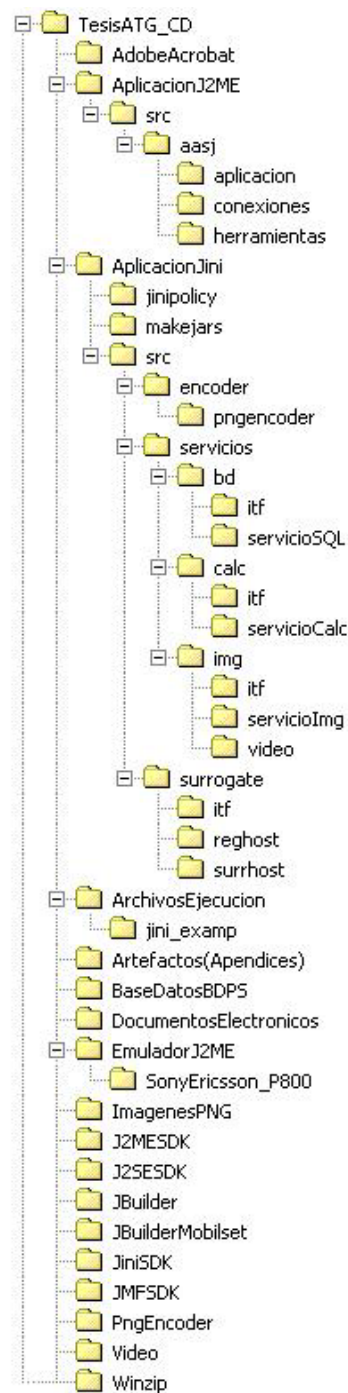


Figura 62. Contenido Tesis CD-ROM.

El contenido del CD-ROM visto en la figura 1, es descrito a continuación:

- AdobeAcrobat: Adobe Acrobat reader v.5.0, software que permite acceder a archivos en formato PDF.
- AplicacionJ2ME: Código fuente de la aplicación MIDlet.
 - src
 - aasj
 - herramientas
 - aplicacion
 - conexiones
- AplicacionJini: Código fuente de la aplicación Jini (servicios y surrogate host).
 - jinipolicy (política de ejecución Jini)
 - makejars
 - src
 - encoder (codificador PNG)
 - pngencoder
 - servicios (servicios Jini)
 - bd
 - itf
 - servicioSQL
 - calc
 - itf
 - servicioCalc
 - img
 - itf
 - servicioImg
 - video
 - surrogate (funciones surrogate host)
 - itf
 - reghost
 - surrhost
- ArchivosEjecucion: Archivos de ejecución de la aplicación Jini.
- Artefactos(Apéndices): Artefactos de software desarrollados en el proyecto y guía de usuario.
- BaseDatosBDPS: Archivos de la base de datos BDPS.

- DocumentosElectronicos: Todos los documentos electrónicos mencionados en la bibliografía.
- EmuladorJ2ME:
 - SonyEricsson_P800: Archivos ejecutables del emulador y manual de instalación.
- ImagenesPNG: Imágenes en formato PNG.
- J2MESDK: Sun Microsystems, J2ME Wireless Toolkit v.1.0.4_01, y archivo de manual de instalación.
- J2SESDK: Java 2 Enterprise Edition SDK v.1.4.0_03 y archivos de manual de instalación.
- JBuilder: Borland JBuilder v.7.0. y archivo del manual de instalación.
- JBuilderMobilset: Borland JBuilder Mobileset v.3.01 y archivo del manual de instalación.
- JiniSDK: Sun Microsystems, Jini Technology Starter Kit v. 1.2.1_001 y archivo del manual de instalación.
- JMFSDK: Sun Microsystems, Java Media Framework v.2.1.1b y archivo del manual de instalación.
- PngEncoder: Contiene al API para la implementación del codificador de imágenes al formato PNG.
- Video: Archivo de video en formato MPEG.
- Winzip: Winzip Computing, Winzip v.8.1. software que permite acceder a archivos en formato ZIP.

Advertencia: La licencia de uso del software contenido en el CD-ROM puede ser gratuita, de libre distribución o de evaluación, por lo que se recomienda visitar la pagina de Internet del fabricante, para verificar los términos de la licencia de uso. Pueden existir nuevas versiones del software mencionado anteriormente, por lo que queda a criterio de la persona que requiera replicar la implementación del sistema, acceder a las paginas de los fabricantes correspondientes y obtener las nuevas versiones de software. El uso de nuevas versiones del software anteriormente mencionado no garantiza un adecuado funcionamiento del sistema.