Agentes Java para Dispositivos Móviles con Conexión Bluetooth*

J. A. Royo**, E. Mena, and R. Acero

Departamento de IIS, Univ. de Zaragoza
Maria de Luna 3, 50018 Zaragoza, España
{joalroyo, mena}@unizar.es, roaceroc@prometeo.cps.unizar.es

Resumen La tecnología de agentes móviles cobra especial importancia en entornos inalámbricos, donde la computación y los recursos en general pueden llevarse allá donde sean necesarios.
Sin embargo, actualmente existe un salto tecnológico entre los requerimientos software de una plataforma de agentes móviles y las prestaciones de los dispositivos móviles actuales, incluso los PDAs (Personal Data Assistant) más avanzados. Entre los problemas a resolver podemos citar los protocolos de comunicación inalámbrica, las limitaciones de los sistemas operativos preinstalados, la capacidad de almacenamiento, etc.
En este trabajo presentamos el tipo de tecnología, basada en Java, Linux y software libre, que hemos utilizado para construir sistemas de agentes móviles sobre PDAs con conexión Bluetooth. En estos sistemas los agentes móviles son capaces de viajar a un PDA, a través del enlace inalámbrico, desde un PC de sobremesa o incluso desde otro PDA.

Palabras clave: Java en dispositivos móviles, Bluetooth, agentes móviles.

1 Introducción

Actualmente los dispositivos inalámbricos tales como PDAs están adquiriendo un gran protagonismo en la vida cotidiana. Sobre todo en entornos empresariales donde la movilidad es una necesidad, entornos asociados a comerciales, directivos de empresas, etc. Además no basta con que estos dispositivos tengan capacidad de almacenamiento sino que además deben poseer una elevada capacidad de interconexión.

En nuestro caso nos vamos centrar en la utilización de la tecnología de agentes móviles en PDAs que utilizan una conexión inalámbrica para conectarse con otros dispositivos, bien sean otros dispositivos inalámbricos u ordenadores de sobremesa que les permitan acceder a internet. A lo largo del artículo presentaremos cuáles son las ventajas de los agentes móviles

* Trabajo subvencionado por los proyectos CICYT TIC2001-0660 y DGA P084/2001.
** Trabajo subvencionado por la beca B131/2002 del Gobierno de Aragón y el Fondo Social Europeo.

cuando se están ejecutando en entornos heterogéneos y distribuidos con conexiones inalámbricas, que por su propia naturaleza son más lentas e inestables que las cableadas.

El contexto en el que nos encontramos presenta fuertes restricciones en cuanto a la capacidad de almacenamiento, procesamiento, visualización e interconexión, que son muy limitados. Es decir, nos encontramos ante un dispositivo de pequeño tamaño al que le vamos a exigir unas altas prestaciones. Debido a la utilización de los agentes móviles se pueden reducir los requisitos que debe poseer el PDA porque podemos trasladar la ejecución de los programas a un ordenador de sobremesa. Por ejemplo, como la tecnología de agentes móviles realiza una menor utilización de la red el ancho de banda necesario es menor. Además la conexión puede ser más inestable debido a que los agentes utilizan conexiones asíncronas y por lo tanto la utilización de la red es más puntual. Sin embargo para poder instalar una plataforma de agentes necesitamos poder ejecutar código Java 2 [32] y una conexión de red TCP-IP, tal como se muestra en la Figura 1. Para utilizar agentes móviles necesitamos Java 2 y una conexión TCP-IP y para poder instalar una máquina virtual que soporte Java 2 necesitamos que el sistema operativo sea Linux Familiar. Esto nos impone una restricción más si queremos configurar una red TCP-IP en un sistema Linux que utilice el protocolo Bluetooth, que es la utilización de Bluez como implementación de la pila Bluetooth [25].

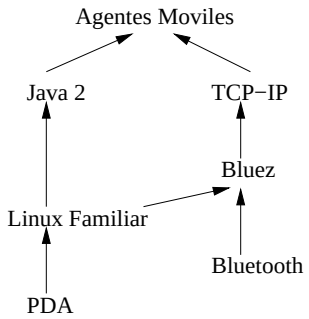

Figura 1. Restricciones para la instalación de una plataforma de agentes en un PDA

El resto del artículo está estructurado como sigue: en la sección 2 introducimos brevemente la tecnología de agentes móviles, su relación con Java y los requisitos software para disponer de dicha tecnología. En la sección 3 comenzamos por enumerar las características de los principales dispositivos móviles y los sistemas operativos (SO) de que se dispone

así como las distintas plataformas Java que podemos utilizar sobre ellos. Finalmente se muestran las mejores combinaciones PDA, Sistema Operativo y versión de Java, destacando sus ventajas e inconvenientes. En la sección 4 describiremos brevemente el procedimiento para crear una conexión TCP-IP en un dispositivo utilizando una conexión Bluetooth. En la sección 5 enumeramos distintos detalles de nuestra experiencia con la utilización de agentes en entornos inalámbricos, sus ventajas e inconvenientes, así como los principales problemas encontrados. En la sección 6 se citan algunos trabajos relacionados con este artículo. Finalmente, en la sección 7 se resumen las principales aportaciones de nuestro trabajo así como el trabajo futuro sobre este contexto.

2 Agentes móviles

En esta sección describiremos brevemente los aspectos más destacables de la tecnología de agentes móviles, así como las características que les hacen interesantes frente a soluciones basadas en una aproximación cliente/servidor.

Comenzaremos por definir que un *agente software* [13,28] es un módulo software que se ejecuta en un cierto contexto de ejecución o *place* creado utilizando un *sistema de agentes*. Un Sistema de Agentes [24] es una plataforma que puede crear interpretar, ejecutar, transferir y liberar agentes. Un agente posee las siguientes propiedades principales[1]: 1) *autonomía*, posee el control sobre sus propias acciones; 2) *finalidad*, gestiona una agenda de objetivos; 3) *cooperación*, un agente es capaz de comunicarse con otros agentes; 4) *aprendizaje*, cambia su comportamiento de acuerdo a su experiencia previa; 5) *movilidad*, puede viajar de un ordenador a otro (en realidad, de un *place* a otro); 6) *reactividad*, siente los cambios de su entorno y reacciona ante ellos; y 7) *persistencia*, para poder interrumpir su ejecución y continuarla más adelante.

2.1 Agentes móviles en Java

En lo referente a agentes móviles, para que un agente pueda trasladar su ejecución de un ordenador a otro necesita que su código pueda ser ejecutado en ambos ordenadores. Es decir los agentes móviles necesitan utilizar un lenguaje que sea multiplataforma, como es Java, o utilizar un lenguaje interpretado. Una de las ventajas de utilizar Java es que la mayor parte del código esta precompilada y por lo tanto es más eficiente

[1] En algunos contextos concretos no tienen porque concurrir todas las propiedades.

que la utilización de código interpretado. Por esta razón, entre otras, se han desarrollado las principales plataformas de agentes móviles utilizando Java, como por ejemplo Aglets [17], Voyager [26] y Grasshopper [12]. Tambien existen otras aproximaciones como D'Agent [10] que permite la programación de agentes en TCL y Python, aunque estan añadiendo la posibilidad de que se puedan programar en Java, la razón para ello es la mayor utilización de las plataformas de agentes móviles basadas en Java.

Como conclusión puede decirse que el lenguaje Java influye de forma determinante en el desarrollo de aplicaciones basadas en agentes, independientemente de que estos sean móviles o no. Esto se debe a que Java proporciona al desarrollador de aplicaciones un entorno de programación que es independiente del entorno, encapsulando problemas como: la heterogeneidad de las distintas arquitecturas y sistemas operativos.

2.2 Agentes móviles: una alternativa para la computación distribuida

Antes de que la tecnología de agentes móviles se hiciese realidad, sólo existía una aproximación para el diseño de sistemas distribuidos: la aproximación cliente/servidor, donde un módulo denominado "cliente" invoca servicios remotos de un módulo remoto que acepta dichas llamadas, denominado "servidor". Este procedimiento también se denomina llamada a procedimientos remotos o *Remote Procedure Calling (RPC)*. Existen diferentes protocolos de comunicaciones que implementan esta idea, como RMI [23] y CORBA [11].

Sin embargo, con la llegada de los agentes móviles existe una nueva posibilidad. Los agentes móviles son módulos inteligentes y autónomos que se mueven por sí mismos de un ordenador a otro, llevando consigo su estado y continuando su ejecución en el ordenador remoto. Algunas ventajas del uso de los agentes móviles, relacionadas con el acceso a información remota y la capacidad de interconexión, son las siguientes:

- *Encapsulan el protocolo de comunicaciones.* El agente móvil sabe cómo moverse por si mismo de un lugar a otro, no hay código de transferencia de información entre ordenadores.
- *Son asíncronos.* No necesitan comunicaciones síncronas para trabajar, dada su naturaleza autónoma, aunque por supuesto pueden sincronizarse con otros módulos o agentes.
- *Permiten reducir el uso de la red.* En el modelo RPC clásico, cuando un servicio es invocado remotamente, la conexión de la red debe estar abierta desde la invocación del método remoto hasta que se obtienen

los resultados. En el caso de los agentes móviles, cuando el agente ha llegado al ordenador destino, la conexión de red ya no se necesitará hasta que el agente necesite viajar de nuevo o comunicarse remotamente.

- *Interacción local.* En lugar de realizar llamadas a procedimientos remotos pueden viajar al ordenador adecuado e interactuar localmente con el sistema servidor lo cual puede ser crítico en aplicaciones como las basadas en tiempo real.
- *Adaptabilidad al contexto.* Pueden comportarse de forma diferente en sitios diferentes, adaptándose a los recursos disponibles.

2.3 Importancia de los agentes móviles en entornos inalámbricos

Todas estas características les hace muy interesantes para el diseño de aplicaciones en entornos inalámbricos debido a que encapsulan el protocolo de comunicaciones haciendo transparente al programador el tipo de red utilizada. La asincronía de las comunicaciones en la tecnología de agentes móviles nos permite obtener mejores prestaciones en redes inalámbricas. La asincronía de las comunicaciones implica una menor duración de las mismas por lo tanto pueden producirse errores mientras se esta procesando la información, lo cual no podría suceder utilizando una aproximación síncrona en las comunicaciones. Además debido a que se reduce el número de reintentos por errores de comunicaciones se reduce el tráfico de red.

Otra característica intrínseca de las redes inalámbricas es que son más lentas que las redes cableadas, por consiguiente si se debe interactuar con un agente o proceso remoto, el agente móvil puede viajar a la máquina en la que se está ejecutando el otro agente/proceso e interactuar localmente más rápidamente que si lo hiciese de forma remota.

En el contexto en el que nos encontramos tenemos el inconveniente adicional de que se dé una gran heterogeneidad *hardware* y *software*, lo cual puede solucionarse o paliarse gracias a la tecnología de agentes móviles. Por ejemplo, como los agentes móviles son programados en Java 2 estos pueden viajar a máquinas con distinta arquitectura porque la heterogeneidad es encapsulada por la máquina virtual de Java.

3 Máquinas virtuales Java (JVM) en Personal Digital Assistants (PDAs)

En esta sección comentamos cuáles son las características que deben tener los PDAs y las posibles opciones de configuración que podemos adoptar cuando se quiere diseñar un sistema que utilice agentes móviles (Java) sobre una conexión Bluetooth. Primeramente detallaremos cuáles son las características de algunos de los principales PDAs disponibles en el mercado. En segundo lugar trataremos cuáles son los sistemas operativos más utilizados por este tipo de dispositivos y mostraremos al lector cuál es la versión de Java que los usuarios pueden instalar en el PDA.

3.1 PDAs

Los PDAs son dispositivos de prestaciones y tamaño reducido, en comparación con los ordenadores de sobremesa. El tamaño de este tipo de dispositivos posiblemente no varíe mucho en el futuro debido a que se pretende que sean de un tamaño abarcable por la mano del usario[2]. Por lo tanto la capacidad de visualización en este tipo de dispositivos seguirá siendo limitada debido a su tamaño. También la limitación de tamaño implica que la capacidad de almacenamiento sea menor que en el caso de los portátiles u ordenadores de sobremesa. Aunque las prestaciones de este tipo de dispositivos varía rápidamente, nunca podrán alcanzar las prestaciones de un ordenador de sobremesa. Por ejemplo: Compaq en un año ha pasado de tener procesadores de 200 Mhz a 400 Mhz, pero los ordenadores de sobremesa han pasado 1.7 GHz a 2.5 Ghz.

Sin embargo, podría decirse que se está volviendo a la prehistoria de la informática, debido a que los distintos fabricantes de PDAs utilizan distintas arquitecturas . Es decir, cuando se está desarrollando un sistema operativo para un PDA, éste tiene que ser compilado para cada modelo. También sucede lo mismo con los accesorios y memorias de los diferentes modelos de PDAs, por ejemplo las antenas desarrolladas para un modelo no sirven para otros.

En cambio si utilizamos un lenguaje de programación como Java la aplicación puede desarrollarse independientemente de la arquitectura del sistema debido a que la heterogeneidad es encapsulada por la máquina virtual de Java. También hay que considerar en este entorno los teléfonos móviles de altas prestaciones, debido a que los últimos modelos que han

[2] Algunos fabricantes de PDAs incrementan el tamaño de los mismos convirtiendolos en micro ordenadores.

aparecido en el mercado soportan la ejecución de programas Java (ver Figura 2).

Figura 2. PDA y Teléfono móvil de altas prestaciones

3.2 Sistemas operativos para PDAs

En esta subsección vamos a dar una pequeña introducción a los diferentes sistemas operativos utilizados por los PDAs.

- *Windows Pocket PC 2002* [6]: este sistema operativo es desarrollado por Microsoft, su precursor fue Windows CE [5]. Esto se puede observar cuando se ve en funcionamiento el sistema operativo debido a que posee un entorno similar al de las distintas versiones de *Windows* para ordenadores de sobremesa o portátiles. Por ejemplo, este sistema operativo posee una versión reducida de *Internet Explorer*, que no posee la capacidad de ejecutar *applets* o que únicamente puede realizar operaciones básicas de manejo de correo electrónico. Por ejemplo, este sistema operativo posee una versión de *Word*; pero hay que tener sumo cuidado en su utilización debido a que el formato de los ficheros utilizados no es compatible con las versiones utilizadas en los ordenadores de sobremesa o portátiles.
- *PalmOS* [27]: este sistema operativo es el que instala Palm en sus dispositivos, es decir, estamos ante un sistema operativo hecho a medida.

Aunque de este sistema operativo hay que destacar que la utilización de los recursos es mucho más eficiente que en el caso de los sistemas operativos desarrollados por Microsoft. Sin embargo, tiene el inconveniente de cambiar el interfaz al que están acostumbrados los usuarios de Windows. También hay que tener en consideración que este sistema operativo esta desarrollado para PDAs de bajas prestaciones.

- *Symbian OS* [19]: este sistema operativo tiene unas menores prestaciones que los presentados anteriormente debido a que se utiliza principalmente en teléfonos móviles. Debido a la utilidad del sistema operativo las aplicaciones desarrolladas para el mismo son mucho menos numerosas y tienen disponibles un menor número de recursos.
- *SavaJe XE OS* [35]: es el primer sistema operativo capaz de ejecutar Java 2, este sistema operativo es soportado por los iPAQ de Compaq. Otra de las ventajas de este sistema operativo es que ha sido desarrollado por Sun Microsystems. Aunque este software no es propiamente un sistema operativo debido a que debe ser instalado sobre Pocket PC. El inconveniente del mismo es los altos requerimientos hardware que tiene (Sólo se puede instalar en el IPAQ de Compaq).
- *Linux Familiar* [18]: este sistema operativo procede de la versión de Linux para ordenadores de sobremesa. Por lo que tiene en común muchas de las características del sistema operativo linux. Entre las características de éste, podríamos destacar:
  - Está basado en XFree86 [36], incluyendo las últimas extensiones de redibujado de pantalla.
  - Permite conectarse al PDA mediante el protocolo SSH [2].
  - El sistema de ficheros utilizado para leer y escribir en la Compact Flash es JFFS2 [18]. La utilización de este sistema de ficheros permite optimizar los accesos a disco que se realizan debido a que se está accediendo a una compact flash y no a un disco duro.
  - Proporciona lenguajes de programación como Python y Gtk. Además de C++ que sería el equivalente a Visual C++, proporcionado por Microsoft para los sistemas operativos fabricados por dicha compañía.
  - Debido a las limitaciones de espacio de almacenamiento muchas utilidades del sistema como por ejemplo los comandos cp, rm, ln, tar, etc, están implementados dentro del propio interprete de comandos, *busybox*.

En esta subsección se han comentado los diferentes sistemas operativos que pueden ser instalados en un PDA, aunque hay que considerar que no

todos los sistemas operativos pueden ser instalados en todos los PDAs y que no todos los PDAs proporcionan soporte para Java 2 (ver Tabla 1).

3.3 Distintas plataformas Java para PDAs

En esta sección comentamos cuáles son las posibles plataformas Java que se pueden instalar en un PDA o en un teléfono móvil que soporte Java. La primera opción es instalar Java Micro Edition (J2ME) [33,38] que es la versión de Java que está adaptada para ser instalada en un PDA. La segunda opción es instalar una versión de Java 2 que instalaríamos en un PC de sobremesa pero adaptada a la arquitectura del PDA.

3.3.1 Java 2 Micro Edition (J2ME). Esta versión de la máquina virtual de Java está pensada pare poder ser ejecutada en dispositivos de reducidas prestaciones y con un tamaño de memoria limitado. Aunque debido a la amplia gama de dispositivos que admiten la ejecución de programas Java existen dos tipos de configuración:

- *CLDC (Connected Limited Device Configuration)*: es una configuración muy limitada y que está pensada para dispositivos con conexiones intermitentes, procesadores de bajas prestaciones y con memoria limitada. Es decir, esta pensada para teléfonos móviles y PDAs de bajas prestaciones.
- *CDC (Connected Device configuration)*: este tipo de configuración está diseñada para dispositivos con más memoria, PDAs de altas prestaciones y con un mayor ancho de banda. Dentro de este tipo de dispositivos encontramos los PDAs de última generación, sistemas telemáticos dentro de vehículos, etc. Contiene todas las características de una maquina virtual de Java 2 aunque como inconveniente requiere una arquitectura de 32 bits y 2MB de memoria disponible.

Para poder utilizar J2ME necesitamos la máquina virtual que está formada por dos componentes el CLDC o CDC y el MIDP (*Mobile Information Device Profile*) en el que se almacena la interface de usuario, tipo de conexión de red, tipo de almacenamiento de datos y control de ejecución.

Utilizando estos dos componentes ya tenemos todo lo necesario para poder ejecutar aplicaciones Java aunque con una versatilidad de programación reducida. Por ejemplo no podemos ejecutar ningún sistema de *agentes móviles* que se haya desarrollado hasta el momento.

3.3.2 Java 2 Standard Edition (J2SDK). En el proceso de instalación de Java 2 [34], en un PDA, solamente debe configurarse de forma obligatoria el JRE (*Java Runtime Edition*), es decir, el conjunto de ficheros necesarios para poder ejecutar un programa Java 2. Cuando en un dispositivo como en un PDA no instalamos el J2SDK sino el JRE perdemos la capacidad de compilación, es decir, podemos ejecutar cualquier programa Java 2 pero no podemos compilarlos en el PDA. La arquitectura del J2SDK descrita por Sun Microsystems puede verse en la Figura 3.

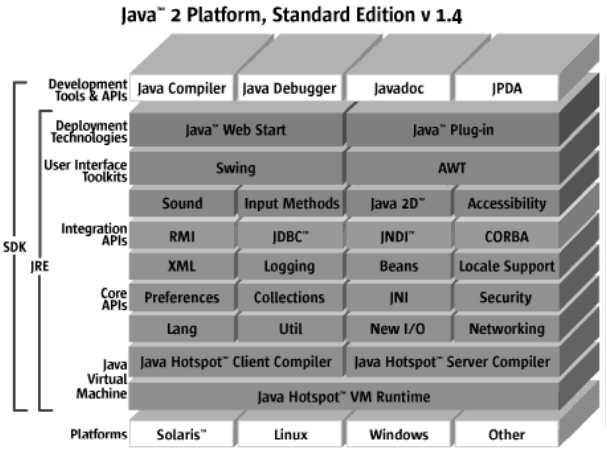

Figura 3. Arquitectura del J2SDK

En nuestro caso debido a que queremos utilizar agentes móviles necesitamos que la máquina virtual instalada en el PDA sea capaz de comunicarse mediante una conexión TCP-IP. Además, si es posible poseer todas las prestaciones ofrecidas por una máquina virtual de la versión J2SDK la diversidad de aplicaciones que podrán desarrollarse para el PDA será mucho mayor, es decir, se pueden programar las mismas aplicaciones que para un ordenador de sobremesa.

3.4 Las mejores combinaciones: Java 2 con comunicación TCP-IP sobre Bluetooth

En esta subsección vamos a presentar las mejores configuraciones que podemos adoptar para configurar un PDA de tal forma que dicha configuración nos proporcione: Java 2 y un interfaz de red TCP-IP. En el caso del interfaz de red debido a que estamos utilizando sistemas con una bate-

ria limitad trataremos que la potencia consumida por las comunicaciones sea minimizada.

1. *HP Jornada con Microsoft Windows CE como sistema operativo*: en este caso para tener soporte para Java se tiene que instalar el software *CHAI* pero éste no proporciona una máquina virtual estándar para Java 2 sino que es compatible con la versión 1.1. La ventaja de esta configuración es que se pueden probar los desarrollos realizados utilizando un entorno emulado.
2. *iPAQ H3870 con sistema operativo Microsoft Windows CE*: en esta configuración hay que instalar un software adicional denominado Jeode que nos permite ejecutar código Java, aunque la máquina virtual no es totalmente compatible con la desarrollada por Sun Microsystems.
3. *iPAQ H3870 con el sistema operativo Pocket PC*: en este caso para soportar la versión J2SE necesitamos instalar sobre el sistema operativo Savaje XE. Después de instalar este sistema se tiene soporte completo para CORBA, RMI, JNI y Bluetooth mediante código nativo. Es decir, Savaje nos proporciona una máquina virtual J2SE pero no nos permite acceder al iPAQ mediante el puerto USB ni mediante una conexión Bluetooth.
4. *iPAQ H3870 con el sistema operativo Linux Familiar*: este tipo de configuración nos permite instalar en el sistema operativo una máquina virtual J2SE y además nos habilita opciones de acceso al PDA a través de los puertos USB y Bluetooth del iPAQ. Al igual que con Savaje el acceso al puerto Bluetooth se hace mediante código nativo. En este caso el código utilizado por el PDA pertenece a proyectos de software libre.

En esta sección hemos comentado cuáles son las mejores combinaciones de Sistema Operativo y PDAs disponibles en el mercado, esto queda reflejado en la Tabla 1. Al realizar dichas consideraciones hemos tenido en cuenta la capacidad de interconexión del PDA y la posibilidad de ejecutar Java 2, siempre teniendo como finalidad la utilización de la tecnología de agentes móviles. Debido a que la configuración compuesta por un IPAQ H3870 con un sistema operativo Linux es la que nos proporciona todas las características deseadas, ésta será la configuración adoptada y la que supondremos en el resto del artículo.

4 Sistema de comunicaciones para PDAs

Debido a las características de un PDA como son el espacio bastante limitado de almacenamiento y la ausencia de un sistema de entrada de

SO/PDA	J2ME	J2SE	Bluetooth Nativo	TCP-IP sobre Bluetooth
1) HP Jornada con Windows CE y CHAI	Sí	Java 1.1	No	No
2) iPAQ H3870-H3970 con Windows CE y Jeode	Sí	No compatible con JVM de Sun	Sí	No
3) iPAQ H3870-H3970 con Pocket PC y Xavaje XE	Sí	Sí	Sí	No
4) Linux Familiar	Sí	Sí	Sí	Sí

Tabla 1. Características disponibles en las diferentes configuraciones

datos tipo disquetera, CD-ROM, etc es necesario dotar de algún sistema de comunicaciones al dispositivo móvil para permitir la transferencia de información con otros dispositivos móviles o fijos.

Existen varias alternativas en la actualidad, desde las tradicionales basadas en conexiones a través de un cable conectado a un puerto serie de un PC o las más modernas que no necesitan conexión a través de un cable físico. Entre estas nuevas tecnologías se pueden citar IRDA, tecnología de infrarrojos que precisa contacto visual entre los dispositivos a comunicar para establecer la comunicación o las tecnologías basadas en radio frecuencia. Entre éstas se pueden destacar 802.11b [30] y Bluetooth [25]. Ambas utilizan el mismo rango de frecuencias para establecer la comunicación pero presentan diferencias entre si, que ahora analizaremos. Pero un método de comunicación basado en radio parece la elección más adecuada en el momento tecnológico en el que vivimos.

4.1 Bluetooth: la mejor alternativa de comunicaciones inalámbricas de corto alcance en PDAs

Las alternativas tecnológicas de comunicaciones inalámbricas existentes actualmente son 802.11b y Bluetooth. Bluetooth es un protocolo de comunicaciones inalámbrico basado en radio frecuencia con tasas de transferencia de hasta 721 kbps y tiene un alcance que varía entre 10 y 100 metros. Esta diseñado para consumir la menor potencia posible debido a que ajusta la misma dependiendo de la distancia entre emisor y receptor. Además proporciona servicios de alto nivel para descubrimiento de dispositivos Bluetooth activos.

El protocolo de comunicaciones 802.11b, más conocido como WI-FI permite una comunicación inalámbrica basada en radio frecuencia con velocidades de transferencia de hasta 11Mbps.

Bluetooth se considera que es más indicado para las comunicaciones de corto alcance y además proporciona servicios de descubrimiento de dispositivos (servicios) a los que conectarse y por esta razon lo hemos elegido como protocolo de comunicaciones. Estas características nos facilitan el desarrollo de servicios orientados a usuarios que se mueven por un entorno en el que hay distintas antenas Bluetooth y éstos se conectan a las mismas para acceder a internet o a cualquier otro servicio de valor añadido disponible, detectando las antenas que les dan cobertura automáticamente gracias al servicio de descubrimiento que posee Bluetooth. Permitiendo de esta forma la configuración automática de la topología de la red. Existen dos variantes posibles, para las configuraciónes maestro esclavo:

- Las *piconets*: en las que hay un nodo maestro y el resto son esclavos(Ver Figura 4.a).
- Las *scatternets*: formadas por la superposición de las áreas de cobertura de varias piconets (Ver Figura 4.b)

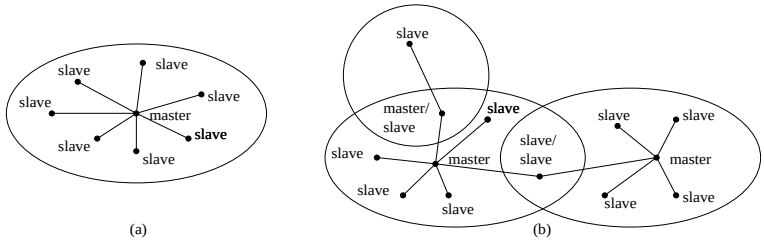

Figura 4. Topologías en redes Bluetooth: (a) piconet y (b) scatternet

Bluetooth está definido a modo de pila, similar al esquema ISO/OSI. Como se muestra en la Figura 5, esta pila queda dividida por la interfaz HCI en dos partes: la parte inferior que usualmente es incorporada por hardware dentro de los dispositivos Bluetooth y la parte superior que se implementa por software. En algunos dispositivos empotrados, la parte superior puede estar implementada dentro del propio chip, pero no es lo habitual.

La funcionalidad de cada nivel es la siguiente:

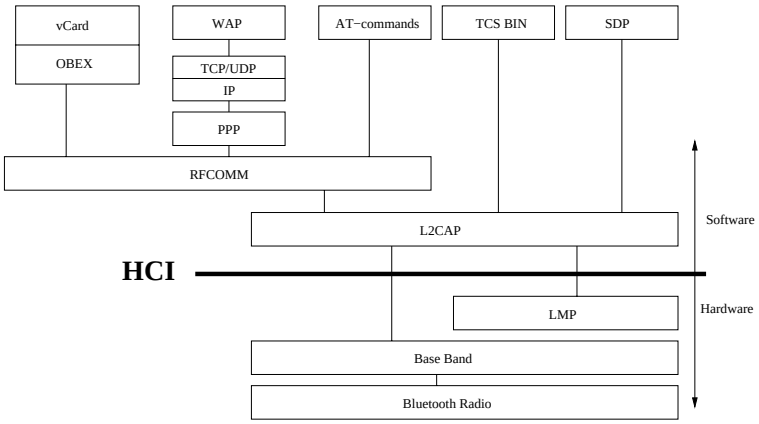

Figura 5. Pila Bluetooth Dividida por el Interfaz HCI

1. *Bluetooth radio*: es el nivel encargado de la emisión-recepción de la señal de radio. Trabaja en el rango de frecuencias de la banda ISM (Médico-Científica Internacional) en 2.45Ghz.
2. *Baseband*: es el encargado de la sincronización, control de flujo, transmisión de la información, corrección de errores, división lógica de canales y seguridad. Permite la formación de dos tipos de conexiones: 1) asíncronas (ACL[3]): utiliza conmutación de paquetes como protocolo de comunicaciones y se usan para la transmisión de datos; y 2) síncronas (SCO[4]): el protocolo de comunicaciones utilizado es por conmutación de circuitos.
3. *Protocolo de control de enlace (LMP):* este nivel traduce los comandos de niveles superiores a los niveles inferiores de la pila.
4. *Host Controller Interface (HCI):* es la interfaz que une el dispositivo Bluetooth con el PDA u ordenador.
5. *Link Layer Control and Adaptation Layer Protocol (L2CAP)*: esta capa toma los datos de los niveles superiores y se los pasa a las capas inferiores. De esta forma permite: 1) multiplexación de varias capas superiores (incluso protocolos diferentes) sobre un mismo enlace ACL, 2) segmentación y reensamblaje de paquetes de gran tamaño en paquetes del tamaño de las capas inferiores y 3) calidad de servicio (QoS) para capas superiores.
6. *Service Discovery Protocol (SDP)*: protocolo para descubrir que servicios tienen disponibles los dispositivos Bluetooth próximos.

[3] *Asychronous Connectionless.*
[4] *Synchronous Connection-Oriented.*

7. *RFCOMM*: es un protocolo que implementa una emulación de puertos serie RS232 sobre un canal L2CAP. Permite configurar una conexión punto a punto (PPP) sobre la que se establecerá un enlace TCP-IP.

En esta sección se han definido las funcionalidades que ofertan cada una de las capas de la pila Bluetooth. En los prototipos desarrollados hemos utilizado Bluez [3] como implementación concreta de la pila Bluetooth. En el Apéndice A se detalla el proceso de configuración de Bluez en una máquina Linux. En los siguientes apartados vamos a definir como se puede establecer un enlace TCP-IP sobre un dispositivo Bluetooth.

4.2 TCP-IP sobre Bluetooth

Hasta ahora hemos hablado de Bluetooth como una pila de protocolos de comunicaciones inalámbricas, pero esta pila hasta ahora sólo permite la conexión de un dispositivo móvil con otros. Es momento de poder integrar esos dispositivos móviles con las redes ya existentes, sean fijas o móviles, permitiendo de esta forma nuestro objetivo final que es la utilización de agentes móviles. Para ello tenemos que utilizar un protocolo estándar a todas estas redes (TCP-IP) que se ejecute sobre la pila que ya tenemos definida. Esto aporta grandes ventajas a los dispositivos móviles, debido a que permite acceder a cualquier red del mundo a través de su enlace inalámbrico.
Entre las características de estas nuevas redes se pueden citar:

- Utilizan TCP-IP estándar, no una versión modificada como 802.11b, con lo cual se integra perfectamente en las redes ya existentes. Al ser estándar, nuestras aplicaciones Java que lo utilizan como protocolo de comunicaciones no tienen que ser adaptadas para ejecutarse en un PDA que se comunica usando Bluetooth.
- Permite la formación de redes ad-hoc. Estas redes se crean y se destruyen dinámicamente y no tienen necesidad de ninguna infraestructura previa, en modo de estaciones base, para su creación. Bastan dos dispositivos móviles que se conecten entre sí para su formación. Esto resulta muy útil para transferir datos entre un dispositivo y otro cuando se encuentran en el mismo área de cobertura.
- Las redes creadas pueden ser completamente heterogéneas. La compatibilidad entre distintas máquinas y distintos sistemas operativos es total.

Hay cuatro modos principales de configurar IP sobre la pila de protocolos Bluetooth: 1) IP sobre L2CAP, 2) PPP sin RFCOMM, 3) PPP sobre

RFCOMM y 4) *Bluetooth Network Encapsulation Protocol* (BNEP). Los dos primeros no tienen mucha importancia debido a que el acceso a los mismos se realiza mediante código nativo y no permite la configuración de una red TCP-IP. Por lo tanto los dos modos más comunes son los dos últimos, que ahora comentamos en más detalle.

Para el establecimiento de una conexión *PPP sobre RFCOMM* es necesario configurar sobre el nivel RFCOMM una conexión punto a punto (PPP) y sobre ella establecer los niveles de red habituales, IP y TCP. En la Figura 6.a se puede ver claramente como se comunican entre sí los diferentes niveles de la pila.

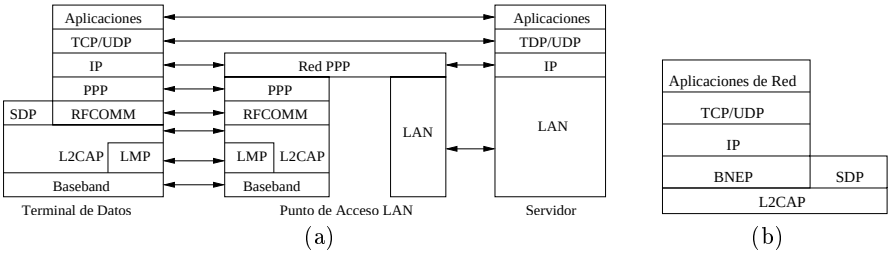

Figura 6. Pila de una conexión PPP sobre RFCOMM (a) y una BNEP (b)

El método *Bluetooth Network Encapsulation Protocol (BNEP)* es más simple que RFCOMM. En BNEP directamente se transmiten los paquetes IP dentro de los paquetes L2CAP con una cabecera que identifica los paquetes BNEP. Se puede utilizar para la formación de redes ad-hoc debido a su simplicidad a la hora de establecer la comunicación. En la Figura 6.b se observa como queda la pila de protocolos.

Se pueden dar dos tipos de escenarios:

- *Network Access Point* (NAP): dispositivo que actúa como una puente para conectar una piconet y una red cableada.
- *Group ad-hoc Network* (GN): dispositivo que conecta uno o más dispositivos Bluetooth en una piconet. Estas redes no permiten la comunicación con el exterior. En este tipo de configuración cada usuario puede tener su propia red de area local[5].

[5] Personal Area Network User (PANU).

5 Ejemplo: Servicio de Obtención de Software

En esta sección vamos a presentar un caso de ejemplo para que se vea la potencia de las técnicas explicadas en este artículo. Para lo cual vamos a explicar un Servicio de Obtención de Software [20] (*Software Retrieval Service* (SRS)), desarrollado para PDAs con conexión Bluetooth y que utiliza agentes móviles. Este servicio se ha desarrollado como parte del sistema ANTARCTICA [9].

Una de las tareas más frecuentes entre los usuarios de ordenadores es obtener nuevo software, para mejorar las prestaciones o dar mayor funcionalidad a sus ordenadores o PDAs. Actualmente un procedimiento habitual para obtener software es visitar sitios web que ofrecen versiones shareware o demos de diferentes programas (tales como Tucows [37] y CNET Download.com [4]); Debido a que este procedimiento no considera cuestiones importantes para los usuarios de los PDAs como pueden ser desconexiones, capacidad de almacenamiento muy limitada, facilidades de búsqueda de software, etc se ha desarrollado el SRS.

5.1 Agentes del sistema

En esta sección presentamos brevemente los principales agentes que constituyen la arquitectura del SRS, el esquema de la arquitectura se muestra en la Figura 7. Este servicio esta localizado en un servidor concreto que nosotros denominamos *proxy* y es uno de los servicios que provee el sistema ANTARCTICA [9].

A continuación vamos a detallar cuales son los agentes más importantes del servicio y cuál es su función:

- *Alfred*: es un eficiente mayordomo que sirve al usuario y es el encargado de almacenar tanta información del usuario y de su dispositivo móvil como sea posible. Este agente ayuda al usuario a localizar y ejecutar los servicios disponibles.
- *El Gestor de Software*: que es el especialista en manejo de catálogos de software. Este agente es el encargado de adaptar el catálogo enviado a cada uno de los usuarios dependiendo de las necesidades del mismo y del entorno de ejecución.
- *El Navegador*: que es el agente encargado de visualizar el catálogo de software en el PDA (ver la Figura 8) y de ampliar o reducir el catálogo mostrado al usuario dependiendo de su comportamiento.
- *El Vendedor*: es el agente encargado de descargar el software al ordenador o PDA del usuario y de ayudarle a instalarlo. En el caso de

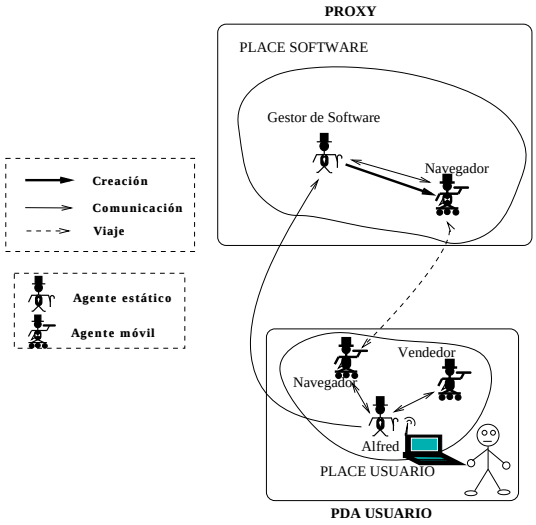

Figura 7. Arquitectura del SRS

que la descarga y posterior instalación del software implique alguna transacción comercial, este agente se encargará de comunicarse con Alfred para pedirle la información necesaria relativa al usuario.

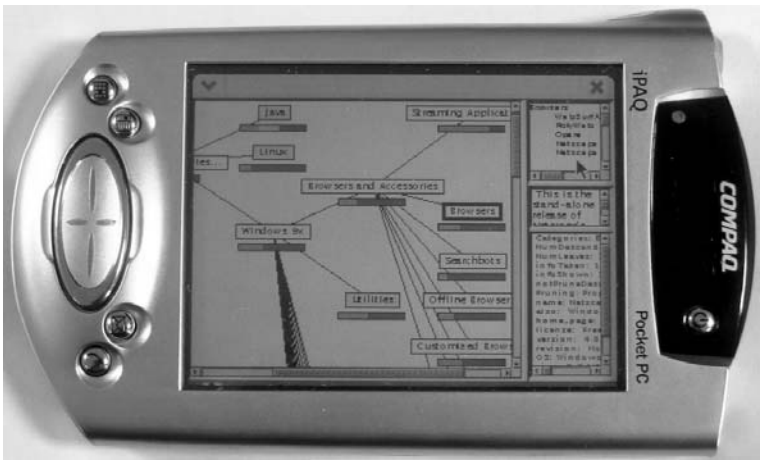

Figura 8. Búsqueda en un catálogo de software utilizando el SRS

Como toda búsqueda de software, la misma finaliza cuando la aplicación solicitada es encontrada. El SRS, además, de ayudar a los usuarios

inexpertos en la búsqueda de software también les proporciona ayuda en el proceso de descarga del software y en su posterior instalación.
El sistema desarrollado presenta las siguientes ventajas:

- Optimización de la utilización de las comunicaciones. Los agentes del SRS gracias a su "inteligencia" envían catálogos personalizados de software para minimizar el uso de la red.
- Adaptabilidad al contexto. Los agentes del SRS personalizan los catálogos de software que son enviados al PDA del usuario dependiendo de la velocidad de las comunicaciones, la historia anterior, etc.
- Búsqueda en catálogos locales. El agente Navegador trae un catálogo de software personalizado al PDA del usuario. Después de que el catálogo es mostrado al usuario, éste puede realizar búsquedas de forma local.

El SRS se desarrolló sobre una arquitectura hardware basada en PCs y estaciones Sun conectados por una red ethernet. Posteriormente, debido a la aparición de PDAs con prestaciones suficiente para ejecutar una máquina virtual de Java 2 se adaptó para poder ejecutarse en un PDA. Para realizar esta adaptación únicamente tuvimos que desarrollar una clase que modificase el tamaño de los elementos gráficos utilizados dependiendo del tamaño de la pantalla que estabamos utilizando. Sin embargo el mecanismo de comunicaciones [21] y la inteligencia de los agentes [22] es la misma que cuando el sistema se ejecutaba en un ordenador de sobremesa. También hay que considerar que el tamaño de los catálogos se ha adaptado al dispositivo automáticamente, debido a que se había desarrollado una técnica que permitía al sistema adaptarse al contexto dependiendo de la capacidad de procesamiento del dispositivo y de la velocidad de red.

6 Trabajos relacionados

En esta sección presentamos los trabajos relacionados conocidos más relevantes. En el caso de trabajos relacionados nos centraremos en comentar aquellos que están focalizados en la utilización de Java en PDAs y en aquellos que tratan la utilización de agentes en entornos inalámbricos. Considerando todo tipo de comunicaciones, por ejemplo: Bluetooth [25], GSM [29], GPRS [1,8], UMTS [15], WI-FI [30], comunicaciones por infrarrojos, etc.

Existen otros trabajos relacionados en los que no se utilizan las plataformas de agentes móviles desarrolladas para ordenadores de sobremesa

sino que se diseña la plataforma de agentes móviles para soportar acceso a PDAs. Este es el caso por ejemplo del proyecto TACOMA [14].

Otras aproximaciones permiten la integración de J2ME con la tecnología Bluetooth. Este es el caso de *Technology Licensing Kit* (TLK) desarrollado por Roccosoft [7]. Es decir TLK implementa la API Java que permite la comunicación entre CLDC-MIDP y la capa de conexión Bluetooth. Anteriormente a este desarrollo Rococosoft implementó el simulador impronto que simulaba un dispositivo Bluetooth y la API Java para acceder al mismo. Mediante la utilización del TLK se podría utilizar una plataforma de agentes siempre que ésta estuviese desarrollada para comunicarse mediante la API Java desarrollada por Rococosoft que es compatible con JABWT[6].

En [31] se comentan las ventajas de la utilización de agentes móviles en entornos basados en dispositivos móviles y comunicaciones inalámbricas. En este artículo no solamente se comentan las ventajas de esta aproximación sino que también se desarrolla un emulador que permite simular las comunicaciones de los agentes.

7 Conclusiones

En este artículo hemos presentado una tecnología basada en Java 2, Linux y software libre que nos permite desarrollar sistemas multiagente en PDAs con conexión Bluetooth. En nuestro caso debido a que nos encontramos en un contexto inalámbrico las comunicaciones son inestables y lentas por lo que nos hemos centrado en la utilización de agentes móviles porque estos ayudan a la optimización de las comunicaciones. Los agentes móviles favorecen la optimización de la utilización de las comunicaciones y nos proporcionan un nuevo paradigma de computación distribuida.

Tras el desarrollo de todo el proceso de instalación que incluye incluso el cambio del protocolo Bluetooth utilizado por el PDA (reprogramar la antena Bluetooth), debido a que anteriormente Bluez no soportaba de forma completa el estándar Bluetooth. Se puede instalar la máquina virtual de Java 2 y la plataforma de agentes móviles.

Como conclusión final podemos decir que los agentes móviles son una buena aproximación para el desarrollo de aplicaciones en entornos heterogéneos y distribuidos con comunicaciones inalámbricas porque: 1) se adaptan al contexto, 2) reducen el uso de comunicaciones respecto a anteriores aproximaciones como la cliente/servidor, 3) encapsulan el protocolo

[6] *Java APIs for Bluetooth Wireless Technology.*

de comunicaciones y 4) pueden ser ejecutados en distintas plataformas debido a que utilizan Java como lenguaje de programación.

Nuestra experiencia en el desarrollo de sistemas de acceso a datos desde PDAs ha sido muy satisfactoria debido a la facilidad con la que se adaptan los sistemas desarrollados para otras arquitecturas a las características proporcionadas por un PDA. Esta ventaja es conseguida gracias a la utilización de un lenguaje multiplataforma como Java. Los costes de software han sido mínimos debido a la utilización de proyectos de software libre como Linux y Bluez.

Nuestra línea de investigación continuará con el desarrollo de aplicaciones basadas en agentes móviles que nos permitan determinar la localización de un dispositivo móvil dependiendo de la potencia de la señal bluetooth que éste recibe de las antenas que le proporcionan cobertura. Esto permitirá el desarrollo de muchas aplicaciones basadas en la localización. Además también queremos ampliar los protocolos inalámbricos utilizados, por ejemplo desarrollando redes que posean nodos Bluetooth y nodos WI-FI.

Referencias

1. Christoffer Andersson. *GPRS and 3G Wireless Applications.* Wiley, 2001.
2. Daniel J. Barrett and Richard Silverman. *SSH, The Secure Shell: The Definitive Guide.* O'Reilly & Associates; 1st edition, ISBN: 0596000111, February 2001.
3. BlueZ project. Official Linux Bluetooth protocol stack, 2003. http://bluez.sourceforge.net/.
4. CNET Inc., 1999. http://www.download.com.
5. Microsoft Corporarion. Windows CE. http://www.microsoft.com/windowsce.
6. Microsoft Corporarion. Windows Pocket PC 2003. http://www.microsoft.com/windowsmobile/products/pocketpc/default.mspx.
7. Rococo Software Corporation. Technology licensin kit, 2003.
8. European Telecommunications Standards Institute (ETSI). General packet radio service (gprs): Service description, 1998.
9. A. Goñi, A. Illarramendi, E. Mena, Y. Villate, and J. Rodriguez. ANTARCTICA: A multiagent system for internet data services in a wireless computing framework. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems, Scottsdale, Arizona (USA)*, pages 119–135. Lecture Notes in Computer Science (LNCS 2538) 2002, ISBN 3-540-00289-8, October 2001.
10. Robert S. Gray. Agent Tcl: A transportable agent system. In *Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95)*, Baltimore, Maryland, December 1995.
11. Object Management Group. http://www.omg.com.
12. IKV++ technologies. Grasshopper - a platform for mobile software agents. http://www.grasshopper.de/download/doc/GrasshopperIntroduction.pdf.
13. T. Imielinski and H.F. Korth, editors. *Mobile Computing.* Kluwer Academic Publishers, 1996.

14. Dag Johansen, Kare J. Lauvset, and Keith Marzullo. An extensible software architecture for mobile components. In *In Proceedings of the 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems Lund, Sweden*, April 2002.
15. Heikki Kaaranen, Siamak Naghian, Lauri Laitinen, Ari Ahtiainen, and Valtteri Niemi. *UMTS Networks: Architecture, Mobility and Services.* John Wiley & Sons; 1st edition, ISBN: 047148654X, August 2001.
16. Inc. Kernel.Org Organization. The Linux Kernel Archives, 2003. http://www.kernel.org/.
17. D. Lange and M. Oshima. *Programming And Deploying Java Mobile Agents with Aglets.* Addison-Wesley, 1999.
18. Linux Familiar. http://familiar.handhelds.org/.
19. Symbian Ltd. Symbian OS. http://www.symbian.com/index.html.
20. E. Mena, A. Illarramendi, and A. Goñi. A Software Retrieval Service based on Knowledge-driven Agents. In *Fith IFCIS International Conference on Cooperative Information Systems (CoopIS'2000), Eliat (Israel)*, pages 174–185. Springer series of Lecture Notes in Computer Science (LNCS), ISSN 0302-9743, ISBN 3-540-41021-X, September 2000.
21. E. Mena, J.A. Royo, A. Illarramendi, and A. Goñi. Adaptable software retrieval service for wireless environments based on mobile agents. In *2002 International Conference on Wireless Networks (ICWN'02), Las Vegas, USA*, pages 116–124. CSREA Press, ISBN 1-892512-30-0, June 2002.
22. E. Mena, J.A. Royo, A. Illarramendi, and A. Go ni. An agent-based approach for helping users of hand-held devices to browse software catalogs. In *Cooperative Information Agents VI, 6th International Workshop CIA 2002, Madrid (Spain)*, pages 51–65. Lecture Notes on Artificial Intelligence (LNAI), ISBN 3-540-44173-5, September 2002.
23. Sun Microsystems. http://java.sun.com.
24. D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. MASIF, the OMG mobile agent system interoperability facility. In *Proceedings of Mobile Agents '98*, September 1998.
25. Robert Morrow. *Bluetooth: Operation and Use.* McGraw-Hill Professional; 1st edition, ISBN: 007138779X, June 2002.
26. ObjectSpace, 1999. http://www.objectspace.com/.
27. Inc. Palm Source. Palm OS. http://www.palmos.com.
28. E. Pitoura and G. Samaras. *Data Management for Mobile Computing.* Kluwer Academic Publishers, 1998.
29. M. Rahnema. Overview of the gsm system and protocol architecture. *IEEE Communications Magazine 31(4)*, pages 92–100, April 1993.
30. Neil P. Reid and Ron Seide. *Wi-Fi (802.11) Network Handbook.* McGraw-Hill Osborne Media, ISBN: 0072226234, December 2002.
31. Ichiro Satoh. A testing framework for mobile computing software. *To appear in IEEE Transactions on Software Engineering*, 29, 2003.
32. Inc. Sun Microsystems. http://java.sun.com.
33. Inc. Sun Microsystems. Java 2 Platform, Micro Edition (J2ME). http://java.sun.com/j2me.
34. Inc. Sun Microsystems. Java 2 Platform, Standard Edition (J2SE). http://java.sun.com.
35. Savaje Technologies. SavaJe XE SO. http://www.savaje.com.

36. The XFre86 Project Inc. http://www.xfree86.org/.
37. Tucows.Com Inc., 1999. http://www.tucows.com.
38. J. White and D. A. Hemphill. *Java 2 Micro Edition*. Manning Publications Company; 1st edition, ISBN 1930110332, April 2002.

A Bluez: Implementación de la Pila Bluetooth

Bluez [3] es una implementación de la pila de protocolos Bluetooth gratuita y disponible para Linux, que ha sido adoptada como la implementación de la pila Bluetooth que incorporan los nuevos kernel de Linux [16] a partir de la versión 2.4.20.

La implementación de la pila comienza en la interfaz HCI, en la *Controller Transport Layer*, donde se define el tipo de conexión entre el dispositivo físico y el ordenador o PDA. Tiene definidas tres tipos de conexiones: RS232, USB y UART. Esto permite poder conectar distintos tipos de antenas (USB, PCMCIA) a la misma máquina, ya que sólo cambiando un nivel intermedio se puede enlazar con los niveles superiores.

Bluez implementa los niveles de la pila de mayor nivel, como son L2CAP, SDP, RFCOMM y BNEP. Los niveles inferiores están incluidos dentro del chip del dispositivo con lo que no entran dentro de esta implementación. Sin embargo se implementan los comandos de nivel superior que se envían a estos niveles inferiores.

A.1 Utilidades proporcionadas por Bluez

Los comandos y demonios asociados a la pila Bluez [3] son:

- hcid: es el demonio que gestiona los comandos de nivel HCI (crear enlace ACL, romper enlace...) es obligatorio que este programa esté ejecutandose para poder ejecutar cualquiera de nivel superior.
- hciattach: es el comando utilizado para activar el enlace bluetooth.
- hcitool: es la implementación de los comandos HCI. Este programa permite ejecutar estos comandos desde la línea de comandos.

```
hcitool cc 11:22:33:44:55:66
     crea una conexión ACL con el dispositivo con esa dirección.
hcitool scan
     Escanea en busca de dispositivos enlazables dentro del área de
 cobertura.
```

- hcidump: este programa es un *sniffer* a nivel HCI, captura todo el tráfico que pasa por la interfaz HCI.

- sdpd: es un demonio para la gestión de los servicios que ofrece un dispositivo Bluetooth.
- sdptool: este programa implementa el mecanismo para definir los servicios que ofrece un dispositivo, así como buscar los que ofrecen el resto de dispositivos.
- pand: es el demonio de la implementación del protocolo BNEP. Se puede configurar según parámetros para escuchar peticiones de conexión o intentar conectar a un dispositivo. Obtiene la información sobre las IP que tiene que asignar a la conexión según los ficheros de configuración de las interfaces de red de Linux.
- dund: es el demonio de la implementación del protocolo RFCOMM. Se configura en los mismos ficheros de configuración que el resto de conexiones PPP del sistema Linux. Es más complejo de configurar que las conexiones BNEP ya que requiere parámetros, como por ejemplo la velocidad de conexión entre otros.

Para el funcionamiento de todos estos programas es necesario que estén cargados en el sistema los módulos de kernel asociados a Bluez. Estos módulos son: bluez, l2cap, hci_usb y hci_uart.

A.2 Creación de una conexión Bluetooth utilizando Bluez

La implementación de Bluez se integra perfectamente dentro del sistema de interfaces de red de Linux. Al crearse una nueva conexión y asignarle una dirección de red a ambos extremos se da de alta una nueva interfaz de red en el sistema. Estas interfaces se llaman ppp0, ppp1, ..., pppN para el caso de las conexiones creadas usando RFCOMM y bnep0, bnep1, ..., bnepN para las creadas con BNEP. Esto simplifica mucho el trabajo en redes inestables, ya que estas interfaces se activan o desactivan automáticamente según la conexión esté o no disponible, con lo que en caso de estar en una zona de muchas interferencias o mala cobertura simplifica y automatiza el trabajo. Estas conexiones pueden automatizarse todavía más a través del sistema *Hotplug de Linux*, permitiendo que la configuración de la red se haga automáticamente cuando se conecta una antena Bluetooth al dispositivo (ordenador/PDA).

Los pasos necesarios para crear una conexión entre dos dispositivos bluetooth son:

1. Instalar los módulos de kernel requeridos usando *modprobe* o *insmod*

```
modprobe bluez
modprobe l2cap
```

```
modprobe hci_usb    (en caso de utilizar este tipo de interfaz)
modprobe rfcomm
modprobe bnep
```

2. Lanzar los demonios necesarios

   ```
   hcid
   ```

3. Existen dos opciones de configuración del protocolo de conexión:
   - Utilizando RFCOMM:
     (a) Ejecutar el comando que inicializa la escucha de conexiones basadas en RFCOMM en un dispositivo
     ```
     dund --listen --channel 1
     ```
     (b) Ejecutar el comando que intenta la conexión RFCOMM en otro dispositivo
     ```
     dund --connect 11:22:33:44:55:66 --channel 1
     ```
   - Utilizando BNEP:
     (a) Ejecutar el comando que inicializa la escucha de conexiones basadas en BNEP en un dispositivo
     ```
     pand --listen
     ```
     (b) Ejecutar el comando que intenta la conexión BNEP en otro dispositivo
     ```
     pand --connect 11:22:33:44:55:66
     ```

En este punto el sistema conecta los dos dispositivos Bluetooth y basándose en los parámetros de configuración de la conexión PPP, asigna una dirección IP a los dos dispositivos y da de alta la interfaz de red en el sistema.