UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías
Informática y de Telecomunicación

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Movilización de Información Multimedia e Interacción con Entornos Inteligentes

Memoria de tesis presentada por

Javier Medina Quero

para optar al grado de Doctor en Ingeniería Informática
por la Universidad de Granada

Directores de la Tesis:
Dr. Miguel Delgado Calvo-Flores y Dra. Amparo Vila Miranda
Granada, 2010

Editor: Editorial de la Universidad de Granada
Autor: Javier Medina Quero
D.L.: GR 2008-2011
ISBN: 978-84-694-1175-9

La memoria de tesis titulada '**Movilización de Información Multimedia e Interacción Interacción con Entornos Inteligentes**', que presenta **Javier Medina Quero** para optar al grado de Doctor en Informática, ha sido realizada en el **Departamento de Ciencias de la Computación e Inteligencia Artificial** de la Universidad de Granada bajo la dirección de los doctores **Miguel Delgado Calvo-Flores** y **Amparo Vila Miranda**.

Javier Medina Quero
Doctorando

Miguel Delgado Calvo-Flores
Director

Amparo Vila Miranda
Director

Agradecimientos

Esta tesis es posible gracias
a la educación de mis padres,
a la confianza de mis tutores,
y al amor de Mariola.

Resumen

El trabajo mostrado en esta tesis avanza en las formas de comunicación entre las personas y los espacios que habitamos. Para ello, hemos desarrollado técnicas que permiten a los usuarios conectarse con las cámaras y micrófonos situados en el entorno usando sus dispositivos móviles. También hemos integrado la comunicación entre las personas, incluyendo a los dispositivos como fuentes multimedia móviles en tiempo real.

Además, el espacio y recursos que rodean a los usuarios se muestra mediante mapas en sus computadores de mano, permitiendo operar y efectuar consultas en los mismos. Además, estos mapas permitirán conocer el estado y cambio en el ambiente gracias a la comunicación con el entorno y los usuarios. Esta comunicación ha sido contemplada de forma bidireccional, permitiendo propagar información y consultar datos con los usuarios desde los mapas semánticos que son mostrados en sus dispositivos móviles.

Movilización de Información Multimedia e Interacción con Espacios Inteligentes

Javier Medina Quero

2010

Índice

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

La transformación más vertiginosa y espectacular de la Edad Contemporánea ha tenido lugar en el ámbito de la Tecnología y Conocimiento. En particular, los años comprendidos entre finales de los 90 y principios del nuevo milenio han estado marcados por la Revolución de la Información que provocó el acceso a Internet.

Paralelamente, gracias al enriquecimiento industrial y económico de las sociedades avanzadas, se han acumulado multitud de dispositivos electrónicos que conviven con nosotros. Entre ellos, destacan los computadores de mano. Gracias al crecimiento espectacular de móviles y de agendas electrónicas, han surgido nuevos modelos de gestión de Información, como la Computación Ubicua.

De forma análoga, existen otros tipos de dispositivos que empiezan a ser instalados en nuestro entorno y que nos permiten interaccionar con las personas y edificios de forma remota. Un ejemplo cotidiano de los mismos, son las cámaras web y de seguridad, aunque actualmente existe una gran variedad de elementos capaces de monitorizar hogares y otro tipo de instalaciones más complejas. Las disciplinas que estudian la interacción del hombre con estos dispositivos, se enmarcan en el paradigma de la Inteligencia Ambiental.

Dando un paso más hacia la interacción con este tipo de ambientes, los Entornos Inteligentes evalúan y analizan el estado de los sensores y dispositivos de un espacio de forma conjunta. De esta manera, pueden ofrecer una respuesta inteligente al operador humano, guiando al usuario por el edificio o incluso, previniendo posibles situaciones de riesgo. Por ello, es crítico representar el espacio en un modelo simbólico común de comunicación, entre las personas y el Entorno Inteligente.

Esta tesis pretende ofrecer unas herramientas para que estas tecnologías puedan integrarse satisfactoriamente y den cabida a los paradigmas y las aplicaciones de un futuro cada vez más cercano.

Escenario

Para ilustrar la investigación de esta tesis, vamos a describir un escenario que ponga en relieve los objetivos y aplicaciones que se desean obtener de la misma. A continuación, presentamos una escena que describe las funcionalidades que pretendemos otorgar a los Sistemas Móviles y a los Entornos Inteligentes, así como destacar su posible impacto en el día a día de las personas.

Como idea general, deseamos dotar de una infraestructura tecnológica que permita comunicar a las personas con los edificios que visitan, y a los edificios con las personas que los habitan; además las personas también podrán interaccionar entre ellas usando computadores de mano, véase la figura 1.1.

Imaginemos, que vamos de vacaciones y cuando nos encontramos desplazándonos en el viaje queremos saber cómo es el hotel donde vamos a hospedarnos. Inmediatamente podemos consultar, desde nuestro móvil, las cámaras exteriores que el hotel tenga como públicas y conocer las infraestructuras e instalaciones que vamos a visitar. Una vez que llegamos al hotel, el Sistema nos ofrecerá la posibilidad de descargar el mapa del edificio y, una vez instalado, nos mostrará una ruta desde la recepción hasta nuestra habitación. Además, este mapa también podrá indicarnos cómo llegar hasta otras instalaciones, como la piscina o los restaurantes del complejo.

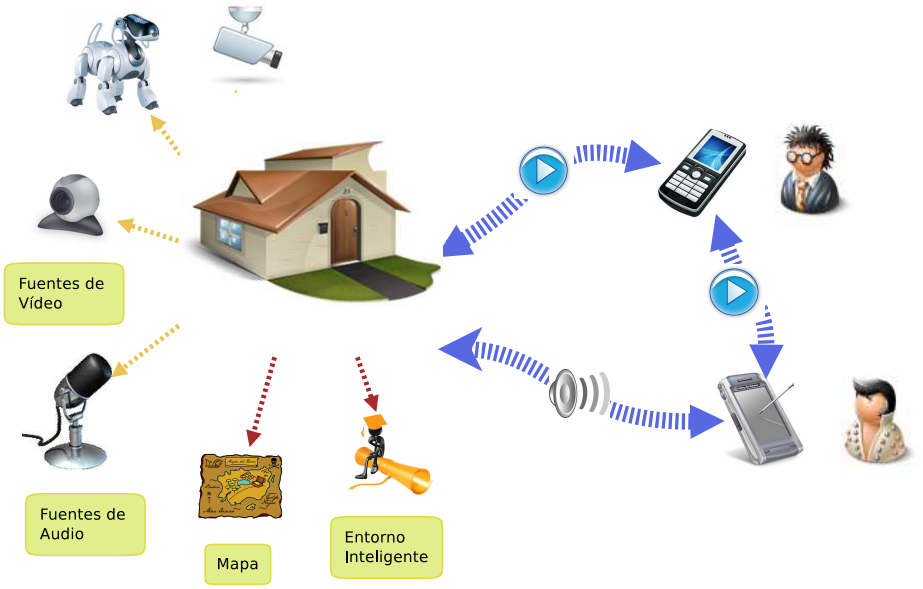

Figura 1.1: Interacción entre las personas y el edificio

Después de dejar las maletas y acomodarnos, nos gustaría hablar con nuestros familiares para informarles de que hemos llegado al hotel. Para ello, usamos el dispositivo móvil que llevamos en nuestro bolsillo y además podemos mostrarle la decoración y las vistas de nuestra habitación usando la cámara del teléfono. Afortunadamente no tenemos prisa en colgar, porque la conferencia navega por la red y es gratuita, evitando los costes de las vídeo llamadas de las compañías de teléfono.

A mediodía, después de ver la previsión meteorológica, nos sentimos algo preocupados porque no sabemos si el mal tiempo habrá afectado al jardín de nuestra casa. La inquietud dura poco tiempo porque volvemos a sacar el móvil del bolsillo y nos conectamos de forma privada a las cámaras de nuestra casa. Comprobamos el estado del jardín y de la piscina. Parece que por ahora no ha habido ningún desperfecto, y además estamos tranquilos porque el Sistema de Vigilancia está activado y nos avisará al móvil ante cualquier situación de emergencia, explicándonos dónde y por qué ha ocurrido

un problema. Finalmente, llega un aviso a la aplicación de nuestro computador de mano: es el sistema del hotel que nos recuerda que es la hora de almorzar. También nos sugiere que marquemos nuestra localización en el mapa para ofrecernos la mejor ruta hasta el restaurante que habíamos contratado.

Este idílico escenario parece ser descrito en un futuro lejano. Sin embargo, si lo analizamos con detenimiento, observamos que en la actualidad disponemos de todos los ingredientes necesarios para hacerlo realidad. El acceso a los teléfonos móviles está más que generalizado, y ya existen infinidad de dispositivos multimedia capaces de describir nuestro alrededor. En esta tesis se propondrá la integración y desarrollo de tecnologías que permitan comunicar estos dispositivos y ofrecer servicios que sean accesibles y mostrados desde la palma de la mano de los usuarios.

Es importante aclarar, que no pretendemos resolver un escenario concreto o construir un Sistema de Monitorización para un uso específico, como la publicidad, la gestión o la vigilancia; sino desarrollar herramientas genéricas que nos permitan comunicar estos subsistemas y poder ofrecer soluciones que sean fácilmente adaptables a cualquier escenario y a cualquier campo de aplicación.

1.2. Objetivos Generales

De acuerdo a la situación descrita en el apartado anterior, nuestro trabajo se va a centrar en estudiar las formas de interacción entre los usuarios y el entorno mediante dispositivos electrónicos. En concreto, dotar de herramientas para que los usuarios puedan visualizar la representación de los edificios mediante mapas y acceder a los recursos audio-visuales de los mismos en tiempo real. A su vez, estudiaremos la importancia de incorporar a los propios usuarios en el sistema, permitiendo que añadan información a los mapas e integrando sus dispositivos móviles como fuentes multimedia accesibles.

Esta funcionalidad ha sido concretada en una serie de objetivos que son descritos a continuación y que representan de forma detallada

las características más importantes que deseamos incorporar en nuestro trabajo:

- El primer objetivo de esta tesis es **desarrollar una aplicación que nos permita integrar los diferentes recursos multimedia de nuestro entorno**. Hoy día existen multitud de cámaras y micrófonos a nuestro alrededor, pero cada uno ofrece una conectividad y características diferentes por lo que acceder a las fuentes de forma homogénea no es trivial.

  Como detallaremos posteriormente, para abordar esta problemática es necesario realizar un proceso previo de integración, que unifique la reproducción de las fuentes multimedia existentes en las instalaciones. A su vez, sería interesante establecer unos protocolos y formatos de salida que adaptasen el flujo multimedia a cada cliente y, a su vez, configurasen una solución escalable y ampliable a otras plataforma.

- Otro requisito, que dota de atractivo y potencialidad al sistema, es que **la visualización de las fuentes multimedia ambientales se realice mediante dispositivos móviles**, de forma que la comunicación sea accesible por los usuarios desde cualquier punto del entorno, o incluso fuera de las propias instalaciones donde se encuentren los recursos. Evidentemente, la visualización de vídeo y la reproducción de audio en tiempo real requieren de un tratamiento diferente, ya que la decodificación del flujo multimedia es distinta y la presentación de los resultados se realiza en la pantalla o en el altavoz del dispositivo móvil.

- Una vez presentado el proceso de integración de las fuentes multimedia y su reproducción en dispositivos móviles, es realmente novedoso **la introducción de los propios dispositivos como fuentes multimedia de información, ofreciendo flujos de vídeo o audio en tiempo real** en función de las características de los mismos. Para ello, es necesario acceder a la cámara y al micrófono de los computadores de mano de forma eficiente y continua, generando la transmisión a la vez que se realiza la captura

de muestras. Esta característica dota a los sistemas de un gran atractivo y potencialidad.

- Además de desarrollar técnicas de propagación multimedia entre el ambiente y los dispositivos móviles, hemos estudiado mecanismos de comunicación e interacción entre el hombre y el Entorno Inteligente. Para ello, hemos establecido **el uso de mapas como marco de representación y comunicación con los edificios**. Los mapas establecen un modelo apropiado porque en ellos es posible determinar la localización, forma y elementos involucrados en los mismos. La definición gráfica del entorno es representable desarrollando un modelo geométrico apropiado y las transformaciones asociadas al mismo. Además, para ofrecer una interacción atractiva a los usuarios, es necesario integrar la navegación por los mapas con operaciones de escala y traslaciones; así como, ofrecer un mecanismo de selección de los objetos representados.

  Por otro lado, nuestro objetivo es diseñar mapas que aporten, además de las características geográficas, una definición de los elementos y cualidades de los mismos. La incorporación de semántica permitirá a los dispositivos móviles visualizar todo tipo de recursos y operar con los elementos representados en los mapas, ofreciendo una navegación por plantas e interactuar con los objetos en función de sus características. Estos mapas con representación gráfica y semántica han sido denominados en nuestro trabajo como *Mapas Semánticos*.

- También estudiaremos las posibilidades para **incluir el Cálculo de Rutas en interiores dentro de los dispositivos móviles**. Estos algoritmos han sido muy útiles y populares en exteriores, y pueden ser trasladados, como veremos bajo algún matiz, en edificios. Para minimizar los tiempos de búsqueda y ofrecer un cálculo más ligero dentro de la computación móvil, se incluirá una heurística original para el enrutamiento basada en la intuición humana, y formalizada como una minimización de ángulos.

- Los Mapas Semánticos son un modelo perfecto para representar

la información del entorno y para compartir esos datos con otros computadores, ya que disponen de una parte gráfica y otra semántica. Por eso, hemos propuesto a **los mapas como eje de comunicación bidireccional entre el usuario y los Entornos Inteligentes**. Por ejemplo, los Mapas Semánticos pueden mostrar la localización de las personas y objetos móviles que los sensores recogen, o ser usados para que un operador humano determine su localización de forma cómoda sobre los mismos. Finalmente, extenderemos el uso de los Mapas Semánticos en dispositivos móviles para **mostrar la localización de usuarios en tiempo real** o para **representar las alertas que el Entorno Inteligente puede emitir**.

- Aunque el crecimiento de los dispositivos en los últimos años ha sido impactante, las técnicas y herramientas para manejar homogéneamente la tecnología no han progresado a la misma velocidad. Los principales problemas han sido provocados por las barreras de integración entre plataformas, cuyas principales causas son el enorme mosaico de dispositivos y protocolos, agravados por la falta de estandarización de los fabricantes. Esto ha provocado que las limitaciones se traduzcan en importantes dificultades para desarrollar Sistemas Móviles: a mayor número de computadores de mano y sensores, mayores problemas para comunicarlos.

  Para ofrecer una solución genérica y maximizar el número de dispositivos sobre los que aplicar los resultados de la tesis, emplearemos **Máquinas Virtuales**, las cuales nos permiten garantizar la portabilidad de las aplicaciones y de los resultados a otros dispositivos, ya sea parcial o totalmente. Una Máquina Virtual ofrece una solución que abstrae la máquina física de un dispositivo, construyendo otra "virtual" que hace de intermediaria entre la aplicación y el hardware. Gracias a esta abstracción una misma aplicación puede ser ejecutada en varios dispositivos, siempre y cuando éstos dispongan de una Máquina Virtual que interprete el código genérico a instrucciones nativas.

  Aunque el uso de las Máquinas Virtuales aumenta la portabilidad

parcial o total de los desarrollos obtenidos, sin embargo presenta limitaciones en cuanto a la velocidad de cómputo y las posibilidades para acceder a los recursos multimedia de los dispositivos móviles. En este trabajo veremos cómo enfrentarnos a esa problemática y evaluaremos su capacidad para realizar procesos de reproducción y transmisión de flujos multimedia en tiempo real que requieren de altos costes computacionales y que pondrán a prueba las capacidades de las Máquinas Virtuales.

1.3. Organización de la Memoria

Una vez definidos los objetivos generales de la tesis, vamos a describir la organización del documento para facilitar la lectura del mismo.

El contexto de los dispositivos móviles en la actualidad, junto a *los requerimientos del sistema y la propuesta* de desarrollo serán estudiados en profundidad en el **capítulo 2**.

Inicialmente, daremos una visión de los fabricantes, los trabajos y herramientas para dispositivos móviles bajo la temática de la tesis. Posteriormente, se describirá la plataforma de desarrollo elegida: Java Micro Edition, y pondremos en relieve las ventajas que ofrece respecto a desarrollos de aplicaciones nativas; pero también las limitaciones que impone el uso de Máquinas Virtuales sobre el uso de los formatos y protocolos multimedia.

Continuaremos describiendo brevemente las características de las fuentes multimedia ambientales que van a ser integradas, así como los protocolos y formatos para transmitir los flujos de vídeo y audio de forma homogénea.

En este mismo capítulo, trataremos el acceso a estos y otros servicios usando un Middleware compatible con los requerimientos de la movilidad, y que nos permitirá comunicar dispositivos y computadores de forma transparente.

Finalmente, y a modo de resumen, representaremos la organi-

zación de los todos los componentes del sistema bajo una *Arquitectura* que incluirá Servicios Multimedia para la reproducción de fuentes en tiempo real y Servicios Remotos para la sincronización de los Mapas Semánticos.

Los desarrollos referentes a las capacidades multimedia en dispositivos móviles serán presentados en el **capítulo 3**.

En primer lugar, se presentará el Servidor Multimedia para *recursos multimedia ambientales*, encargado de generar un flujo de vídeo y audio desde las fuentes del edificio usando los protocolos y formatos especificados en el capítulo anterior. La reproducción de estas fuentes podrá ser capturada desde *aplicaciones móviles* desarrolladas ad hoc. Los resultados de la reproducción de audio y vídeo se presentan de forma independiente, para mostrar las peculiaridades de cada formato. Además, para aprovechar las capacidades de los dispositivos, se han realizado trabajos separados para dispositivos ligeros, como teléfonos móviles y para computadores de mano, como PDAs.

Finalmente, y como uno de los puntos más destacados de la tesis, describiremos los desarrollos sobre la *generación de flujos multimedia desde los propios dispositivos móviles*, dividiendo los resultados, a su vez, en dispositivos ligeros o computadores de mano, y por formatos de audio o de vídeo.

La representación e interacción con el espacio será estudiada en el **capítulo 4**.

Inicialmente presentaremos los *Mapas Semánticos como modelo de representación de interiores*, con una parte gráfica asociada a la localización y otra semántica que etiqueta los elementos definidos en la ontología. Posteriormente, introduciremos el Algoritmo A* para ofrecer un *Cálculo de Rutas* en interiores, presentando una novedosa heurística basada en la minimización de ángulos.

Las últimas secciones del capítulo harán referencia a los Servicios Remotos que permiten a los usuarios *interactuar con el Entorno Inteligente* a través de Mapas Semánticos, desarrollando un par de ejemplos asociados a la Localización de elementos móviles y a la Notificación de eventos en Tiempo Real.

Finalizaremos la tesis con las conclusiones más importantes derivadas de nuestro trabajo. En ellas, destacaremos los problemas que han surgido durante el desarrollo y las soluciones propuestas para resolver las limitaciones de los dispositivos móviles y del uso las máquinas virtuales. En esta última sección incluiremos las reflexiones sobre trabajos futuros, así como definiremos unas líneas de investigación que deben seguir desarrollándose a nivel personal, pero también haremos hincapié en los avances que se esperan del resto de la comunidad y de las compañías para poder desarrollar sistemas móviles con mayor potencialidad.

Capítulo 2

Estado del Arte y Propuesta

Una vez congregados, procura seguir el consejo del que opine mejor;
necesario nos es uno bueno y prudente,
ahora que junto a nuestros navíos encienden tanta hoguera, mas,
¿Quién estas cosas verá alegremente?
Esta noche se habrá de salvar o arruinar el ejército.

La Ilíada.

En el primer capítulo hemos introducido la temática en la que se ve inmersa la tesis y también hemos descrito los objetivos generales que hemos propuesto abarcar.

En primer lugar, ofrecer una solución homogénea para integrar diferentes dispositivos multimedia que se encuentren instalados en los edificios, tales como cámaras por red o web. Además de los recursos de vídeo, vamos a incluir micrófonos que permitan conectarnos y oír cualquier rincón del entorno.

Los encargados de reproducir estos formatos multimedia serán los dispositivos móviles de los usuarios. Además, la transmisión debe ajustarse al tiempo real y producirse de forma continua, siguiendo

una filosofía *streaming*. La forma de resolución de estos propósitos estará determinada por el uso de Máquinas Virtuales, con prestaciones computaciones bajas pero que nos permiten trasladar los resultados a varios dispositivos móviles.

Además de las capacidades multimedia entre dispositivos móviles, vamos a estudiar las posibilidades de interacción con nuestro alrededor. En esta tesis, hemos denominado como Entorno Inteligente a los diferentes procesos que permiten monitorizar nuestras actividades, ya sea, ayudarnos a realizar alguna labor concreta o bien velar por nuestra seguridad. En la literatura hemos podido encontrar sistemas de muchos tipos y características para sectores tan diversos como la salud o la videovigilancia. La definición de Entorno Inteligente abarca cualquier tipo de tecnología referente a estos aspectos.

Uno de los aspectos más interesante, es que el intercambio de información entre los Entornos Inteligentes y los usuarios sea intuitivo y ubicuo. En este sentido, hemos propuesto el uso de Mapas Semánticos como modelo para definir espacios interiores en los que se encuentra el usuario, pudiendo aplicar sobre ellos procesos locales como el Cálculo de Rutas.

Además, los Mapas Semánticos van a permitir intercambiar y mostrar información entre los usuarios y el Entorno Inteligente de forma bidireccional. Para establecer este intercambio de información, integraremos Servicios Remotos ubicuos, en el que los dispositivos pueden actuar como sumideros o generadores de cambios en el ambiente en tiempo real.

A continuación, presentamos la organización en secciones de este capítulo.

En la **sección 2.1,** se describe el contexto actual de la tecnología y las propuestas referentes a la integración multimedia y la transmisión de la misma en tiempo real. Como veremos, gran parte de los desarrollos son producidos por compañías privadas que ofertan novedosas formas de comunicación móvil. También nombraremos algunos artículos sobre Entornos Inteligentes, destacando la monitorización de personas o el control de espacios, y cómo es posible mantener a los

usuarios informados usando los dispositivos móviles.

Una vez presentados los objetivos generales de la tesis y el contexto de los dispositivos móviles hoy día, vamos a especificar *los requisitos del Sistema* en el **apartado 2.2**. En base a estos requisitos, en las siguientes secciones del capítulo, vamos a describir la propuesta de trabajo junto con los distintos componentes software y hardware que estructuran el Sistema.

En la **sección 2.3** analizaremos los distintos *lenguajes* que pueden emplearse en el desarrollo de dispositivos móviles y explicaremos ventajas y limitaciones de las Máquinas Virtuales. También, describiremos a detalle la plataforma de desarrollo elegida, *Java ME*, que dará cabida tanto a computadores de mano, muy potentes, como a dispositivos ligeros, con menores prestaciones.

Posteriormente, en la **sección 2.4**, indicaremos los *dispositivos multimedia ambientales* que se van a integrar en el trabajo, tales como cámaras y micrófonos. Estos dispositivos serán transmitidos y recibidos mediante a unos *protocolos sensibles al tiempo real* y que se detallan a profundidad en las **secciones 2.5 y 2.6**.

La solicitud y resolución de peticiones remotas entre los dispositivos del Sistema se formalizará mediante un *Middleware*. El Middleware permitirá establecer potentes mecanismos de comunicación para notificar y propagar la información entre el Entorno Inteligente y los usuarios. En la **sección 2.7**, comentaremos las principales características de los mismos enfatizando los requisitos más importantes a tener en cuenta en nuestra propuesta.

La representación del entorno que vamos integrar en los mapas será introducida en la sección **sección 2.8** junto a las dos propuestas de comunicación con el Entorno Inteligente basadas en la localización de objetos y en la notificación de eventos.

Finalmente, en la **sección 2.9**, mostraremos la arquitectura del Sistema. Aquí, se organizan de forma conjunta los componentes que hemos estudiado en el capítulo. Para ello, explicaremos el lugar que ocupan cada uno de los componentes mencionados, así como las comunicaciones e intercambio de información que se producen entre

ellos.

2.1. Contexto actual de los dispositivos móviles

En esta sección detallamos el contexto actual de los dispositivos móviles, en cuanto a plataformas, sistemas y tecnologías que se encuentran en el mercado y en la literatura. Además, como la diversidad de las tecnologías integradas en la tesis es amplia, es necesario describir el avance y estado de varios sectores novedosos. En primer lugar, abordaremos el impacto de los dispositivos móviles de los últimos años y los sistemas multimedia desarrollados sobre los mismos. Posteriormente, destacaremos las últimas tendencias en Entornos Inteligentes, enfatizando las posibilidades de interacción con los dispositivos móviles y en particular con las líneas de trabajo la tesis.

La mayor parte de los objetivos de este trabajo son posibles gracias a la reciente difusión de los dispositivos móviles. El gran avance de la telefonía móvil entre los consumidores se produjo entre finales de los 90 y aún llega con fuerza a nuestros días. Para ser conscientes de la magnitud del fenómeno sólo hay que decir que, en pocos años en España, el número de líneas de móviles ha superado al de las personas, o que se han vendido más de cien millones de dispositivos en menos de quince años según la Comisión del Mercado de las Telecomunicaciones [CMT, 2009], véase figura 2.1. Esto no ha pasado inadvertido entre los investigadores y las empresas que se han hecho eco de los los avances que esto produce en las Ciencias de la Computación.

Motivada por estos cambios, nació la Computación Ubicua [Weiser, 1991], que introducía el concepto de accesibilidad a la información desde cualquier lugar [Greenfield, 2004]. Esta filosofía ha revolucionado al sector tecnológico porque permite introducir y consultar datos desde la palma de la mano de los usuarios, usando un computador ligero que puede ser transportado en un bolsillo. La ubicuidad de la computación ha sido posible gracias al amplio despliegue de los dispositivos que ya hemos comentado. La competitividad del mercado y la aceptación de los usuarios ha

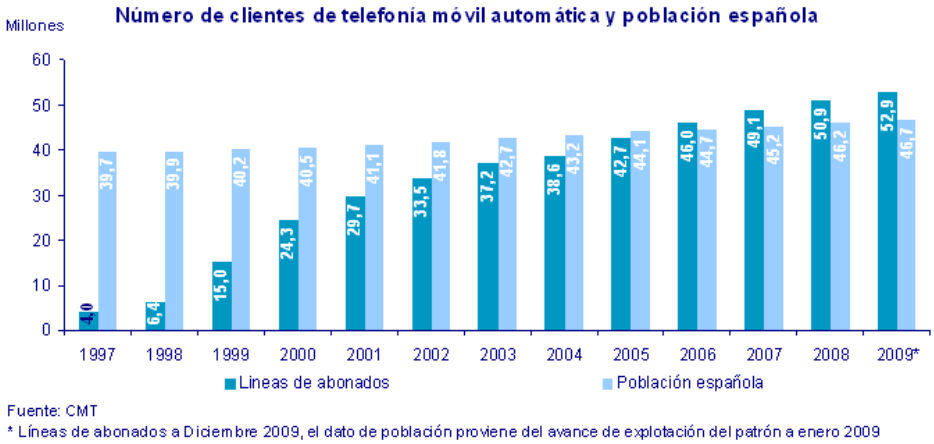

Figura 2.1: Número de clientes móviles en comparación al de personas

impulsado la creación de dispositivos móviles más potentes a la vez que pequeños.

Las posibilidades de los teléfonos han favorecido el desarrollo de aplicaciones que permiten leer el correo, acceder a internet o gestionar los recursos de una empresa. El mercado que se abre a los computadores de mano abarca desde televisión móvil a publicidad dirigida. Por este motivo, las grandes compañías de la Información y las Telecomunicaciones apuestan por lanzar al mercado nuevos y más potentes productos.

Paradójicamente, hay que decir que no abundan las aplicaciones que saquen un gran partido a los recursos que aporta la movilidad. Las causas se derivan de las grandes dificultades y limitaciones que los dispositivos actuales tienen para integrarse y programarse. Existen muchos esfuerzos por parte de las empresas para paliar esta situación. Por un lado, debemos destacar la versión reducida de Java, conocida como Java Micro Edition (Java ME) [Muchow, 2002] que lanzó Sun MicroSystems, ahora propiedad de Oracle Corporation. Describiremos las propiedades de Java ME a mayor detalle en la sección 2.3.1. Esta plataforma ha proporcionado un desarrollo común para los teléfonos

Tabla 2.1: Fabricantes de dispositivos móviles

Posición	Fabricante	Millones de dispositivos	Porcentaje
1	Nokia	432	38
2	Samsung	227	20
3	LG	117	10
4	SonyEricsson	57	5
5	Motorola	55	5
6	ZTE	50	4.5
7	Kyocera	45	4
8	RIM	35	3.5
9	Sharp	29	2.6
10	Apple	25	2.2
-	Otros	56	5
-	TOTAL	1.130	-

y agendas electrónicos de muchos dispositivos gracias a su Máquina Virtual. El concepto de portabilidad es más que esencial en los casos de los teléfonos móviles, ya que su mercado se encuentra repartido y fragmentado entre los fabricantes. Evidentemente, aunque exista una gran diversificación en los mismos, destacan importantes fabricantes como Nokia, Samsung o LG. En la tabla siguiente 2.1 se muestran los datos de los fabricantes a nivel mundial en el año 2009 [Ahonen, 2010].

La tabla la encabeza Nokia, cuyos orígenes son tan remotos que hacen referencia al siglo XIX como compañía maderera. En la actualidad se ha convertido en el mayor fabricante de teléfonos móviles. Lidera el desarrollo del Sistema Operativo Symbian [Community, 2008] que se encuentra en gran parte de sus dispositivos, y también ha apoyado y fomentado el desarrollo de Java ME. En Junio de 2008, Nokia se convirtió en participante mayoritario y liberó la plataforma. Symbian ya se encuentra en la versión 9.5 junto con sus paquetes de desarrollo gratuitos.

Tampoco debemos olvidar a la compañía BlackBerry que ofrece dispositivos móviles desde 1996. Fue creada por la compañía canadiense Research In Motion (RIM) y ha tenido un gran auge gracias a la incorporación del correo electrónico móvil; lo que ha facilitado su pen-

etración en el sector empresarial. Ambas compañías empezaron desarrollando teléfonos sencillos, que cada vez más incluyen potentes características como WiFi, cámara o GPS.

Como las capacidades de los computadores de mano han crecido, y con ellas las posibilidades de desarrollar aplicaciones, las grandes compañías se han involucrado con mucho interés en el mercado. Windows fue una de la más previsora ofreciendo sus computadores de bolsillo *Pocket PC* desde el año 2000. El Sistema Operativo de estas agendas ha variado mucho y hoy se conoce como Windows Mobile. Su última versión es la 7. El desarrollo en Windows Mobile es gratuito, sin embargo, es necesario adquirir una licencia para la comercialización de los productos. Los fabricantes que usan Windows para sus dispositivos son numerosos, desde Toshiba a HTC, pasando por LG y muchos más; incorporando cada dispositivo mayores o menores prestaciones en función del producto.

Por su parte, Apple presentó en 2007 un poderoso dispositivo, el IPhone, que se ha hecho muy popular en el mercado. El gran diseño, tan presente en la compañía, junto con una cuidada e intuitiva interacción del operador con el dispositivo, lo convierten en uno de los mejores teléfonos móviles, y computadores de mano, del mercado. Tiene integrados acelerómetros, WiFi y una gran pantalla multi touch. Como siempre Apple ofrece el hardware y Sistema Operativo en conjunto. La compañía además, controla las aplicaciones y plataformas de desarrollo de forma celosa.

Finalmente, Google se unió al mercado presentando Android en 2007, pero tardó un año en ofrecer comercialmente el producto. A diferencia del IPhone, Android es un Sistema Operativo basado en Linux que puede ser integrado en los dispositivos de varios fabricantes, aunque el gran aliado por el momento es la compañía taiwanesa HTC. Android también integra acelerómetros, comunicaciones WiFi o cámaras. Además la libertad y posibilidades de desarrollo que ofrece Google lo sitúan como una potente opción en crecimiento. En la siguiente tabla 2.2, podemos ver la penetración de las grandes compañías en los computadores de mano, también llamados Smartphones. También presentamos a continuación, otra tabla 2.3

Tabla 2.2: Fabricantes de computadores de mano

Posición	Fabricante	Millones de dispositivos	Porcentaje
1	Nokia	68	39
2	RIM	35	20
3	Apple	25	15
4	HTC	8	5
-	Otros	35	21
-	TOTAL	175	-

Tabla 2.3: Sistemas Operativos en computadores de mano

Posición	Fabricante	Porcentaje
1	Symbian	45
2	RIM	20
3	Apple	15
4	Windows Mobile	6
5	Google Android	4
-	Otros	10

que incluye los porcentajes de los Sistemas Operativos que usan los mismos.

Junto a la CU, surgió el término de Inteligencia Ambiental [Zelkha and Epstein, 1998], como el estudio de dispositivos empotrados en el entorno que ayudan a las personas en el desarrollo de sus actividades cotidianas. De esta forma, la CU e IAm permiten construir sistemas móviles y empotrados que pueden ser usados por el hombre en el ejercicio de sus labores, y que serían inviables bajo la perspectiva estática y pesada de los primeros computadores personales. En resumen, la IAm trabaja para integrar los dispositivos electrónicos que cohabitan en nuestro entorno.

La IAm ha estado muy unida al término Domótica, que hace referencia al conjunto vivienda y dispositivos que están integrados de forma automática para garantizar el bienestar de las personas que los habitan. En la actualidad existen comercialmente pequeños componentes electrónicos que nos permiten conocer la temperatura

de una habitación o el nivel de ruido de la misma. Un sector muy interesado por esta tecnología es la vigilancia, que ha introducido elementos como sensores de presencia o movimiento. Pero en IAm, no sólo son importantes las recepciones de las señales, cobra igual relevancia la Inteligencia del Entorno, y cómo éste es capaz de actuar y predecir situaciones. Entre los actuadores más populares, podemos destacar los motores de persianas, las cerraduras de puertas, y otros elementos de cierre adaptados a la maquinaria de la empresas.

En el caso de esta tesis, los dispositivos que vamos a incorporar son cámaras y micrófonos ambientales que están situados en el edificio. Como detallaremos más adelante, es necesario realizar un proceso de integración para poder trabajar con estos dispositivos de forma homogénea. A su vez, veremos cómo aplicar técnicas específicas de transmisión en tiempo real.

Para el desarrollo de propuestas ambiciosas en el ámbito de la CU, es necesario trabajar aspectos de telecomunicación e información de diversa índole. Esto ha provocado que las investigaciones en CU e IAm hayan proliferado en diferentes niveles tecnológicos, destacando por su transversalidad. La potencialidad de esta capa radica en la abstracción de redes de comunicación, tales como Wi-Fi o Bluetooth, la lectura de datos desde dispositivos como GPS, RFID o la integración de recursos multimedia, como cámaras y micrófonos empotrados en el entorno. Por ejemplo, a un bajo nivel, en las capa de sesión o transporte los esfuerzos se han concentrado en la integración de dispositivos y sensores, resolviendo las limitaciones de los servicios en las redes de sensores, destacando el trabajo de [Moya et al., 2009] para integrar la orientación a objetos en los mismos. También se ha trabajado en la integración de seguridad en la red de comunicaciones en entornos ubicuos [Yamada and Kamioka, 2005].

Capacidades Multimedia de los dispositivos móviles. De forma paralela a estos avances, surge el uso de los formatos multimedia en Internet. Gracias a ellos, en la actualidad disfrutamos de tecnologías muy atractivas que han revolucionado la difusión por vídeo y audio. Entre éstos, destacan las videoconferencias, la transmisión de sonido

en las conversaciones o de la visualización de vídeos en la red. La integración multimedia de los computadores ha supuesto una verdadera revolución para las comunicaciones, y como veremos en esta sección tiene una especial atractivo cuando se une a la movilidad.

Desde el año 2000 podemos encontrar literatura que empiece a integrar pequeños avances multimedia en las computadoras de mano. Uno de los primeros trabajos lo encontramos en [Li et al., 2002] y nos permitía ponernos en situación de las restricciones para el desarrollo multimedia móvil con las que podemos encontrarnos. El Sistema que se propuso está orientado a vigilancia y constaba de una aplicación para PDA que permitía consultar un Servidor y visualizar mensajes. Los mensajes estaban representados por texto y podían contener una imagen que había sido tomada por una cámara del edificio. En el artículo se discuten los tipos de imágenes que soporta Java ME. Esto se alejaba bastante de las pretensiones de muchos sistemas de sobremesa que ya contaban con transmisión de vídeo y sonido, pero hasta ese momento era lo que se podía hacer.

Para paliar las necesidades de las aplicaciones en los computadores de mano, Java ME diseñó una librería sobre formatos multimedia, MMAPI. En [Tierno and Campo, 2005] se evalúan las mejoras de los formatos en las imágenes (JPG, BMP, PNG), pero todavía no se habla de vídeo o sonido. Poco después, en [Serpa and Rodriguez, 2007] se analizan todos los detalles de la nueva librería MMAPI y la posibilidad de visualizar vídeo y audio en tiempo real desde los móviles. Sin embargo, las capacidades para reproducir y visualizar formatos multimedia en tiempo real, pasan por llamar de forma nativa a librerías o terceras aplicaciones que residen en el dispositivo.

Los motivos de esta delegación de funciones viene derivada de las limitaciones de cómputo de las Máquinas Virtuales que no pueden trabajar con codificaciones complejas. Por ejemplo, la solución depende de cada dispositivo y de la Máquina Virtual, de forma que sólo la gama alta de Nokia y algún otro fabricante incluye la visualización bajo estos protocolos. De la misma forma, ocurre con la decodificación: cada dispositivo tiene unos formatos diferentes al resto. Además, esta limitación no depende de la potencialidad del dispositivo, más bien

de la benevolencia del fabricante por dotar de potencialidad a Java ME. Por otro lado, tampoco es posible generar un flujo de audio o vídeo en tiempo real desde los computadores de mano, con lo que los sistemas móviles no pueden actuar como fuentes multimedia, limitando sobremanera el desarrollo de las comunicaciones.

En los trabajos de esta tesis, detallaremos como podemos resolver esta limitación y desarrollaremos aplicaciones móviles que se encargan de recoger y decodificar los formatos sin delegar estas tareas en otras soluciones.

Una versión más reciente y con relación con las Redes Sociales, la encontramos en los primeros Sistema Móviles de Vigilancia en [Wu et al., 2005], donde los usuarios pueden enviar fotos de su móvil a otros usuarios usando MMAPI y un protocolo seguro de e-mail. En este caso, para poder usar los dispositivos como fuentes de información los autores usan imágenes estáticas como formatos multimedia, pero no pueden incluir el audio o el vídeo como flujos constantes de información en tiempo real.

Probablemente una de las referencias más cercanas a la línea de nuestro trabajo es [Menkens et al., 2007]. Además, en este artículo se introduce la necesidad de incluir mapas y la localización de los usuarios como complemento a la transmisión de voz. Respecto a la transmisión multimedia, los clientes pueden enviar a un servidor información multimedia obtenida desde teléfonos móviles. Esta información es reenviada a otros móviles posteriormente. Como se indica en el artículo, la limitación de los dispositivos no permite hacer una visualización en directo de los recursos, sino que han de acabar de grabarse en el dispositivo para poder ser enviados y visualizados por el resto de usuarios.

Esta filosofía de comunicación se conoce como *Pulsar para Hablar* o *Push to Talk* (PTT), y permite la transmisión de sonido apretando un botón y transmitir lo capturado durante ese tiempo al liberarlo. La transmisión representa un esquema simplificado de comunicación *hall-duplex*, donde sólo es posible recibir o enviar el sonido, y no efectuar ambos procesos simultáneamente. Además, en

la propuesta de [Menkens et al., 2007] tampoco es posible transmitir en tiempo real los datos capturados, sino que se almacenan localmente en el dispositivo móvil y, cuando se finaliza la grabación, se envían al completo. La solución hace a la propuesta insensible al tiempo real, ya que si una grabación dura cinco minutos, los oyentes deberán esperar al menos el mismo tiempo para empezar a escucharla. En nuestra propuesta veremos cómo acceder a esta información al vuelo, sin que previamente haya sido almacenada usando técnicas de *streaming*.

En cuanto a los métodos para la encapsulación multimedia en dispositivos pequeños, encontramos adaptaciones de los métodos actuales, como en [Zhang et al., 2008] donde se estudia la inclusión de marcas de tiempo relativas entre paquetes. En [Yoo et al., 2008], se presenta una adaptación a las reservas de canales y rutas para las transmisiones en tiempo real ad hoc dispositivos móviles en base a la calidad de servicio de cada cliente. Estas y otras propuestas se centran en las técnicas sobre las capas de comunicación y no sobre el nivel de aplicación, y nos sirven para comprobar que la transmisión multimedia móvil es muy reciente y las tecnologías para aumentar las prestaciones siguen actualizándose y evolucionando.

La generación de flujos multimedia desde los computadores es una herramienta muy atractiva porque ofrece la posibilidad de realizar llamadas que navegan por la red sin tener que usar las llamadas de pago de los teléfonos y a expensas de la compañía que tarifica el servicio. Para ello, se introdujeron sobre las redes IP la implantación de protocolos ligeros y no orientados a conexión, como UDP [Postel, 1980] o RTP [Schulzrinne et al., 2003], que produjeran una transmisión rápida sobre la que enviar los datos.

También se estudiaron con detalle los formatos para codificar y comprimir el audio y el vídeo, ya que las características de transmisión en tiempo real no permitían trasladar, tal cual, las codificaciones de los ficheros multimedia clásicos. Los formatos de audio más populares que permiten el envío de forma adaptada al tiempo real son G.711, G.723.1, G.729. Sobre estos protocolos estándar se han desarrollado multitud de variaciones que permiten adaptarlos mejor a anchos de banda reducidos o mejorar sus prestaciones para la mejorar la calidad,

[Hiwasaki et al., 2006] y [Hiwasaki and Ohmuro, 2009]. Los formatos de audio serán estudiados con más detalle en el capítulo 3 donde se propone su implementación dentro de las Máquinas Virtuales de Java ME.

Gracias al potencial que ofrecen las llamadas por la red, muchas compañías se interesaron rápidamente por la tecnología de voz ip. Una de las más conocida es Skype, que fue fundada en 2003. Esta compañía ofrece unos mecanismos de codificación propietarios, además de garantizar la seguridad cifrando las solicitudes de las llamadas, como los propios datos de voz. La compañía lucha por ampliar la portabilidad de sus productos a todo tipo de dispositivos y redes. En la literatura encontramos un estudio [Caizzone et al., 2008] sobre los problemas de escalabilidad de Skype al aumentar los usuarios y el uso de sus servicios multimedia.

De forma paralela, también se han desarrollado otros estándares abiertos que permiten el acceso libre de los usuarios a recursos multimedia remotos efectuando llamadas IP. Entre ellos:

- El más destacado es Session Initiation Protocol SIP [Rosenberg, 2002] que permite iniciar, modificar o finalizar sesiones de voz en tiempo real. Fue creado por Internet Engineering Task Force, en un esfuerzo por unificar y estandarizar el acceso a las sesiones multimedia.

- SDP [Handley and Jacobson, 1998] es el protocolo más antiguo, concretamente en 1998, y permite la inicialización con parámetros de flujos multimedia. Existen trabajos en la literatura que han usado este protocolo.

- H.323 es una recomendación del ITU-T (International Telecommunication Union) y fue el primer estándar de VoIP en adoptar el protocolo RTP.

- IAX2 (RFC 5456) es un protocolo creado por la plataforma Asterisk. Actualmente se ha mejorado la versión inicial IAX, especificando la versión dos que constituye el estándar RFC.

En lo referente a la investigación, existen muchos trabajos en la literatura que han estudiado la interoperabilidad de estos protocolos de acceso en distintas redes y plataformas. En concreto, se han realizado varios estudios para adaptar, de la forma más eficiente, la transmisión en tiempo real sobre la limitada red móvil. Entre ellos se encuentra [H. et al., 2000], que evalúa el protocolo H.323 sobre GSM. También se han realizado avances en la comunicación con los ya mencionados protocolos SDP o SIP, etc, y entre las que destacamos la evaluación sobre las nuevas redes de telefonía UMTS [Pous et al., 2005].

En contraposición a la privacidad de Skype, han surgido varias plataformas libres para soportar las llamadas IP, tales como QuteCom [OpenWengo, 2005] o Asterisk [Spencer, 2002]. Asterisk es un popular programa que ha ampliado la funcionalidad de las llamadas IP, incluyendo buzón de voz, conferencias, o distribución automática de llamadas; pero gracias a la liberación de su código, puede adaptarse para propósitos específicos. Además dispone de versiones para Windows, Linux y MacOS.

Tampoco las grandes compañías como Google o Microsoft no han querido quedarse atrás y disponen de herramientas para realizar llamadas por la red. La primera de ellas, conocida como GoogleTalk [Google, 2005], ha sido integrada con Gmail y el chat en tiempo real que ofrece la compañía. A su vez, Microsoft también las ha incluido en la popular aplicación del Messenger. Ambas incorporan una codificación privada, e incluyen videoconferencias en equipos de sobremesa.

Los primeros resultados experimentales producidos al respecto de la voz IP, se realizaron desde computadores de sobremesa que permitían hablar por el micrófono y reproducir la señal de audio del destino por los auriculares. Sin embargo, rápidamente se vio la utilidad de introducir la movilidad sobre las llamadas IP. Por ejemplo, se han diseñado varios dispositivos, móviles o no, que integran directamente esta tecnología, en lugar de la clásica de los teléfonos, el *Global System for Mobile Communications* (GSM) [Cept, 2002]. Sin embargo, se ha visto con mayor utilidad la de incorporar las llamadas IP a los teléfonos ligeros comunes mediante una aplicación ex profeso, de forma que no es necesario cambiar de dispositivo, sino simplemente instalar una

aplicación en un teléfono clásico.

Los teléfonos conectados por Wi-Fi pueden realizar y recibir llamadas IP, usando su tarjeta de red, aunque no es el único camino para conectarse a internet. Para poder ampliar la oferta de servicios del Sistema Global para las Comunicaciones Móviles (GSM), se definió el *General Packet Radio Service* (GPRS), gracias al cual podemos acceder a Internet. Los servicios que amplían las posibilidades de la voz a internet, son conocidos como telefonía de tercera generación (3G). De forma más reciente, se está trabajando en la implantación del Sistema Universal Mobile Telecommunications System (UMTS) que soporte un mayor ancho de banda para internet móvil. Gracias a los servicios 3G, podemos conectarnos a Internet en cualquier lugar con la cobertura apropiada usando nuestro móvil.

La adaptación de las redes 3G para usos de llamadas IP, choca con los intereses de las compañías telefónicas, ya que con la navegación de internet los usuarios podrían efectuar llamadas sin usar las tarifas clásicas de voz. Para evitarlo, las compañías de telefonía prohíben explícitamente el uso de esta tecnología, y fabricantes como IPhone han llegado a eliminar aplicaciones que lo permiten. Estas restricciones han sido denunciadas públicamente [Reventos, 2009], [Jiménez, 2003]. Afortunadamente, las trabas empiezan a desaparecer, gracias también a la intervención de la Unión Europea en el asunto.

Las llamadas mediante IP en dispositivos móviles son una tecnología reciente, y de la que rápidamente se han hecho eco compañías como Skype o Fring [Shechter, 2006]. Fring es un programa gratuito para terminales móviles que permite realizar llamadas de voz y mantener conversaciones de mensajería instantánea a través de la conexión a Internet del teléfono, ya sea WiFi, 3G o incluso GPRS. Se conecta a las principales redes de VoIP: IAX, SIP, GoogleTalk o MSN. Los dispositivos sobre los que opera actualmente Fring son Symbian, Android o IPhone, creando aplicaciones de forma nativa para cada uno de ellos. Además, ha desarrollado una versión reducida *MiniFring* para Java ME, pero que sólo permite establecer mensajes en tiempo real entre los usuarios.

Las prestaciones que han acompañado a los equipos de sobremesa junto a la popularidad y bajo coste de las cámaras web, proporcionó a las llamadas por IP el acceso al vídeo de los participantes que intervenían en la conversación. Esta tecnología, junto a la voz IP, se conoce como videoconferencia. De forma análoga a la voz, existen una serie de formatos para vídeo que se han hecho populares; los más destacados son H.261, H.263 y el reciente H.264 conocido como MP4. Estas codificaciones permiten la reducción del tamaño de los vídeos efectuando una compresión de los datos en tiempo. Para ser transmitidos por protocolos ligeros de tiempo real se han definido los estándares H.261 [Even, 2006] o H.264 [Wenger et al., 2005] que determinan cómo empaquetar el vídeo según el protocolo RTP.

Tenemos que puntualizar que uno de los aspectos más negativos de estos codecs es la licencia privativa de los mismos y el elevado precio de la misma, derivados de la patente que lo protege. Esto ha hecho por ejemplo, que el popular navegador Firefox no incluya a H.264 como códec. En la literatura científica [Cha et al., 2003] [Gualdi et al., 2008] podemos encontrar estudios sobre la generación de flujos de vídeo adaptados a los dispositivos móviles.

Debido a las limitaciones de las licencias, Google compró el verano del 2009 a ON2, una corporación especializada en desarrollar tecnologías de compresión de vídeo. ON2 tiene como producto estrella al códec VP3 [Theora, 2004], que aunque surgió con licencia propietario, fue donado al público como código abierto y la compañía rechazó todos los derechos que tenía sobre éste, incluyendo el uso de patentes. Actualmente han surgido muchas versiones derivadas de VP3 que mejoran su comportamiento, hasta llegar a la VP8. Según la ON2, este tipo de formatos presentan una mejor adaptación en procesadores limitados, como el de los dispositivos móviles, que el del resto de competidores. Además, estos formatos pretenden resolver el problema de la licencia y patentes: en Febrero de 2010 la Free Software Foundation solicitó a Google la liberación del códec VP8, lo cual ha sido concedido en Marzo de este mismo año.

Como veremos en el capítulo siguiente, 2.6, el uso de estos formatos tan pesados hace inviable su decodificación en las Máquinas

Virtuales, por lo que se propondrá una compresión ligera que permitirá, no sólo la decodificación del vídeo en tiempo real, sino generar y codificar el mismo desde los dispositivos móviles.

Desde el punto de vista móvil, la incorporación del vídeo en tiempo real ha permitido ofrecer, por ejemplo, televisión móvil. Esta tecnología sitúa a los dispositivos móviles como sumideros multimedia. Los reproductores que se encargan de visualizar los flujos de vídeo en tiempo real, suelen estar integrados de forma nativa en los dispositivos. Es el caso de *RealPlayer* para los Sistemas Operativos Symbian o Windows Mobile, o *TVUPlayer* en IPhone.

De forma independiente a las plataformas, debemos destacar el reproductor de código libre conocido como VideoLan (VLC) que integra muchos protocolos y formatos, incluyendo los comentados sobre tiempo real, audio y vídeo. La construcción de reproductores se realiza sobre el proyecto *ffmpeg* [Lantau, 2000] que convierte y transforma formatos multimedia. La gran ventaja del mismo, es que está disponible para Windows y Linux, y empieza a ser portado parcialmente en algunas plataformas móviles, pero, a día de hoy, de modo experimental.

De forma inversa, los dispositivos móviles pueden verse también como generadores de vídeo y no sólo como sumideros. Esta filosofía se conoce como *broadcast*. Aunque formalmente, *broadcast* hace referencia a un tipo de comunicación en el que un emisor envía un paquete a varios receptores. En el caso de la generación de vídeo desde los computadores móviles, se entiende como la capacidad de enviar el flujo de imágenes de la cámara del dispositivo en tiempo real a varios receptores. Esta novedosa técnica es usada, junto al sonido, para emitir en vivo conciertos, entrevistas o cualquier experiencia de los usuarios. Actualmente, los avances realizados por estas compañías esperan ser muy lucrativos, por ejemplo en sectores como la televisión, y que la capacidad de generar alta calidad sobre las transmisiones desde móviles permita a los reporteros cubrir noticias y espacios cómodamente, usando sus computadores de mano. A su vez, el broadcast puede ser una revolución para el usuario que desee compartir escenas en directo desde cualquier parte del mundo a través de las redes sociales.

También han sido compañías privadas quienes empiezan a ofertar productos *broadcast* desde los dispositivos móviles. En concreto destacamos Ustream [Ham and Feher, 2007], Qik o Bambuser. Ustream es una plataforma multimedia cuya popularidad creció cuando permitió visualizar televisión en internet. A día de hoy, ha realizado una versión de broadcast móvil que permite la transmisión desde terminales Symbian, Android e IPhone. Bambuser amplía la oferta a los numerosos dispositivos con Windows Mobile y finalmente, Qik representa el repertorio más amplio añadiendo Blackberry.

Como podemos observar los desarrollos se han realizado sobre los Sistemas Operativos, creando aplicaciones nativas que no pueden ser portable. Esto es debido a los costes computacionales del proceso. Evidentemente, los dispositivos compatibles cambian en cada momento debido al efervescente mercado móvil y por las apuestas de ampliación de las compañías a otras plataformas. Todos estos productos pueden sincronizarse con muchas plataformas de redes sociales, como Facebook o Twitter.

Análisis multimedia en dispositivos móviles. Dejando a una lado la transmisión multimedia de las fuentes móviles en tiempo real, también existen trabajos para procesar y analizar las imágenes móviles de forma inteligente. Gracias a la potencialidad de los computadores de mano, como las PDAs, los procesos de extracción se están desplazando hacia los propios dispositivos. Un ejemplo puede verse en [Muftah and Mustafa, 2006] donde se desarrolla un Sistema capaz de procesar la información de las cámaras de computadores empotrados y detectar el movimiento en el dispositivo en sí. La aplicación está basada en PDAs con Windows Mobile 2003 o superior. Aunque nuestra propuesta no introduce mecanismos de detección en el vídeo, el artículo nos da una idea de cómo los computadores de mano pueden realizar algoritmos semi pesados sobre flujos multimedia.

Gracias al procesamiento de imágenes en tiempo real y a la capacidad de cómputo de los dispositivos, se ha relanzado una tecnología muy novedosa que ha encajado perfectamente con la movilidad: la Realidad Aumentada, Augmented reality (AR). El

término fue acuñado en 1992 por Tom Caudell [Barfield and Caudell, 2001] definiendo una tecnología capaz de recoger la visualización procedente del mundo físico, y sobreponer sobre ella meta datos informáticos asociados a los objetos reconocidos en tiempo real. Esta filosofía es muy atractiva y práctica porque permite establecer un vínculo entre el mundo físico y virtual [Wagner et al., 2005]. Sin embargo, requiere de procesos complejos de cómputo, como la captura de imágenes en tiempo real, la recogida de sensorización como acelerómetros, el reconocimiento de los objetos de la escena, la incorporación semántica de los mismos y la presentación de la escena combinada. Un ejemplo de esta tecnología puede verse en la aplicación KARMA (Knowledge-based Augmented Reality for Maintenance Assistance) [Feiner et al., 1993]. El sistema KARMA diseña mundos virtuales que explican cómo operar, mantener y reparar equipamiento del mundo real.

Muchas aplicaciones clásicas de AR presentan dispositivos electrónicos que se adaptan a las personas, permitiendo recoger y mostrar la información en el mismo campo de visión. El coste de los mismos y la incomodidad para llevar continuamente las prendas han alejado estas propuestas de los ciudadanos de a pie. Esta limitación parece haber sido resuelta por la Computación Ubicua, ya que permite disfrutar de la AR usando el pequeño computador que llevamos en el bolsillo. En la figura 2.2 pueden verse dos imágenes referentes a la AR clásica y la móvil.

La AR en dispositivos móviles usa la cámara fotográfica para recoger las imágenes del mundo real y los acelerómetros y posicionamiento GPS, para finalmente presentar los resultados combinados en la pantalla. Como podemos observar, existen varios puntos comunes con los objetivos multimedia de la tesis, ya que es necesario establecer mecanismos de captura de imágenes y procesamiento de las mismas en tiempo real. Es evidente que las capacidades de cómputo en computadores de mano son más limitadas, pero pueden realizarse aplicaciones muy atractivas que irán mejorando en el futuro, gracias al avance de la tecnología y de la técnica. Por ejemplo, Wikitude AR Travel Guide [Mobilizy, 2009], una aplicación de

realidad aumentada para Android que permite introducir información turística de interés sobre los edificios del entorno.

Figura 2.2: Realidad Aumentada junto y versión móvil de la misma

Propuestas de movilización en Entornos Inteligentes. En este trabajo vamos a estudiar las formas de comunicación entre los usuarios y los Entornos Inteligentes usando dispositivos móviles. Los sistemas inteligentes situados en el entorno deben recoger datos del ambiente, operar y razonar con ellos y finalmente, presentar los datos a los usuarios en función del contexto en que se encuentre.

En la primera etapa, los sistemas inteligentes se centran en integrar y describir semánticamente datos heterogéneos en un lenguaje común. En este sentido, se han descrito varias tecnologías para co-

municar y definir la información, prevaleciendo el uso de estándares de representación de conocimiento, como XML, el paradigma de Orientación a Objetos, o definiciones más complejas y semánticas, como las Ontologías, que permiten enriquecer y relacionar conceptos. El uso de Ontologías en informática viene determinado por estándares de representación del conocimiento, como Resource Description Framework (RDF) u Ontology Web Language (OWL). Se ha escrito mucho en la literatura acerca del razonamiento lógico de las representaciones ontológicas, y cómo trabajar con herramientas automáticas para representar y razonar, por ejemplo, usando la plataforma Protegeé.

En cuanto a la integración de las ontologías en computadores de mano, los estudios delegan el razonamiento en servidores, debido a las limitaciones de cómputo y a la falta de herramientas para integrar los razonadores dentro de los dispositivos móviles. Los esfuerzos, por tanto, se centran en la comunicación con los razonadores y en presentar los resultados en los dispositivos de mano [Bobillo et al., 2008].

En referencia a la representación de interiores, una propuesta relacionada con nuestro trabajo, se encuentra en [Ramdoyal, 2003] y hace referencia a la necesidad de la semántica en los planos. En este trabajo se introducen las estructuras para definir planos interiores: posiciones, habitaciones y comunicación entre ellos. En esta tesis, detallaremos una Ontología de Edificios que representa dichos conceptos, y veremos cómo razonar con ellos para resolver el Cálculo de Rutas en los mismos dispositivos.

Los mecanismos capaces de simular el razonamiento se sitúan en el nivel superior de aplicación, configurando la capa de Inteligencia. En esta capa se ofrece el mayor incremento de abstracción, alimentándose de los distintos datos que es capaz de recoger el sistema, y que serán usados en la toma de decisiones. En el contexto de la CU e IAm, los Sistemas Inteligentes ofrecen mecanismos para centralizar y gestionar los datos recogidos por los sensores, monitorizando el entorno y comunicándose con los usuarios para guiar sus actividades.

Por ejemplo, como referencia a Sistemas que controlan y supervisan el entorno, [Fraile et al., 2008] realiza un proceso

de vigilancia en casas usando agentes para resolver los procesos de razonamiento y extracción de la información. Este sistema, aunque sigue una filosofía de Agentes implementa agentes en el sentido estricto de FIPA. Para comunicar los diferentes módulos de razonamiento se usan mensajes ACL mediante conexiones bajo el modelo SOA (Service-oriented architecture). La notificación del sistema a los usuarios se realiza mediante Short Message Service (SMS) que son recuperados por móviles y PDAs.

Otra interesante propuesta de Entorno Inteligente la encontramos en [Delgado et al., 2009]. En ella, se introducen elementos de identificación como Radio Frequency IDentification (RFID) para etiquetar los elementos del hogar. Los usuarios disponen de un dispositivo móvil, en el futuro pulseras, para recibir y procesar las lecturas de los objetos. Gracias a un sistema de aprendizaje, es posible detectar acciones peligrosas, como olvidarse las llaves al cerrar la puerta, y notificar a los usuarios mediante su dispositivo móvil.

Un gran sector susceptible de beneficiarse de la monitorización y de los Entornos Inteligentes es el sector de la salud. Actualmente existen sistemas experimentales [Lee et al., 2008a] y [Lee et al., 2008b] que controlan las constantes vitales de los usuarios y notifican de posibles problemas. En estos trabajos se hace uso de dispositivos electrónicos conectados a los usuarios, y que emiten las señales por Bluetooth a los teléfonos móviles de los usuarios. Los resultados son comunicados a los profesionales de la salud de forma telemática usando SMS o GSM, notificando en de la gravedad a los profesionales con unos roles u otros.

En la línea del trabajo anterior se pone de manifiesto los problemas de comunicación sobre el trabajo con los profesionales de la salud. La resolución del Proceso de Enfermería mediante dispositivos móviles fue propuesta en [Medina and Ruiz, 2007] para la correcta interacción y monitorización de los protocolos de Enfermería. En el mismo, situaciones de riesgo eran resueltas entre enfermeros y médicos, minimizando los críticos tiempos de respuesta con el uso de computadores de manos. Para ello, los mensajes y eventos generados por el Sistema o por los profesionales podían ser visualizados mediante

la transmisión y recepción de mensajes.

En esta tesis vamos a presentar el envío de notificaciones en dispositivos móviles incluyendo la representación de la información sobre los Mapas Semánticos. Para dar universalidad a la solución, este problema se ha resuelto de forma genérica, sin especificar notificaciones dependientes de un dominio concreto y pudiendo ser adaptado a cualquier Sistema Inteligente que monitorice el entorno. Este tipo de mensajes son muy útiles, por ejemplo, cuando el Entorno Inteligente se encarga de realizar procesos de vídeo vigilancia y donde es crítico que el usuario pueda situar rápidamente la zona de interés que se encuentra afectada, así como otros parámetros importantes, tales como la ruta de evacuación. El uso de los dispositivos móviles como receptor de dicha información, parece ser complemento idóneo. Este tipo de comunicaciones ha sido estudiado mediante la incorporación de Mapas Semánticos y notificaciones en tiempo real asociadas a los mismos en [Delgado et al., 2007].

La centralización de la localización de los usuarios, tanto en interiores como en exteriores, representa una problemática de gran interés en la CU e IAm, porque nos permite determinar la posición de los usuarios en las escenas a monitorizar. Muchos de los computadores móviles, incluso teléfonos ligeros, incluyen un dispositivo receptor compatible con el Global Positioning System (GPS), de forma que es posible determinar la posición de los usuarios gracias a los veintisiete satélites que le dan soporte. Su uso está prácticamente restringido a exteriores, donde no existen interferencias de onda con edificios y la precisión del error puede ser de metros. Actualmente, gracias a la existencia de comunicación 3G, también es posible determinar la situación de un dispositivo de forma aproximada sin el uso del GPS. Para ello, se usan como referencia las estaciones base que dan cobertura a los teléfonos móviles, efectuando procesos de triangulación para determinar la localización. Éstas y otras técnicas para aproximar la localización [Bruck et al., 2009] pueden aplicarse a las redes de sensores. El uso de muchas de estas tecnologías puede resumirse en el artículo [Zhao, 2002].

El uso de la localización ha sido integrado en varias herramien-

tas. Una de las más actuales es Latitude [Google, 2009], desarrollada por Google, y que permite publicar la localización de los usuarios de nuestra red social en el Google Maps. Unas de las características más importantes es la integración de diferentes protocolos de posicionamiento de forma transparente. Una de las grandes críticas a este tipo de sistemas es la visibilidad de los usuarios y el constante seguimiento que pueden tener de los usuarios, aunque existen trabajos [Cheng et al., 2006] que permiten acceder y consultar estos servicios garantizando la privacidad.

La localización en interiores emplea otro tipo de tecnologías, como los ultrasonidos [Piontek et al., 2007], cuyo estudio es más cercano al área de las telecomunicaciones. En esta tesis, podremos describir la localización de usuarios en interiores a nivel de aplicación, usando los Mapas Semánticos, e incluiremos la posibilidad de integrar servicios implícitos que podrá solicitar el Entorno Inteligente a los operadores humanos, y también servicios explícitos que notificará el usuario. La propagación en tiempo real de las localizaciones podrá visualizarse sobre los Mapas Semánticos de los dispositivos móviles y así, los usuarios puedan percibir la movilidad de los otros usuarios y objetos de forma continua.

2.2. Requerimientos funcionales y no funcionales

En la sección anterior hemos ilustrado el contexto actual de los dispositivos móviles, y los trabajos relacionados con la temática de la tesis. A continuación, especificaremos los objetivos generales de forma más precisa y técnica, para ofrecer, en las siguientes secciones, una propuesta de componentes y organización que cumpla con nuestros requerimientos.

Como ya hemos comentado en el primer capítulo, los principales ejes del trabajo radican en las comunicaciones entre los usuarios y el entorno; en concreto, las transmisiones multimedia y el intercambio de datos mediante mapas. A partir de estos objetivos, hemos definido una

serie de **requisitos funcionales** que debe validar el sistema y que enumeramos a continuación:

- Integración de los recursos multimedia ambientales. Los recursos multimedia serán transmitidos según las especificaciones de cada cliente y el flujo generado deberá ser configurado bajo protocolos en tiempo real.
- Reproducción en tiempo real de los recursos multimedia ambientales en dispositivos móviles. Se desarrollarán clientes ubicuos que permitan recibir y presentar los flujos de vídeo y audio en computadores de mano.
- Generación de estos flujos desde la cámara y el micrófono de los computadores de mano. Se integrarán los recursos multimedia móviles, cámaras y micrófonos, bajo las mismas características que los recursos multimedia ambientales.
- Manipulación y consulta de Mapas Semánticos desde dispositivos móviles. Los mapas del entorno serán visualizados desde los dispositivos móviles de los usuarios, sin necesidad de una conexión permanente con el sistema central.
    - Cálculo de Rutas en Mapas Semánticos. La ruta entre dos puntos del mapa será resuelta desde los propios dispositivos usando una heurística ad hoc espacios interiores.
- Interacción con el Entorno Inteligente mediante Mapas Semánticos.
    - Localización de objetos en tiempo real. La localización de elementos móviles del entorno será presentada en los Mapas Semánticos, permitiendo al operador humano rectificar y ampliar los datos desde el propio dispositivo móvil.
    - Notificación de eventos en tiempo real. Los eventos analizados por el Entorno Inteligente se mostrarán en los Mapas Semánticos, describiendo de forma visual las zonas de impacto, una ruta recomendada para actuar y otra información de interés.

La construcción de un Sistema de estas características, implica tomar decisiones, entre otras cuestiones, sobre la plataforma y el lenguaje de programación elegidos para los dispositivos móviles. Esta elección no es trivial, ya que influirá notablemente en los problemas surgidos durante el desarrollo del trabajo. Esta problemática es descrita en la sección siguiente 2.3.

Además, no sólo es necesario estudiar cuestiones técnicas relacionadas con la movilidad, también debemos estudiar los formatos y protocolos incluidos en los componentes de integración de las fuentes multimedia del edificio porque, como detallaremos a continuación, determinarán la eficiencia y comportamiento de las transmisiones en tiempo real.

Por otra parte, debemos establecer unos Servicios Remotos para comunicar los sistemas móviles y los servidores centrales de monitorización. Las soluciones más básicas respecto comunicación, pasan por construir y manejar las conexiones por nosotros mismos, y posteriormente establecer algún formato para codificar la solicitud y recepción de datos. Sin embargo, existen herramientas capaces de abstraer esta funcionalidad, permitiendo, incluso, la comunicación con independencia de la plataforma o el protocolo con el que se solicitan los datos. En Computación Ubicua e Inteligencia Ambiental tienen un valor añadido porque permite comunicar el conglomerado de dispositivos móviles y del entorno. Esta tecnología es conocida como Middleware y será estudiada en la sección 2.7.

También podemos destacar una serie de **requisitos no funcionales** que de alcanzarse parcial o totalmente, dotarían al Sistema de una potencialidad extra. Por ejemplo, como ya hemos comentado en la introducción, sería interesante:

- Garantizar la *portabilidad a un sector importante del mercado de los dispositivos móviles*, o que al menos, la solución propuesta pueda ser adaptada a otros Sistemas Operativos, o en el peor de los casos, *los componentes desarrollados puedan ser trasladados fácilmente a otras plataformas* de programación. Para ello, desarrollaremos las as aplicaciones móviles bajo Máquinas Vir-

tuales. Como veremos en los capítulos siguientes, las Máquinas Virtuales pueden presentar limitaciones y restricciones para nuestros requisitos, por lo que, en algunos casos, deberemos construir nosotros mismos los componentes software, y no delegar funcionalidades en terceras aplicaciones que sean independientes de nuestras aplicaciones.

- También es interesante que las plataformas y lenguajes de desarrollo no se encuentren limitadas por *licencias económicas* o restrinjan su uso, por lo que la *liberación de código* y/o su gratuidad serán valorados positivamente. Finalmente, deberemos evaluar el impacto y aceptación de las tecnologías dentro de los *estándares*, ya sean por el uso extendido y generalizado de los mismos, *de facto*; o por la legitimación de los mismos a través de organismos, de *iure*.

- Además, para crear una solución adaptada a la potencialidad de cada Máquina Virtual, es conveniente desarrollar las aplicaciones para los dos grupos de dispositivos más populares: los dispositivos ligeros tales como teléfonos móviles, y los computadores de mano, como PDAs, con mayor capacidad de procesamiento.

2.3. Análisis de los lenguajes y plataformas para Dispositivos Móviles

En esta sección detallaremos las características de los lenguajes y de las plataformas que se han estudiado para el desarrollo de la tesis. Gracias a este análisis, hemos podido construir las aplicaciones que cumplan los requerimientos del Sistema y resolver los problemas y limitaciones de la programación en dispositivos móviles.

Durante los años 80 y 90 los lenguajes de programación sufrieron una profunda transformación. La aparición de la Programación Orientada a Objetos y el auge de C y Java, en orden temporal, acapararon gran parte del sector haciendo del paradigma una referencia imprescindible en Ciencias de la Computación. Actualmente, otros lenguajes

como Phyton o Rubby ocupan un sector importante en las fases de implementación, ya que el esquema común de los lenguajes Orientados a Objetos nos permite aprender con mayor facilidad otros lenguajes que compartan este paradigma. La Orientación a Objetos ha sido ampliamente aceptada en la programación porque traslada la lógica de los objetos del mundo real dentro de la computación y establece unos esquemas intuitivos sobre cómo debe estructurarse la información dentro de la computadora.

Con toda seguridad, la mayoría de los desarrolladores estarían de acuerdo en que dos de lenguajes más influyentes de la programación son Java y C. Sin embargo, entre ambos existen importantes diferencias que vamos a comentar a continuación. Los desarrollos de estas tesis han sido posibles gracias a las ventajas que ofrecen cada uno de ellos.

El lenguaje C para implementación móvil. En primer lugar, describiremos brevemente el lenguaje C y destacaremos las aportaciones al lenguaje de otras versiones más recientes que se realizaron sobre el mismo: C++, C# y Objective-C. En 1972 el físico estadounidense Dennis M. Ritchie presenta el lenguaje de programación C. Éste fue creado como un lenguaje para desarrollo basado en la potencialidad de implementación sobre Sistemas Operativos. Por ello, C fue pensado para producir un código compilado que estuviese adaptado a una máquina concreta. Una de las ventajas de este tipo de compilación es la eficiencia, ya que se crean aplicaciones que están optimizadas específicamente bajo código ejecutable.

Por otra parte, de forma muy resumida, C++ surgió ocho años después, en 1980, para extender la funcionalidad de C e incorporar los mecanismos del paradigma de la Orientación a Objetos. Tanto C como C++ permiten la creación de aplicaciones para dispositivos móviles y plataformas sobre computadores de escritorio o servidores, siendo indiscutible su uso dentro de los Sistemas Operativos. Una de sus virtudes de este lenguaje, radica en la rapidez, ya que no necesita una Máquina Virtual para ser ejecutado. Sin embargo, esta característica lo ha alejado del desarrollo de aplicaciones genéricas

móviles, donde las diferencias entre los Sistemas Operativos móviles son muy heterogéneas. Un programa en C o C++ no puede ser ejecutado en otros dispositivos, no sólo en el sentido de que deba ser compilado para otros procesadores, sino que muchas de las librerías, por ejemplo referentes a interfaz gráfica o conexiones de red, son específicas de un Sistema Operativo y no pueden ser trasladados a otro. Esta característica hace muy difícil ,en la práctica, la portabilidad de una aplicación a otro dispositivo empotrado.

Para paliar estos inconvenientes, y para incorporar las ventajas de Java (que a continuación detallaremos), en 2001 Microsoft lanzó C#. Esta plataforma se sustenta sobre una Máquina Virtual y ofrece una programación más sencilla que C++, con lo que se esperaba atrajese a los desarrolladores. Además, Microsoft apostó por crear una versión más reducida, *C# Compact*, destinada al desarrollo de móviles que dispusiesen de su Sistema Operativo, Windows Mobile. Sin embargo, C# no se ha instaurado como se esperaba de forma mayoritaria dentro los Sistema Móviles o tradicionales. Una de las causas puede ser por las restricciones de las licencias de Microsoft, pero también ha influido el auge de Java, la liberación de Java ME recientemente (2008) y la incorporación de Android al mercado.

Otra variante de C orientada a objetos es *Objective-C*. Fue creado en 1980 por Brad Cox y se definió como un superconjunto de C. La peculiaridad del mismo radica en tener un paso de mensajes parecido a Smalltalk en lugar de las clásicas llamadas a métodos. En 1992 fue liberado bajo licencia GPL para el compilador *GCC*. Objective-C permite el desarrollo de aplicaciones para Apple y su Sistema Operativo Mac OS X, incluyendo el famoso dispositivo móvil IPhone.

La licencia para el desarrollo en IPhone podría evaluarse como contradictoria. Por un lado, el paquete de desarrollo es gratuito, aunque el coste para la difusión de aplicaciones en dispositivos reales, en la actualidad, es de 99 dolares al año, y 299 dolares para las empresas. Sin embargo, la licencia se vuelve totalmente restrictiva en la práctica, ya que se establecen restricciones sobre los lenguajes a utilizar y se impide el uso de herramientas externas diferentes a las que proporciona la compañía. Además la difusión del producto es controlada por Apple

porque las aplicaciones sólo pueden ser difundidas mediante el App Store de la compañía, aunque existen herramientas que permiten saltar este paso bajo el coste de la pérdida de garantía del dispositivo. Además, Apple debe aprobar la aplicación revisando su uso y también se reserva el derecho de eliminar aplicaciones de App Store en función de unos parámetros subjetivos sobre el concepto de lo que es *desagradable*.

El lenguaje Java para implementación móvil. De forma más reciente a C, Java vio la luz en 1995 gracias a Sun Microsystem, y rápidamente se hizo un hueco entre los programadores. La curva de aprendizaje del mismo resultaba cómoda respecto C++ y ofrecía la ventaja de la portabilidad, gracias a la Máquina Virtual que caracterizan los lenguajes interpretados y los de código intermedio.

La mayor parte de la implementación de las aplicaciones móviles en esta tesis ha sido realizado usando Máquinas Virtuales. A continuación, describiremos brevemente qué es una Máquina Virtual y cuáles son las características más importantes de las mismas. De forma general, una Máquina Virtual representa una abstracción de un dispositivo físico concreto que es capaz de ejecutar una aplicación como si fuese una máquina real. Gracias a ello, es posible ejecutar una aplicación en un computador aunque no se encuentre compilada para el Sistema Operativo de un dispositivo concreto. El concepto de Máquina Virtual fue acuñado por IBM en 1959 para describir uno de los primeros Sistemas Operativos que existieron en la historia de la computación, el *VM/CMS*.

Gracias a la abstracción de las Máquinas Virtuales, su uso nos proporciona la importante ventaja de la *portabilidad*. Las Máquinas Virtuales se ofrecen como traductores entre el código de una aplicación y las instrucciones del procesador concreto. Si queremos portar esa misma aplicación sin ningún tipo de cambios a otro Sistema Operativo, sólo es necesario construir otra Máquina Virtual que convierta las instrucciones originales a las nativas.

Es importante destacar diferentes formas de aplicar la *virtual-*

ización. Una de ellas es realizando una interpretación completa del código en tiempo de ejecución, que agrupan los lenguajes Interpretados. Para ejecutar un programa escrito en un lenguaje completamente interpretado, el intérprete debe realizar el análisis léxico y sintáctico en el momento de ejecución el programa. Sin embargo, el uso de este tipo de lenguajes interpretados presentan un problema de eficiencia derivado del proceso léxico y sintáctico, que puede hacer que el tiempo sea hasta cien veces más lento que un lenguaje compilado.

Para resolver esta problemática, surgen los lenguajes de *Código Intermedio*, que realizan las tareas de análisis léxico y semántico previamente a la ejecución, para generar unas instrucciones más cercanas a las instrucciones máquina. La Máquina Virtual recibe dicho código y traduce las operaciones a instrucciones nativas del Sistema Operativo. Por ejemplo, en Java, estas instrucciones intermedias son conocidas como *bytecode*, que recibe su nombre porque cada operación tiene una longitud de un byte seguido de los parámetros de operación, tales como los registros o las direcciones de memoria. Además, siempre y cuando sea compatible, el código generado por el bytecode puede ser interpretado por cualquier Máquina Virtual, sea la de un computador de sobremesa o la de un teléfono móvil (véase figura 2.3). Esta filosofía puede resumirse bajo la máxima que describe a Java: “escribe una vez y ejecuta en todas partes”, [Baratz, 1996], aunque como acabamos de explicar debería citarse “compila una vez y ejecuta en todas partes”.

Curiosamente, Java tuvo sus orígenes entre dispositivos empotrados, en concreto, para programar electrodomésticos. Para potenciar la portabilidad en este sector, Sun lanzó Java 2 Micro Edition (J2ME), ahora conocido como Java ME, para permitir el desarrollo de aplicaciones en computadores de mano. La solución de Java ME ha sido ampliamente adoptada para la creación de aplicaciones móviles, porque ofrece un marco común en un enorme mosaico de Sistemas Operativos y plataformas móviles diferentes que conviven en el mercado. También ha sido productivo para el desarrollo de aplicaciones móviles porque ha proporcionado a los fabricantes la posibilidad de ofrecer aplicaciones a los usuarios de forma genérica. Cuando una compañía de teléfonos móviles incorporaba la especificación de una Máquina Virtual, todas

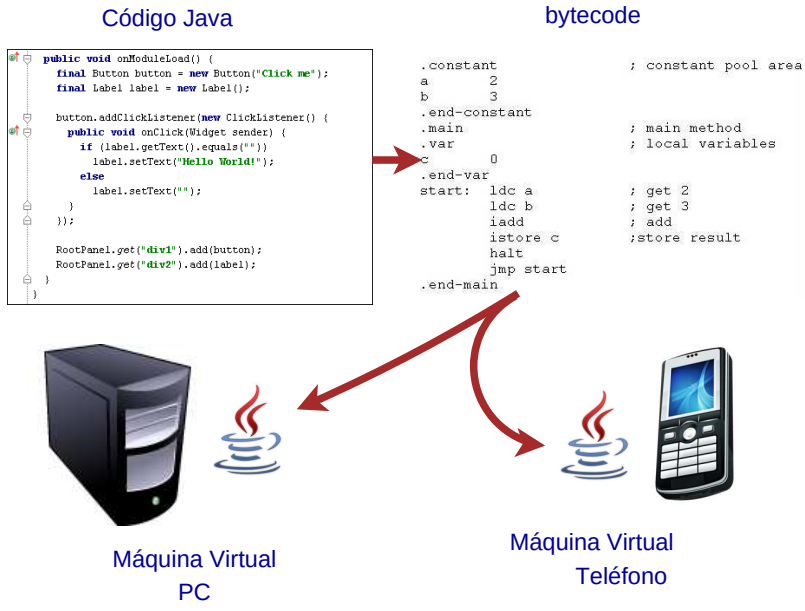

Figura 2.3: Máquina Virtual Java, bytecode y fuente

las aplicaciones creadas para la misma podían ser ejecutadas, a priori, en sus terminales.

Así, la Máquina Virtual más popular en los dispositivos móviles ha sido Java Micro Edition, que ha sido integrada en multitud de dispositivos, la más completa por IBM pero también existen versiones de otros fabricantes, como Jeode. También encontramos referencias para mejorar las prestaciones del uso de las Máquinas Virtuales en computadores de mano en la literatura [M et al., 2006]. Las características más importantes de Java Micro Edition serán detalladas en el siguiente capítulo, así como las configuraciones y perfiles que se han desarrollado para adaptar la plataforma a los dispositivo empotrados o móviles.

Otra ventaja de las Máquinas Virtuales es que permiten trabajar con dispositivos como cámaras y otros sensores en un sentido abstracto, de forma que cada Máquina Virtual resuelve las dependencias y limitaciones del Sistema Operativo sin que el desarrollador tenga

que conocer los pormenores de cada dispositivo. Sin embargo, esta abstracción ha hecho que la portabilidad real no siempre pueda estar garantizada. Cuando la abstracción de la Máquina Virtual era insuficiente para obtener todas las prestaciones de un dispositivo concreto, se determinaban nuevos requerimientos que empezaban a funcionar en unos modelos, pero en otros todavía no. Por ejemplo, la capacidad para usar y sincronizar dispositivos en Bluetooth es un requerimiento que algunas Máquinas Virtuales de Java ofrecen, pero no todas pueden garantizar. En Java ME, este tipo de requerimientos se conocen como Java Specification Requests (JSR), clasificando al Bluetooth, por ejemplo, como una especificación optativa llamada JSR-R82.

Además, siguiendo con el mismo ejemplo, aunque el dispositivo posea Bluetooth no significa que automáticamente la Máquina Virtual permita trabajar con él; para ello es necesario traducir correctamente las operaciones abstractas a unas instrucciones máquina válidas. Como veremos, este tipo de problemas ocurren con las bibliotecas multimedia, ya que en lugar de definir una reproducción a bajo nivel dentro de la Máquina Virtual, se ha definido un modelo abstracto que delega la reproducción en el Sistema Operativo y que pocos dispositivos móviles ofrecen.

Aunque como ya hemos comentado, los inconvenientes de la Máquina Virtual se derivan de una relativa lentitud con respecto los lenguajes compilados, que puede ser crítica en el contexto de los dispositivos móviles. En cualquier caso hay que destacar que las Máquinas Virtuales para dispositivos móviles se encuentran muy optimizadas. Además, muchos de los recortes de Java ME respecto a Java, por ejemplo la carencia de la reflexión y alguna potencialidad de las hebras permiten ejecutar estas aplicaciones más rápidamente en tiempo de ejecución.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL v2, incluyendo a Java ME dentro del proceso de apertura. Gracias al acceso de las fuentes de Java, la comunidad podría modificar y modificar la plataforma para adaptarla a sus necesidades. Para

fomentar el mantenimiento y portabilidad de Java ME, surgió el proyecto PhoneME [Oracle and Java, 2007], el cual mantiene las actualizaciones de código y algunas adaptaciones a procesadores ARM para los Sistemas Operativos Linux y Windows Mobile.

No sólo Java ME ha presentado una solución Java sobre Máquinas Virtuales para el desarrollo de dispositivos móviles. Otra plataforma que ha integrado esta filosofía y que debemos mencionar es Waba [Waba, 1999], una opción de desarrollo basada en un lenguaje Java más restringido, pero con unas librerías ad hoc para manipular y gestionar las entradas y salidas de los dispositivos móviles. Para optimizar el código a cada plataforma, Waba incorpora otra etapa en la generación de código intermedio. Una vez construido el bytecode típico de Java, éste es adaptado a un código intermedio que es optimizado para una plataforma concreta. A diferencia de Java, a partir de un código fuente se genera una aplicación para cada plataforma concreta; en lugar de una única aplicación que ha de ser ejecutada en diferentes plataformas.

Finalmente, la Máquina Virtual de Waba de cada dispositivo realiza el proceso de traducción del bytecode optimizado a instrucciones nativas. La filosofía de esta metodología podría pronunciarse bajo el lema “escribe una vez y compila en todas partes” porque el mismo código puede usarse para varias plataformas, pero la compilación de una no es útil para todas las plataformas.

Derivado de este proyecto, surge en SuperWaba [Corp., 2002] con la finalidad de ampliar las capacidades de comunicación e interfaces de usuario de Waba. Actualmente ha resurgido bajo el nombre de Totalcross. Aunque es poco conocida, la potencialidad de sus librerías es extraordinaria y está pensada para dispositivos móviles de mano potentes. Existen Máquinas Virtuales para Windows Mobile, Palm Os, iPhone y BlackBerry, y recientemente Android. La licencia de SuperWaba es GPL, siendo necesario hacerse con la licencia LGPL si queremos desarrollar aplicaciones privativas. En [Medina and Ruiz, 2007] se describe un completo sistema de movilización del Proceso de Enfermería bajo esta plataforma, comprobando la gran eficiencia y portabilidad de las aplicaciones móviles resultantes.

La plataforma para desarrollo de aplicaciones móviles basada en Java más reciente ha sido Android. Ésta fue presentada por Google en 2007, y está compuesta por un Sistema Operativo ad hoc dispositivos móviles basado en un núcleo Linux. A su vez, incluye un conjunto de paquetes de desarrollo propios que permiten y facilitan el desarrollo en esta plataforma. Al igual que para IPhone, las aplicaciones de Android no pueden ser ejecutadas en otros Sistemas Operativos diferentes. Android integra una Máquina Virtual adaptada a su Sistema Operativo y que se conoce con el nombre de Dalvik.

Dalvik, que fue diseñada por Dan Bornstein. En realidad, como Android esta basado en Linux, puede programarse en lenguaje nativo gracias a la herramienta de desarrollo, Android NDK para implementar librerías y aplicaciones en C++. Sin embargo, para facilitar la programación a un nivel más abstracto, la Máquina Virtual Dalvik permite escribir aplicaciones usando como lenguaje Java; al menos en sintaxis, porque esta implementación nada tiene que ver con Java ME. Incluso que el bytecode es diferente y está orientado a registros.

Además, la Máquina Virtual de Android se encuentra optimizada para ofrecer unos tiempos de ejecución menores delegando en el Sistema Operativo Android operaciones como la gestión de procesos, memoria e hilos.La mayoría del código fuente de Android ha sido publicado bajo la licencia de software Apache, una licencia de software libre y código fuente abierto.

La descarga de aplicaciones puede realizarse usando una aplicación construida para ese propósito, Android Market, al estilo del App Store de IPhone. Sin embargo, en gran parte por el uso de una licencia libre, Android se ha presentado como un producto alternativo a IPhone. Las diferencias entre las licencias es profunda, y los desarrolladores de Android no se encuentran limitados por las revisiones que la compañía haga a sus productos.

Como hemos visto en esta sección, existen varias alternativas para el desarrollo de aplicaciones en dispositivos móviles. En algunos casos encontramos el uso de lenguajes compilados, como en IPhone, y

en otros podemos hacer uso de Máquinas Virtuales, tales como Java ME y Dalvik. Las apuestas por construir aplicaciones para Sistemas Operativos concretos ofrecen una mejor adaptación a los dispositivos concretos, pero no pueden portarse a otros dispositivos.

2.3.1. Java ME

Finalmente, el desarrollo de las aplicaciones de esta tesis se realizarán bajo la plataforma Java. La elección se ha decantado por Java Standard Edition (Java SE) para la implementación de aplicaciones en computadores de sobremesa y Java Micro Edition (Java ME) para el desarrollo en dispositivos móviles. Como ya hemos visto, las principales ventajas que ofrece Java ME es su portabilidad e implantación en el mercado. En concreto:

- Está integrado en 2.100.000.000 dispositivos entre teléfonos y PDAs [Ovum, 2005].
- Ofrece una mayor portabilidad sobre dispositivos móviles (BlackBerry, Nokia, Samsung, Windows Mobile e incluso Android).
- Es código libre.
- Cuenta con una amplio repertorio de librerías y aceptación entre los programadores.
- Puede ser portado a otra plataforma, o incluso lenguaje, con menor esfuerzo que otras plataformas, ya que la especificación para las Maquinas Virtuales es más abstracta que el código nativo.

A continuación, estudiaremos a mayor detalle las características que ofrece Java ME, ya que la plataforma permite segregar varios perfiles en función de los dispositivos a los que orientemos nuestra aplicación. Los motivos de que existan diferentes configuraciones, y como veremos incluso perfiles, permite aprovechar y adaptar las Máquinas Virtuales a la heterogénea diversidad de los computadores

de mano en función de las características de cada grupo de dispositivo. En primer lugar, Java ME posee en dos configuraciones para el desarrollo de dispositivos móviles y empotrados.

- La primera configuración fue bautizada como *Java ME CLDC (Connected Limited Device Configuration)* y está orientada a teléfonos móviles. Esta configuración presenta un subconjunto de Java mínimo. Para ejecutar aplicaciones de esta configuración, es necesaria una Máquina Virtual mínima, llamada en este caso Kilo Virtual Machine (KVM). KVM puede ser ejecutada en un procesador con al menos 16 bit/16 MHz, ocupando unos discretos 160-512 KB de la memoria para la plataforma Java.

  Java ME CLDC ha llegado a ser muy popular porque permite el desarrollo para teléfonos ligeros y muchos fabricantes incluyen la Máquina Virtual en sus dispositivos. Para ampliar las capacidades básicas de esta configuración, existen varias especificaciones que permiten, por ejemplo, manejar la conectividad Bluetooth o acceder a la cámara del dispositivo móvil. Sin embargo, estas especificaciones, conocidas como Java Specification Requests (JSRs), son incorporadas en función del fabricante y de la potencia del dispositivo. En muchos casos, las librerías usan llamadas internas al teléfono, que no pueden ser accedidas desde Java ME.

  Las especificaciones extra representan los únicos mecanismos para ampliar la capacidades de Java ME CLDC. Por tanto, la Máquina Virtual KVM es cerrada y no puede modificarse de forma arbitraria. Debido a esta limitación, la portabilidad de las aplicaciones dependerá de la calidad de la KVM que ha implementado el fabricante. En algunos casos, podremos construir una solución usando código 100 % Java ME y que no dependa de ninguna especificación extra; en otras, no tendremos más remedio que depender de la Máquina Virtual de cada dispositivo.

  Las limitaciones de Java ME CLDC repercuten en una restricción sobre los usos de Java a los que estamos acostumbrados. El caso más drástico es la ausencia de punto flotante, pero ha

sido resuelta en la versión 1.1 respecto a la 1.0, debido a las limitaciones computacionales que derivaba. Como ya hemos comentado, el uso de herramientas para realizar llamadas a librerías nativas, como Java Native Interface, no está incluido. Tampoco es posible contar con la reflexión: definición, creación y manipulación de objetos en tiempo de ejecución. Por ello, la llamada a interfaces remota y el envío de objetos por la red usando Remote Method Invocation queda eliminada. También el manejo de hebras es limitado, porque no es posible construir grupos o procesos conjuntos de ejecución.

Dentro de la configuración ligera de Java ME, existen varios perfiles que incluyen la especificación CLDC. Estos perfiles usan como núcleo a Java ME CLDC, incorporando las características que ya hemos comentado, pero además aportan otras funcionalidades propias de cada máquina, como por ejemplo las interfaces de usuario. A continuación, destacaremos tres de los perfiles más populares:

- *Mobile Information Device Profile (MIDP)* aporta una interfaz de usuario básica para construir aplicaciones, llamadas MIDlets, y operar con los usuarios. A su vez incluye un pequeño repositorio donde almacenar y recuperar registros en memoria (Record Management Store), pero sin las funcionalidades complejas de un sistema de ficheros complejo. Este perfil es el más popular, ya que está incluido en los teléfonos móviles de muchos fabricantes (Nokia, Samnsung, Windows Mobile, Ericsson, ...).
- *Information Module Profile (IMP)* está orientado a dispositivos empotrados que necesitan conectarse entre sí. El perfil es apropiado para el desarrollo en cajeros y máquinas de venta con una interfaz de usuario mínima.
- *DoJa* es un perfil propietario de Java ME CLDC que, aunque es poco conocido en Europa, en Japón es de uso común y que fue desarrollado dentro del propio país. Ofrece una interfaz de usuario amigable, al estilo de MIDP (véase figura 2.4).

Figura 2.4: Perfil Doja y MIDP de JavaME

- La segunda configuración *Java ME CDC (Connected Device Configuration)* fue pensada para computadores de mano con mayores posibilidades, como las agendas electrónicas. Es prácticamente un antiguo Java (1.4) junto con algunas librerías propias de CLDC referentes a la conectividad por red, Bluetooth u otros. La integración de estas librerías permite que muchos desarrollos puedan ser ejecutados indistintamente en Java ME CLDC o CDC.

  Al igual que la configuración ligera, si deseamos garantizar la portabilidad de nuestras aplicaciones, las librerías que incorporemos de Java, deben ser *cross all*, esto es, no tener dependencias con la Máquina o el Sistema Operativo y ser compatibles con Java 1.4 o 1.3.1. Sin embargo, y a diferencia de la configuración CLDC, otra forma de suplir las carencias de la máquina virtual, es creando una nueva librería nativa que pueda ser llamada desde Java ME CDC. De este propósito se encarga JNI (Java Native Interface), una interfaz de programación que incorpora la Máquina Virtual para realizar llamadas a librerías nativas desde una aplicación Java. Además, la configuración CDC solventa la mayoría de las limitaciones de CLDC incorporando por ejemplo, RMI, la reflexión o el manejo de hebras completo.

De forma análoga a la configuración CLDC, existen varios perfiles que añaden a esta configuración más funcionalidades, como por ejemplo, conectividad o interfaces de usuario.

- *Foundation Profile* incorpora el acceso a la red y al sistema de archivos del dispositivo.
- *Personal Basis Profile* es un superconjunto del anterior, que permite introducir componentes *Xlet* en los dispositivos. Los Xlet son unos componentes para aplicaciones interactivas de televisión digital, que pueden ser instalados y configurados remotamente en un computador empotrado, gracias a la integración de métodos remotos y de la reflexión dentro de la Máquina Virtual.
- *Personal Profile* es otro superconjunto de *Foundation Profile* que incluye, parcialmente, la interfaz de usuario Abstract Windowing Toolkit (AWT). Esta interfaz gráfica de usuario presenta una potencialidad importante sin hacer un uso excesivo de los recursos. Además, se encuentra integrada dentro del Java para escritorios, Java SE. A parte de la interfaz de usuario, *Personal Profile* permite conectarse con otros motores de Bases de Datos usando los adaptadores Java Database Connectivity (JDBC). Además da soporte para los populares Applet de Java en los navegadores web o permite incorporar componentes de desarrollo como Beans.

Por último debemos indicar que, por lo general, en las agendas electrónicas donde podemos usar la configuración CDC, es posible a su vez, encontrar Máquinas Virtuales para CLDC. Esto hace que Java ME CDC sea más restrictivo en cuanto a portabilidad, pudiendo encontrarse en Windows Mobile o algunos Linux para PDAs; mientras que la versión CLDC podría ser ejecutada en teléfonos móviles, agendas u otros computadores, para los que existan Máquinas Virtuales para esos dispositivos.

Además de las configuraciones que acabamos de detallar, la familia de Java abarca otra tecnología que aunque alejada de la tesis, conviene mencionar: Java Card. Ésta tecnología nos permite construir

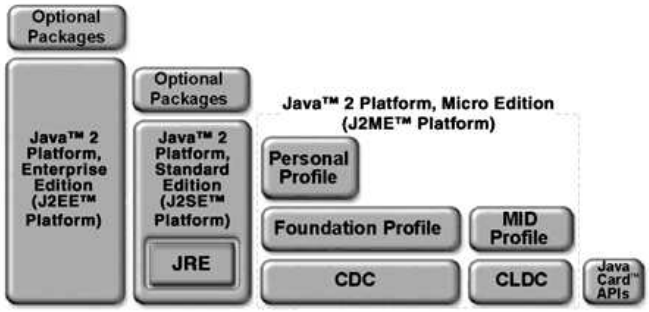

Figura 2.5: Plataformas Java

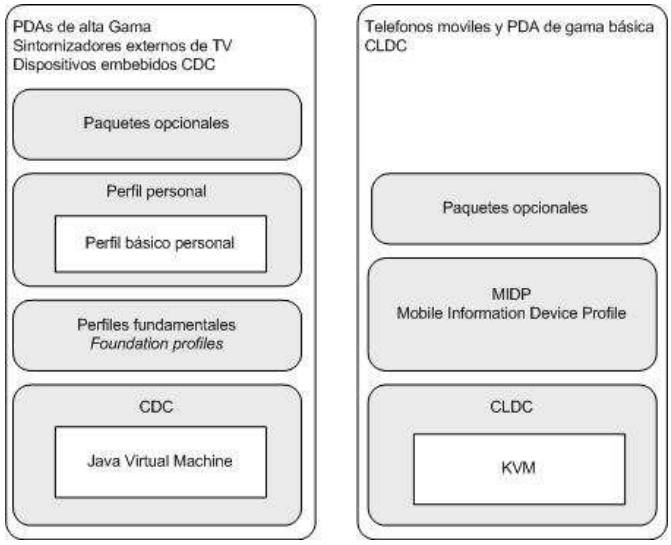

Figura 2.6: Configuración y perfiles de Java ME de la propuesta

un pequeño Applet muy básico que puede ser ejecutado en el chip de una tarjeta inteligente. El desarrollo de Java Card es tan restrictivo que no dispone, por ejemplo, de recolector de basura o no permite construir matrices de más una dimensión.

A modo de resumen, en las figuras 2.5 y 2.6 se detallan las familias y relaciones de cada uno de los perfiles, configuraciones y plataformas de Java y Java ME.

Perfiles y Máquinas Virtuales del desarrollo. Los trabajos realizados en esta tesis se han desarrollado para las dos configuraciones de Java ME: CLDC para dispositivos ligeros y CDC en computadores de mano. De forma más concreta, se ha escogido el perfil MIDP dentro de Java ME CLDC, porque es el más implantado dentro de los abundantes teléfonos móviles y proporciona una interfaz gráfica de usuario sencilla. La versión de la configuración y del perfil, respectivamente, son CLDC 1.1 y perfil MIDP 2.0.

Ya hemos comentado que el uso de la configuración CLDC viene sustentada por una Máquina Virtual limitada llamada Kilo Virtual Machine. Cada fabricante de dispositivos móviles que esté interesado en integrar la tecnología Java ME MIDP, es el encargado de dar soporte de la misma a su producto; por lo que cada compañía y modelo suelen presentar una Máquina Virtual con las mínimas garantías de ejecución. El resto de potencialidad puede variar y está estructurada en paquetes o librerías optativos llamados Java Specification Requests (JSRs). Por ejemplo, la capacidad para manejar Bluetooth en el teléfono es conocida como JSR-82 y no cualquier dispositivo la incluye.

Las Máquinas Virtuales, por tanto, dependerán de la especificaciones de cada fabricante. Para testar los desarrollos de estas tesis, hemos escogido un teléfono móvil Nokia. La elección de este fabricante es muy recomendable porque, como se introdujo en 2.1, siempre ha estado muy ligada al desarrollo de Java ME e incorpora Máquinas Virtuales con grandes garantías de funcionamiento. La elección del modelo es un dispositivo de gama media-alta, en concreto, un Nokia N81 a 369 MHz, que dispone de WiFi y nos permite conectarnos por red a otros computadores de sobremesa o móviles.

Respecto al desarrollo para los computadores de mano o agendas electrónicas más potentes, se ha usado la configuración Java ME CDC. El perfil más extendido para la construcción de aplicaciones para usuarios no especializados, es el de *Personal Profile* que incluye la interfaz de usuario AWT. Aunque esta configuración es muy potente, desafortunadamente no existen muchas Máquinas Virtuales que la soporten. La versión de desarrollo de *Personal Profile* es la 1.1. Los dispositivos más comunes que incorporan este perfil son: una gama de

móviles Nokia (N800/N900), Linux empotrado y Windows Mobile.

Los primeros, la serie N de Nokia, la incluyen íntegramente, de forma que es la propia compañía quien ofrece la compatibilidad. Por otra parte, algunas de las pocas agendas electrónicas con Linux incluyen una Máquina Virtual para Java. Por ejemplo, el modelo Sharp Zaurus incluye el perfil Personal Profile gracias a la incorporación de una Máquina Virtual que proporciona la plataforma Jeode. Poro otra parte, los dispositivos con Windows Mobile, suelen disponer de una pequeña KVM para MIDP, pero no traen de serie la potencia de la configuración CDC. Son otras compañías las que se han encargado de dar este soporte. La más conocida es J9 desarrollada por IBM que incluye el perfil Personal Profile en la versión 1.1 para dispositivos Windows Mobile. El uso de J9 con fines comerciales tiene un coste económico, por dispositivo, de 6 dolares.

Como hemos detallado en 2.3, gracias a la liberación de Java ME, se creó el proyecto PhoneME, [Oracle and Java, 2007], para difundir y adaptar el código de fuente de la Máquina Virtual a otros computadores de mano. Fruto del mismo, se han desarrollado varias Máquinas Virtuales Java en sus diferentes perfiles para los Sistemas Operativos de Linux y Windows Mobile.

Los desarrollos realizados para computadores de mano han sido realizados para Windows Mobile usando PhoneME. Como veremos más adelante, en algunos casos se han encontrado muchas limitaciones cuando se ha trabajado con esta Máquina Virtual, pero todos ellos han sido solventados desarrollando librerías nativa para el Sistema Operativo de Windows Mobile. En esta situación, parte del cómputo es ejecutado nativamente en el dispositivo y devuelto a la Máquina Virtual.

Estas librerías que amplían la potencialidad de las Máquinas Virtuales han sido implementadas en C++, y se han compilado para su ejecución en Windows Mobile. Una ejecución dividida, entre Java y C++, debe ser cuidadosamente resuelta, más aún en procesos concurrentes tales como la reproducción de flujos multimedia en tiempo real.

A modo de resumen, los desarrollos de la tesis se van realizar de forma mayoritaria bajo la plataforma **Java ME**, en sus dos configuraciones CLDC y CDC. Puntualmente, se resolverán algunas limitaciones de la segunda configuración con implementaciones nativas en **C++** para el Sistema Operativo de Windows Mobile.

2.4. Dispositivos Multimedia Ambientales

Uno de los puntos más importantes de nuestra propuesta es el acceso multimedia a un espacio interior o exterior usando un dispositivo móvil. Nuestra herramienta permitirá *ver* y *oír* cualquier rincón de un edificio, o del mundo, que esté conectado al Sistema. En este sentido, podemos definir este proceso como Percepción Virtual Movilizada, donde la realidad del usuario es trasladada a otro lugar pero sin necesidad de usar cascos, guantes u otros artificios (no inmersiva). Las aplicaciones prácticas del Sistema son innumerables: seguridad, control del hogar, publicidad, turismo virtual, etc.

En esta sección vamos a describir el tipo de cámaras más comunes para una transmisión de vídeo. En primer lugar, debemos mencionar los sistemas de cámaras más clásicos, que fueron denominados Circuitos Cerrados de Televisión (CCTV). Esta tecnología nos permite transmitir las imágenes capturadas por una cámara a una línea de televisión, de forma que varios televisores pueden recibir y mostrar por la pantalla el vídeo capturado.

De forma reciente, gracias a la expansión de las redes en Internet, las empresas de vídeo vigilancia han visto que el complemento de ambas tecnologías permitía construir sistemas altamente escalables y fácilmente configurables. Bajo estas características surgen las *Cámaras IP*, que incluyen un microprocesador propio que las monitoriza y que transmite un flujo de vídeo a la red a través de un adaptador de red ethernet o incluso inalámbrico.

Gracias a esta conectividad con la red, este flujo de vídeo puede ser visualizado, por ejemplo, usando un ordenador. Actualmente existen multitud de dispositivos de vídeo derivados de éstos en función

de sus características. Algunas de ellas incorporan, a su vez, un micrófono que captura el sonido ambiente que envuelve a la cámara.

En el ámbito doméstico, también han tenido una gran acogida las populares cámara web, que permiten recoger imágenes y realizar videoconferencias mediante un ordenador o portátil, por lo que muchas de ellas incorporan un micrófono que facilita la transmisión simultánea de audio y vídeo. Las cámaras web suelen ser usadas para mostrar a la personas que se encuentran detrás de un computador, pero también pueden ser usadas para controlar el entorno de forma remota. En general, las cámaras web se conectan a los computadores mediante un USB. Como anécdota, la primera cámara web no fue diseñada para describir a la gente, sino para conocer el estado de una cafetera en el trabajo, y quiénes eran las personas que no se molestaban en rellenarla.

Recientemente, otros dispositivos modernos que incorporan cámaras son los robots. Entre los robots domésticos vamos a destacar comercialmente a Aibo, porque introdujo al mercado un potente dispositivo con importantes comportamientos inteligentes a un precio relativamente asequible. Así, Aibo fue presentado por Sony en 1999 como un robot mascota con forma de perro que muestra el resultado de la combinación de la Inteligencia Artificial y componentes físicos avanzados. Además de un amplio número de sensores y actuadores, para los propósitos de captura de imágenes, cuenta con una sofisticada cámara de vídeo CMOS de 350.000 píxeles.

Esta es sólo una muestra del tipo de dispositivos multimedia que existen en el mercado. Como podemos observar, existen dispositivos muy dispares en características y conectividad. Para desarrollar uno de los objetivos de la tesis: la visualización de estos recursos desde dispositivos móviles en tiempo real, podríamos pensar en ofrecer una visualización específica para cada tipo de cámara o micrófono. Sin embargo, esta solución sería poco recomendable, ya que cada vez que incorporásemos un nuevo dispositivo, tendríamos que crear un reproductor diferente para esa cámara concreta.

Por establecer una solución escalable, la opción que se propone es

diseñar un Servidor Multimedia cuyas entradas fuesen los diferentes dispositivos multimedia y la salida produjese un **flujo homogéneo** en tiempo real y que fuera independiente del recursos hardware que lo genera. Para ello, es necesario encapsular la información multimedia y obtener un canal en **tiempo real** que transmita y codifique los datos con formatos de vídeo o de audio de forma apropiada.

En los objetivos de la tesis no sólo se ha establecido la generación flujos multimedia desde computadoras de escritorio y fuentes estáticas, además pretendemos incorporar a los dispositivos móviles como fuentes multimedia. Bajo este esquema, otorgamos transparencia no sólo sobre los recursos hardware que los generan, sino también sobre el tipo de computadores que los monitorizan, ya sean de escritorio o móviles. La transparencia de transmisión se obtiene estableciendo los mismos protocolos y formatos para el Servidor Multimedia y para la generación de flujos desde los dispositivos móviles. Así, los receptores reproducirán con total similitud los flujos multimedia de unos y otros de forma transparente.

A continuación vamos a presentar los formatos y protocolos que se han estudiado para la propuesta de la tesis.

2.5. Protocolos en Tiempo Real

En este apartado vamos a centrarnos en la temática que caracteriza a las transmisiones multimedia en tiempo real, y explicaremos el uso de formatos y protocolos apropiados para establecer una correcta emisión y recepción de los mismos.

Como adelantamos en la introducción, el tipo de transmisiones deben de ser rápidas, ya que resulta crítico minimizar los retrasos en el envío y en la recepción de datos. Para ello, veremos cómo es necesario sacrificar la verificación o pérdida de paquetes y cómo el flujo continuo debe recomponerse ante estas situaciones. Posteriormente presentaremos los formatos de codificación de vídeo y audio con buenas características de decodificación para las Máquinas Virtuales de los dispositivos móviles.

En primer lugar, describiremos los protocolos apropiados para enviar vídeo y audio de forma estándar y eficiente. Entre ellos presentamos los dos principales protocolos sobre los que soportamos la transmisión por la red en tiempo real: UDP [Postel, 1980] y RTP [Schulzrinne et al., 2003].

UDP (User Datagram Protocol) es un protocolo a nivel de transporte de Internet que permite enviar mensajes sin que exista una conexión previa entre los participantes. Además, no ofrece confirmación de control, flujo o recepción, por lo que los paquetes pueden perderse o desordenarse sin que sean conscientes los emisores y receptores. Aunque parezca contradictorio, son justo estas características, que podrían evaluarse como limitaciones, las que lo sitúan como el protocolo a nivel de transporte más apropiado para las transmisiones en tiempo real. Como el flujo multimedia se transmite en continuo y bajo una frecuencia de transmisión alta, no es conveniente esperar a que cada uno de los paquetes sea verificado y rectificado en caso de error.

Esto provocaría un retraso o desfase de la señal que es inconcebible para las exigencias del tiempo real. Por otra parte, si valoramos la pérdida de algún posible paquete, vemos que no es determinante porque sabemos que el flujo multimedia no se ve drásticamente afectado debido al elevado y frecuente volumen de la transmisión. Sin embargo, el formato encargado de decodificar el flujo debe ser robusto a la pérdida de los mismos y saber reestructurar la secuencia de vídeo o audio.

Para solventar algunas de las limitaciones, suele situarse sobre UDP al protocolo RTP. RTP (Real-time Transport Protocol) es un protocolo de nivel de sesión que añade una cabecera con información extra de los paquetes que se envían. Fue diseñado en 1996 y es ampliamente conocido por su uso en las transmisiones de vídeo y audio en tiempo real. Como acabamos de comentar, suele situarse sobre protocolos ligeros, el más común UDP, porque en este tipo de transmisiones los retardos pueden resultar críticos. Gracias a la combinación de ambos, puede corregirse el orden o establecerse un manejador de tiempo real en base a la marca de tiempo, sin perder la velocidad caracterizada por UDP. Sin embargo, no es excluyente de esta

combinación y puede complementar protocolos orientados a conexión como TCP. La cabecera de RTP incluye, entre otros:

- Un número de secuencia que representa la posición del paquete respecto los anteriores. Gracias a ello, podemos ordenar posibles desordenes producidos por UDP.
- Una marca de tiempo que indica el tiempo en el que tomó la muestra.
- El tipo de formato es un número que establece el formato en que son codificados los datos.
- El tamaño de los datos.
- Una bandera que indica si ha sido necesario rellenar con ceros el final de los datos porque sobrase espacio.
- Un identificador de la fuente para diferenciar las fuentes que se transmiten concurrentemente.

Como los paquetes RTP son enviados desde un emisor hacia los clientes sobre protocolos no orientados a conexión, al emisor le es imposible saber si los paquetes han sido procesado correctamente, se han perdido o sobrepasan la capacidad de recepción de los receptores. Para solventar esta limitación, el protocolo RTP suele estar complementado con otro canal que envía las estadísticas de la transmisión bajo el conocido RTP Control Protocol (RTCP). RTCP se encarga de proporcionar el estado de la Calidad del Servicio, Quality of Service (QoS); así como informar de otras características de la fuente que no son proporcionadas por RTP. Los diferentes tipos de mensajes son:

- *Informe de Emisión* informa periódicamente de los paquetes perdidos desde el último informe entre el receptor y el emisor, así como la marca de tiempo del último paquete recibido.

- *Informe de Recepción* es similar al anterior, pero es usado cuando existen participantes que sólo reciben información, por lo que el informe sólo cuenta con información de la recepción.

- *Descripción de una nuevo participante* permite dar a conocer el nombre del emisor que transmite una fuente determinada. Es arbitrario, pero puede usarse por ejemplo, el nombre relacionado con la dirección IP.

- *Finalización de un participante* notifica que un emisor ha dejado de transmitir los datos de una fuente determinada. Hemos de tener en cuenta que este mensaje no es trivial, ya que si el receptor deja de recibir paquetes, desconoce si es por el cierre, por una pausa o por una pérdida temporal de la transmisión.

- *Mensajes de aplicación* pueden incluirse otros mensajes que sirvan a los propósitos concretos de una aplicación.

El protocolo RTCP es totalmente opcional, no siendo imprescindible para el uso de RTP. Los casos en los que es apropiado, es cuando los emisores tienen la potencialidad de adaptar el flujo de la fuente según las características de la recepción. Por ejemplo, en el caso de transmitir las imágenes de una cámara, si la transmisión se ve afectada por pérdidas podría reducir el número de captura por segundo; y viceversa, aumentarlas si no se pierde ningún paquete. En situaciones más complejas, incluso es posible modificar los tipos de codificación, ya que como detallaremos en la siguiente sección, afectan al tamaño y tiempo de decodificación drásticamente.

2.6. Formatos para el Tiempo Real

En general, todo proceso de transmisión en Tiempo Real tiene unas fases bien delimitadas. La primera es la recogida de los datos a transmitir, esto es, las imágenes o el sonido recogido por las fuentes. En general, estas fuentes suelen ser estáticas, por ejemplo, ficheros que residen en memoria y que van a ser enviados en tiempo real.

Sin embargo, en el contexto de nuestra propuesta las fuentes hacen referencia a datos que se generan dinámicamente describiendo un mundo en directo.

Eso significa, que a diferencia de los archivos que residen estáticamente en memoria, los datos se han de capturar y transmitir concurrentemente. Además, tampoco podemos esperar a que el flujo haya finalizado para enviarlo, si no que se construye continuamente desde que comienza la sesión. Gracias a este mecanismo, el cliente puede visualizar el contenido multimedia desde el principio, sin esperar a que finalice la captura completa del vídeo o del audio.Esta filosofía de envío se conoce como "streaming" y es ampliamente usada en las aplicaciones comerciales y en la literatura [Gregory, 2002].

El *Streaming* permite el envío de información en continuo a un cliente que es capaz de recibir y reproducir simultáneamente los datos. Una de las ventajas de esta técnica es el ahorro en memoria que se produce, ya que no es necesario completar la descarga de un archivo y situarlo en memoria para visualizarlo. Para la recepción, el cliente dispone simplemente de un vector que retiene los datos de forma temporal y que se define formalmente como *buffering*. El reproductor accede temporalmente a las muestras almacenadas y construye un pequeño fragmento de audio o vídeo, liberando posteriormente los recursos del vector ya consumidos.

Como puede observarse, el *streaming* puede verse como un problema clásico de consumidor y productor. Esto es, el receptor recoge los paquetes que son recibidos de la red y los introduce en un vector; mientras que el consumidor es el reproductor que recoge los mismos. La filosofía de esta propuesta es realizar los dos procesos en hebras separadas, de forma que ambos se ejecutan en paralelo. Es extremadamente importante que el acceso al recurso del vector se realice de forma segura. Para protegerlo, las operaciones han de realizarse en exclusión mutua, evitando que la escritura y lectura se realicen concurrentemente.

La estructura más recomendable para formalizar el acceso a los datos bajo estas restricciones es un vector circular, en el que

exista una posición de lectura y otra de escritura que avanzan progresiva y circularmente. Es crucial salvaguardar que la posición de lectura nunca adelante a la de escritura. La gran ventaja de la estructura circular es que si la escritura avanza rápidamente, esto es que se reciben muchos datos, el tamaño del vector no crece indefinidamente porque a lo sumo se sobreescribirían datos del mismo. Incluso en este caso, la pérdida no es dramática, porque correspondería a los datos más antiguos, y por lo tanto, a los que menos nos interesan. Evidentemente, en este caso se pierde información, pero si existen saturaciones es conveniente ignorar los paquetes antiguos y recomponer la reproducción.

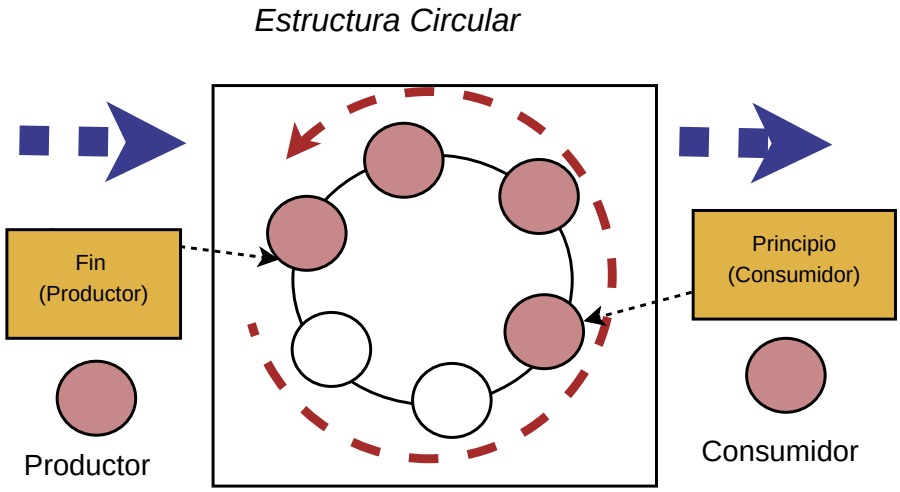

Figura 2.7: Estructura circular para la recepción en tiempo real

Existen dispositivos en los que la información de las fuentes puede estar codificada y comprimida siguiendo algún formato determinado, como un vídeo en formato AVI o sonido en MP3. En esta circunstancia habría que efectuar primero un proceso de descompresión y, posteriormente, transmitir los datos de forma apropiada a las especificaciones del tiempo real; esto es, volviéndolos a comprimir para la retransmisión usando una codificación compatible con RTP. Aunque parezca innecesario que en muchas ocasiones no podamos transmitir los datos tal cual son recogidos, hay que tener en cuenta que bajo el protocolo RTP los paquetes son susceptibles de perderse, sin que el cliente

que los visualiza sea consciente de ello. Esto implica que la forma de compresión tiene que ser compatible con estas circunstancias de envío en RTP y pueda recuperarse ante susceptibles pérdidas de datos.

Como hemos visto en la sección 2.4, en el caso de nuestra propuesta no es necesario decodificar el vídeo, ya que no está codificado previamente, sino que partiremos de la base de captura de imágenes para desarrollar nosotros el vídeo completo. En concreto, trabajaremos con la captura de una ráfaga de imágenes JPG producidas por la cámara. El audio por su parte, tampoco viene codificado, ya que es recogido directamente desde el micrófono como una secuencia de bytes secuenciales, esto es, en formato lineal.

Estos datos multimedia de la fuente original son codificados posteriormente para generar un formato de vídeo o audio que ocupe menos tamaño y reduzca la carga en la red. A su vez, serán transmitidos sobre los protocolos detallados anteriormente: UDP, situado en la capa de transporte y RTP, a nivel de sesión. Cuando los datos llegan al dispositivo encargado de la visualización, son decodificados y visualizados en la pantalla o reproducidos en el micrófono.

También es determinante el estudio de la complejidad y latencia de los algoritmos de compresión/descompresión porque el cuello de botella se estrecha en la reproducción del flujo en el dispositivo móvil. En la práctica y ante esta arquitectura móvil, los parámetros a controlar son el número de muestras por segundo, la frecuencia, la calidad y el códec de comprensión. Evidentemente la latencia, a su vez, dependerá de la cobertura de la red inalámbrica y el procesador del dispositivo. Por ejemplo, si obtenemos una visualización lenta de la cámara y tenemos una potente red inalámbrica podemos usar un algoritmo de codificación que comprima menos los datos pero que viaje más rápido, a costa de aprovechar el mayor ancho de banda de la red. En definitiva, estos parámetros deben ser ajustados a cada caso hasta obtener los mejores resultados que nos posibiliten los recursos.

El formato de vídeo que proponemos es Motion- JPEG Compressed Video (M-JPEG) [Berc et al., 1996]. Esta ligera codificación

ofrece una baja latencia en la fase de codificación y decodificación porque no comprime los frames en tiempo. Gracias a ello, no es necesario esperar a que varias capturas estén disponibles y se reducen los estrictos tiempos de espera. Además M-JPEG ha sido diseñado específicamente en base a la fragmentación del protocolo RTP. En cada datagrama, la cabecera de M-JPEG contiene información sobre el tamaño de la imagen o numeración, así como el desplazamiento que representa el paquete respecto el frame (véase la figura 2.8). Este desplazamiento es necesario porque los frames suelen ser típicamente mayores al tamaño máximo del datagrama.

Otra gran ventaja es que debido a su simplicidad, puede ser incluido en las máquinas virtuales de los dispositivos móviles de forma implícita en el código, haciendo que la portabilidad de la aplicación esté completamente garantizada. La única desventaja que ofrece M-JPEG es el alto gasto de información y ancho de banda que produce la no comprensión en tiempo. Otros formatos más potentes, como H.261 o H.263, sí que reducen el tamaño del vídeo, aunque su complejidad hacen que sean poco recomendables para ejecutarse en las limitadas Máquinas Virtuales de los dispositivos móviles.

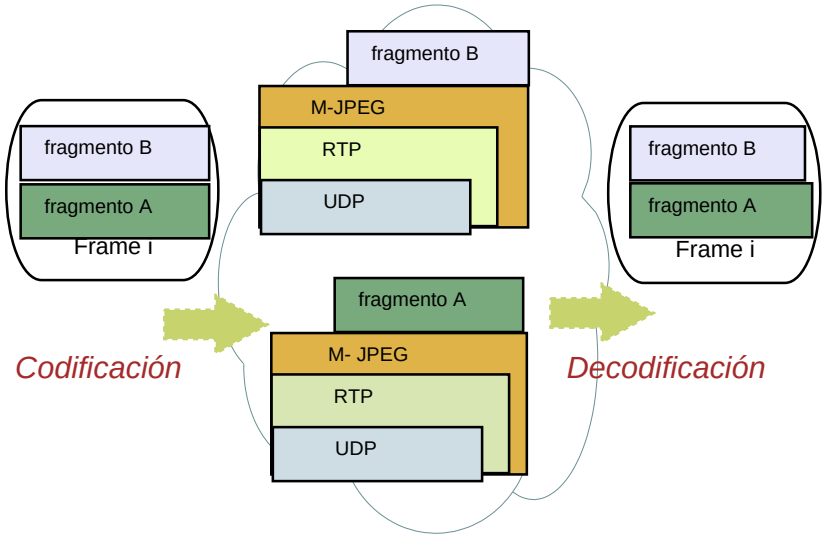

Figura 2.8: Códec de Vídeo para el Tiempo Real

Por otra parte, en el caso de la codificación del audio, hemos estudiado los dos algoritmos clásicos para la compresión de audio que se encuentran disponibles en el estándar G.711 [Zopf, 2002]: Mu-law y A-law. Este estándar ofrece un formato de compresión logarítmico que comprime una señal de audio capturada a 8000 muestras por segundo. Finalmente, hemos optado por elegir Mu-law porque ofrece una ligera mejor compresión de los datos de audio (véase la comparativa de la figura 2.9). La adaptación del estándar para el protocolo RTP es sencilla. En cada datagrama se incluye un fragmento con la codificación de una muestra de sonido, que es independiente de otros fragmentos. Esto permite extraer el sonido lineal de cada paquete, haciendo que la posible pérdida de sus paquetes vecinos no afecte al menos a aquellos que se han recuperado correctamente (véase la figura 2.10).

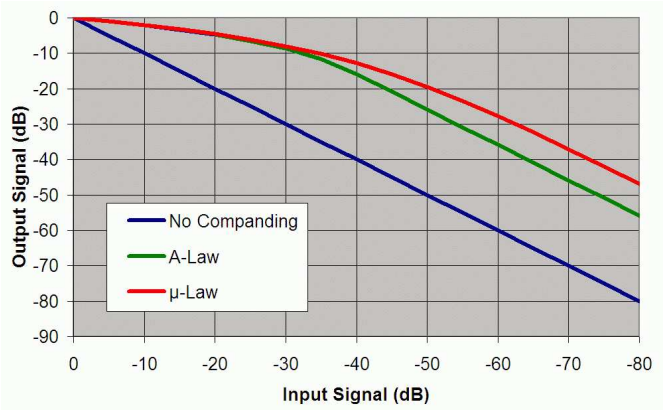

Figura 2.9: Comparativa Mu-law y A-law

2.7. Middleware

En esta sección vamos a hacer referencia un conjunto de herramientas que permiten la comunicación de componentes por red. Este tipo de herramientas pretenden facilitar la integración de aplicaciones distribuidas y que se han convertido en una etapa común en los desarrollos de muchos proyectos. El intercambio de datos por red entre aplicaciones empezaba a resultar muy costoso, porque, por

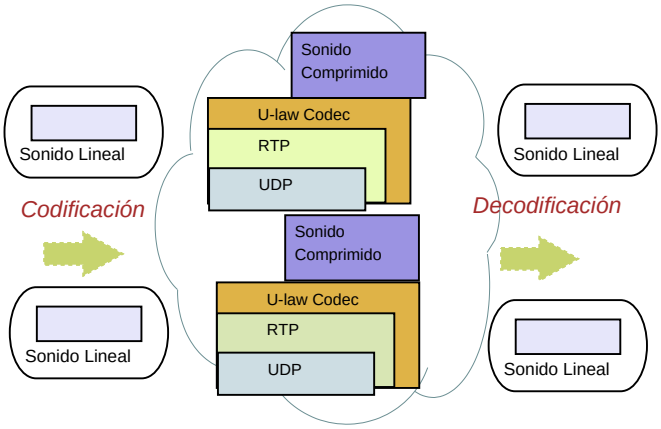

Figura 2.10: Códec de Audio para el Tiempo Real

ejemplo, había que manejar y construir conexiones bajo un protocolo de red, y establecer un mecanismo para codificar y decodificar los datos por la red para cada una de las plataformas implicadas en el desarrollo. Además, estos mecanismos había que replicarlos bajo otros parámetros en otros sistemas. Ante esta problemática, surge la necesidad de crear herramientas que faciliten la comunicación entre componentes distribuidos.

En base a esta demanda, en estos últimos años se ha investigado de forma extensa sobre las comunicaciones distribuidas, desarrollando conceptos como *Servicios Web*, *Remote Call Procedure*, *Service-Oriented Architecture* o *Publicación y Subscripción* dentro de la definición de Middleware, [Orfali, 2002]. Este concepto es muy genérico y varía en función de las demandas de conectividad que se requieran, por lo que, como veremos a continuación, algunos Middleware ofrecen unas características diferentes a otros.

En general y bajo nuestro punto de vista, un Middleware debe permitir la comunicación entre aplicaciones con independencia del Sistema Operativo, protocolo de red o lenguaje de programación. Sin embargo, la complejidad de los sistemas hace que, en general, no todos estos requerimientos se cumplan. Por otra parte, gracias a la influencia de la filosofía de la Orientación a Objetos, algunos Middlewares facilitan la comunicación mediante la publicación de

Objetos y Servicios Remotos en la red. Esta característica también es interesante para nuestra propuesta porque el desarrollo de nuestras aplicaciones se realizan en Java, y una filosofía Orientada a Objetos permitiría integrar un Middleware de estas características con mayor facilidad.

Sobre los conocidos Servicios Web, debemos decir que más que ofrecer las características completas de un Middleware, permiten un paso de comunicación estándar entre plataformas, pero no abarcan otras características importantes como las referencias remotas o la transparencia de protocolos. Por ejemplo, el formato para codificación es XML, el cual está pensado para ser entendido por un operador humano más que para la codificación eficiente de un objeto. Además, suelen estar muy ligados a un protocolo concreto, como HTTP o HTTPS, y sólo trabajos experimentales [Gehlen et al., 2006] ofrecen alternativas.

Actualmente, casi todos los lenguajes de nueva generación han hecho un esfuerzo por incluir estos conceptos en sus plataformas. En concreto, Java integra la tecnología *Java Remote Method Invocation* para publicar e invocar a métodos de forma remota, y Microsoft incluye *.Net Remoting* con los mismos fines. En general, ambos ofrecen una abstracción sobre la solicitud y acceso a métodos; la publicación y paso por parámetro de objetos; o la transparencia sobre protocolos de red. Sin embargo, el punto más excluyente radica en que éstos Middlewares sólo permiten la comunicación de sus plataformas entre sí. Eso significa que los desarrolladores de un lenguaje diferente no pueden no publicar o llamar estos Servicios Remotos; o, lo que es lo mismo, no son multi-plataforma, ya sea por una limitación intrínseca de RMI o Net. Remoting, o bien porque no les interese comercialmente que otras plataformas puedan comunicarse con ellos.

Sin embargo, otros Middlewares, sí son independientes del lenguaje de programación y permiten la abstracción sobre plataformas y lenguajes, entre los que debemos destacar al clásico Corba [Slama et al., 1999] y al novedoso Ice ZeroC [Henning, 2004]. Corba se ha convertido en un estándar gracias a:

- La incorporación de la filosofía Orientada a Objetos.
- La definición de la semántica de la comunicación, usando el formato Interface Description Language.
- La integración de varios protocolos de forma transparente, como TCP o SSL.
- El paso de referencia de objetos remotos.
- Puede usarse con garantías de estándar en los lenguajes: Ada, C, C++, Lisp, Ruby, Smalltalk, Java, COBOL, PL/I y Python.

Por su parte, ZeroC Ice extiende las grandes ventajas de Corba e incorpora nuevos conceptos para hacerlo más intuitivo y accesible a las nuevas tecnologías. Además de las características de Corba, incluye:

- El mapeo a los lenguajes C++, Java, .Net, Python, PHP, Objective-C, Ruby
- Integración de forma transparente de los protocolos TCP, UDP, SSL o HHTP.
- Una versión, algo más limitada, ZeroC Embedded, para operar con Java ME CLDC, Android o IPhone.
- Licencia GPL, siempre es positivo destacar las herramientas de código abierto.

Para los intereses particulares de esta tesis, se han evaluado estas herramientas en función de nuestros requerimientos. Evidentemente, Net. Remoting fue descartado al decantarnos por el desarrollo en Java. RMI fue una opción interesante porque permitía la comunicación entre tecnologías Java, como era nuestro caso, aunque al ser incompatible con Java ME CLDC (la versión para teléfono móviles), lo descartamos rápidamente. Otro punto flaco de RMI es que no puede usarse en otras plataformas distintas a Java, haciendo que posibles portabilidades de nuestro sistema en el futuro, por ejemplo a IPhone, incomunicaran a los clientes y a los servidores.

Por otra parte, los Servicios Web han sido descartados por las limitaciones que ofrecen para dispositivos ligeros, en nuestro caso, Java ME CLDC. Su aspecto positivo radica en que muchas plataformas de desarrollo facilitan la comunicación bajo Servicios Web desde los clientes móviles que soliciten servicios a un servidor central. La principal carencia estriba en que no incluye a los dispositivos móviles como emisores de información, de forma que no es posible desplegar un Servicio Web dentro del propio dispositivo.

El estudio de estas características debe ser considerado en profundidad, porque condiciona drásticamente el tipo de arquitecturas que pueden desarrollarse. Por ejemplo, la restricción que imposibilita a algunos Middleware para desplegar un Servicio en los dispositivos móviles, es incompatible con nuestros propósitos, donde el Entorno Inteligente ha de contactar de forma directa con los dispositivos móviles de los usuarios. Sin esta característica, los dispositivos tienen que conectarse periódicamente con el servidor central para comprobar si existen cambios en el estado del ambiente, tal como ocurre con la versión para dispositivos ligeros de JXTA [Montoya and Piedrahita, 2005]. Evidentemente, esta filosofía consume conexiones innecesarias y provoca retardos en la propagación de la información.

Si por el contrario, el Middleware permite desplegar Servicios dentro de los computadores de mano, y además, permite el paso de referencia de objetos remotos, la arquitectura del sistema puede integrar la propagación de información mediante Canales de Eventos. Este modelo es muy apropiado para arquitecturas de pizarra, donde un sistema central registra a los suscriptores de forma dinámica y puede enviar información a los mismos. Las características de los Canales de Eventos se detallarán en la sección 2.7.1.

Dentro de tanta variedad de nomenclaturas, hemos denominado como *Servicios Remotos* a los métodos y operaciones que han sido desarrolladas en la tesis y que son accedidos de forma distribuida. En nuestro contexto los Servicios Remotos pueden ser desplegados y solicitados, tanto en equipos de sobremesa como en dispositivos móviles.

Finalmente, hemos optado por el desarrollo distribuido bajo ZeroC Ice por todas las ventajas comentadas anteriormente. Como única desventaja, debemos destacar que existen algunas limitaciones de la plataforma para realizar desarrollos en Java ME. Vamos a destacar las dos más importantes para los propósitos de esta tesis. En primer lugar, sólo es posible usar estructuras para definir objetos, esto es, la herencia de clases en la definición de objetos remotos no está permitida. En segundo lugar, no es posible establecer comunicaciones seguras en la red, tales como Secure Sockets Layer (SSL), sino simplemente conexiones simples bajo el protocolo TCP.

2.7.1. Canales de Eventos

En este apartado vamos a hacer referencia a un modelo de arquitectura muy conveniente, en cuanto a organización y comunicación se refiere, en los entornos móviles: los Canales de Eventos. Un Canal de Eventos es una forma de comunicación basada en la suscripción y propagación de información. En ella intervienen dos actores: el Administrador y los Suscriptores. Por un lado, el Administrador del Canal, es el encargado de recibir y recordar a los Suscriptores que desean recibir notificaciones del canal, así como propagar la información a los Subscriptores pertinentes cuando un evento ocurra y deba notificarse.

Por el otro lado, los Suscriptores pueden ser vistos como pequeños agentes que en un momento determinado quieren recibir los eventos que se generan en un canal de interés. Estos clientes procesan los eventos, que llegan de forma asíncrona a la suscripción. Cuando los Suscriptores no deseen recibir nuevos eventos, sólo deberán desuscribirse del canal. Un canal robusto, además, debe eliminar a los suscriptores que estén fuera de línea porque se hayan caído de la red o su dispositivo deje de estar operativo.

Actualmente, esta filosofía ha sido muy reveladora y puede reconocerse, por ejemplo, en la publicación de noticias por Really Simple Syndication (RSS) o en conversaciones simultáneas de los chat en tiempo real entre uno o varios usuarios. Técnicamente, para que un Middleware pueda manejar la suscripción y propagación de clientes

correctamente, debe poder pasar por referencia Objetos Remotos en la llamadas a suscripción. De esta forma, el Administrador recibe directamente el Objeto Remoto al que tiene que notificar los eventos, sin tener que crear una conexión implícita en cada propagación. ZeroC Ice incluye no sólo el paso de referencia de objetos remotos, sino herramientas que facilitan el uso y mantenimiento de los canales de eventos, en concreto, *Ice Storm*.

En nuestro caso, los Canales de Eventos son muy apropiados para representar la interacción entre elementos móviles y el entorno. Por un lado, el Entorno Inteligente, actuaría como Administrador del Canal, esperando a que los clientes móviles fuesen inscritos, y por el otro, los dispositivos móviles representarían a los Suscriptores que se registran en el Sistema. De esta forma, el Entorno Inteligente es consciente de los computadores de mano que están conectados al entorno, pudiendo solicitar información a los mismos, razonar con ella y propagar los resultados en el canal, ver figura 2.11.

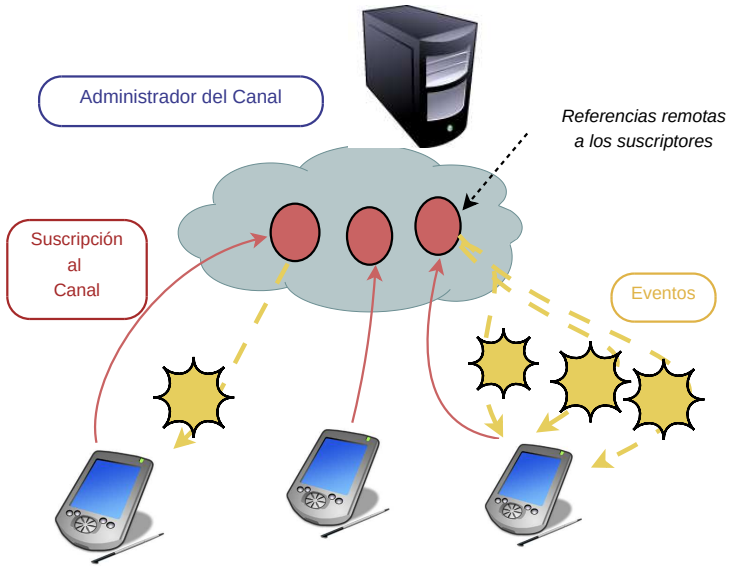

Figura 2.11: Canal de Eventos

En determinados contextos, los Canales de Eventos pueden hacer

visible a los suscriptores entre sí. Para ello, el Administrador puede informar al resto de participantes del canal cuándo un nuevo cliente se ha añadido o se elimina del mismo.

En entornos ubicuos, esta información es muy valiosa. Pensemos que un cliente móvil no puede conocer los parámetros de conexión de todos los clientes móviles, de forma que las conexiones punto a punto entre ellos serían muy difíciles de establecer. Sin embargo, si la suscripción y desuscripción de los usuarios es propagada al resto de participantes del canal, éstos podrían conocer la referencia remota al nuevo usuario operativo y solicitar un Servicio Remoto desde un dispositivo a otro en una conexión punto a punto. Este tipo de conexiones punto a punto entre dispositivos móviles serían invisibles al resto, esto es, sin que medie el servidor central entre ellos, lo cual puede resultar apropiado o no, según el contexto de seguridad y escenario se desarrolle el sistema.

Por lo tanto, los Canales de Eventos centralizan y propagan la información que sea pertinente entre los suscriptores del mismo. Además, pueden mostrar la visibilidad de los Servicios Remotos que aporta cada suscriptor, facilitando comunicaciones no centralizadas y privadas entre los clientes del canal.

2.8. Mapas

Dejando a un lado los mecanismos de comunicación entre el Entorno Inteligente y los dispositivos, es igualmente importante el estudio de las estructuras de intercambio de información entre ambos. En nuestro caso, surge la necesidad de detallar un modelo de representación del espacio que nos rodea y poder trasladar esa representación a un dispositivo móvil. Esta necesidad determinó la base para desarrollar los Mapas Semánticos.

Los mapas deben incluir una representación gráfica, para que los usuarios puedan determinar y reconocer las dimensiones del entorno. En ellos, los usuarios podrán visualizar el espacio que les rodea desde la palma de la mano y acceder a otros dispositivos, como las

fuentes multimedia conectadas en el ambiente. La construcción de los mapas se realizará en tiempo de ejecución leyendo un archivo con la configuración gráfica y semántica del espacio. Esto nos permite, poder cargar los Mapas Semánticos de diferentes edificios, sin necesidad de instalar una nueva aplicación. De esta forma, cuando una persona llega a un edificio podrá sincronizar su mapa con la información del Entorno Inteligente y visualizar los elementos y plantas del espacio que le rodea.

Los Mapas Semánticos integrarán una vista cenital de los edificios que se desarrollará mediante un pequeño motor gráfico para dispositivos móviles que se encargará de representar el edificio. Además, los mapas incluirán otro tipo de información asociada a la gráfica, para determinar las cualidades de los elementos que se están representado. Esta semántica nos permitirá operar con los elementos según sus propiedades. Por ejemplo, integrarán dentro de su representación a los recursos estáticos multimedia: las cámaras y los micrófonos del entorno. Para facilitar el acceso a las fuentes multimedia, los usuarios podrán seleccionar estos recursos desde los Mapas Semánticos, y una vez seleccionada la cámara o micrófono, la aplicación permitirá conectarse con al Servidor Multimedia y mostrar el contenido del mismo de forma automática.

A su vez, los Mapas Semánticos servirán como integración entre los Entornos Inteligentes que monitorizan el ambiente y la comunicación con los dispositivos móviles de los usuarios. La interacción entre sistema y usuarios será moldeada por un Middelware que nos permitirá introducir Servicios Remotos en el Entorno Inteligente y en los dispositivos móviles, además de propagar la información del ambiente a los usuarios usando Canales de Eventos. En concreto, se van a desarrollar servicios de Localización y Notificación en tiempo real.

Mientras que los servicios multimedia desarrollados son válidos para cualquier escenario con recursos multimedia, para la comunicación entre los dispositivos móviles y el Entorno Inteligente es necesario realizar unos servicios ad hoc en función del contexto en que se trabaje. Sin embargo, los propósitos de esta tesis no son desarrollar unos servicios para un contexto determinado, sino estudiar las diferentes

arquitecturas y formas de comunicación entre los clientes móviles y un sistema centralizado.

Por este motivo, se han implantado unos servicios lo más genéricos posibles, pero que pueden ser adaptados a muchos escenarios concretos. En particular se ha estudiado la **localización** y **notificación** de los usuarios usando dispositivos móviles.

No todas la operaciones sobre los Mapas Semánticos deben restringirse a la comunicación con el Entorno Inteligente. Es conveniente aprovechar la capacidad de cómputo de los dispositivos móviles e integrar esa potencialidad en operaciones locales. Para ello, hemos integrado el Cálculo de Rutas en los Mapas Semánticos como un proceso local que puede resolverse sin acceder a Servicios Remotos. El Cálculo de Rutas permitirá conocer el camino para acceder a los diferentes recursos que se muestran en el mapa. En este trabajo, se estudian los algoritmos clásicos de búsqueda y se ofrece una propuesta adaptada a las rutas en edificios e interiores.

Los requisitos y temática de los Mapas Semánticos y la comunicación con el Entorno Inteligente se estudiarán en profundidad en el Capítulo 4.

2.9. Arquitectura

Una vez enumerados los componentes que compondrán la solución y que se han descrito a lo largo del capítulo, en esta sección vamos a describir cómo se organizan todos los componentes software y hardware que hemos elegido para nuestra propuesta. En el primer apartado 2.9.1, organizamos los componentes software dentro de cada dispositivo, mientras que en el segundo apartado 2.9.2 exponemos la comunicación que se establece entre los dispositivos y el entorno.

2.9.1. Estructura de Capas

La estructura en capas permite mostrar el orden en que operan los Sistemas Operativos, Máquinas Virtuales, Middleware y flujos mul-

timedia en tiempo real en los computadores de sobremesa o móviles. La estructura que proponemos permite una gran interoperabilidad. En primer lugar, porque las características de las Máquinas Virtuales permiten abstraer los Sistemas Operativos de los dispositivos, pudiendo trasladar los resultados a otras Máquinas Virtuales sin depender de aplicaciones nativas.

En segundo lugar, a nivel de conectividad, podemos distinguir dos formas de comunicación: Servicios Remotos dentro del Middleware y el streaming multimedia en tiempo real. Ambas comunicaciones permiten que exista independencia respecto a una plataforma de desarrollo concreta, ya que el Middleware posibilita la llamada a los Servicios Remotos desde otras plataformas y porque los flujos multimedia han sido codificados bajo protocolos y formatos estandarizados. Esta división se representa en la figura 2.12.

Además, esta flexibilidad en la comunicación va a permitir que los dispositivos móviles se conecten a las fuentes multimedia de forma transparente al dispositivo que la genera. Gracias a esta transparencia, es posible introducir a los dispositivos móviles, a su vez como fuentes multimedia. Para ello, los dispositivos móviles deberán generar los flujos multimedia con las mismas características con que se transmiten las fuentes ambientales. De esta forma, cualquier receptor podrá conectarse indistintamente a los recursos multimedia ambientales del Entorno Inteligente o a los recursos multimedia móviles de un dispositivo de mano.

2.9.2. Comunicación del Sistema

En la sección anterior, mostramos la organización de los componentes dentro de cada dispositivo. En esta sección, estableceremos una organización para que los dispositivos móviles se comuniquen con el entorno y entre sí de forma apropiada. En este apartado vamos a introducir la distribución de todos los componentes explicados previamente, es decir, la Servicios del Sistema.

Para ello, hemos definido unos requisitos de comunicación que debe validar el sistema, y que han sido agrupados en los dos grandes

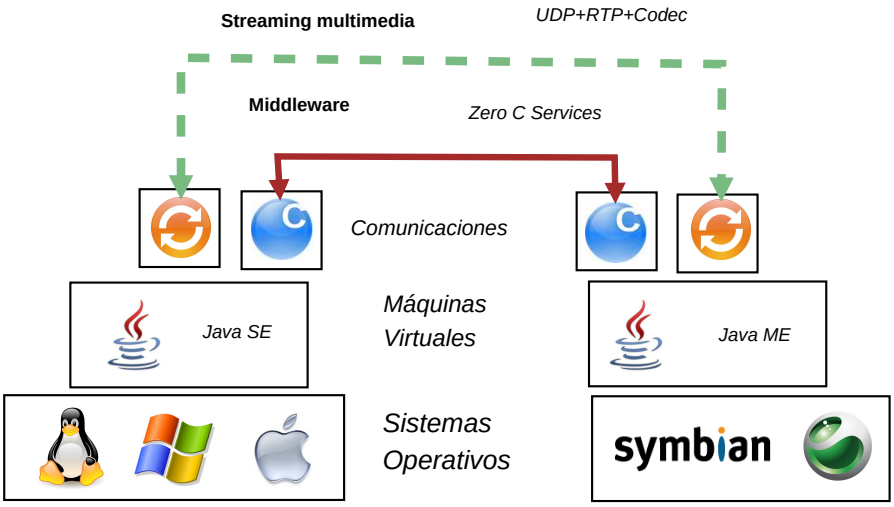

Figura 2.12: Arquitectura de Capas de la propuesta

bloques de la tesis: la Movilización de los Fuentes Multimedia y la Interacción con Entornos Inteligentes mediante Mapas Semánticos.

En primer lugar, los requisitos referentes a la *Movilización de las Fuentes Multimedia* son:

- *Ofrecer un flujo homogéneo multimedia en tiempo real a partir de dispositivos heterogéneos*. En una capa primaria de datos se encuentran las fuentes multimedia estáticas, que representan los dispositivos con capacidad para recoger muestras de audio e imágenes del entorno. Estas fuentes deben ser integradas para trabajar homogéneamente con ellas mediante un componente al que llamaremos Servidor Multimedia. El Servidor Multimedia se ofrece como intermediario entre los dispositivos móviles que desean reproducir el vídeo o audio de las fuentes estáticas.

- *Recepción de los flujo multimedia en los dispositivos móviles*. La petición de los clientes para solicitar la reproducción de un recurso multimedia se definirán bajo Servicios Remotos. Como hemos comentado, las solicitudes de los clientes se realizarán

como un paso previo dentro del Middleware, y posteriormente se transmitirá el flujo multimedia, usando los protocolos ligeros que optimizan los resultados en tiempo real. Esta organización aporta varias ventajas. La prioritaria, recordemos, es la seguridad, ya que los clientes no tienen acceso directo a los recursos, y el Servidor Multimedia puede denegar el acceso a un recurso por motivos de seguridad.

Los formatos y protocolos elegidos para transmitir los flujos multimedia responden a dos requerimientos concretos. El primero es ajustar las transmisiones de las fuentes multimedia a la sensibilidad del tiempo real usando los protocolos UDP y RTP. El segundo, poder traducir el flujo de las imágenes y el sonido desde las Máquinas Virtuales sin delegar en las librerías nativas de los dispositivos. Además, estos mismos formatos y protocolos serán usados posteriormente para generar los flujos multimedia desde la cámara y micrófono de los propios dispositivos. Por ello, deben ser formatos sencillos y computacionalmente poco complejos.

- *Incorporar a los dispositivos móviles como fuentes multimedia en tiempo real del sistema*. También se van a incluir a los propios dispositivos móviles como fuentes multimedia en tiempo real. Para ello, los dispositivos han de acceder al micrófono y a la cámara de los mismos, generando un flujo continuo que es recibido por los clientes. Para potenciar una mayor integración de los componentes, el flujo generado por los dispositivos móviles debe presentar las mismas características que las transmisiones desde Servidor Multimedia. Para ello, será necesario transmitir las fuentes móviles bajo los protocolos en tiempo real y con los formatos de vídeo y audio mencionados. Gracias a la unificación de los flujos multimedia, los receptores en tiempo real de los dispositivos móviles podrán reproducir de forma transparente e idéntica, por ejemplo, la cámara del robot Aibo o la cámara de una PDA.

La Arquitectura y desarrollos de **las Fuentes Multimedia Ubicuas se describen en el capítulo 3**.

En segundo lugar, hemos definido unas operaciones para la *Interacción con Entornos Inteligentes*. Estas operaciones tienen como base de representación los *Mapas Semánticos* de nuestro entorno, que son mostrados en los dispositivos móviles y que son actualizados y consultados mediante una comunicación entre los usuarios y el Sistema. En concreto, se han definido dos Servicios Remotos sobre los Mapas Semánticos:

- *Comunicar los dispositivos con el Entorno Inteligente para recibir la localización en tiempo real de los objetos del sistema.* Este tipo de servicio es bidireccional: el Entorno Inteligente puede solicitar información a cualquiera de los elementos conectados al Sistema, y cuando la localización es procesada y analizada, se reenvía a todos los usuarios inscritos en el Canal de Eventos. Gracias a esta organización, cualquier cambio de localización es conocida por todos los clientes. Evidentemente, esta política podría incluir restricciones de visibilidad al estilo de la herramienta de localización Latitude [Google, 2009], donde el usuario puede ocultar la ubicación o mostrarla a sus amigos de forma especializada.

  En la tesis se van estudiar otras formas de comunicación con los Entornos Inteligentes, a parte de la propagación de información desde los Canales de Eventos, que además no son excluyentes. Por ejemplo, podemos incluir conectividad bidireccional punto a punto entre el servidor centralizado y un cliente móvil determinado. Es el caso cuando un usuario quiere actualizar su localización sin una solicitud previa, o cuando el Entorno Inteligente realiza una petición de posicionamiento a un cliente concreto. Además, la resolución de localizaciones no es exclusiva de un operador humano, por lo que los Servicios Remotos que se han definido abren la puerta a la integración de otro tipo de sensores, de movimiento o localización que pueden alimentar al Sistema y cuyos datos pueden ser propagados a los usuarios suscritos al canal.

- *Mostrar notificaciones en los dispositivos móviles que haya*

analizado el Entorno Inteligente. Además de representar la localización en los Mapas Semánticos, también vamos a estudiar la notificación de alertas a los usuarios usando los Mapas Semánticos en el dispositivo móvil. En este caso, también es idóneo el uso de un Canal de Eventos, donde el Administrador es el Entorno Inteligente y los suscriptores los clientes móviles. Pero en este caso la comunicación es unidireccional: por un lado los sensores y otras fuentes de datos ajenas a los suscriptores deben informar al Sistema, para que éste analice la información y estime las posibles situaciones de interés. Finalmente, el Administrador del Canal genera una alerta que es propagada en tiempo real a los clientes móviles integrados en el entorno. Debemos destacar, que este tipo de alertas no tienen por qué ser siempre peligrosas: pueden determinar desde el peligro por fuego hasta información turística de interés. En nuestro caso, hemos asociado a la alerta una ruta y elementos de objetivo sobre los que se refiere, de forma que, ya sea por emergencia o por simple publicidad, el usuario sabe a dónde y cómo debe ir en caso de interés. La organización en Canales de Eventos y la sincronización de los Mapas Semánticos puede verse en la figura 4.11.

La Arquitectura y desarrollos de los **Mapas Semánticos y Entorno Inteligente se describen en el capítulo 4**.

Finalmente, en la figura 2.13 podemos observar cómo interactúan todos elementos del sistema que hemos descrito en el capítulo bajo una Arquitectura global que integra los Servicios para Fuentes Multimedia y los Servicios desplegados sobre los Mapas Semánticos.

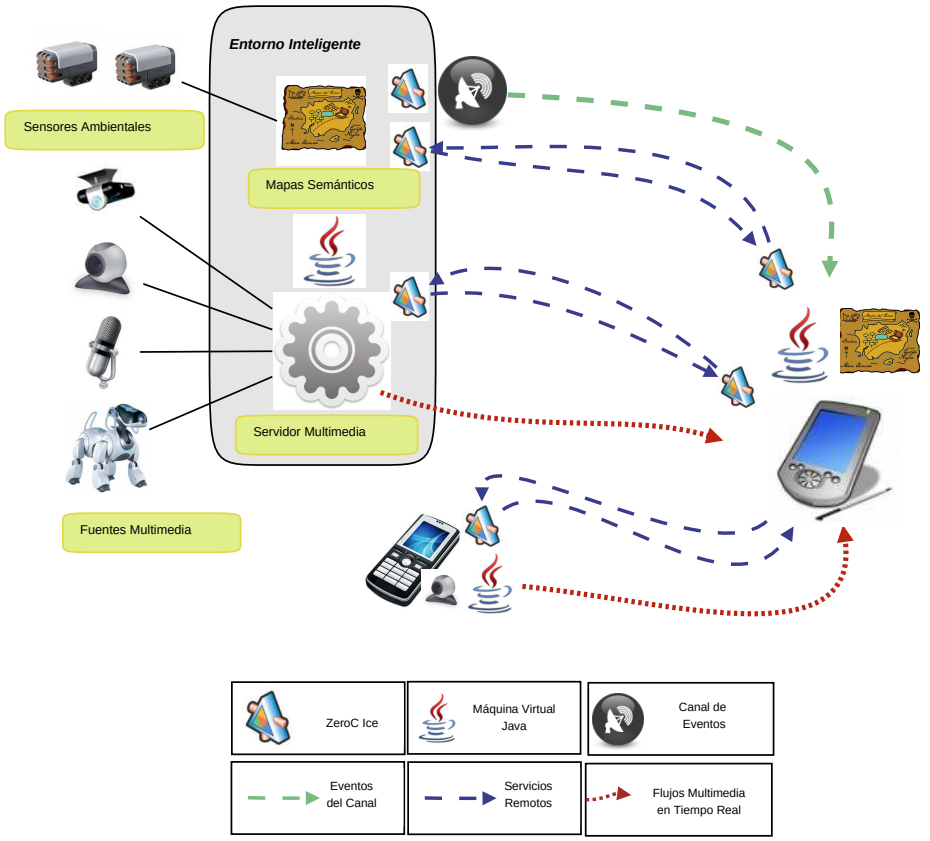

Figura 2.13: Arquitectura integrada

Capítulo 3

Fuentes Multimedia Ubicuas

Usa el método científico: probando varias veces, llegarás a la verdad.

Marco Tulio Cicerón.

Una vez descritos los componentes y la arquitectura que intervienen en nuestra propuesta, en este capítulo vamos a incluir los desarrollos realizados sobre la movilización de fuentes multimedia, que incluirán a los recursos ambientales situados en el entorno y a las fuentes móviles de los computadores de mano.

En la primera **sección 3.1** describimos la Arquitectura Multimedia, que explica la comunicación entre los componentes multimedia dentro el Sistema indicando las peticiones mediante Servicios Remotos y las transmisiones de los flujos en tiempo real.

La **sección 3.2** se centrará el desarrollo del Servidor Multimedia para integrar los recursos multimedia del ambiente, los cuales se caracterizan por su naturaleza estática dentro del entorno.

A continuación, en **la sección 3.1**, detallaremos las aplicaciones móviles que se han construido para realizar una correcta reproducción

de los recursos multimedia en tiempo real. Como veremos, en algunos casos, podremos usar herramientas software que faciliten la visualización o reproducción, mientras que en otros, será necesario partir desde cero construyendo nuestras propias librerías.

Posteriormente, **la sección 3.4** introducirá la misma funcionalidad del Servidor Multimedia en los computadores de mano, generando flujos multimedia en tiempo real desde recursos móviles, estos son, la cámara y el micrófono de los mismos. Para aumentar las posibilidades de comunicación, se han usado los mismos los protocolos y formatos de codificación móviles, pudiendo visualizar cualquier recurso, estático o móvil, desde cualquier computador de sobremesa o de mano.

3.1. Arquitectura Multimedia

El primer objetivo a resolver en la movilización multimedia, es la integración de las fuentes heterogéneas que se encuentran instaladas en el entorno. Para ello, se ha diseñado un Servidor Multimedia en Java SE que pueda conectarse a los recursos estáticos multimedia y ofrecer una salida estandarizada en tiempo real y compatible con las restricciones de los dispositivos móviles. Este Servidor monitoriza las cámaras IP, las cámaras Aibo, cámaras Web y micrófonos, de forma que los clientes no pueden acceder a ellas directamente, sino a través de una sesión que se solicite previamente.

Los clientes que deseen visualizar los recursos multimedia deben realizar una petición previa al Servidor Multimedia, y éste posteriormente creará una sesión en tiempo real hacia el dispositivo que ha solicitado la reproducción del recurso. La petición se corresponderá con una llamada a un Servicio Remoto dentro del Middleware, y el flujo de respuesta se transmitirá bajo los protocolos ligeros (UDP, RTP) comentados en la sección 2.5 y los formatos comentados en 2.6. Los parámetros y detalles de las peticiones al Servidor Multimedia serán estudiados en la siguiente sección. Los desarrollos en dispositivos móviles se realizarán bajo las dos configuraciones de Java ME: CLDC y CDC, por lo que desde ambas aplicaciones debemos solicitar la reproducción de

los recursos multimedia siguiendo la misma filosofía.

Como acabamos de especificar, la petición se formalizará estableciendo una comunicación previa con el Servidor Multimedia. Este paso puede verse como una solicitud de permiso para acceder a los recursos ambientales. Gracias a esta solicitud inicial, pueden establecerse políticas de seguridad o de límite de usuarios en el Servidor, aceptando o rechazando los peticiones de los clientes. Para desarrollar esta sincronización previa entre los clientes y el servidor, estudiamos inicialmente la posibilidad de incorporar una ampliación del protocolo RTP, conocida como *RTSP* (Real time Transport Streaming Protocol). El protocolo RTSP envía unas tramas determinadas para solicitar/iniciar/parar la transmisión, lo cual es muy cómodo para acceder directamente a las fuentes sin pasar por el Middelware. Sin embargo, bajo esta solución los recursos multimedia se hacen públicos y accesibles.

Para protegerlos explícitamente, hemos especificados unos Servicios Remotos que definen el inicio y parada de las sesiones mediante el Middleware. Gracias a ello, podemos asegurar que los flujos multimedia en tiempo real sólo pueden generarse mediante el servicio concreto de ZeroC Ice y que el Servidor Multimedia pueda incluir políticas de seguridad de acceso a los recursos. De esta forma, incluso es posible dividir el acceso en redes diferentes: la primera entre los recursos multimedia estáticos y el Servidor Multimedia; y la segunda entre los dispositivos móviles y el Servidor Multimedia.

Otra ventaja es la posibilidad de transmisiones multidifusión, o *broadcast*, que permiten que varios clientes reciban un mismo flujo multimedia. Cuando el Servidor Multimedia recibe una solicitud sobre un recurso en uso, simplemente ha de añadir al nuevo cliente como destinatario de la sesión. Si los clientes accediesen directamente a los recursos multimedia sin un Servidor Multimedia que las monitorizara, no podrían emplearse este tipo de técnicas.

Posteriormente, cuando el Servidor acepta la petición y genera el flujo multimedia, los clientes se encargan de recibir, decodificar y presentar el *streaming* en el dispositivo móvil. Como veremos en

este capítulo, para los computadores de mano, como PDAs, existen herramientas que facilitan la reproducción en tiempo real, mientras que para los teléfonos móviles debemos construirlas nosotros mismos.

Uno de los objetivos más novedosos de nuestro sistema, es la incorporación de los dispositivos móviles como fuentes multimedia. Eso implica, generar un flujo de datos en tiempo real desde la cámara o micrófonos de los computadores de mano. Para ello, cada dispositivo fuente debe implementar un Servicio Remoto que otros clientes invocarán para solicitar la reproducción del recurso móvil. Es importante que este servicio sea el mismo que implementa el Servidor Multimedia para el inicio/parada de las sesiones en tiempo real, porque nos permite ampliar las posibilidades de interconexión de los dispositivos. En este caso, los clientes podrán conectarse entre ellos en una comunicación punto a punto sin pasar por el Servidor Multimedia. Además, el cliente podría conectarse y visualizar los recursos móviles multimedia con transparencia al emisor que lo genera, ya sea el Servidor Multimedia que monitoriza un hogar, o cualquier dispositivo móvil conectado al sistema.

3.2. Servidor Multimedia para Recursos Ambientales en Tiempo Real

Como se introdujo en la sección 2.1, hoy día, existen numerosos dispositivos a nuestro alrededor con capacidad de describir el mundo visual o acústicamente. Estos recursos suelen estar instalados en ordenadores e instalaciones para poder visualizar a las personas y entorno que los rodean. En esta sección integraremos los recursos ambientales para que puedan ser accesibles de forma transparente y la transmisión de los datos sea sensible al tiempo real. Sin embargo, resulta complejo ofrecer una solución que integre distintos componentes multimedia ambientales, porque el acceso a los dispositivos presenta sendas diferencias, como por ejemplo, la cámara de un robot respecto a una cámara web.

A modo de resumen, los objetivos generales del Servidor

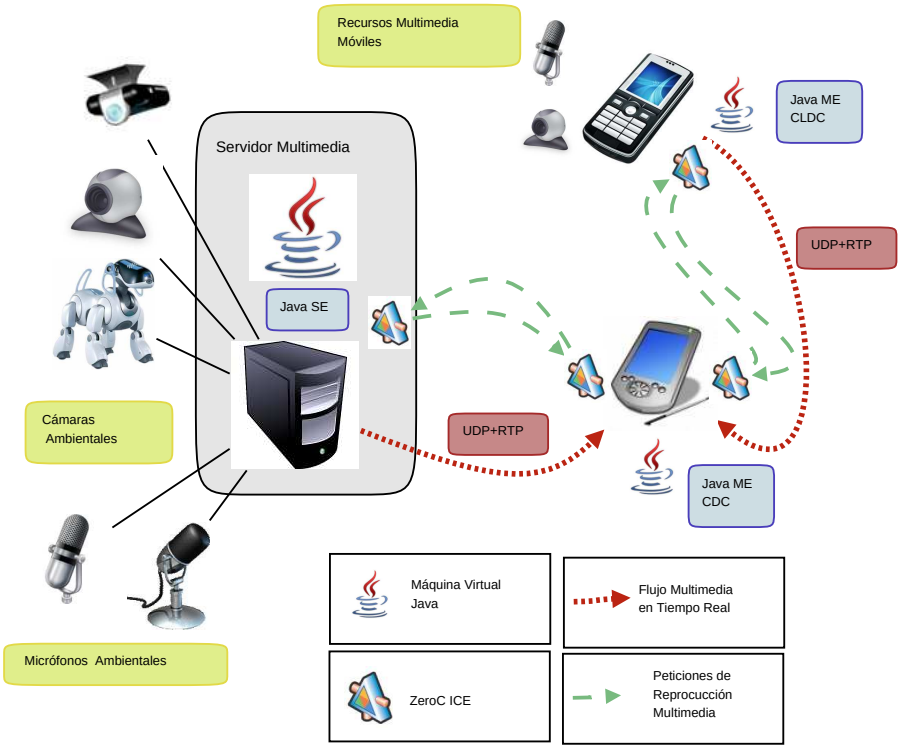

Figura 3.1: Esquema de Movilización Multimedia

Multimedia son:

- Generar un flujo de multimedia homogéneo independiente de las fuentes que lo generen.
- Incluir características del tiempo real en el flujo multimedia. En concreto, introduciremos los protocolos y formatos que han sido estudiados en la propuesta 2.5 y 2.6.
- Adaptar la transmisión en función de las pretensiones de cada dispositivo móvil, de forma que los clientes puedan establecer una serie de parámetros en el envío.

El esquema general del Servidor Multimedia puede verse en la figura 3.2. Como vemos, para solicitar la transmisión de una fuente los

clientes tienen que enviar una petición previa que determine la fuente y parámetros con que desean que se genere el flujo. Los parámetros permiten adaptar la transmisión a las capacidades de cada dispositivo móviles. Evidentemente, estos ajustes tienen valores distintos si la fuente multimedia es de vídeo o de audio.

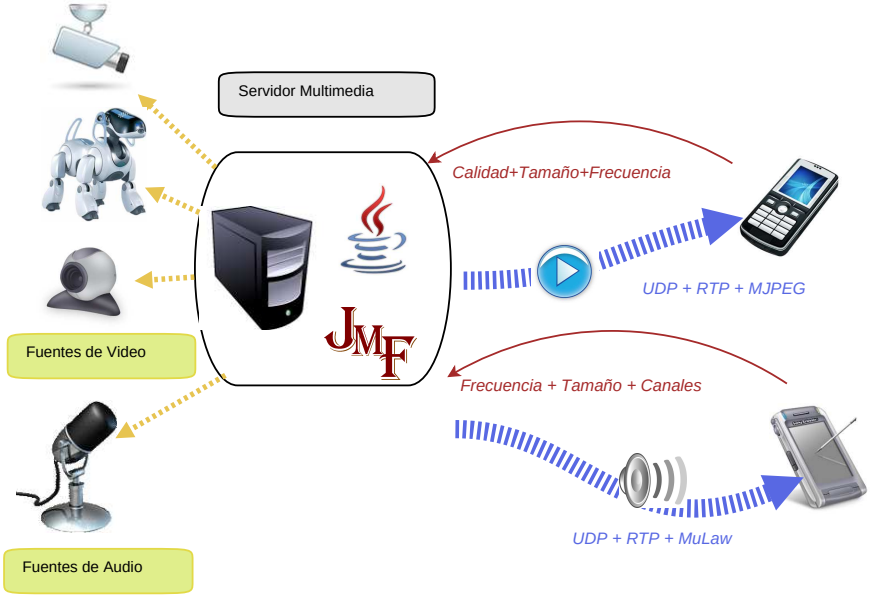

Figura 3.2: Funcionalidad del Servidor Multimedia

Para resolver estos requisitos generales, se han seleccionado un conjunto de requisitos funcionales que el Servidor Multimedia debe incluir para cubrir los objetivos establecidos. Estos requisitos son:

- *Conectividad con las fuentes multimedia*. El Servidor Multimedia es el encargado de monitorizar los dispositivos de audio y sonido, así como de ofrecer un flujo de datos multimedia en tiempo real. De esta forma, los clientes no tienen porque tener acceso a las fuentes y aumenta la seguridad y privacidad del Sistema.
- *Negociación de puertos*. Es importante que una entidad controle los puertos que son usados en cada conexión multimedia. Así, se

evitan situaciones de conflicto en las que varios clientes intentan solicitar un mismo puerto del servidor para la recepción del flujo multimedia, haciendo inviable la transmisión. Por tanto, el Servidor Multimedia es el encargado de gestionar puertos libres para que puedan establecerse conexiones concurrentes satisfactoriamente.

- *Integración homogénea*. Cada fuente multimedia tiene unas características diferentes al resto. Por ejemplo, una cámara web suele estar conectada por USB, mientras que una cámara IP se conecta por red; sin embargo, el Servidor debe ofrecer un flujo de vídeo transparente a las características de cada una de las fuentes.

- *Ajuste del flujo en función del contexto*. Cada cliente móvil se caracteriza por disponer de unos recursos de cómputo y tamaño de pantalla diferentes. Esto hace que, en la práctica, el flujo en tiempo real de un dispositivo sea inapropiado para otro. Para solventar esta problemática, cada cliente puede establecer unos parámetros de ajuste sobre la transmisión, ya sea de audio o de vídeo. Será el Servidor Multimedia quien configure y genere la conexión en función de la frecuencia, tamaño y calidad requeridas por los clientes.

- *Gestión de* **[broadcast??]** *sobre los receptores*. En el caso de que un cliente solicite la conexión a una fuente multimedia, que ya se está enviando previamente, y cuyos parámetros de ajuste sean idénticos, el Servidor añadirá a este cliente en canal multidifusión. En particular, la difusión multi-unicast, nos permite enviar a varios receptores un único flujo multimedia de datos. De esta forma, no se satura la fuente porque sólo se realiza una conexión para obtener las capturas, pero el mismo flujo puede ser recibido por varios clientes.

El ciclo de vida de los flujos multimedia en tiempo real ha sido definido en tres fases. La primera, corresponde a una solicitud a través del Middleware en la que se incluye el identificador del

recursos y los parámetros a los que debe ajustarse el flujo. En caso de que los parámetros e identificador sean válidos, el Servidor Multimedia devuelve la dirección IP y el puerto donde va a producirse la transmisión. La segunda fase se corresponde con la generación del flujo según el contexto del cliente. En esta fase, se recogen muestras de la fuente, se comprimen y se envían en tiempo real bajo las restricciones que han sido solicitadas. La última fase es activada por el cliente cuando desea que el Servidor Multimedia cierre el flujo multimedia.

Esta etapa de finalización es necesaria porque debido a las características de los protocolos ligeros, el Servidor no es consciente de la recepción de los datos del cliente. Sin una finalización explícita, los receptores podrían dejar de reproducir el flujo, y el Servidor seguiría enviando los datos de la fuente. Para evitar este consumo innecesario de recursos, el cliente debe solicitar la finalización del flujo, para que, posteriormente, el Servidor Multimedia cierre la transmisión.

3.2.1. Arquitectura del Servidor Multimedia

Para ampliar la potencialidad del Sistema, según los escenarios donde opere el Servidor Multimedia, hemos dividido la funcionalidad del mismo en componente distribuidos que realizan diferentes funciones y que otorgan mayor flexibilidad al sistema. Esta división puede verse en la figura 3.4.

En primer lugar, hemos definido la conectividad de las fuentes multimedia en la interfaz *MultimediaSources*. Este componente se encarga de representar un nodo que está conectado a un conjunto de recursos multimedia y que puede generar un flujo multimedia a partir de los mismos. Cada nodo *MultimediaSources* especifica los recursos que monitoriza mediante un archivo XML de entrada para que los usuarios puedan gestionar cómodamente la inclusión o eliminación de recursos. Un ejemplo de este fichero puede verse a continuación, donde se especifica una fuente de sonido y una cámara IP.

Imaginemos por ejemplo un par de ordenadores, cada uno con su cámara web y micrófono diferentes, existirían por tanto un

```
<resources>
        <resource id="microphone1" type="Sound">
                <url>sound://</url>
        </resource>

        <resource id="camara2" type="Video">
                <url>http://161.67.108.7/axis-cgi/jpg/image.cgi?resolution=320x240</url>
        </resource>

        <resource id="camara3" type="Video">
                <url>vfw://0</url>
        </resource>

        <resource id="camara4" type="Video">
                <url>aibo://192.168.0.102:8000</url>
        </resource>
</resources>
```

Figura 3.3: Especificación en XML de recursos multimedia ambientales

nodo *MultimediaSources* en cada uno de ellos. Si en lugar de un par de ordenadores, estuviésemos en una oficina de trabajo que deseamos monitorizar, el número de nodos crece. Sin embargo, no parece cómodo que un dispositivo móvil, que desee visualizar cualquier recurso multimedia de un edificio, deba conectarse a uno u otro individualmente. Es necesario, por tanto, especificar un mecanismo de agrupamiento para que todos los nodos sean accesibles conjuntamente.

Para ello, la interfaz *MultimediaSourcesGroup* agrupa a varios nodos de forma dinámica, y permite la agregación o segregación del grupo en tiempo de ejecución. Con esta distribución, una fuente multimedia puede ser transmitida por varios nodos. Esto aumenta la potencialidad del sistema, ya que por un lado, varios clientes pueden transmitir una misma fuente, y además la interfaz *MultimediaSourcesGroup* puede buscar otro nodo que transmita el recurso, en caso de que uno de ellos se encuentre caído.

Finalmente, la interfaz *MultimediaSession* resuelve las peticiones de solicitud de los recursos multimedias, junto con los parámetros de ajuste de cada cliente. Ésta representa, por tanto, la interfaz pública que ven los clientes, siendo el resto de componentes y la arquitectura multimedia totalmente transparentes para ellos.

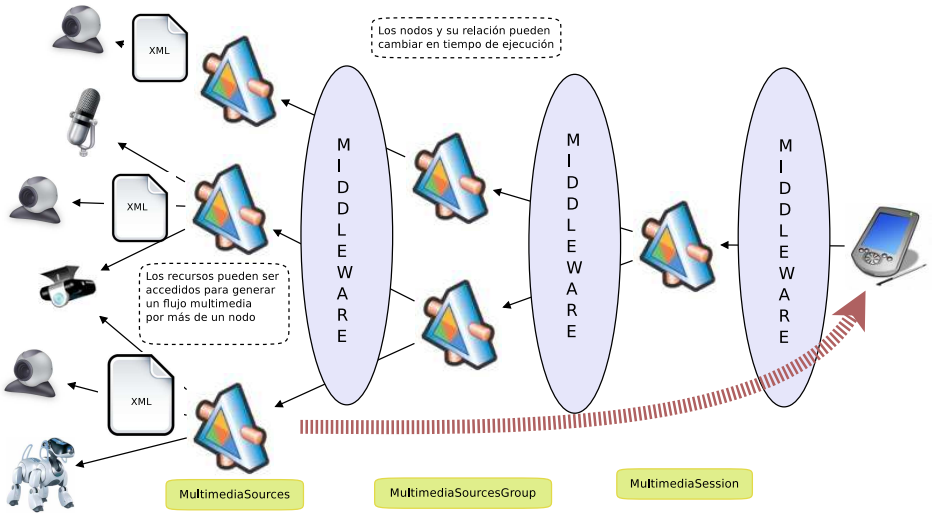

Figura 3.4: Arquitectura del Servidor Multimedia

Ajuste del flujo multimedia en tiempo real Como hemos comentado, en la petición de reproducción multimedia, los clientes pueden incluir una serie de parámetros de configuración del flujo multimedia que ajusten la recepción a las limitaciones de cada dispositivo. Esto parece sensato, ya que cada dispositivo tiene por ejemplo, un tamaño de pantalla determinado, y es apropiado solicitar un flujo de vídeo que se adapte a la misma. De esta forma, no se desperdicia una transmisión en una escala demasiado grande, o por el contrario, se generan imágenes demasiado pequeñas para un dispositivo concreto. En particular, cada visualizador de vídeo puede establecer tres propiedades:

- *El tamaño de las imágenes*. Deben corresponder a tamaños múltiplos de 8, para ser ajustados correctamente al formato M-JPEG, por ejemplo 320x240 o 160x120.

- *La frecuencia*. El cliente puede limitar la frecuencia máxima con la que se transmite cada imagen. De esta forma, los clientes pueden prevenir una posible saturación de imágenes debido a las limitaciones de cómputo de los mismos. Es importante

detallar que, por el contrario, el Servidor Multimedia ofrece esta frecuencia en base a sus posibilidades y las de la fuente; es decir, en caso de que la fuente transmita a menor frecuencia que la solicitada por el cliente, la transmisión se corresponderá con la limitación de la fuente.

- *La calidad*. El formato de compresión de imágenes M-JPEG puede ser ajustado gracias a este parámetro, aunque evidentemente afecta a la nitidez y granularidad del vídeo generado. Este parámetro es independiente de la fuente visual, es decir, la transformación de la escala la realiza el propio Servidor Multimedia.

Por su parte, si el recurso al que se hace referencia es una fuente de audio, el cliente que la solicita puede especificar:

- *Número de muestras*. Este parámetro especifica el número de muestras por segundo que pueden recogerse. Para una mayor compatibilidad con G.711 este formato debe ser de 8000 muestras, o al menos un múltiplo de éste.
- *Tamaña de las muestras*. Cada muestra puede obtenerse con menor o mayor calidad (8 bits o 16 bits).
- *Número de canales*. El cliente puede establecer el flujo de audio en mono o estéreo.

Implementación del Servidor Multimedia En el capítulo anterior, se describieron las bondades y limitaciones de las Máquinas Virtuales en el ámbito de los dispositivos móviles. El Servidor Multimedia no se encuentra restringido por el desarrollo móvil, ya que a priori, puede ser construido para equipos de sobremesa. Además, el uso de las Máquinas Virtuales y sus ventajas son extensible a cualquier computador. Por ello, la implementación de Servidor Multimedia se ha realizado en Java Standard Edition (Java SE), que nos permite usar las variadas librerías de la plataforma y portar la solución a Sistemas Windows, Linux o Mac OS.

Una de las librerías más populares en Java para potenciar las posibilidades multimedia, es Java Media Framework (JMF) [Gordon and Talley, 1999], la cual fue creada para dar soporte a muchas de las cuestiones tratadas en esta sección. Esta librería ha sido usada en el Servidor Multimedia, y ha permitido desarrollar los requisitos funcionales que se han especificado en el sistema. Según el contexto de nuestro trabajo, algunos de los puntos fuertes de JMF son:

- *Codificación*. Los códecs son definidos como codificadores, decodificadores o ambos. Además dispone de un Buffer a alto nivel, que abstrae estos procesos con independencia del protocolo de acceso (file, http, etc...). Los códecs pueden ampliarse y registrarse en JMF de forma escalable, estableciéndose una colección de formatos que nuestras aplicaciones Java puedan usar.

- *Protocolos acceso/envío*. Para el envío/recepción de información disponemos de protocolos de acceso, tales como file, HTTP, o RTP. En los casos de los ordenadores de sobremesa, además, se incluyen protocolos de acceso para la cámara web o para el micrófono, de forma que es posible operar con los recursos a alto nivel.

- *Abstracción en la presentación*. Se han construido clases que permiten abstraer el proceso de configuración, realización y presentación de la información multimedia de forma transparente al programador. Por ejemplo, es capaz de reconocer una configuración de codecs que permita traducir los datos en función de las características del sistema, decodificando automáticamente los datos y reproduciéndolos en el recurso apropiado. Evidentemente, si el Sistema no dispone de los codecs, protocolos o recursos necesarios, se produce un error.

- *Ortogonalidad*. Aunque se automaticen las fases de reproducción o envío, es posible especificar y cambiar cualquier módulo de las etapas. Por ejemplo, podemos diseñar nosotros mismos un protocolo o un decodificador o especificar el envío de una fuente de vídeo por RTP en algún formato diferente a los que se ofrecen por defecto.

- *Portabilidad*. JMF es portable, al menos una versión del mismo: *cross-all*, la cual incluye los formatos y protocolos íntegramente en la librería. Esta versión realiza todos los procesos de codificación y decodificación sobre la Máquina Virtual, pudiendo portar la solución a cualquier plataforma Java compatible. Sin embargo, la versión JMF *cross-all* no soporta algunos formatos, sobre todo de vídeo, mientras que la versión para Windows o Linux permite reconocer y usar los drivers instalados en el propio Sistema Operativo. Afortunadamente, para los propósitos de este trabajo, la versión *cross-all* nos permite cubrir todos los formatos y protocolos especificados en los requisitos.

3.2.2. Dispositivos de vídeo

En este apartado se presentan los desarrollos realizados para integrar diferentes fuentes de vídeo en el Servidor Multimedia. Para probar la escalabilidad del Servidor, hemos realizado pruebas con diferentes tipos de fuente de vídeo: cámaras IP, cámaras web y cámaras del robot Aibo, aunque no son excluyentes y puede incluirse otro tipo de dispositivos de vídeo gracias a la modularidad del Servidor Multimedia y de la propia librería JMF.

Una de las características de nuestros dispositivos multimedia, es que los recursos deben ser solicitados a demanda, en función de la capacidad de la transmisión RTP y de la velocidad de captación de imágenes de los mismos. La clase de JMF idónea para este tipo de fuentes es *PullBufferStream*. Esta clase representa las fuentes que deben de llamarse de forma explícita para recoger datos multimedia. A diferencia de un archivo físico, como una película de vídeo donde el número de imágenes y frecuencia es constante, las imágenes de las cámaras ambientales pueden ser capturadas en función de la capacidad y parámetros de los clientes. Bajo esta filosofía, es el Servidor Multimedia el que solicita una nueva imagen a la cámara, y no la cámara la que acumula las imágenes en el Servidor. El formato de captura de las imágenes se ha establecido como JPEG, de forma que pueden integrarse otras fuentes que permitan recoger capturas bajo

este formato.

De esta forma, para poder encapsular cualquier fuente de información JPEG en la librería JMF, hemos implementado una clase, *JPEGCameraStream*, que hereda las propiedades de las fuentes *PullBufferStream*. Gracias a esta herencia, la captura de imágenes es invocada automáticamente por el manejador RTP para construir el flujo de vídeo. Esto nos exime de grandes complicaciones en la sincronización de RTP, ya que cada fuente es trasmitida a la velocidad que su canal permite sin influir en el resto de transmisiones. La clase *JPEGCameraStream* debe indicar a JMF cada track del vídeo y escribirlos dentro del canal del buffer. Para ello, es necesario recoger los bytes de la imagen que la cámara está visualizando actualmente y establecer el formato de entrada de la fuente como JPEG. Este formato especifica la naturaleza de los datos de entrada, aunque es posible especificar otros formatos en el caso de que las fuentes de vídeo capturasen las imágenes bajo otra codificación.

La clase *JPEGCameraStream* además de encapsular los accesos de fuentes JPEG, incluye un controlador de tiempo capaz de ajustar la frecuencia de envío según la solicitud de los clientes. Para este propósito se almacena el tiempo de la última captura, y en caso de ser necesario, se incluye una espera que despierta el proceso en el momento justo en que debe tomar la nueva imagen. Evidentemente, si la frecuencia solicitada por el cliente es mayor que la de captura, no es necesario esperar.

Además de este ajuste temporal, los datos recogidos del dispositivo de vídeo pueden ser adaptados al tamaño y la calidad que deseen los receptores. Para ello, se incluye una fase posterior a la captura, que permite construir una nueva imagen JPEG con otro tamaño y calidad diferente a la original. Este proceso es esencial en la movilidad, porque muchas veces la calidad de captura de una cámara está pensada para pantallas de sobremesa, y gracias a este ajuste el envío y recepción de imágenes se ciñe a las limitaciones de los computadores de mano.

Las primeras cámaras que fueron incluidas como fuentes de vídeo fueron las cámaras IP porque la captura de las mismas venía

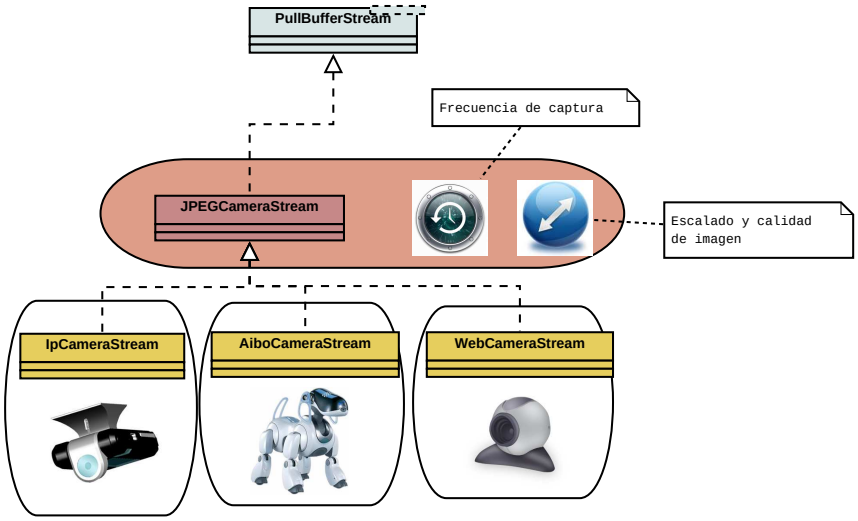

Figura 3.5: Clases para el envío de fuentes JPEG

codificada en JPEG y hacía sencilla su encapsulación en M-JPEG. En general, cualquier cámara IP ofrece una ráfaga de imágenes JPEG mediante HTTP, aunque existen algunas más modernas que permiten otro tipo de codificaciones más complejas, como MP4 o la propagación de vídeo bajo el propio protocolo RTP. Sin embargo, nosotros vamos a trabajar con la funcionalidad básica de codificación en JPEG y transmisión bajo el protocolo HTTP porque representa el requisito mínimo de las mismas y, de esta forma, pueden integrarse cualquiera de ellas dentro del Servidor Multimedia. Finalmente, hemos definido un identificador basado en URL para definir y parametrizar los recursos de las cámara IP: "ipcamera://ip:puerto", de forma que en la cadena se especifica la dirección y puerto donde el Servidor Multimedia debe conectarse para realizar las capturas. La clase que implementa las especificaciones de las cámaras IP, ha sido nombrada *IpCameraStream*.

De forma parecida, hemos incluido una configuración de vídeo para integrar la cámara de los robots Aibo y transmitir su visualización mediante la red Wi-Fi del perro. Los robots Aibo disponen de una cámara para capturar y analizar el entorno, por lo que nos pareció

novedoso introducirlos como visualizadores del Sistema Multimedia, y ofrecer una perspectiva dinámica de los espacios del edificio. Aunque la captura de imágenes es similar a las cámaras IP, hay que establecer unos pasos previos. La plataforma de desarrollo Aibo cuenta con herramientas para capturar y almacenar las imágenes, sin embargo éstas no pueden ser accedidas, a priori, por la red. Para que esto sea posible, existe una pequeña aplicación, llamada *W3C*, que permite acceder a la cámara de Aibo usando el protocolo HTTP, y convertirlo en una fuente accesible de vídeo. De esta forma, el Servidor Multimedia puede encapsular las imágenes del Aibo de forma similar a las IP, aunque tenemos que detallar que no son iguales, ya que Aibo incluye una cabecera con meta información asociada a cada imagen. El Servidor Multimedia se encarga de obtener los datos JPEG y eliminar la cabecera extra que introduce el robot.

La lectura de las muestras Aibo, se especifica en una clase independiente *AiboCameraStream*, que se conecta con los robots y captura los frames para que JMF pueda configurar el vídeo en tiempo real. También hemos definido un identificador basado en URL para estos robots: "aibo://ip:puerto". Podríamos matizar, que bajo una filosofía puramente distribuida, las capacidades de Aibo permitirían transmitir el flujo de vídeo en cada robot, sin la intervención del Servidor Multimedia, para lo que sería necesario desarrollar el proceso de transmisión en tiempo real y codificación en el procesador del robot. Sin embargo, esta filosofía versión punto a punto no es posible, a día de hoy, porque no se ha encontrado una librería que manipule sesiones RTP en los robots.

En último lugar, las cámaras web suelen estar conectadas mediante un puerto USB, que las comunica con un computador de sobremesa. Para que este tipo de dispositivos sean accesibles deben de instalarse unos drivers específicos que integren y configuren las cámaras dentro del Sistema Operativo. Gracias a este controlador, otras aplicaciones pueden comunicarse con estos dispositivos de vídeo. Este proceso es el que incorpora JMF, que permite el acceso a las cámaras web buscando los controladores de vídeo que estén disponibles en el Sistema Operativo. Por ello, la conectividad de las cámaras web

Tabla 3.1: Número máximo de imágenes por segundo

Cámera IP	Cámera Aibo	Cámara web
30	18.1	44.2

con la librería JMF es dependiente del Sistema Operativo y no está disponible en la versión *cross-all*. Sin embargo, sí es portable a los dos entornos Windows o Linux, gracias a las versiones de JMF adaptadas a los mismos.

La clase que implementa la funcionalidad de las cámaras web ha sido notada como *WebCameraStream*. Para poder capturar las imágenes de la cámara web es necesario inicializar el dispositivo creando una instancia del vídeo de la cámara. Este proceso debe realizarse sólo una vez, en el constructor de la clase, porque el tiempo de inicialización y configuración es costoso. Posteriormente, cada vez que se solicite una captura, podremos acceder eficientemente a las imágenes de la misma, y escribir en el buffer de JMF los frames en formato JPG. La URL que identifica a la cámara web es *vfw://n*, donde *n* es el número asociado a una determinada cámara, ya que pueden existir varios dispositivos conectados a un mismo ordenador.

Finalmente, en 3.1, mostramos los resultados de captura máxima que cada dispositivo de vídeo genera con una conectividad óptima, esto es, y sin otros problemas externos de recepción. Como vemos, el dispositivo con mejor comportamiento son las cámaras web, ya que la conectividad por USB no se produce por la red como en el caso de las cámaras IP, o incluso por la red inalámbrica en el caso del robot Aibo. Aún así, estos resultados son más que óptimos para la recepción en dispositivos móviles, porque en los casos más limitados, un tasa de refresco de 6 u 8 imágenes por segundo puede representar el límite de saturación en algunos dispositivos móviles ligeros.

3.2.3. Dispositivos de sonido

Aunque las cámaras ofrecen una atractiva percepción del entorno basado en imágenes, también son muy importantes las fuentes de sonido; ya que recogen los datos de audio del ambiente para que puedan ser reconocidos por un operados humano, o incluso un sistema experto que reconozca patrones. En el contexto de nuestro trabajo, existen dos tipos de funcionalidades derivadas del uso de micrófonos. Por un lado, dispositivos estáticos situados en las instalaciones y que recogen muestras del sonido ambiental, de forma que puedan ser transmitidas y reproducidas en los dispositivos móviles en tiempo real. Por otro lado, la generación de audio desde los propios dispositivos multimedia, que veremos en secciones siguientes, que añaden la funcionalidad de establecer conversaciones de voz en tiempo real desde redes inalámbricas y de forma gratuita.

En esta sección, vamos a centrarnos en el primer caso, en dispositivos de sonido conectados al entorno y que pueden ser accedidos en tiempo real desde computadores de mano. De esta forma, a las diferentes fuentes de vídeo comentadas anteriormente, vamos a incluir recursos de otra naturaleza como los de sonido. Para realizar las pruebas sobre el ambiente, hemos usado un micrófono instalado a un ordenador por USB, al estilo de las cámaras web. Las muestras de sonido del micrófono son recogidas por el Servidor Multimedia, usando la versión para Windows o Linux de JMF. Al igual que en el caso de las cámaras web, la manipulación de estos dispositivos pasa por la correcta configuración del micrófono en el Sistema Operativo, y por lo tanto, sólo esta disponible en las versiones de JMF nativas.

Gracias a la abstracción que otorga JMF, el programador no tiene que abrir el micrófono y efectuar lecturas periódicas de los datos recogidos, sino que el proceso es efectuado automáticamente por la librería multimedia. Sin embargo, como veremos en la generación de audio desde los dispositivos móviles, no contamos con esas herramientas en las PDAs, donde hemos tenido que obtener los datos del micrófono a muy bajo nivel.

Una vez capturados los datos del micrófono, es necesario en-

viar los mismos en tiempo real en paquetes codificados a los receptores. Para ello, hemos usado el formato de compresión comentado en 2.1. Recordemos que el formato G.711 ofrece una rápida compresión/descompresión de los datos y que reduce el tamaño de envío del audio. Además cada muestra o paquete tiene independencia del resto, de forma que una pérdida de paquetes en la red no afecta a los sonidos enviados anterior o posteriormente, reduciendo así los tiempos de silencio.

En caso de incluir en trabajos futuros otras fuentes de sonido, como terminales de voz IP, éstos podrían integrarse fácilmente dentro del sistema. Para ello, el Servidor Multimedia tendría que recoger datos a 8000 muestras por segundo de los mismos y a 8 bits por muestra, para que éstos puedan ser codificados y enviados automáticamente usando JMF. Recordemos que esta frecuencia y tamaño de captura de audio es necesaria si trabajamos con el formato G.711.

Resumen El Servidor Multimedia es un componente distribuido que permite integrar y comunicar diversas fuentes multimedia de forma homogénea. A su vez, puede estar formado por varios nodos, llamados *MultimediaSource*, que agrupan uno o varios recursos multimedia. La estructura y organización es transparente a los clientes, por lo que es necesario un componente que haga de intermediario entre los recursos multimedia y que ofrezca un control de los puertos y permisos de acceso. Este componente se denomina *MultimediaSession* y es el único que establece comunicación con los clientes móviles.

Los diferentes tipos de recursos que actualmente monitoriza el Servidor Multimedia son: cámaras web, cámaras IP, cámaras Aibo y micrófonos integrados. Los tres primeros recursos ofrecen un vídeo en tiempo real que es homogéneo respecto a las fuentes que las genera, esto es, independientemente de si es una cámara web o IP, el flujo tiene las mismas características. La codificación y envío en tiempo real se ajusta a las elecciones realizadas en 2.6 y 2.5. Además, cada cliente puede especificar unos parámetros de ajuste para la creación de los flujos multimedia, pudiendo amoldar la transmisión al contexto de

cada receptor.

3.3. Dispositivos Móviles como Receptores Multimedia en Tiempo Real

Una vez concluido el estudio sobre integración de recursos multimedia ambientales, los objetivos marcados recaen sobre la movilización de los flujos multimedia generados. Estas herramientas móviles deberán sincronizarse correctamente con el Servidor Multimedia, trasladando la visión de las cámaras y el sonido de los micrófonos a cualquier localización de los usuarios.

Por lo tanto, en esta sección se presentan las aplicaciones para mostrar los flujos multimedia en tiempo real y que han sido realizadas especialmente para dispositivos móviles. En primer lugar, incluimos los resultados asociados a la recepción de vídeo y posteriormente incluimos los de audio. Además, como veremos a continuación, se han desarrollado reproductores tanto en dispositivos ligeros, como teléfonos móviles, y en computadores de mano más potentes, al estilo de PDAs. El motivo es que ofrecen características y potencialidades diferentes, por lo que las plataformas y herramientas usadas son diferentes. También es importante enfatizar el carácter de portabilidad que se ha perseguido para las aplicaciones, realizando un esfuerzo extra para que estos resultados abarquen al mayor número de dispositivos móviles.

3.3.1. Receptores de Vídeo

En primer lugar, vamos a detallar los mecanismos y peculiaridades de la recepción de los flujos de vídeo en tiempo real. Como hemos comentado anteriormente, el flujo generado por el Servidor Multimedia es transparente a las fuentes que lo generan, por lo que las aplicaciones móviles y los resultados mostrados a continuación son independientes del tipo de cámara que desarrolla el vídeo en tiempo real. Gracias a la adaptación del flujo a la frecuencia, tamaño y calidad de los clientes, se ha podido realizar un estudio para evaluar el umbral de saturación de

las herramientas que hemos desarrollado.

3.3.1.1. Receptores de Vídeo en computadores de mano

Recientemente, han surgido nuevos dispositivos de mano llamados agendas electrónicas, PDAs u ordenadores de bolsillo, con mayor potencia de procesamiento y mejores prestaciones que las ofrecidas por los teléfonos móviles. Por lo general, éstos ofrecen una comunicación por pantalla táctil que facilita la interacción entre el usuario y la máquina y que permite manejar el dispositivo con la mano y/o un lápiz.

Debemos recalcar que existe un importante sesgo entre la programación de teléfonos móviles y ordenadores de bolsillo, que se deriva de las diferencias intrínsecas entre unos dispositivos y otros. Mientras las PDAs mantienen muchas características hardware y software de los ordenadores convencionales, motivo por el que en ellas pueden ser instalados Sistemas Operativos como Windows, Linux o MacOS; los móviles ligeros suelen integrar Sistemas Operativos ad hoc que han sido desarrollado expresamente para unos fines concretos. Esto se traduce en grandes diferencias para desarrollar aplicaciones entre computadores de mano y dispositivos ligeros.

En esta sección vamos a centrarnos en el desarrollo para computadores de mano. Como se describió en 2.3, para este tipo de dispositivos, hemos trabajado con la configuración Java ME CDC. En concreto, se ha elegido el perfil Personal Profile 1.1, del cual podemos recordar, se han construido Máquinas Virtuales para Windows Mobile, Linux empotrados como Zaurus y algunos móviles de Nokia. Además, resulta muy familiar para los desarrolladores Java, ya que es es prácticamente una versión 1.4 reducida de Java SE.

La relación de Java ME CDC con Java SE, amplía las posibilidades multimedia para los dispositivos, porque permite el uso y compatibilidad de la librería JMF dentro de la plataforma móvil. Sin embargo, hay que matizar la potencialidad de JMF en computadores de mano. Como hemos indicado en el capítulo anterior, la librería multimedia JMF reconoce e incorpora una serie de dispositivos, como

cámaras web o micrófonos, además de integrar una serie formatos y protocolos de acceso a la información. Esta gran capacidad viene en parte derivada por el propio Sistema Operativo que reconoce estos dispositivos o que tiene integrado unos determinados codecs. Por eso, JMF dispone de una versión *cross-all* que incorpora sólo los formatos y protocolos que es capaz de manejar sin ayuda del Sistema Operativo.

Esta versión *cross-all* sí es compatible con la configuración Java ME CDC. Afortunadamente, incluye la funcionalidad para reproducir el flujo de vídeo en tiempo real sin necesidad de otras herramientas. La librería multimedia JMF *cross-all* incorpora, por lo tanto, la recepción y generación de sesiones RTP de forma automática. Además, los formatos de vídeo en tiempo real que pueden ser decodificado en la PDA, son M-JPEG o MPEG, aunque en esta propuesta trabajaremos únicamente con el primero. En nuestro caso, el formato de compresión que encapsula el vídeo procedente del Servidor Multimedia es M-JPEG. Si el flujo multimedia navega bajo el protocolo RTP y el vídeo es codificado en M-JPEG, la librería multimedia nos permite construir en el dispositivo móvil un componente gráfico a partir de la IP y el puerto de la sesión en tiempo real.

Como indicamos en la sección anterior, el ajuste de estos parámetros se realiza mediante una solicitud a un Servicio Remoto mediante el Middleware; en la que se incluye el recurso a visualizar. En el caso de que la petición sea válida, los parámetros compuestos por la IP y el puerto son devueltos por el Servidor Multimedia cuando los clientes soliciten la visualización de una cámara. Recordemos que la integración del Middleware ZeroC Ice en dispositivos empotrados es factible gracias a su versión reducida, Embedded Ice (Ice-E), que nos permite solicitar y desplegar Servicios Remotos en Java ME.

Por tanto, en un primer paso interviene una petición al Servidor Multimedia realizada a través de Ice-E, la cual devuelve la dirección y el puerto donde recibir la sesión en tiempo real. En segundo lugar, será la librería multimedia JMF, también en su versión reducida *cross-all*, la que decodificará el flujo de vídeo y lo presentará en la pantalla de computador de mano. Esta organización queda representada visualmente en la figura 3.6.

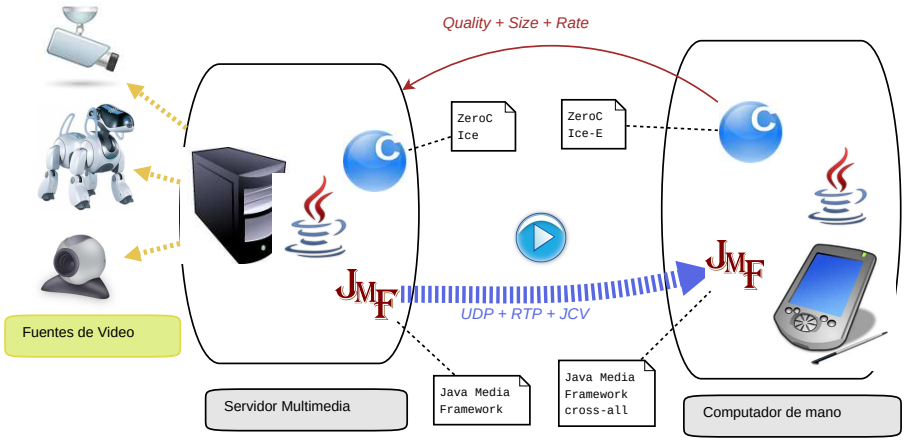

Figura 3.6: Componentes en la Visualización de Vídeo en Tiempo Real en computadores de mano

Para poner a prueba el Sistema, hemos diseñado un visualizador de hasta cuatro recursos multimedia concurrentes, como puede verse en 3.7. Estos reproductores trabajan de forma independiente entre sí, de forma que la tasa de transmisión de uno, o incluso la desconexión del mismo, no afectan al resto. En general, pueden incluirse más de cuatro reproductores multimedia, pero no tiene mucho sentido por las limitadas dimensiones de las pantallas de los computadores de mano. Los resultados derivados de la reproducción de vídeo son bastantes aceptables, teniendo en cuenta el uso de las Máquinas Virtuales y de las limitaciones computaciones de los dispositivos.

En la gráfica 3.8 se muestran la comparativa entre el número de imágenes que envía el Servidor Multimedia, respecto a las imágenes que es capaz de decodificar correctamente el cliente móvil. Después de realizar la recepción combinando los parámetros de ajuste, podemos llegar a la conclusión de que el dispositivo ofrece una barrera respecto a la combinación del número de frames por segundo y el tamaño de los mismos. Esto quiere decir que la saturación no se sitúa simplemente en función de la frecuencia de los frames, sino sobre la tasa de decodificación por segundo.

Figura 3.7: Visualización de Vídeo en Tiempo Real en computadores de mano

En la figura 3.8 puede verse que existen diferentes umbrales en función del tamaño de las imágenes. A mayor calidad y tamaño, menor umbral de saturación. Este tamaño de la imagen puede ser ajustado por el tamaño y calidad de imagen que especifique cada receptor.

Sin embargo, sí es posible establecer una constante independiente de estos parámetros, y es la tasa de KB/s de flujo de vídeo que es capaz de decodificar correctamente el computador de mano y que dependerá de cada dispositivo. En concreto, podemos observar que para el modelo de prueba, la constante se sitúa en torno a 20 KB/s.

Por lo tanto, un flujo de vídeo cuyos frames ocupen 1 KB saturarán al dispositivo en torno a los 20 fps, mientras que apenas podemos llegar a 10 fps si las imágenes superan los 2 KB. Superar este umbral conlleva una retraso en la decodificación de los fragmentos y una pérdida de los paquetes de envío, reduciendo el número de imágenes por segundo que son procesadas correctamente. Además, estos valores son constantes para el número de cámaras reproducidas

concurrentemente, es decir, la transmisión máxima de visualización para un recurso, es la mitad cuando se reproducen dos, un tercio del máximo cuando son tres, y así sucesivamente.

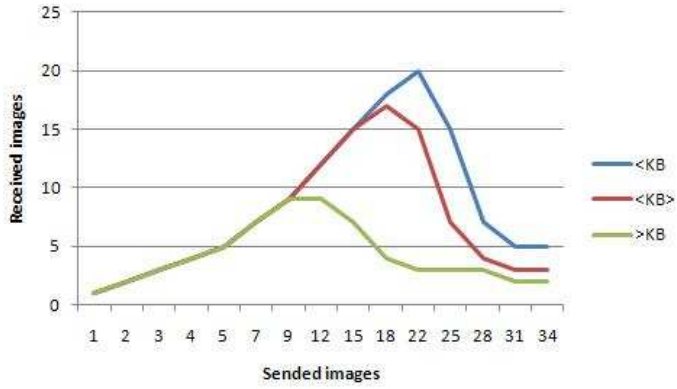

Figura 3.8: Imágenes enviadas/recibidas en Tiempo Real en computadores de mano

3.3.1.2. Receptores de Vídeo en dispositivos ligeros

El desarrollo de reproductores de vídeo en tiempo real en dispositivos ligeros, ha sido diseñado e implementado de forma muy diferente a los computadores de mano. Las diferencias se derivan de la falta de herramientas para manejar los flujos en tiempo real, y en el caso de existir, se ven afectadas por la especificación y estandarización de cada fabricante de móviles.

La plataforma de desarrollo para el desarrollo de aplicaciones para dispositivos ligeros, como teléfonos móviles, es la configuración Java ME CLDC, en concreto la versión 1.1. Sobre esta configuración, vamos a trabajar con el estándar de interfaces de usuario que integra el perfil MIDP 2.0. Como ya hemos comentado en 2.3.1, este tipo de aplicaciones móviles corren en una Máquina Virtual muy limitada, conocida como Kilo Virtual Machine (Kilo VM) por la reducción y simplicidad del núcleo de ejecución. Aunque la Kilo VM mantiene muchas características de Java a las que estamos acostumbrados, tan

potentes y sorprendentes como la sincronización y exclusión mutua de código o el manejo de hebras; también tiene muchas limitaciones, entre las que destacamos la reflectividad o el manejo limitado de hebras.

Estas limitaciones ponen en relieve las pretensiones de la visualización de vídeo en tiempo real y del esfuerzo para desarrollar una solución bajo esta plataforma. Describiremos en primer lugar, las herramientas de que disponemos a priori para manejar formatos multimedia en Java ME CLDC. La opción inicial por analogía con su plataforma hermana Java ME CDC, era usar la librería multimedia JMF, para que decodificara y presentara el flujo de vídeo (véase el apartado anterior). Sin embargo, la versión JMF *cross-all* de la librería es totalmente incompatible con las especificaciones de los teléfonos móviles, haciendo inviable tanto su uso, como su portabilidad.

La segunda herramienta multimedia que estudiamos fue Multimedia Mobile API (MMAPI), la cual ha sido presentada brevemente en 2.1 como una opción usual dentro de la literatura. Esta librería ha sido incorporada a Java ME CLDC para integrar protocolos, codecs y dar acceso a los recursos como la cámara o el micrófono. Parece, por tanto, la herramienta idónea para reproducir el flujo de vídeo desde un móvil. Sin embargo, MMAPI delega casi toda su funcionalidad en llamadas nativas al dispositivo o a terceras aplicaciones; y por tanto, no incluye los protocolos o formatos de descompresión de forma portable. De esto se deduce que cada dispositivo móvil puede reproducir unos archivos diferentes a otros, y no podemos garantizar la reproducción de los recursos de forma genérica.

La falta de herramientas que satisficiesen los requisitos de portabilidad, relegaron como única solución la construcción de una herramienta de recepción y decodificación por nosotros mismos. El desarrollo de esta herramienta incluye la ardua tarea de integrar el protocolo de tiempo real RTP y el formato M-JPEG en código Java ME. Para evitar empezar desde cero, hemos usado una librería de código abierto destinada a integrar el protocolo RTP dentro de Java y que es conocida como *jLibRTP*.

Nuestro trabajo ha consistido en traducir esta librería, para que

pueda ser ejecutada en la limitada plataforma Java ME, en función de las posibilidades que ofreciese esta última. *jLibRTP* además de ofrecer una herramienta para codificar o decodificar el protocolo RTP, integra un pequeño manejador de tiempo real que almacena los datos de forma ordenada, y que nosotros hemos definido bajo un vector circular, véase la figura 3.9.

Este tipo de filosofía es muy recomendable para el tiempo real, ya que los datos se mantienen temporalmente en memoria y a la vez, quedan eliminados automáticamente los más antiguos con la llegada de nuevos paquetes. Hemos de destacar, que alguna funcionalidad de *jLibRTP* no ha podido ser incluida dentro de Java ME CLDC, debido a las limitaciones de comunicación de la plataforma móvil. Por ejemplo, ha sido necesario eliminar la opción de realizar el envío simultáneo a un grupo de múltiples receptores, conocido como multicast, en tiempo real.

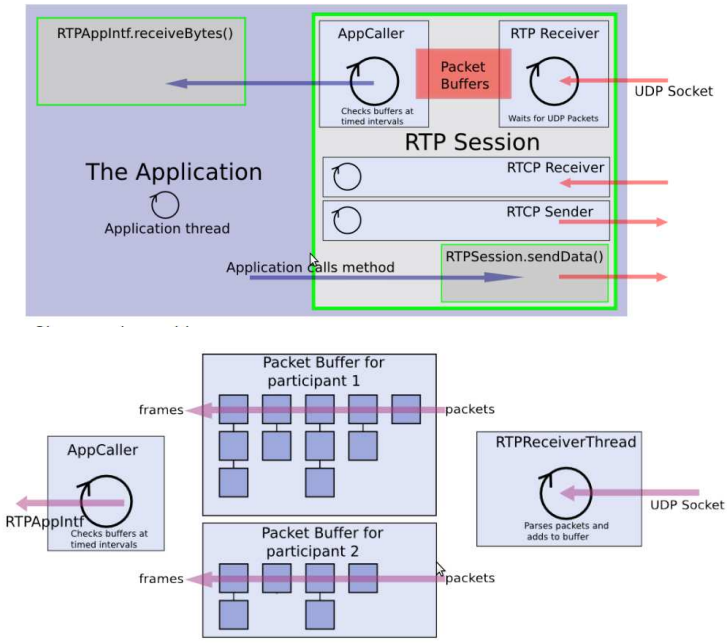

Figura 3.9: Arquitectura de jLibRTP

Una vez desarrollada y testada la recepción bajo el protocolo RTP en los teléfonos móviles, ha sido necesario incluir una segunda etapa que agrupase y mostrase las imágenes en el dispositivo móvil. Es decir, esta etapa se encarga de decodificar los paquetes en formato M-JPEG y agruparlos hasta formar una imagen completa, la cual debe ser descomprimida y presentada en la pantalla del dispositivo. Como se ha descrito en la sección 2.6, el formato M-JPEG transmite cada imagen en varios paquetes independientes.

El motivo de la fragmentación viene determinado porque el tamaño máximo recomendable de un paquete UDP no debe superar los 1024 bytes. Sin embargo, es más que probable que los frames superen este tamaño máximo aconsejable. Por lo tanto, es necesario partir cada frame en varios paquetes, cada uno codificado según el formato M-JPEG. Este formato añade, entre otros parámetros, una cabecera que determina el frame que se envía, y el fragmento que ocupa dentro de ese frame.

Bajo esta filosofía de envío es aconsejable, aunque no imprescindible, indicar qué paquete ofrece los últimos bytes del frame, de forma que el receptor pueda agruparlo junto con los anteriores. Esta señalización es indicada en el formato M-JPEG activando el bit de marca opcional que se encuentra en el paquete RTP. Para detectar los casos de pérdida, M-JPEG incluye en la cabecera la numeración, el tamaño y el desplazamiento de cada paquete dentro de la imagen. En caso de detectar la pérdida de algún paquete, la aplicación reinicia el buffer hasta que una secuencia de paquetes se complete correctamente y genere un frame completo.

Si después de todo, la recepción de los datos ha sido correcta, el cliente podrá entonces:

- Recuperar el frame completo. Si la secuencia de los paquetes es correcta y está ordenada, cuando llegue la marca de fin de frame, debemos recuperar el frame correcto.

- Decodificar la imagen. Los frames completos son incluidos en un buffer circular para que la imagen real pueda ser descomprimida

a partir del formato JPEG. Siguiendo con las limitaciones de la plataforma Java ME CLDC, hay que decir que el descompresor de imágenes para JPEG no siempre puede encontrarse en los dispositivos móviles, aunque bien es cierto que muchos teléfonos móviles sí lo incluyen. Nuestra aplicación está diseñada para detectar la existencia de este decodificador en tiempo de ejecución. En caso de no existir, hemos incluido la herramienta GPL creada por [Dersch, 2009] y que nos permite descomprimir imágenes JPEG en dispositivos ligeros Java.

- Mostrar la imagen en la pantalla del dispositivo. Como la plataforma y el perfil de Java ME que hemos utilizado es MIDP, el componente gráfico resultante de todo este proceso debe ser compatible con esta interfaz de usuario. Para ello, hemos usado uno de los componentes básicos: *javax.microedition.lcdui.Canvas* que representa un lienzo sobre el que pintar gráficos. Cuando el decodificador termina el proceso de descompresión, la matriz de colores básica (rojo, verde, azul) es impresa sobre este lienzo, dando como resultado la imagen que ha transmitido el flujo de vídeo.

Estas tres fases deben realizarse de forma independiente a la recepción de nuevos paquetes para no interferir en la llegada de otros frames. Para ello, se generan hebras independientes que realizan la recepción, la decodificación y la presentación de forma segura, es decir respetando la exclusión mutua de los datos. Véase la figura 3.10.

Al igual que la aplicación para computadores de mano, el paso previo a obtener el flujo de vídeo se produce mediante la solicitud al Servidor Multimedia. Afortunadamente la librería Ice-E es compatible con las configuraciones Java ME CDC y CLDC, de forma que la petición del Servicio Remoto es idéntica a la de los computadores de mano.

Finalmente, los resultados de la recepción de vídeo nos revelan que el comportamiento es más que aceptable para un dispositivo ligero, como un teléfono móvil. A su vez, el reproductor de vídeo en tiempo real ha sido testado en dispositivos reales, no sólo en emuladores. En la tabla 3.11 se muestran los resultado obtenidos en un Nokia N81, un

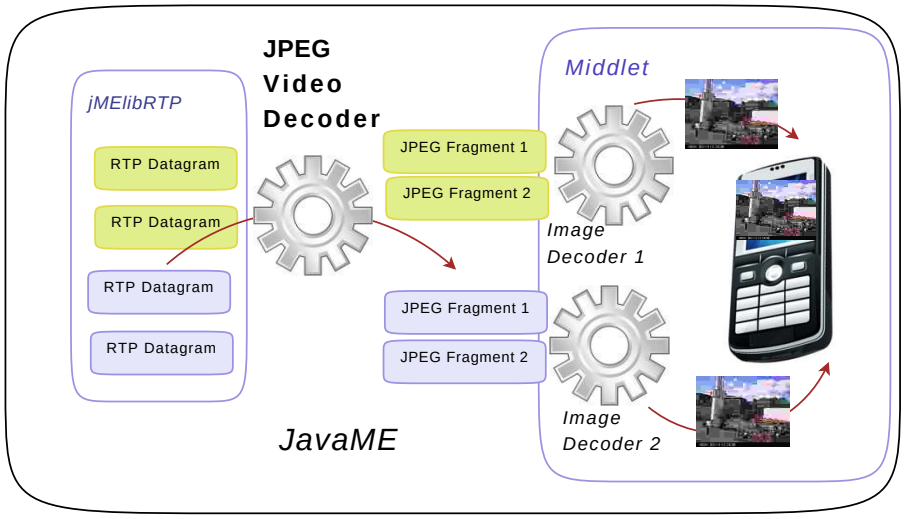

Figura 3.10: Componentes del Visualizador de Vídeo en Tiempo Real en dispositivos ligeros

dispositivo de gama media con un procesador a 360 MHz, el cual ha recibido un flujo de vídeo en tiempo real con imágenes de diferentes cámaras Aibo, ip o web. Este dispositivo concreto ha situado su tasa de saturación entorno a 5 KB/s, lo que se corresponde 5 imágenes por segundo de 1KB, creando un vídeo constante y eficiente para flujos de vídeo que respeten este umbral. Esta frecuencia óptima se reduce a la mitad si el tamaño de las imágenes crece al doble, porque por ejemplo, se mejore la calidad o se amplifique el tamaño de la imagen. Estos resultados pueden verse gráficamente en 3.12.

Además del comportamiento mencionado, es importante enfatizar los resultados no funcionales que se han conseguido y que se pueden resumir en desarrollar el proceso de reproducción íntegramente en la Máquina Virtual KVM. Este requisito ha supuesto un esfuerzo importante, ya que no se han usado paquetes adicionales, como MMAPI, pudiendo construir una solución portable a la plataforma con la que hemos trabajado: Java ME, configuración CLDC 1.1 y perfil MIDP 2.0.

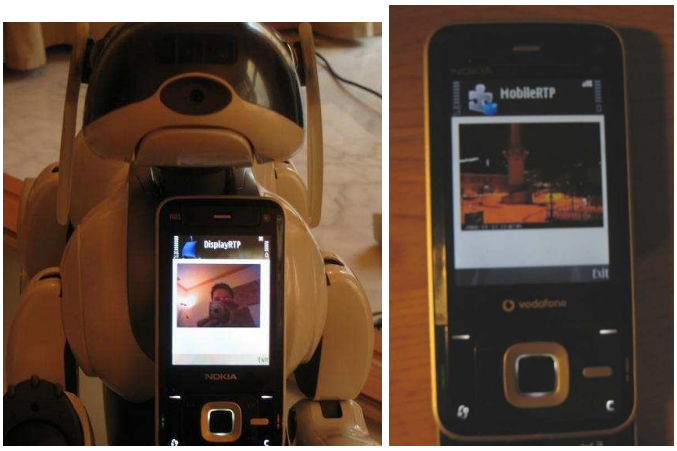

Figura 3.11: Imágenes enviadas/recibidas en Tiempo Real en dispositivos ligeros

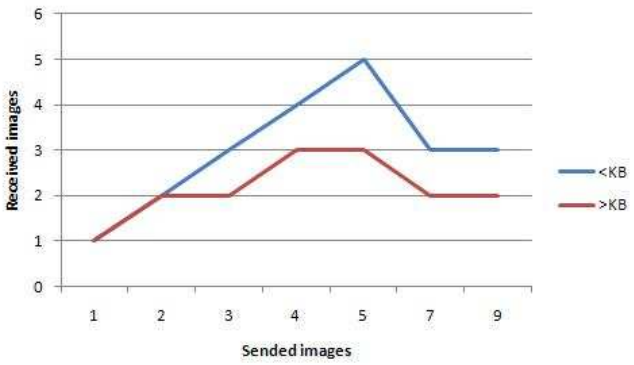

Figura 3.12: Resultados de la Visualización de Vídeo en Tiempo Real en dispositivos ligeros

3.3.2. Receptores de Audio

Aunque las cámaras suelen ser los recursos multimedia más comunes que se encuentran instalados en los edificios y en el entorno, también es habitual encontrar micrófonos empotrados que nos permitan acceder al sonido ambiente de una determinada zona. Además, muchos equipos de sobremesa cuentan con la incorporación de micrófonos integrados, o bien las mismas cámaras web suelen incluirlas. En esta sección vamos a detallar cómo podemos recibir flujos de audio desde estas fuentes de sonido en tiempo real usando dispositivos móviles, tanto computadores de mano de mayor potencia como teléfonos ligeros.

3.3.2.1. Receptores de Audio en computadores de mano

Al igual que en el apartado sobre vídeo 3.3.1.1, para aprovechar la potencialidad de los computadores de mano y de una Máquina Virtual más eficiente, hemos construido una aplicación específica para estos dispositivos. Como hemos detallado previamente, la librería multimedia JMF es compatible con la configuración de Java ME para computadores de mano. En concreto, su versión *cross-all* permite manejar varios formatos y protocolos multimedia en tiempo real. Gracias a esta herramienta, es posible recibir un flujo multimedia, pudiendo automatizar la recepción bajo el protocolo RTP.

Por otra parte, la versión *cross-all* sólo permite manejar algunos formatos que están incluidos implícitamente en la misma. Afortunadamente, el estándar de sonido G.711, que había sido propuesto en la sección 2.6, pertenece a la colección de codecs portables. Gracias a estas facilidades, parecía que la reproducción del audio en tiempo real iba ser delegada totalmente en la librería JMF.

Sin embargo, nos encontramos nuevamente ante las limitaciones y problemas de la Máquina Virtual cuando trabajamos en dispositivos móviles. En este caso, la librería que controla y maneja el sonido de los computadores de mano no está conectada con el Sistema Operativo, siendo inviable la reproducción de las líneas de audio desde la Máquina

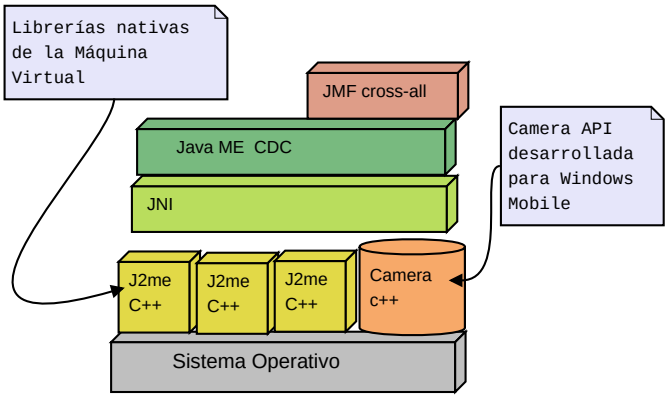

Figura 3.13: Arquitectura JNI

Virtual. La forma para solventar estas limitaciones fue introducida en la presentación de la plataformas Java ME 2.3.1. La configuración Java ME CDC permite solventar las limitaciones de la Máquina Virtual CDC, desarrollando librerías nativas sobre el Sistema Operativo y efectuando llamadas a las mismas de la aplicación. De estos propósitos se encarga la interfaz JNI, que permite la comunicación de librerías en lenguaje nativo, como C++ o C, y las Máquinas Virtuales Java (véase la figura 3.13).

En nuestro caso, la solución pasa, por tanto, en construir una librería en C++ que nos permita escribir datos sonido en el altavoz. Como la etapa de decodificación es realizada por la librería multimedia JMF, la librería sólo debe encargarse de escribir el audio por los altavoces del dispositivo en formato lineal , esto es, sin codificar o bruto. Cuando el flujo de sonido comienza, la aplicación móvil inicializa la línea de sonido según la frecuencia, tamaño de las muestras y canales del mismo. Esta línea permanecerá abierta mientras continúe la transmisión de sonido, ya que los datos se recibirán en pequeños fragmentos que se escribirán secuencialmente en el altavoz. El mantenimiento temporal de estos fragmentos se realiza con una estructura circular. Finalmente cuando el usuario finalice la transmisión, la línea de sonido se cerrará.

Para que la librería multimedia JMF llame automáticamente

a nuestra librería, es necesario redirigir los formatos de sonido lineales hacia nuestra nueva librería en C++. El mecanismo para integrar y hacer visibles "nuestros altavoces", pasa por implementar un *Render*, en nuestro caso lo hemos llamado *LinealRender*, que efectúe las llamadas a la librería nativa. Los Render (o interpretadores) simbolizan a los sumideros de datos multimedia en la librería JMF. Finalmente, es necesario registrar nuestro Render como el encargado de reproducir el audio lineal. Con este último paso, cuando se recibe un flujo de sonido en tiempo real, la reproducción del mismo se realiza de forma automática.

La organización de todos los componentes involucrados puede verse gráficamente en la figura 3.14:

- El flujo de audio en tiempo real es recibido por JMF, la cual se encarga de decodificar los paquetes y obtener el fragmento de sonido lineal.
- La clase *LinealRender* recibe el fragmento de audio y lo envía a la nueva librería nativa usando la interfaz JNI.
- El fragmento lineal es escrito en los altavoces, y será reproducido cuando los fragmentos de sonido previos hayan finalizado.

Los resultados efectuados para evaluar la transmisión de sonido se han basado en varias mediciones. La primera es el tiempo que se emplea en transmisión y recepción de paquetes bajo el protocolo UDP en la capa de transporte, notado como tR. La segunda, hace referencia al tiempo de decodificación del protocolo RTP y del formato Mu-law, para obtener las muestras lineales, definido como tD. Finalmente, se ha medido el tiempo entre la disposición del sonido lineal y la escritura final en la línea de sonido, al que hemos llamado tE. El computador de mano con el que se han realizado las pruebas integra un procesador a 624 MHz.

El tiempo de transmisión de paquetes y la recepción de los mismos en dispositivos móviles ha sido medido con paquetes de longitud 1024 bytes que han sido transportados bajo una transmisión

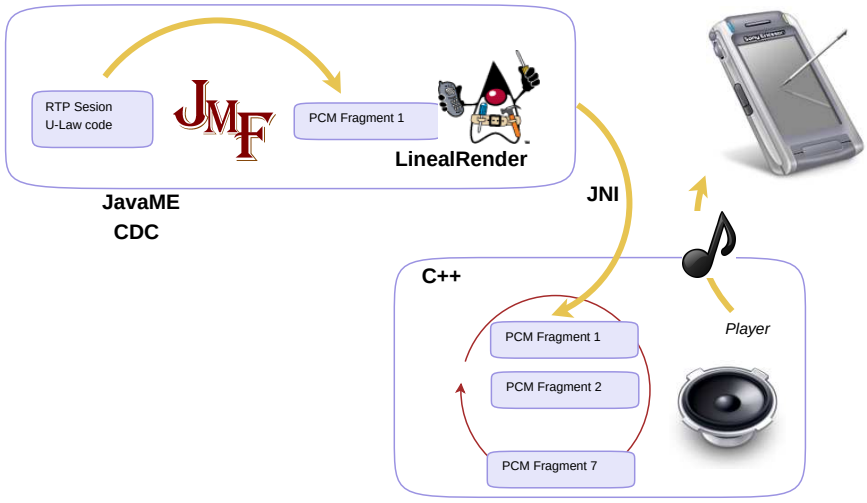

Figura 3.14: Arquitectura de la Reproducción de Audio en Tiempo Real en computadores de mano

no orientada a conexión como UDP. Este tamaño es el que usa el formato G.711 para el envío de sonido en tiempo real. Para efectuar una correcta medición entre los retardos, el computador de mano ha sido sincronizado con un equipo de sobremesa usando la herramienta *ActiveSync*. Esta sincronización, entre otras funcionalidades, incluye la igualdad del tiempo de los relojes de cada la máquina de sobremesa y móvil. Este tiempo tR para nuestro dispositivo móvil se ha establecido en una media de 230 ms.

El segundo tiempo que se ha medido cuantifica la fase de descompresión de bytes que han sido codificados bajo el protocolo RTP y decodificados bajo el formato Mu-law. El tiempo para la recuperación de los datos en RTP tiende a cero, ya que éste apenas añade una pequeña cabecera a los datos fuente. El mayor coste de esta etapa radica en la descompresión Mu-law de los paquetes UDP y que producirán sonido lineal. Recordemos, que la notación de los algoritmos Mu-law y A-law era logarítmica (véase la sección 2.6). El tiempo de ambos procesos tD con el tamaño de paquetes de 1024 bytes se ha situado en una media 53 ms.

Finalmente, una vez obtenidos los datos lineales, es necesario medir el tiempo que transcurre desde que están disponibles hasta que son reproducidos en el altavoz. Los datos lineales que se obtienen del proceso de decodificación de un único paquete establecen 960 muestras a 16 bits por medición en mono canal. Por tanto, la llegada de estos paquetes se realiza a una frecuencia de 60 ms. En esta etapa se realiza la llamada a la librería nativa que hemos construido para resolver las carencias del audio. El tiempo de escritura tE de un fragmento de sonido en la misma se ha estimado en 20 ms, lo que incluye la llamada a las funciones de la librería y la escritura en la línea de audio. Es importante destacar que gracias a que el tiempo de escritura es inferior a la frecuencia de llegada de los paquetes, no se acumulan silencios entre las tramas. Si por el contrario, el tiempo de escritura en la línea de sonido superase los 60 ms, no daría tiempo a consumir los paquetes respecto su producción, con lo que se acumularían los fragmentos de sonido en cada llamada provocando retardos y pérdidas de audio.

Si sumamos todos los tiempos medios obtendremos el retardo total medio que sufren los paquetes, desde que son recibidos en el dispositivos hasta que son escritos en la línea de sonido:

$$tD + tR + tE = 230ms + 63ms + 20ms = 313ms$$

Es muy importante destacar que el retardo real desde que se recoge una muestra de sonido en un micrófono hasta que el receptor humano oye el sonido por el altavoz, difiere de los tiempos medidos en este apartado. En concreto, estos tiempos miden el retardo en el receptor móvil, pero análogamente habría que añadir los tiempos que acumula en emisor en recoger, codificar y transmitir las muestras. En la siguiente sección, cuando los dispositivos móviles actúen como emisores, determinaremos estos tiempos. Por otra parte, la computación puede medir el tiempo de escritura en la línea de sonido del dispositivo, pero la reproducción real puede incluir otros retardos derivados del control del micrófono por el Sistema Operativo.

Respecto los resultados no funcionales asociados a la portabilidad, debemos destacar qué partes de la propuestas son trasladables y

cuáles dependen del Sistema Operativo, ya que en esta solución se ha conseguido una portabilidad parcial. La recepción y decodificación de las muestras se realiza dentro de la Máquina Virtual usando la librería multimedia JMF, con lo que el proceso es totalmente portable a otras configuraciones CDC. Sin embargo, la librería nativa (.dll) para Windows Mobile no es compatible con otros Sistemas operativos. Por ejemplo, para ampliar la portabilidad a un dispositivo Linux empotrado, debemos desarrollar una nueva librería, por ejemplo (.so), que integre la funcionalidad de la interfaz JNI, en nuestro caso, la escritura de sonido lineal . El resto, incluyendo la interfaz JNI que hemos definido y el uso de JMF, son componentes totalmente portables.

3.3.2.2. Receptores de Audio en dispositivos ligeros

En este apartado se describen los desarrollos para integrar la reproducción de sonido en tiempo real sobre dispositivos ligeros. Esta aplicación esta destinada a cubrir el amplio abanico de usuarios con teléfonos móviles para que puedan conectarse remotamente a micrófonos ambientales.

Como hemos comentado en varias ocasiones, la Máquina Virtual KVM que se incluye en los dispositivos ligeros dispone de unas prestaciones muy reducidas. Además, si pretendemos realizar un desarrollo que no dependa de un modelo concreto de dispositivo móvil, debemos hacer uso de herramientas portables para la configuración Java ME CLDC, sin integrar librerías optativas que sólo algunos teléfonos incluyen. Recordemos que éste era el caso de Multimedia Mobile API, que permite la reproducción en tiempo real de algunos recursos, pero que delega la funcionalidad sobre otras aplicaciones nativas y que se encuentra disponible en algunos dispositivos de gama alta de Nokia.

Por tanto, si deseamos garantizar una aplicación genérica, es imprescindible basar la solución en unos requisitos mínimos de reproducción de sonido. En nuestro caso, el requisito básico viene determinado por la reproducción de sonido en formato de salida lineal o bruto. Para poder escribir en la línea de sonido del dispositivo, la

configuración Java ME CLDC permite volcar un flujo de audio a la salida básica del micrófono.

Este flujo de audio puede estar codificado en varios formatos, tales como MP3, en función de la capacidad de decodificación del dispositivo. Sin embargo, para relajar las restricciones sobre el formato de compresión, hemos realizado la descompresión en una fase previa, delegando al altavoz de esta responsabilidad y obteniendo directamente los datos en formato lineal. El formato para escribir los datos en formato lineal en Java ME CLDC, es conocido como WAV. Este formato añade a los datos del sonido una pequeña cabecera donde se indican la frecuencia de muestras por segundo, si es mono o estéreo o el tamaño de cada muestra y el número de muestras totales que componen el audio. Es importante enfatizar que la Máquina Virtual espera a recibir todos los bytes especificados en el archivo y que componen el sonido, para, posteriormente, escribir los datos en el micrófono.

Como vemos, esta funcionalidad tan limitada se aleja de nuestras pretensiones de reproducir el sonido en tiempo real bajo el formato G.711. Para resolver estas limitaciones, vamos a hacer uso, en primer lugar, de la librería para la decodificación del tiempo real *jLibRTP* que ya se introdujo en la sección 3.3.1.2 y que ha sido trasladada satisfactoriamente a Java ME CLDC. Esta herramienta nos permite decodificar los paquetes RTP que lleguen al dispositivo móvil y almacenarlos temporalmente en un buffer circular.

Los datos recuperados en tiempo real están codificados en formato Mu-law, por lo cual, hemos de decodificarlos y obtener los bytes en formato lineal. Afortunadamente, este formato es libre por lo que hemos encontrado numerosos decodificadores para Java. Gracias a ellos, hemos podido implementar el codec en Java ME dentro de la propia aplicación. Esta compresión se basa en en el uso de una matriz para que hace corresponder una pareja de bytes consecutivos con un único byte, de forma que se reducen a la mitad el número de muestras capturadas. En el proceso inverso, la descompresión recoge una muestra de bytes en formato Mu-law y usa la matriz inversa para obtener el doble de muestras de sonido. Para no trasladar los retardos

de este proceso a la etapa de recepción, de la decodificación se encarga una hebra independiente que transforma los datos del buffer circular recogidos por jLibRTP y los convierte a sonido lineal.

Una vez obtenido el sonido sin codificación, tenemos que escribir los fragmentos en el micrófono del teléfono. Para ello, hemos creado un *stream* en Java ME que introduce como primeros bytes la cabecera WAV, necesaria para que la Máquina Virtual conozca las características del fragmento de sonido (tamaño de las muestras, número de canales y muestras, etc.); y posteriormente escriba los datos lineales del sonido en el altavoz. Como ya hemos comentado, hasta que todos los datos no son escritos, el sonido no comienza a reproducirse. Para reducir esta espera es conveniente reducir también el tamaño de las muestras. Sin embargo, en contraposición a esta técnica, los teléfonos ligeros padecen un retardo entre la escritura de un *stream* y el siguiente, ya que tanto la inicialización como la finalización de cada reproducción consume un cierto tiempo. Por lo tanto, si los fragmentos son demasiado pequeños, la espera del sonido disminuye en tiempo, pero aumentan los silencios entre los fragmentos.

Por tanto, los resultados con un sólo reproductor que se encargue de la reproducción de las muestras, incluyen estos silencios y retardos. Para solventar el problema de los silencios, se han incluido varios reproductores en el proceso, de forma que cuando uno escribe un fragmento de sonido, el siguiente ya está preparado para escribir los siguientes. De esta forma un reproductor avisa al siguiente instantes antes de finalizar, minimizando el silencio existente entre la finalización del primero y el comienzo de éste. Evidentemente, este proceso conlleva una sincronización extra para que el resultado del sonido sea homogéneo y no se produzcan solapamientos o sonidos. Además, para una mejor cooperación entre los reproductores, éstos configuran una pequeña relación circular, y acceden en exclusión mutua a los datos lineales que han sido producidos por el proceso de decodificación. El número de muestras que cada reproductor puede escribir también ha sido parametrizado, permitiendo agrupar varios fragmentos de audio y escribirlos concurrentemente, produciendo un sonido más limpio.

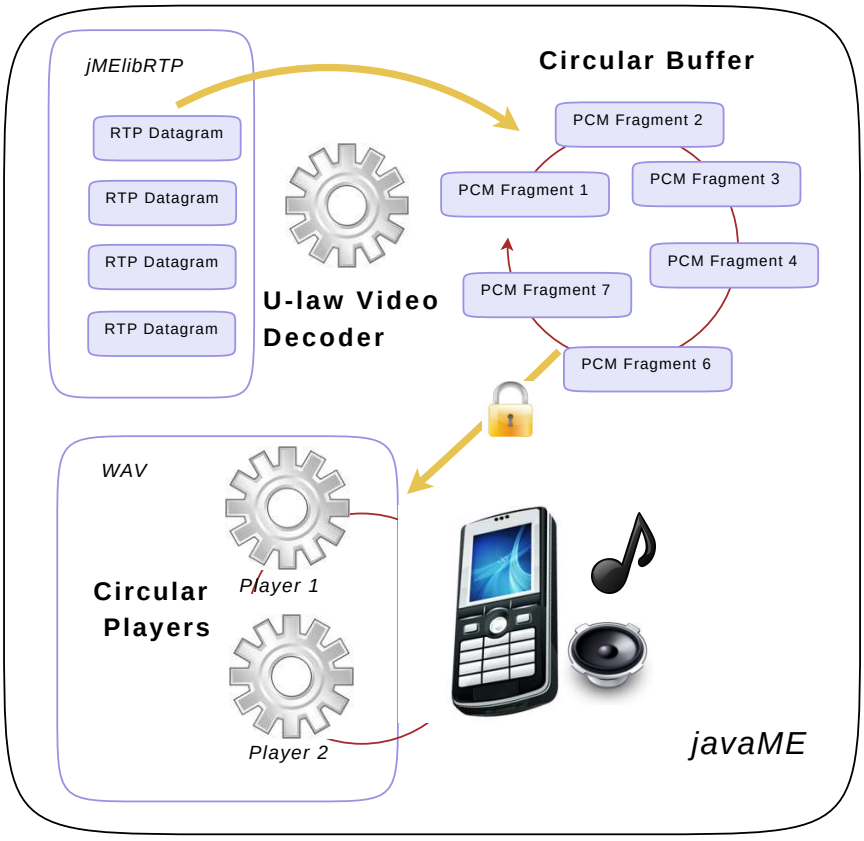

Figura 3.15: Arquitectura de la Reproducción de Audio en Tiempo Real en dispositivos ligeros

Este proceso completo puede verse en la figura 3.15.

Los resultados son bastante aceptables para ser producidos por un teléfono móvil y un proceso íntegro sobre la limitada Máquina Virtual KVM (véase figura 3.16), aunque evidentemente, el comportamiento es más pobre que el derivado de los computadores de mano, donde hacíamos uso de una librería nativa. El modelo concreto con el que se han estimado los datos, es un Nokia de gama media-alta a 360 Mhz. Este dispositivo ofrece unos silencios entorno a 325 ms entre la finalización de un reproductor y el siguiente. Este silencio obtenido es independiente del número de muestras que escribe cada reproductor

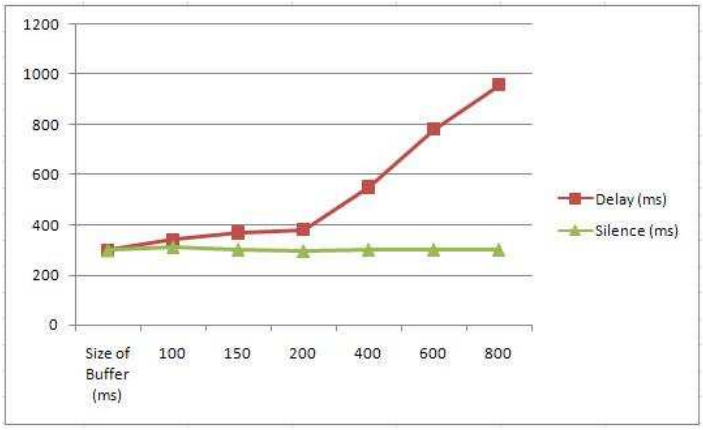

Figura 3.16: Resultados de la Reproducción de Audio en Tiempo Real en dispositivos ligeros

en el altavoz, ya que es intrínseco de la apertura y cierre del mismo. Sin embargo, el aumento del tamaño de escritura en bloque provoca un sonido más limpio y menos fragmentado, aunque como ya hemos comentado, también incrementa los retardos derivados de la espera a que el *stream* de audio finalice.

3.4. Dispositivos Móviles como Generadores Multimedia en Tiempo Real

En la sección anterior hemos descrito los desarrollos en los dispositivos móviles que nos permiten mostrar los flujos multimedia generados desde las fuentes ambientales. Con este Sistema, los usuarios pueden conectarse a los recursos multimedia de los edificios y estar comunicados con el entorno desde cualquier ubicación. En esta sección, descubrimos la potencialidad de situar a los dispositivos móviles como emisores y no sólo como receptores de información de audio o vídeo.

De esta forma, pretendemos desarrollar uno de los objetivos más atractivos de la tesis: la generación de flujos multimedia desde los dispositivos móviles. Esta funcionalidad otorga a los sistemas

movilizados infinidad de posibilidades, entre ellas las conversaciones por red o incluso transmisiones de vídeo conferencias. Así, es posible interconectar a los usuarios no sólo con el entorno, sino con otros usuarios usando sus teléfonos móviles y las redes inalámbricas. Además, las transmisiones desde dispositivos móviles pueden ser recibidas no sólo por los operadores humanos. Por ejemplo, el Entorno Inteligente puede solicitar la captura del vídeo o audio de una zona de interés donde se encuentren los usuarios para almacenarlo o procesarlo.

En definitiva, ésta es una característica novedosa que pocos sistemas comerciales incluyen. A continuación detallamos cómo generar los flujos multimedia desde las Máquinas Virtuales y los resultados y problemas que se han encontrado.

3.4.1. Generadores de Vídeo

El primer apartado sobre generación multimedia en tiempo real va a centrarse sobre la transmisión de vídeo desde dispositivos móviles. Aquí describiremos los problemas asociados a la captura de imágenes desde los mismos, y cómo debemos encapsularlas para transmitirlas en tiempo real. Una de las características que nos gustaría incorporar es la generación del flujo siguiendo los mismos protocolos y formatos usados en las cámaras estáticas. El motivo es mantener la transparencia de las fuentes de vídeo, ya conseguida en el el Servidor Multimedia. En éste, la transmisión del vídeo era independiente de la naturaleza de la cámara lo generaba (IP, Aibo o web), y esta filosofía debería trasladarse a la solución móvil. Con ello, si mantenemos los mismos formatos y protocolos, un cliente podría visualizar desde su ordenador de mano cualquier cámara de vídeo, ya sea estática o móvil, de forma transparente y homogénea.

3.4.1.1. Generadores de Vídeo en computadores de mano

En este apartado vamos a presentar los resultados y desarrollos realizados sobre computadores de mano, tales como PDAs. Para

recordar el contexto de trabajo sobre la creación de aplicaciones móviles en los computadores de mano, hemos elegido como la plataforma de desarrollo a Java ME en su configuración CDC. Esta versión de Java ME, nos permite manejar algunos formatos multimedia usando una versión ligera de la librería Java Media Framework (JMF). Esta herramienta, aunque ligera, incluía la recepción bajo el protocolo RTP y algunos codecs integrados de forma implícita, y que podían ser ejecutados en la Máquina Virtual.

Para este nuevo propósito de emisión, el flujo multimedia debe ser generado desde el propio dispositivo, por lo que en lugar de evaluar las características sobre recepción, hemos de enmarcar aquellas que nos permitan la transmisión de vídeo. El primer punto positivo de la librería JMF, es que integra la codificación del protocolo RTP, esto es, permite la generación de sesiones en tiempo real y además, el encapsulamiento de los datos se realiza automáticamente.

En referencia a la codificación de la fuente de vídeo, la librería multimedia incluye el formato M-JPEG para la transmisión de imágenes como codificador. Esto nos va permitir construir un flujo de vídeo a partir de una secuencia de imágenes. Es importante que este formato sea el mismo que se usó en el Servidor Multimedia, de forma que las fuentes ambientales y móviles posean las mismas características. Sin embargo, el uso de la librería JMF en su versión *cross-all* elimina algunas las características y funcionalidades que sí integraba el Servidor Multimedia.

En concreto, no es posible especificar un tamaño de vídeo diferente al que se recoge en la fuente, eso significa que la transmisión no puede ajustarse en tamaño al que solicitan los clientes. Tampoco es posible reajustar la calidad de la imagen y reducir el ancho de banda del flujo de vídeo. Sin embargo, estas carencias en la calidad del servicio no son tan críticas, porque como veremos en los resultados, el tamaño y la calidad de la imagen no son muy exigentes, ya que son capturadas desde la cámara de un computador de mano.

Sin embargo, la limitación de la plataforma Java ME CDC y de la librería JMF radica en el acceso a la cámara del dispositivo.

Si la Máquina Virtual no ofrecía una conexión con los altavoces del computador de mano, mucho menos permite manejar la cámara del mismo. Para resolver esta carencia, hemos utilizado el mismo método que en la sección 3.3.2.1, creando una comunicación entre la Máquina Virtual y una librería nativa que opere con la cámara.

En primer lugar hemos de desarrollar una librería en lenguaje nativo C++ que permita inicializar el dispositivo, capturar una imagen del mismo y cerrarlo de forma apropiada. Esta librería depende del Sistema Operativo. En el caso de nuestros dispositivos de prueba, la librería ha sido desarrollada para Windows Mobile. El soporte para la cámara se ha realizado usando las clases que incorpora Windows Mobile para capturar y almacenar las imágenes de la cámara: *ICaptureGraphBuilder2*, *IBaseFilter* e *IImageSinkFilter*. El uso de las mismas tiene algunas limitaciones intrínsecas; por ejemplo, no es posible especificar el tamaño de captura, o la imagen de captura es almacenada en la memoria física del dispositivo, y no en forma dinámica bajo un vector de bytes.

Para que la Máquina Virtual pueda llamar a nuestra librería nativa en C++ y recoger las imágenes de la cámara, es necesario desarrollar una interfaz JNI que la conecte con la Máquina Virtual. Esta interfaz define tres operaciones: inicialización, captura y cierre del dispositivo de captura. Finalmente, una vez solicitada la captura de una nueva imagen, el resultado se vuelca sobre un archivo en formato JPEG.

La última tarea consiste en recoger los bytes que configuran la imagen y enviarlos en tiempo real. Este proceso debe estar integrado usando la librería JMF. Para ello, y de forma análoga a la incorporación de las cámaras estáticas en el Servidor Multimedia, hemos desarrollado una clase *PullBufferStream* de JMF, que inicializa la cámara móvil, y almacena las imágenes. Este tipo de flujo de vídeo nos permite que la captura de imágenes sea realizada asíncronamente respecto a la codificación y envío, con lo que se eliminan las esperas innecesarias entre la recogida y la transmisión en tiempo real. El proceso completo puede verse en la figura 3.17.

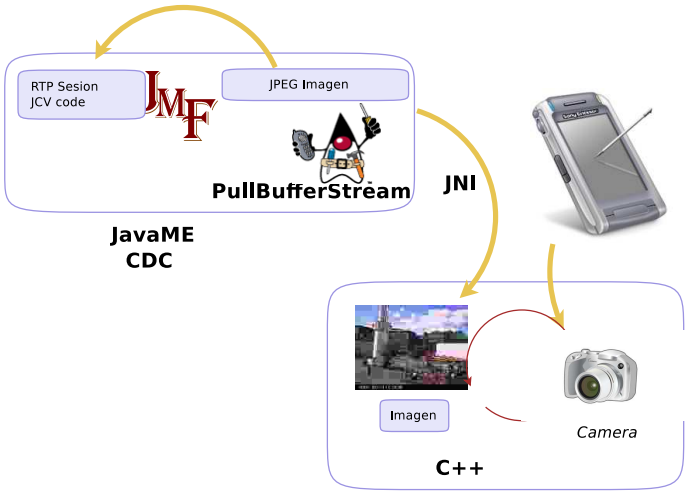

Figura 3.17: Arquitectura de la Generación de Vídeo en Tiempo Real en computadores de mano

Finalmente, se ha testado la solución en un dispositivo real. Los resultados, en el contexto de nuestro dispositivo Windows Mobile a 623 Mhz, ofrecen un flujo de vídeo con una tasa de transmisión máxima de 3.14 imágenes por segundo, véase la figura 3.2. Las características del flujo de vídeo tiene unas dimensiones de 320x240, y un tamaño entorno a los 3KB, generando un flujo de vídeo de hasta 10 KB/s. Estos resultados son más que aceptables para el uso de Máquinas Virtuales que ofrecen una visualización homogénea desde la cámara móvil de forma remota.

Además, la frecuencia podría mejorarse si se redujese el proceso de recogida de muestras. El cuello de botella de la transmisión, se establece en la captura de la imagen, que consume entorno a 300 ms para generar una muestra. Por otra parte, debemos destacar enfáticamente que la rapidez y calidad de las imágenes depende sobre manera del dispositivo Windows Mobile y de la cámara integrada del modelo. Por ejemplo, se han encontrando resultados muy pobres en otros dispositivos antiguos que no disponen de auto enfoque y que ofrecen un tiempo de captura alto y las imágenes recogidas son repetitivas, esto es, no varían durante varias muestras.

Tabla 3.2: Ejemplo del Vídeo generado en Tiempo Real desde computadores de mano

Frecuencia Máxima de Capturas en computadores de mano
3.14 fps

3.4.1.2. Generadores de Vídeo en dispositivos ligeros

No queríamos dejar a los dispositivos ligeros fuera de las transmisiones multimedia en tiempo real. En este apartado vamos a detallar las posibilidades y limitaciones, para producir flujos multimedia de vídeo desde teléfonos.

Como ya hemos comentado en anteriores secciones, la librería multimedia JMF no es compatible con el entorno para teléfonos móviles. Para resolver esta carencia, introdujimos la librería *jLibRTP*, la cuál había sido adaptada a la configuración Java ME CLDC para que pudiese ser ejecutada en dispositivos ligeros. Esta librería nos permite, no sólo recuperar eficientemente paquetes RTP, si no también generar una sesión en tiempo real desde el propio dispositivo. Por tanto para nuestros propósitos, *jLibRTP* será la encargada de encapsular el vídeo en tiempo real.

Los paquetes en tiempo real contendrán las imágenes que han sido capturadas desde los dispositivos móviles, en nuestro caso bajo el formato M-JPEG. Para poder codificar las imágenes en este formato, ha sido necesario implementar el codificador en Java ME. Este codificador

fue elegido porque no consume grandes recursos de tiempo y cómputo en la codificación y decodificación, por lo que afortunadamente, el proceso no es complicado.

En primer lugar, la imagen original es dividida en fragmentos que puedan ser enviados en un datagrama, de forma que se generan uno o varios paquetes por cada imagen. A cada uno de estos paquetes es necesario incluir la cabecera del formato M-JPEG, la cual indica el tamaño de la imagen original, el número de secuencia y el offset del paquete respecto a la imagen total. Finalmente, es necesario especificar el último paquete de cada imagen, activando el bit de marca y que indica a los receptores que ya puede leerse el frame completo.

Debemos destacar que todo el proceso anterior es perfectamente portable y aplicable a otra Máquina Virtual con la configuración Java ME CLDC, ya que hemos implementado nosotros mismos estos resultados sin dependencias del Sistema Operativo. Sin embargo, no tenemos la seguridad para poder garantizar que el proceso de recogidas de imágenes pueda ser realizado en todos los teléfonos móviles. El problema estriba en que la captura desde la cámara depende de las capacidades de la librería Multimedia Mobile API. Como hemos comentado en varias ocasiones, estas características varían drásticamente de un dispositivo a otro. En la práctica, los móviles Nokia de última generación y algún otro fabricante integran en la Máquina Virtual el componente para la captura de imágenes. En caso de no disponer del mismo, las limitaciones en cuanto captura de imágenes son insalvables, ya que dependemos de Máquinas Virtuales cerradas (KVM).

La captura de imágenes, por tanto, depende de MMAPI. Afortunadamente, en nuestro modelo Nokia N81, podemos establecer tanto el tamaño de la imagen como la calidad de la misma, aunque debemos especificar que no toda configuración es aceptable. En general, son válidos los tamaños de imagen 160x120 o 320x240. Incluso es posible establecer el tipo de codificación que tendrá la imagen resultante, por ejemplo: bmp, png o jpeg; pero también en este caso, no todos los dispositivos incluyen todos los formatos. El resultado de la captura es un vector de bytes en el formato que se haya especificado. En

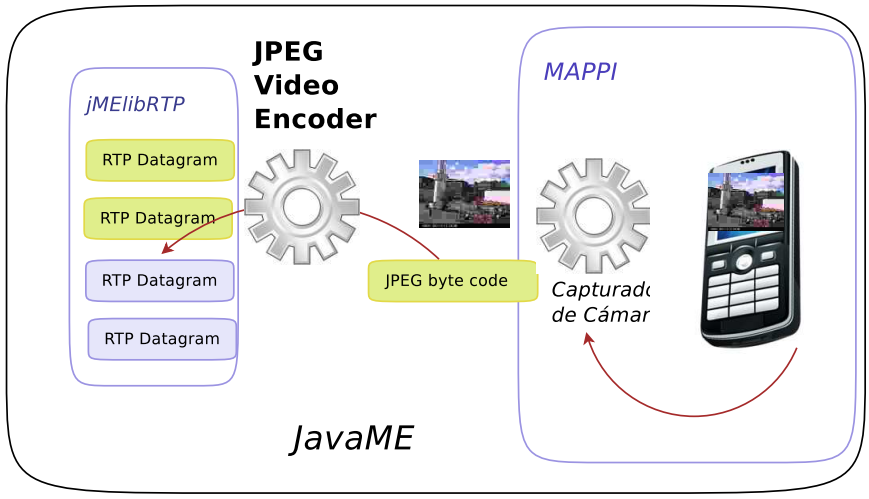

Figura 3.18: Arquitectura de la Generación de Vídeo en Tiempo Real en dispositivos ligeros

nuestro caso, es necesario que la imagen sea codificada en el formato JPEG. En caso de que el dispositivo no permita codificar las muestras bajo este formato, es posible realizar la transformación posteriormente usando el encoder para Java ME de Amini [Amini et al., 1998]. De esta forma, en cualquier caso usando MMAPI y este encoder, podemos obtener una codificación JPEG de la imagen capturada.

A modo de resumen, la captura de imágenes pasa por el proceso de codificación M-JPEG, y finalmente es enviado en tiempo real mediante la librería adaptada *jLibRTP*. Todas las fases del proceso pueden verse en funcionamiento en 3.18.

Finalmente, hemos obtenido un flujo de vídeo constante que puede ser visualizado por un cliente móvil o de sobremesa. Sin embargo, la frecuencia es relativamente baja, menos de una imagen por segundo, usando como teléfono de prueba un Nokia N81 a 360 MHz. Al igual que en los computadores de mano, el cuello de botella viene producido por el tiempo de captura que ofrece el dispositivo intrínsecamente al recoger la imagen. Y es que, estas plataformas no ofrecen la posibilidad de capturar las imágenes sin

Tabla 3.3: Ejemplo del Vídeo generado en Tiempo Real desde dispositivos ligeros

160x240	240x320
0.9	1.4

necesidad de almacenarlas en memoria. Por ejemplo, la plataforma de desarrollo Android incluye eventos que representan las imágenes de la cámara en vivo y cuyo refresco es elevado, por lo que el mismo mecanismo explicado en esta sección debe proporcionar una frecuencia considerablemente más alta para este tipo de Sistemas Operativos.

3.4.2. Generadores de Audio

De forma similar a la captura de imágenes, es posible recuperar los datos del micrófono desde los dispositivos móviles y ofrecer un flujo de sonido en tiempo real desde los mismos. Este funcionalidad unida a la recepción, posibilita la creación de conversaciones a través de la red. Estas conversaciones han resultado muy atractivas en la actualidad, ya que permitirían la comunicación por voz sin tener que pasar por la caja de las compañías de telefonía. Evidentemente, esto no ha gustado a muchas operadores que han visto la tecnología como una amenaza, prohibiendo incluso el uso de *streaming* de audio en redes GRPS.

Dejando a un lado la polémica, en esta sección se van a introducir la generación de audio desde computadores de mano. En este caso, el desarrollo para teléfonos móviles no ha sido realizado debido a las carencias que ofrecen las Máquinas Virtuales de estos dispositivos tan limitados.

3.4.2.1. Generador de Audio en computadores de mano

En esta sección vamos a detallar la generación de audio en tiempo real desde computadores de mano, tales como PDAs. Puede parecer un mecanismo sencillo, ya que al lograr la transmisión de imágenes de la cámara en tiempo real, esta sección puede parecer análoga. Aunque muchos elementos sí serán comunes, es importante destacar la constante que ha seguido a este trabajo para minimizar los silencios y retardos; y es que la generación de audio en tiempo real es más exigente que la de vídeo. Al menos, en cuanto a resultados.

Por ejemplo, si la transmisión en vídeo de una escena se realiza a 8 frames por segundo o a 3, en realidad el operador que la visualiza tiene una percepción relativamente semejante de ambas. Esto no ocurre con el sonido. El audio debe transmitirse de forma constante, evitando generar los silencios y cortes que puedan provocarse, ya que la pérdida o entrelazado de fragmentos distorsiona gravemente la percepción del operador humano que la oye. Es por ello, que se han tenido en cuenta estructuras circulares o el uso de varios buffers de captura concurrentes para proporcionar un sonido limpio y homogéneo.

En primer lugar, como en el resto de secciones, vamos a describir la capacidad de la Máquina Virtual de los computadores de mano sobre la captura y transmisión de sonido en tiempo real. De la misma forma que la interfaz para la reproducción de sonido en PDAs era inexistente, la captura de muestras de audio tampoco está configurada. Eso significa que tendremos que desarrollar una librería nativa en C++ para poder solventar las limitaciones de la Máquina Virtual.

Posteriormente, el proceso de codificación debe respetar el formato de sonido elegido, G.711, y ser encapsulado bajo el protocolo en tiempo real. La codificación y protocolos son iguales los que el Servidor Multimedia realiza en transmisión de audio. Gracias a ello, un receptor podrá suscribirse a una fuente de sonido ambiental o móvil de forma transparente. Al igual que en las transmisión de vídeo, el encapsulamiento sobre RTP es realizado por la librería multimedia JMF. Afortunadamente, también la codificación Mu-law del sonido está integrada en la versión *cross-all* de la librería, por lo que las etapas de

codificación y transmisión puede ser automatizadas.

Vamos a centrarnos, por tanto, en la construcción de la librería nativa para capturar el sonido. Esta librería recoge las muestras procedentes del micrófono en formato lineal, en bruto. Para ello, el primer paso es inicializar el micrófono en base a la frecuencia de lectura, tamaño de las muestras y la posibilidad de establecer sonido estereofónico o mono canal. Hemos de tener en cuenta que no cualquier muestreo es compatible con la codificación G.711. En la transmisión de audio en tiempo real desde los computadores de mano hemos usado una de las tasas más eficientes: la captura a 8000 muestras, 8 bits por muestra y mono canal. Estos parámetros configuran un flujo de audio, a la vez, óptimo y de calidad.

Una vez inicializado el micrófono, debemos realizar una captura eficiente y constante del dispositivo de audio. Para ello, es imprescindible el uso de varios buffers de lectura, de forma que cuando se haya completado la lectura de uno y vayan enviarse los datos por la red, el resto de los buffers sigan capturando muestras. Para ello, hemos usado un tamaño de buffer óptimo de 1000 muestras, lo que se corresponde con 125 ms de audio. Para no perder los datos de audio que puedan generarse, las muestras capturadas son almacenadas en un buffer circular hasta que se soliciten nuevamente los datos. De esta forma, no se pierden fragmentos de sonido y es posible enviar el audio capturado de más de un buffer en una sola lectura.

La comunicación entre la librería nativa y la Máquina Virtual se ha realizado, como en anteriores ocasiones, usando una interfaz JNI común en Java ME y C++. Para representar al micrófono dentro de la librería JMF y automatizar su captura y envío, hemos especificado una clase *PullBufferStream* que representa la captura de sonido en formato lineal. Los *PullBufferStream* representan aparatos de captura en JMF que son invocados asíncronamente y por lo tanto pueden ser codificados bajo el protocolo RTP.

El proceso completo donde interactúan todos los componentes puede verse en la figura 3.19.

Finalmente, se han medido los tiempos de recogida de muestras

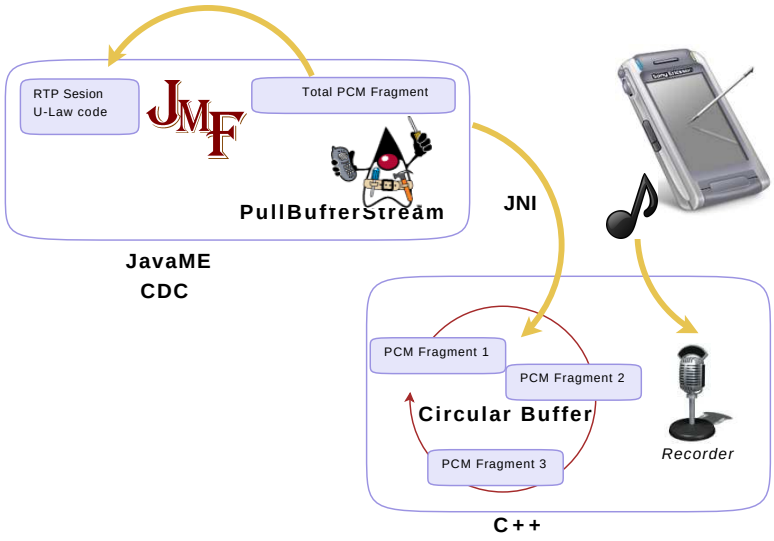

Figura 3.19: Arquitectura de la Generación de Audio en Tiempo Real en computadores de mano

tR, los de codificación tC y envío tE para determinar el retardo introducido por los dispositivos de mano al generar el flujo de audio. La recogida de muestras debe realizarse al menos cada 125 ms, ya que representan las 1000 muestras mínimas que un buffer contiene. A este proceso, hay que añadir el tiempo medio de llamada a la librería nativa que se ha establecido en 28 ms, y que sitúa al tR total en 153 ms. La codificación G.7111 reduce el tamaño del buffer a la mitad, resultado un proceso de tiempo tC para las 1000 muestras de 18 ms. Finalmente, el envío de los 500 bytes resultantes bajo el protocolo ligero UDP consume un tiempo tE de 240 ms.

Si sumamos todos los tiempos medios, obtendremos el retardo total medio que sufren los paquetes desde que son recibidos en el dispositivos hasta que son enviados por la red:

$$tR + tC + tE = 153ms + 18ms + 240ms = 411ms$$

El hecho de integrar a los dispositivos móviles como reproduc-

tores y generadores de flujos de audio en tiempo real, nos permite realizar llamadas por la red. Para ello, sólo hay que establecer una conexión bidireccional de emisión y recepción en tiempo real desde los dispositivos. Las restricciones para esta comunicación en tiempo real son muy exigentes. Si sumamos los retardos de emisión y recepción en computadores de mano, observamos que no alcanzan el segundo de desfase. Este tiempo es aceptable en circunstancias tradicionales, pero apreciable para escenarios muy estrictos. Por ello, las llamadas por red deben realizarse en potentes computadores de mano debido al mejor comportamiento y eficiencia de los dispositivos y de las Máquinas Virtuales.

3.5. Síntesis

En este capítulo hemos expuesto los resultados sobre movilización de fuentes multimedia. Inicialmente, se ha presentado el Servidor Multimedia Ambiental, el cual permite integrar diferentes dispositivos de audio y vídeo instalados en el entorno. La característica más importante del Servidor Multimedia, es la capacidad para adaptar el flujo a los requerimientos de cada cliente, siendo un paso crítico para los visualizadores ligeros de los dispositivos móviles.

La reproducción del audio y del vídeo procedente de las fuentes multimedia tiempo real ha sido desplegado en los dispositivos móviles. El uso de las Máquinas Virtuales ha provocado un esfuerzo extra en la implementación, porque existen limitaciones en cuanto a las capacidades multimedia de las mismas en dispositivos reales. Todos los desarrollos han sido realizados de forma dual, para teléfonos ligeros y computadores de mano, con el objetivo de adaptar las aplicaciones a la potencialidad de cada plataforma. Positivamente, las Máquinas Virtuales han aumentado la portabilidad de nuestros resultados, pero también ha supuesto un problema porque sufren limitaciones para acceder a los recursos multimedia de los móviles, (cámara, micrófono o altavoz). Aquí se han tratado las formas de resolución de esta problemática en la plataforma Java ME para computadores de mano.

La inclusión de los computadores de mano como generadores

de flujos multimedia, permite introducir a las cámaras y micrófonos móviles dentro del sistema. Ha sido muy importante integrar los mismos protocolos y formatos dentro los dispositivos móviles, porque de esta forma un cliente puede visualizarlos con independencia de la fuente que lo genera, sea ambiental o móvil.

Así mismo, se han evaluado y medido los resultados de las Máquinas Virtuales ante un proceso computacionalmente complejo, como la recepción y generación de flujos multimedia.

Capítulo 4

Interacción con Entornos Inteligentes usando Mapas Semánticos

El conocimiento es de dos clases: o sabemos algo por nosotros mismos, o sabemos donde encontrar información sobre ello.

Samuel Johnson.

En este capítulo introduciremos un modelo de representación de la información del espacio que permita al operador humano conocer los elementos que se encuentran a su alrededor. El capítulo se organiza como se explica a continuación.

La **sección 4.1** describe el uso de los planos como eje de representación y comunicación con el Entorno Inteligente, a los que hemos llamado Mapas Semánticos. Para operar con las figuras representados en los planos es necesario construir una Ontología gráfico-semántica que nos permitirá razonar con los elementos que han sido representados en los mismos. Además se ha deslocalizado la visualización de los mismos desarrollando sendas aplicaciones móviles que permitan manejar los Mapas Semánticos desde computadores de mano o teléfonos ligeros.

La semántica del modelo de representación de los planos puede ser procesada por los dispositivos móviles para efectuar operaciones con los elementos del edificio. Por ejemplo, fruto de este análisis semántico, hemos incluido un proceso complejo como el Cálculo de Rutas. El Cálculo de rutas se ha resuelto usando el clásico algoritmo A*, que además ha sido implementado bajo una versión reducida, dentro de los propios dispositivos móviles. Este algoritmo nos permite encontrar una solución al recorrido de grafo mediante heurísticas. Como veremos en el capítulo, hemos resuelto el Cálculo de Rutas pudiendo elegir entre la heurística clásica de distancias, o una propuesta personal basada en la minimización de ángulos hacia la recta.

Finalmente, incluiremos a los Mapas Semánticos como modelo de intercambio de datos entre el Entorno Inteligente y el operador humano en la **sección 4.2**. Esto nos permitirá, por ejemplo, representar cambios en los planos de los usuarios en tiempo real o solicitar información a los usuarios desde sus dispositivos móviles. Para estos fines, integraremos los Mapas Semánticos dentro de Servicios Remotos que pueden estar desplegados tanto en dispositivos móviles como en servidores centralizados, permitiendo una comunicación bidireccional entre el Entorno Inteligente y los usuarios. Como ejemplo de estos servicios, se han desarrollado dos propuestas a modo de ejemplo, que incluyen la localización de objetos móviles por una parte, y la notificación de eventos por otra.

4.1. Mapas Semánticos

Desde que la revolución del acceso a la información penetró en la sociedad y en los sectores tecnológicos, se han dedicado muchos esfuerzos para modelar el mundo y para definir la visión que tiene el hombre sobre el entorno.

Uno de los modelos que mayor atractivo y aceptación ha generado el hombre desde sus orígenes han sido los mapas. Éstos han evolucionado hasta hoy día en modelos complejos, como los

Sistemas de Información Geográfica. Este tipo de herramientas, junto con avances tecnológicos como los satélites, han permitido difundir la visión cenital del mundo. Gracias a ello, el conocimiento que los ciudadanos tenemos de nuestro entorno ha sido enriquecedor. Sin embargo, las herramientas de los mapas no habrían tenido tanto impacto si no se hubiesen incluido representaciones semánticas junto a los datos gráficos que acompañaban las imágenes. La inclusión de meta-información geográfica permite, por ejemplo, realizar consultas de cálculo de rutas, identificar las calles o integrar información sobre comercios.

Como podemos observar, la visión del mundo que ofrecen los satélites se produce a vista de pájaro. En esta sección deseamos trabajar con una representación que defina una perspectiva más cercana al entorno humano, representado el interior de los edificios y los elementos que se encuentran en ellos. Como veremos, esta representación establecerá un marco donde interactuar con los Entornos Inteligentes.

En el primer apartado, introduciremos los conceptos asociados a la representación gráfica y semántica de los edificios, así como su visualización en dispositivos móviles y en computadores de mano, esto es, teléfonos ligeros y PDAs respectivamente. En segundo lugar, mostraremos el algoritmo para el Cálculo de Rutas sobre los planos semánticos y un estudio de las diferentes heurísticas que podemos aplicar sobre los mismos. Finalmente, trataremos los servicios móviles asociados a los planos que comunican los Entornos Inteligentes y los dispositivos. Estos servicios serán vistos desde el punto de vista de solicitudes ubicuas y como servicios integrados móviles que pueden ser resueltos por los operadores humanos desde los computadores de mano.

4.1.1. Ontología de Edificios

De forma general, la representación de edificios y espacios de ámbito local suele estar ligada a formatos de diseño de interiores como *DWG* o *SFX*. Estos formatos son usados por herramientas de diseño gráfico, conocidas como CAD y entre las que podemos destacar Autocad.

Para una primera aproximación a la representación de edificios, pensamos en trabajar bajo estas especificaciones. Para integrarlos dentro de los dispositivos móviles, transformamos una librería con licencia GPL y desarrollada en Java llamada [yCad, 2005]; de forma que pudiese ser ejecutada en Java ME para la configuración CDC orientada a computadores de mano. Esta herramienta nos permitía visualizar en una PDA planos en formato DXF. En la práctica, el tamaño del plano debía ser pequeño porque la especificaciones de edificios en estos formatos están compuestos por innumerables líneas que pueden llegar a detallar las capas interiores de cada muro. En el resto de casos, cuando el plano resulta pesado, era necesario un pre procesamiento que eliminase datos y capas innecesarias, porque ante planos medianos o grandes, la falta de memoria en la Máquina Virtual terminaba con la aplicación.

Sin embargo, el punto más débil de estos planos no era sólo la limitación de memoria dinámica, sino la falta de *semántica*. Estos formatos clásicos de representación sólo especifican la parte gráfica de un edificio y no etiquetan los objetos como habitaciones, pasillos, puertas, escaleras, etc. por lo que es imposible, o bajo procesos muy complejos, determinar qué elementos hay especificados en el plano, y en definitiva, operar con los mismos.

Bajo esta problemática, uno de los requisitos propuestos para el manejo de planos interiores, es desarrollar un modelo que represente edificios y que pueda aplicarse a un escenario genérico. Intuitivamente, sabemos que la especificación de los edificios suele dividirse en plantas y las mismas en habitaciones, posteriormente en mobiliario, y así progresivamente. Además, es necesario representar cómo es posible acceder de un espacio a otro, mediante puertas, pasillos, escaleras o ascensores; de forma que sea posible establecer un recorrido entre los objetos de los planos. Estos conceptos debían de estar incluidos bajo una Ontología de Edificios, que finalmente fue desarrollada desde cero.

4.1.1.1. Representación Geométrica

Suele existir cierto acuerdo sobre la representación gráfica de los objetos. En general, para definir formalmente el espacio usamos matemáticamente espacios euclídeos N-dimensionales. En ellos, podemos representar elementos y situarlos espacialmente. El elemento más básico, el punto, es definido como un vector de N componentes: una por cada dimensión del espacio que especifican la distancia ortogonal a cada uno de los ejes. A su vez, una recta puede ser definida mediante dos puntos representando la línea que los comunica. De forma general, un polígono puede ser descrito con N >3 puntos.

Por otro lado, pueden incluirse espacios curvos que permitan unir los puntos de forma no lineal si no con fragmentos de una circunferencia. También pueden incluirse elipses, definidas con N puntos de localización y N valores que expresen el tamaño de sus arcos, una por cada dimensión del espacio. Los círculos, o esferas en el caso tridimensional, pueden ser vistas como un caso particular de elipses donde los los N arcos y N puntos son iguales.

En lo referente a la dimensionalidad, para resolver el problema de la representación gráfica de edificios barajamos dos opciones. La primera, era definir sólo 2 dimensiones, trabajando con representaciones cenitales de edificios. La segunda, usar un modelo tridimensional que permitiera una navegación virtual y que ofreciese una visión real del edificio. Por mucho que nos atrajo esta última opción, la ardua labor de representar plantas, habitaciones y objetos con 3-Dimensiones fue descartada, ya que un simple polígono en 2-D tiene una representación extremadamente más compleja en 3-D. Eso significa que a las personas encargadas de diseñar un plano para un espacio 3-D concreto, requerirían unos conocimientos y habilidades espaciales complejas, así como elaborar un trabajo desproporcionado para representar cualquier pequeño espacio. Por eso, muchos programas de diseño de interiores trabajan en 2-D y sólo establecen un modelo de capas para representar la superposición y no el complicado espacio tridimensional. Por otro lado, el cálculo del espacio 3D respecto las transformaciones de vista podía resultar demasiado pesado para las Máquinas Virtuales de los

dispositivos móviles.

Por estos motivos decidimos escoger un modelo 2-D, con coordenadas (x, y). Sin embargo, era crítico que contuviera una tercera dimensión que representase la altura de la planta de cada punto. Esta tercera coordenada es esencial para agrupar los elementos por plantas, pudiendo efectuar posteriormente una visualización por capas de las mismas. Además, en el Cálculo de Rutas representa la distancia, esto es en definitiva el trabajo que hay que realizar para trasladarse de una planta a otra.

Gracias al modelo 2-D, obtenemos una representación geográfica completa que nos permite definir el espacio. Sin embargo, para aumentar la escalabilidad de los modelos gráficos e incrementar las posibilidades de cómputo en ordenadores, es posible incluir transformaciones que permitan trasladar los puntos de un grupo de elementos con una simple operación. Para trasladar un punto o un grupo del plano, es necesario conocer las matrices de transformación de vista, en nuestro caso las del espacio euclídeo 2D. Las matrices de transformación permiten obtener una nueva figura geométrica resultante de multiplicar cada punto de la figura original por dicha matriz, en el caso de representación de polígonos. Si los objetos son curvos, como las elipses, las matrices afectan también los arcos que comunican los puntos de la figura.

Para estos propósitos, se incluyen las transformaciones geométricas como conceptos clásicos de la Computación Gráfica, [Trias, 2003] y entre las más importantes dentro la geometría nos encontramos: la Translación, la Rotación y la Escala. Podemos representar cualquier cambio de transformación como una sucesión de estas tres matrices básicas. Es decir, para cualquier valor de M, existe una o varias secuencias de estas transformaciones básicas cuyo resultado sea el deseado.

La Traslación define un cambio de la figura original en el espacio; para ello, es necesario indicar el desplazamiento ortogonal a cada dimensión. La Escala provoca un cambio de medida en cada eje del espacio, aumentando o disminuyendo la distancia entre los puntos; aunque puede aplicarse la misma magnitud a cada dimensión

respetando la proporción de la figura original. La Rotación nos permite desplazar los puntos originales respecto un ángulo ortogonal definido por dos coordenadas del espacio. En nuestro caso 2-D, las matrices para la traslación, escala y rotación son respectivamente:

$$T(tX, tY) = \begin{pmatrix} 1 & 0 & tX \\ 0 & 1 & tY \\ 0 & 0 & 1 \end{pmatrix}$$

$$S(sX, sY) = \begin{pmatrix} sX & 0 & 0 \\ 0 & sY & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R(\alpha) = \begin{pmatrix} cos(\alpha) & sin(\alpha) & 0 \\ -sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Además es posible, y muy útil, aplicar una sucesión de transformaciones para trasladar objetos. Con estas tres transformaciones básicas podemos representar cualquier cambio como una secuencia derivada de ellas. Para ello, se define la matriz M, como el resultado ordenado de aplicar la sucesión de transformaciones. Inicialmente, el valor de M es la matriz identidad, que representa la inmutabilidad de los puntos, ya que al aplicarla, la figura inicial no cambia. La aplicación sucesiva unas transformaciones puede calcularse como el producto de matrices sobre esta matriz identidad, resultando una única matriz que al aplicarla sobre los puntos iniciales representase el movimiento de todas las transformaciones.

Por ejemplo, si queremos desplazarnos 3 unidades en el eje Y, rotar la vista 90 grados y luego escalar por la mitad tendríamos:

$$M = T(0, 3) \otimes R(\pi/2) \otimes S(1/2)$$

Es importante destacar que el orden influye en la aplicación de las transformaciones, ya que por ejemplo es diferente escalar y trasladar que trasladar y escalar.

Evidentemente, estas transformaciones cambian los puntos

originales del plano, por lo que es necesario establecer un mecanismo para conocer el valor inicial de la figura, en caso de que sea necesario. Para ello, es posible deshacer las transformaciones calculando una matriz que represente las operaciones inversas. La matriz inversa de M, se calcula como el producto en orden contrario de la inversa de las transformaciones:

$$T^{-1}(tX, tY) = \begin{pmatrix} 1 & 0 & -tX \\ 0 & 1 & -tY \\ 0 & 0 & 1 \end{pmatrix}$$

$$S^{-1}(sX, sY) = \begin{pmatrix} 1/sX & 0 & 0 \\ 0 & 1/sY & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R^{-1}(\alpha) = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

En el ejemplo anterior, la matriz inversa resultaría:

$$M^{-1} = S(2) \otimes R(-\pi/2) \otimes T(0, -3)$$

Cuando se aplican la matriz M y posteriormente su inversa, el resultado es neutro, resultado la matriz identidad:

$$M \otimes M^{-1} = T(0, 3) \otimes R(\pi/2) \otimes S(1/2) \otimes S(2) \otimes R(-\pi/2) \otimes T(0, -3) = 1$$

4.1.1.2. Representación Semántica

Además de la integración de representaciones geométricas, se han incluido varios conceptos que permiten definir los elementos integrados en el plano y su relación entre ellos.

En primer lugar, se ha diseñado un estructura de clases que permite etiquetar los elementos geográficos usando las clases más comunes utilizadas en interiores 4.1. De esta forma, se añade a

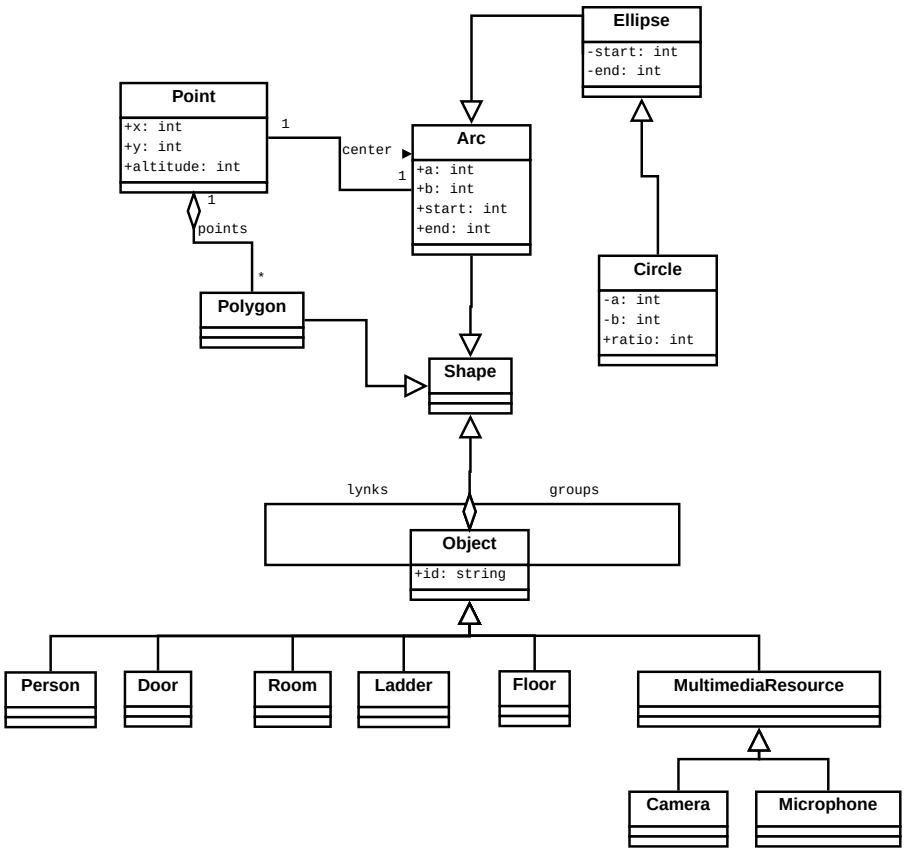

Figura 4.1: Clases para la organización gráfica y semántica de planos

cada objeto del plano una clase de pertenencia. Las clases pueden incluir atributos que especifiquen valores no gráficos del objeto. Por ejemplo, en el caso de las cámaras o micrófonos se incluye un campo identificativo del recurso multimedia. Además, la especificación de la clase permite al motor gráfico dibujar el objeto con unos colores y propiedades visuales implícitos, sin que estos tengan que ser especificados en cada elemento. Esto es muy útil, porque por lo general, todas las habitaciones, puertas u otros elementos tienen una representación conjunta.

Por otro lado, también se ha incluido el concepto de grupo, como un conjunto de elementos que representan un concepto común. Así, un

grupo define una relación entre varios objetos de la Ontología, como puede ser el propio edificio, una planta, o una determinada zona. Estas agrupaciones permiten una visualización compacta, permitiendo al usuario visualizar únicamente los gráficos del grupo de componentes, o bien propagar un transformación a todo un conjunto de elementos. Los grupos pueden tener incluidos otros grupos, formando una jerarquía de elementos dentro del edificio. El ejemplo más sencillo de la potencialidad de unir las transformaciones y los grupos es el navegador de planos que permite efectuar un zoom o mover la vista del observador, con tan sólo aplicar una escala o una traslación en el elemento raíz del edificio.

Otro concepto muy importante introducido en la Ontología de Edificios es el de enlace. La representación gráfica no es suficiente para determinar si un elemento, como una habitación se comunica con otro y en que zona exacta lo hace. Por eso, es necesario introducir implícitamente en la semántica dicha referencia. En particular, cada elemento de la Ontología puede estar enlazado con otro, entendiéndose como la existencia de un camino desde un elemento hacia otro. El nivel de granularidad de los caminos puede especificarse en cada caso, por ejemplo comunicando una habitación con otra, o incluyendo una puerta entre ambos elementos.

Por tanto, el concepto de *Objeto* en la Ontología de Edificios es definido por una 6-tupla (i, g, t, c, h, e) donde:

- **i** es el *identificador del Objeto*. Se trata de una valor que identifica unívocamente al objeto.
- **g** es la información Gráfica, que a su vez está definida por alguno de estos elementos:
  - **polígono**, figura definida por una sucesión de puntos que la configuran.
  - **círculo**, figura definida por un punto y un escalar que corresponde con el radio.
  - **elipse**, figura definida por un punto y una pareja de escalares que definen los arcos.

- **arco**, figura definida por un punto, una pareja de escalares que definen los arcos y una pareja de valores que delimitan el inicio y el fin de la sección a representar.

- **t** define las transformaciones asociadas al objeto. Quedan representadas como una secuencia de elementos, de los que cada uno que puede ser una:

  - **Translación** transformación definida por un punto que representan el desplazamiento de los ejes.
  - **Rotación** transformación definida por un ángulo (en radianes) que determina el giro cenital del observador.
  - **Escala** transformación definida por una pareja de valores que especifican el cambio de escala en el eje de abscisas y ordenadas. Es recomendable la pareja contenga valores idénticos para no deformar la vista (achatar o alargar una dimensión), por lo que se permite pasar un único valor que determina la misma escala en X e Y.

- **c**, clase de objeto que se representa. Es optativa, y queda definida con un valor que contiene una de estas clases de pertenencia:

  - **Habitación.**
  - **Puerta.**
  - **Escalera.**
  - **Planta.**
  - **Ascensor.**
  - **Cámara.**
  - **Micrófono.**
  - **Persona.**

- **h**, pertenencia a un grupo. Es optativo, y especifica la pertenencia a un único objeto padre, que puede agrupar varios elementos. Este valor configura la jerarquía de objetos, de forma que todos los hijos de un objeto pertenecen a un mismo grupo.

- **l**, enlaces a otros elementos. Es optativo y configura un conjunto de valores que hacen referencia al identificador de otro objetos. Estos enlaces determinan la accesibilidad de este objeto con respecto el resto de objetos del plano.

Para realizar la definición de estos elementos de forma estándar y especificar un mapa interior concreto, hemos desarrollado un traductor capaz de construir el mapa en tiempo de ejecución leyendo un archivo. En particular, este archivo está estructurado en formato XML que configura todos los valores de la Ontología de Edificios. Este formato es muy apropiado, porque el traductor de los Mapas Semánticos permite aprovechar la propia estructura jerárquica del documento XML y trasladarla como los grupos de objetos de la Ontología, de forma que no es necesario especificar este campo implícitamente.

Un fragmento de especificación puede verse en 4.2 junto al resultado del mismo en un computador de mano y en un dispositivo ligero 4.3.

Los ejemplos de validación del modelo han sido edificios reales cuya localización ha sido determinada bajo el sistema de coordenadas Universal Transversal de Mercator (UTM). Este sistema de coordenadas está basado en la proyección del globo terrestre sobre un cilindro, conocida como proyección Mercator, pero se ha dividido por Husos en longitud y Zonas en latitud, con el fin de minimizar las deformaciones alejadas del ecuador. Las magnitudes en el sistema UTM se expresan en metros. La lectura de la localización de los objetos bajo este sistema de coordenadas no influye en el modelo 2D porque es compatible con el modelo cartesiano, pero para una correcta visualización es necesario aplicar una translación que centre la figura del edificio y la escale según la pantalla de cada dispositivo.

4.1.2. Navegador de Mapas Semánticos

Para que los usuarios puedan navegar por los Planos Semánticos, se ha añadido un navegador que permite efectuar un zoom y

```
        <element id="141">
                <graphic type="polygon">
                        60836.590979753 4814022.2939114,60844.059470125
4814022.7825042,60843.570877297 4814028.3664223,60847.968212749 4814028.8550151,60848.247408651
4814026.7610458,60849.573589184 4814026.8308448,60849.78298611 4814025.1556694,60853.76152771
4814025.5744633,60854.668914391 4814012.3824569,60850.899769717 4814011.8240651,60851.667558447
4814004.7045696,60848.666202503 4814003.9367809,60849.224594307 4814002.6803993,60845.804444511
4814002.5408014,60845.315851683 4814007.3569307,60838.056758237 4814006.5891419,60836.590979753
4814022.2939114
                </graphic>
                <semantic>
                        <room></room>
                </semantic>
        </element>
        <element id="141">
                <graphic type="polygon">
                        60836.590979753 4814022.2939114,60844.059470125
4814022.7825042,60843.570877297 4814028.3664223,60847.968212749 4814028.8550151,60848.247408651
4814026.7610458,60849.573589184 4814026.8308448,60849.78298611 4814025.1556694,60853.76152771
4814025.5744633,60854.668914391 4814012.3824569,60850.899769717 4814011.8240651,60851.667558447
4814004.7045696,60848.666202503 4814003.9367809,60849.224594307 4814002.6803993,60845.804444511
4814002.5408014,60845.315851683 4814007.3569307,60838.056758237 4814006.5891419,60836.590979753
4814022.2939114
                </graphic>
                <semantic>
                        <room></room>
                </semantic>
        </element>


        <!-- Puertas -->
        <element id="1">
                <graphic type="polygon">
                </graphic>
                <semantic>
                        <door p="60856.40566079 4814032.8321375" l="2"></door>
                </semantic>
        </element>
        <element id="5">
                <graphic type="polygon">
                </graphic>
                <semantic>
                        <door p="60852.556452278 4814043.1420502" l="2"></door>
                </semantic>
```

Figura 4.2: Especificación de Planos Semánticos en XML

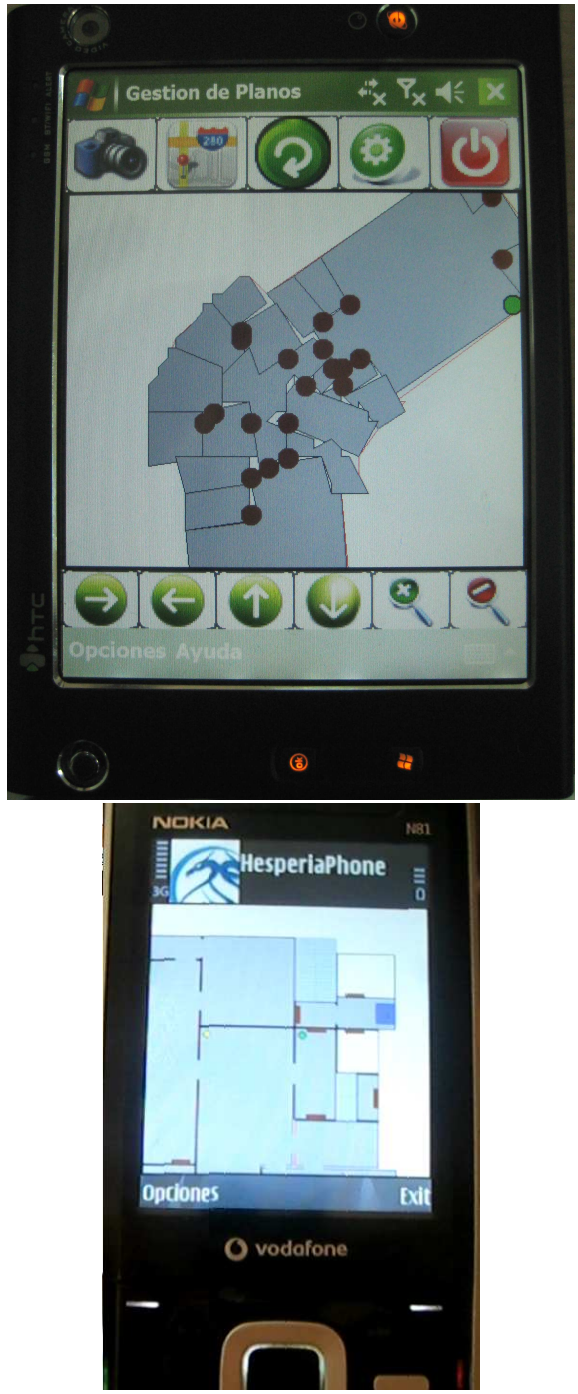

Figura 4.3: Planos Semánticos en los dispositivos móviles

desplazarse por los mismos. Esta herramienta es más que útil cuando hablamos de visualización en dispositivos móviles, ya que el tamaño de la pantalla es muy limitado, y los usuarios pueden ampliar y navegar con el nivel de detalle que requieran. Este navegador ha sido construido para dispositivos móviles ligeros y computadores de mano más potentes. En particular, para la plataforma Java ME para la configuración CDC y el perfil Personal Profile, y también para Java ME configuración CLDC y el perfil MIDP 2.0.

En primer lugar, el navegador permite leer los datos de configuración del plano y construir los objetos gráfico semánticos a partir del mismo. El cálculo de las coordenadas de cada objeto ha de ser calculado en el dispositivo móvil porque puede incluir transformaciones asociadas. Aunque el cálculo de las transformaciones en grupo aumenta la potencialidad de representación, su uso también es complejo porque hemos de transformar los puntos en función de la jerarquía que estructuran el plano. Debido a esta jerarquía, las transformaciones que modifican un objeto del plano son producto, también, de las transformaciones de sus antecesores; y a su vez, es necesario deshacer las transformaciones propias de un nodo para que no se propaguen a los hermanos. En la figura 4.4, vemos un ejemplo que aclara el cálculo de la matriz de transformación M, según el nodo del plano en el que nos encontremos.

El cálculo conjunto de las trasformaciones ha sido resuelto usando un algoritmo que inicializa M a la matriz unidad, y va incorporando las transformaciones de los nodos según baja por el grafo, y las deshace cuando sube por el mismo. El recorrido que efectúa el algoritmo es en *preorden*, también llamado orden previo, el cual consiste en recorrer en primer lugar la raíz y luego cada uno de los hijos $A_1, A_2 \ldots A_k$ en orden previo. Este orden es muy apropiado porque permite realizar la acumulación de transformaciones en la matriz M, según se recorre en profundidad y eliminar las transformaciones cuando se finalizan los hijos y deshacemos el recorrido de un nodo.

Una vez inicializado el Mapa Semántico, el usuario puede navegar por el plano. Para ello, se han definido las operaciones de zoom (ampliación y alejamiento) y desplazamiento hacia los cuatro ejes

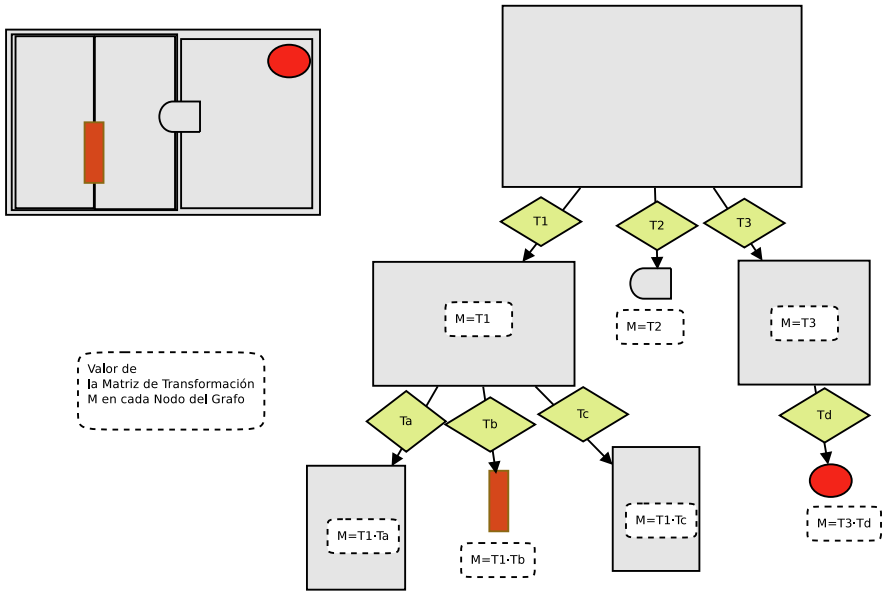

Figura 4.4: Ejemplo de Jerarquía y Transformaciones de cada nodo

Algorithm 1 HierarchicalTransformationAlgorithm G, M

Require: $G \neq \phi$ {Recursive Algorithm, G is a node and M the acumulative transformation matrix}
{First time, when G is the root, we initialize M to Identity}
if $N = root(G)$ **then**
$M = Identity$
end if
$T = transformationsOf(N)$
$P = pointsOf(N)$
$M = M \otimes T$
$P = P \otimes M$
$PAINT(P)$
for i = 1 to $numChildOf(N)$ **do**
$N = ChildOf(i, N)$
$HierarchicalTransformationAlgorithm(N, M)$
end for
$M = M \otimes T^{-1}$

(este, oeste, norte y sur). El resultado de ampliar zoom se formaliza con la aplicación de una transformación en escala s a la raíz del Mapa Semántico. De esta forma, gracias a la propagación del algoritmo jerárquico todos los puntos quedan ampliados. El resultado inverso, el alejamiento de zoom, se realiza con la matriz inversa de la escala $1/s$. De forma análoga, las operaciones que se han definido para navegar hacia el este y oeste resultan de aplicar una translación en el eje de abscisas de tX y $-tX$. Análogamente, el desplazamiento norte y sur se corresponde con una traslación en las ordenadas de tY y $-tY$.

Como los puntos reales del mapa quedan modificados cuando se le aplican estas transformaciones, en el caso de querer obtener las referencias originales del objeto, es necesario almacenar los cambios de zoom y navegación que el usuario ha ido realizando al plano. Para ello, se ha creado una matriz M inicializada a la unidad. A esta matriz se le aplican todas las traslaciones realizadas a los puntos del mapa de forma inversa, de forma que al multiplicar por esta matriz inversa las coordenadas del objeto vuelven a su sistema original.

Imaginemos por ejemplo, que se realiza un zoom al plano y luego desplazamiento en horizontal, todos los puntos del edificio se multiplican por la matriz de la escala s y por la matriz desplazamiento tX por la derecha. Mientras que M^{-1} queda multiplicada por la inversa de la escala $1/s$ y el inverso del desplazamiento $-tX$ por la izquierda.

$$P \otimes S \otimes tX = P', M = tX^{-1} \otimes S^{-1} \otimes 1$$

De esta forma, si queremos obtener los puntos originales, solo debemos multiplicarlos por la matriz M y tendremos la referencia original.

$$P' \otimes M = P \otimes S \otimes tX \otimes tX^{-1} \otimes S^{-1} = P$$

4.1.2.1. Selección de elementos

Uno de los objetivos más importantes en la especificación de los Mapas Semánticos, es la interacción del usuario con los elementos del mapa. Derivado de esta necesidad, hemos desarrollado un proceso que permite recuperar elementos del plano usando la pulsación de pantalla. Esta selección es fundamental para aplicaciones en computadores de mano, donde en algunos dispositivos ya sólo contamos con pantallas táctiles. Para resolver el problema de la selección de elementos, se ha integrado un algoritmo que relaciona una pareja de coordenadas bidimensionales con uno o varios elementos del plano.

Algunos elementos no presentan muchas complicaciones, como es el caso de los círculos, donde sólo hay que evaluar la distancia al centro. En esta sección vamos a resolver el caso genérico para el caso polinomial.

El algoritmo elegido ha sido el Algoritmo de la Cuerda [Peshkin and Sanderson, 1985], ya que ofrece una complejidad lineal al evaluar si un punto es interior a un polígono dado. Este algoritmo calcula el ángulo entre el punto candidato y cada pareja de puntos consecutivos del polígono. Hay que notar que pueden existir ángulos positivos y negativos, en función del sentido del recorrido 4.5. Finalmente, la suma de todos los ángulos representa la solución, pudiendo existir sólo dos valores válidos 0 and $\{0, 2\pi\}$. Si el resultado es $\{0, 2\pi\}$, el punto candidato es interior. La complejidad del algoritmo es $O(N)$. Además, una de las ventajas de este algoritmo es que es robusto a polígonos huecos, esto es, polígonos que contienen una frontera interior cuya superficie no pertenece a la figura.

En la práctica es aconsejable dejar un intervalo de confianza $[2\pi - \alpha, 2\pi + \alpha]$ en función de un valor α. Este intervalo resuelve el problema de la pérdida de precisión de un computador, que puede ser considerable en cálculos de ángulos.

La versión simplificada del algoritmo permite recorre todos los objetos del planos y evaluarlos uno a uno. En caso de que la evaluación del punto candidato y el polígono se encuentre dentro del intervalo de confianza, el objeto se considera seleccionado, por lo que es posible

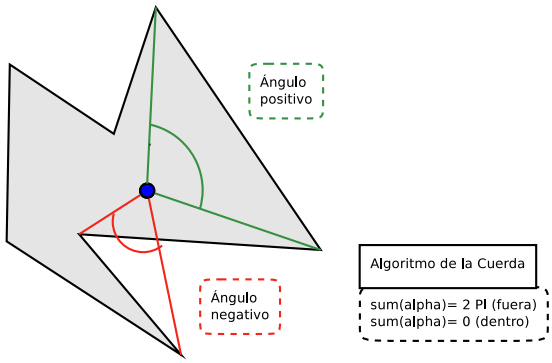

Figura 4.5: Algoritmo de la Cuerda. Ángulos positivos y negativos

Algorithm 2 SelectionAlgorithm N, P

Require: $N \neq \phi$ {Selection Algorithm, N is a node and P the point to evaluate}

$S = 0$
$Ps = pointsOf(N)$
for i = 1 to $|Ps|$ **do**
  $P1 = Ps[i]$
  $P2 = Ps[(i+1) mod |Ps|]$
  $R1 = rectBetween(P1, P)$
  $R2 = rectBetween(P2, P)$
  $S = S + angleBetween(R1, R2)$
end for
if $S == 2\pi$ **then**
  1
end if
0

que varios objetos puedan extraerse de la selección. Esto puede ser producido por solapamiento de objetos cuya es intersección es no vacía, que en nuestro caso es más que probable, ya que dentro del contorno del edificio existen plantas y dentro de ellas habitaciones, y así sucesivamente.

Para restringir el espacio de búsqueda, en el caso de que el número de elementos sea extremadamente extenso, podemos considerar sólo los elementos de una planta, o de una coordenada, descartando el resto. También, gracias a la jerarquía de elementos, podemos realizar una poda del recorrido basada en la inclusión de los hijos dentro de la superficie del nodo padre. Si se cumple esta precondición, no es necesario explorar recursivamente todos los hijos de un nodo, si no sólo aquellos que son seleccionados por la coordenada candidata. Esto nos evita recorrer el grafo por objetos cuyo padre no está seleccionado. Evidentemente, esta selección será correcta y completa siempre que los hijos de cualquier nodo se encuentre incluidos gráficamente dentro de los padres.

Un ejemplo de selección de objetos sobre el plano puede verse en la figura 4.6, donde al seleccionar un elemento, éste cambia a color rojo.

4.1.3. Cálculo de Rutas

El Cálculo de Rutas permite encontrar un camino desde un nodo origen y otro destino entre los nodos de un grafo que están conectados entre sí. En este apartado vamos a detallar el estudio realizado para incluir el Cálculo de Rutas en Mapas Semánticos, así como su desarrollo en dispositivos móviles.

Extracción del Grafo de Rutas El primer paso para efectuar un Cálculo de Rutas, es construir un grafo a partir de los elementos del plano. Gracias a este grafo, podremos automatizar la extracción de caminos desde cualquier mapa. La relación de accesibilidad de cada nodo con el resto se establece gracias a la propiedad *enlace* definida en la Ontología de Edificios.

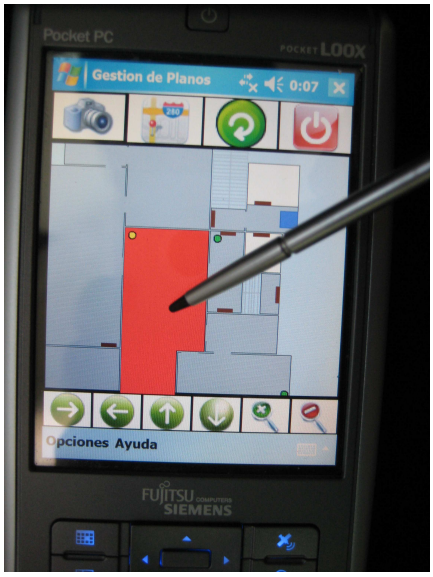

Figura 4.6: Selección de Objetos del Mapa Semántico

El proceso de extracción evalúa cada nodo y establece su posición como centro de gravedad del elemento. El centro de gravedad es calculado como la media geométrica de los puntos en cada valor de los ejes cartesianos:

$$center(P) = \sum P_i/|P|, \forall P_i \in P$$

Posteriormente, relacionamos cada nodo con sus nodos vecinos definidos en el enlace del objeto. Este enlace representa la movilidad de un objeto a otro. Un ejemplo puede verse en la figura 4.7.

Es importante mencionar una nota a parte de este proceso de extracción cuando un elemento del plano representa un espacio extenso. Como podemos observar, la representación de un objeto queda resumida a un nodo localizado en su centro de gravedad. En el caso de que el objeto represente un contorno grande, un único nodo podría suponer una simplificación demasiado severa del objeto. La solución se resolvería si el objeto se dividiese en más de un nodo, haciendo que la granularidad entre el objeto y los nodos que lo representan sea más

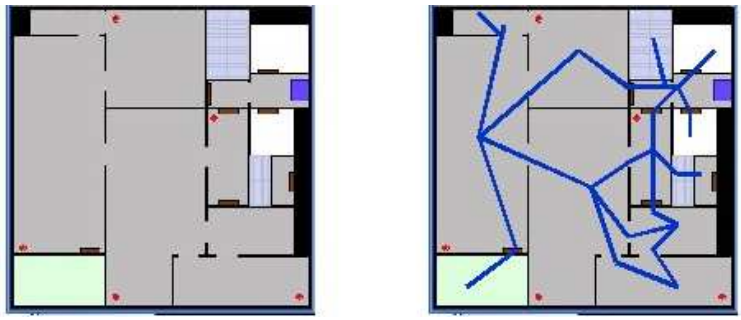

Figura 4.7: Mapa Semántico y grafo de recorrido asociado

densa. De esta forma, al dividir los objetos en secciones los caminos del Cálculo de Rutas serían más precisos, aunque es cierto que el centro de gravedad es más que suficiente para el cómputo de los dispositivos móviles. Con independencia de la localización del objeto, el paso previo de granulación es transparente para el algoritmo de encaminamiento que presentamos a continuación.

Algoritmo A* Una vez obtenido el grafo de rutas, vamos a describir el algoritmo para determinar el camino, si existe, entre un nodo origen y destino del mismo. En la literatura quizá el más conocido es Dijkstra [Dijkstra, 1959], que permite obtener la ruta en complejidad cuadrática en base a las distancias que comunican los nodos.

Por otra parte, el recorrido de grafos ha sido tratado como un problema genérico y configurable, posteriormente, en el Algoritmo A* [Hart et al., 1968]. A* encuentra un camino entre el nodo origen y destino, manteniendo una lista de candidatos ordenados y seleccionados en base a una heurística H*, que evalúa el coste acumulado de viajar hasta ellos. Esta heurística es introducida por el usuario y en base a ella, encontraremos una solución más rápida o mejor $H*$ (desde el punto de vista práctico) y que se asemeje a una evaluación ideal de los esperado H. Muchas veces no conocemos este ideal de búsqueda que puede ser difícil de evaluar (por ejemplo en puzzles).

El algoritmo de Dijkstra es, en realidad, un caso concreto de A*.

Mientras A* resuelve el problema de la búsqueda de forma genérica, Dijkstra realiza el algoritmo en función del cálculo de distancias. El hecho de que A* parezca diferente, se produce por las propiedades que se derivan del uso de las distancias como heurística, que permiten ignorar algunos casos del recorrido genérico de grafos. En concreto, las propiedades de A* para igualar a Dijkstra son:

- *H* (U,V) = Distance(U,V)*. La heurística usada es la distancia entre los nodos.

$$H*(U,V) = Distance(U,V), G(U) = 0 \rightarrow$$
$$F(U,V) = H(U,V)* + G(U) = Distance(U,V)$$

- *H* no sobreestima el valor real H*. Si esto se demuestra, se dice que la heurística es *monótona* o *consistente*. Éste es el caso de las distancias, ya que la heurística se corresponde exactamente con el valor a determinar (H=H*). Si se cumple esta propiedad [Russell and Norvig, 2003], no es necesario reevaluar nodos en el proceso de exploración. Es decir, la primera vez que se explora un nodo (nodos cerrados) es la mejor; por tanto, no es necesario volverlos a evaluar estos nodos, aunque los encontremos de nuevo. En definitiva representa la idea de que el recorrido por el que pasas la primera vez hasta llegar a un nodo, es el óptimo, y si vuelves a pasar, siempre habrás tardado más que la primera.

  Si los nodos cerrados son innecesarios el fragmento de código asociado a $IF(V \in Closed)$ and $IF(V \notin Open \cap V \notin Closed)$ puede ignorarse. De esta forma, obtenemos una versión reducida de A* más eficiente. Además, esta versión puede también usarse con heurísticas no monótonas, pero en este caso, A* no permite garantizar que encontremos un camino que optimice el valor de H*. Esta versión reducida de A* es muy apropiada para el cómputo en dispositivos móviles, ya que la capacidad de los mismos es más limitada.

Algorithm 3 DijkstraAlgorithm G, S, T

Require: $G \neq \phi$ {Dijkstra Algorithm, G is the graph, S is the source and T is the target}
$Dist = \{\infty\}$
$Path = \{\infty\}$
$Q = \{G\}$
$Dist(S) = 0$
{In each iteraction, we get the node in Q that have min value of Dist}
while $Q \neq \phi$ **do**
  $U = min(D, Q)$
  if $U = S$ **then**
    $RETURNPath$
  end if
  $Q = Q - U$
  {If you improve the rote to V, store it}
  for $V = neighbor(U)$ **do**
    if $Dist(U) + Dist(U, V) < Dist(V)$ **then**
      $Dist(V) = Dist(U) + Dist(U, V)$
      $Path(V) = U$
    end if
  end for
end while

Algorithm 4 AStarAlgorithm $G, S, T, H*$

Require: $G \neq \phi$ {A-star Algorithm, G is the graph, S is the source, T is the target and H* is the heuristic}
$F = \{\infty\}$
$Path = \{\infty\}$
$Open = \{S\}$
$Closed = \{\}$
$F(S) = H*(S)$
{In each iteraction, we get the node in Q that have min value of Dist}
while $Open \neq \phi$ **do**
$U = min(F, Open)$
if $U = S$ **then**
$RETURNPath$
end if
{If you improve the rote to V, store it}
for $V = neighbor(U)$ **do**
if $V \in Open \cup V \in Closed$ **then**
if $F(U) + H*(U, V) < H*(V)$ **then**
$F(V) = F(U) + H*(U, V)$
$Path(V) = U$
end if
end if
if $V \in Closed$ **then**
for $W = descendants(V)$ **do**
if $F(V) + H*(W, V) < F(W)$ **then**
$F(W) = F(V) + H*(W, V)$
$Path(W) = V$
$Open = Open \cup W$
end if
end for
end if
if $V \notin Open \cap V \notin Closed$ **then**
$F(V) = F(U) + H*(U, V)$
$Path(V) = U$
$Open = Open \cup V$
end if
end for
end while

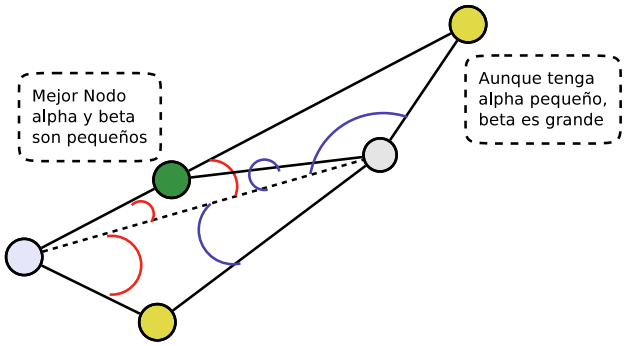

Figura 4.8: Minimización de ángulos sobre la línea origen-destino

Heurística de minimización de ángulos Además de la incorporación de la clásica heurística de las distancias, que como hemos comentado se corresponde al algoritmo de Dijkstra, se ha propuesto otra que se asemeja más a la que realiza consciente o inconscientemente el ser humano. La intuición, y la práctica, nos impulsan a buscar el camino entre dos nodos como el que mejor se aproxima a la línea recta que los une. Esta información es muy valiosa y debería ayudarnos para guiar la búsqueda en el espacio. Para definir una heurística que refleje esta filosofía evaluamos los ángulos que forman los nodos origen, destino y candidato. Al igual que en el caso de las distancias se trata de un problema de minimización.

De forma gráfica, entre un nodo origen, destino y un candidato a ser parte de la ruta, podemos distinguir dos ángulos: α el formado entre origen-candidato, origen-destino y β (candidato-destino, origen-destino). Ambos son necesarios para representar un ajuste con respecto la línea de origen-destino, ya que como vemos en la figura 4.8, existen situaciones en que un ángulo pequeño α puede desembocar en una mala solución porque hace a β muy grande. Sólo cuando ambos valores son pequeños significa que el candidato está cerca de la línea entre el origen y el destino. Además esta heurística se puede considerar como el valor real de búsqueda y puede usarse la versión A* reducida y además garantizar que obtendremos el camino óptimo para la minimización.

Si comparamos las dos heurísticas, distancias y ángulos, vemos que representan planteamientos muy diferentes. Dijkstra usa el valor

de las distancias para representar los pesos de trabajo de ir de un nodo a otro, de forma que la información espacial se pierde y no es usada. Es decir, la localización geográfica en Dijkstra es irrelevante, ya que la relación entre nodos es solo escalar. Esto hace que en los casos promedios, donde los nodos están repartidos homogéneamente, el espacio de búsqueda sea lento, realizando un recorrido hacia la solución en circulo y evaluando más nodos que con los ángulos. En algunos casos, también es una ventaja perder la información geográfica, porque en ocasiones sólo podemos evaluar el trabajo o la velocidad y no la orientación o localización de los nodos. En el peor de los casos, ambas heurísticas se comportan con notación cuadrática que viene derivada de la complejidad de la versión reducida de A*.

Finalmente, en la figura 4.9 mostramos los tiempos de ejecución para una malla de nodos repartida homogéneamente en un espacio bidimensional y cuyo nodo origen es la coordenada (0,0) y el nodo destino (N,N). Podemos observar cómo al crecer el número de nodos, crece el tiempo de búsqueda. Las fluctuaciones en la gráfica de la heurística de los ángulos viene provocada por los casos cuadráticos NxN, donde la línea entre el nodo origen y destino coincide con el camino de la malla, y por lo tanto, la convergencia a la solución es directa porque los nodos del camino hacen que la heurística tenga un valor 0. En el resto de los casos, donde los nodos no están situados justo sobre la línea solución, el espacio de búsqueda es más extenso haciendo que se consuma más tiempo hasta encontrar la solución, aunque nunca se obtienen peores resultados que con Dijkstra.

La versión reducida de A* y la integración de las dos heurísticas ha sido implementada satisfactoriamente para dispositivos ligeros y computadores de mano. La implementación se ha realizado sobre la plataforma Java ME, siendo compatible con las configuraciones CLDC y CDC. En la figura 4.10, podemos ver el resultado del cálculo de rutas sobre un computador de mano.

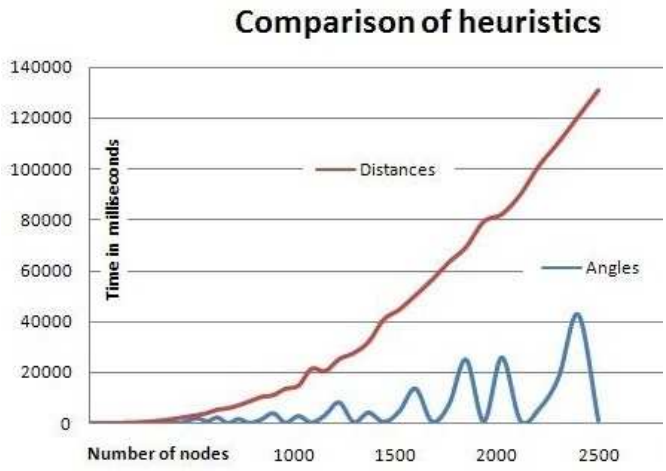

Figura 4.9: Comparativa de las heurísticas de minimización de ángulos y distancias

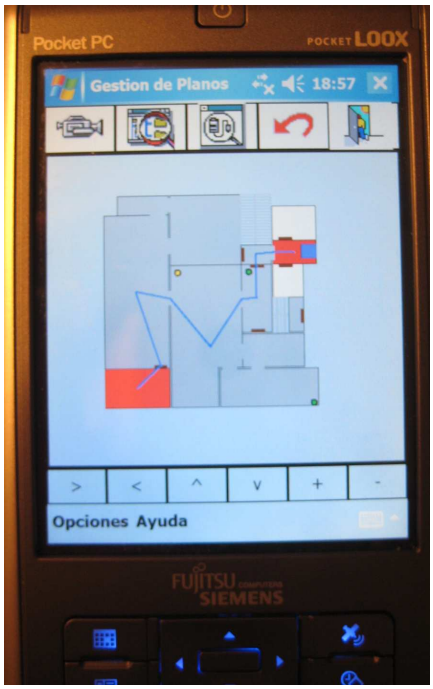

Figura 4.10: Cálculo de rutas en dispositivos móviles

4.2. Arquitectura con Entornos Inteligentes

Una vez especificada la representación y Cálculo de Rutas de los edificios, en esta sección vamos a presentar Servicios Remotos asociados a los mismos y que permitan sincronizar la información del sistema y de los usuarios mediante los Mapas Semánticos. La visualización de los datos ambientales en los Mapas Semánticos es muy útil, porque permite mantener actualizada la información del entorno y mostrarla al usuario de forma visual en su dispositivo móvil.

Es importante destacar, que en esta tesis vamos a centrarnos en los mecanismos de transmisión y visualización de los datos provenientes del Entorno Inteligente en los dispositivos móviles; esto es, sin profundizar en los procesos de análisis y extracción de datos que puedan realizar los diferentes Sistemas Expertos del ambiente.

Como comentamos en la sección 2, en este trabajo se van a incluir dos Servicios entre el Entorno Inteligente y los dispositivos móviles. En concreto, en la tesis se presentan los trabajos relacionados con la Localización y Notificación en Tiempo Real sobre los Mapas Semánticos.

Los Servicios Remotos que se exponen a continuación han sido representado bajo el modelo de propagación de Canales de Eventos que se presentó en la sección 2.7.1. Los Canales de Eventos permiten propagar información a todo un grupo de usuarios suscritos a un canal y conectarse a los mismos para recibir los datos que el administrador emite.

En nuestro caso, la propagación de información al canal la realiza el Entorno Inteligente. El Entorno Inteligente es el encargado de procesar la información del edificio, proveniente de sensores u otras fuentes de información, para procesarla y transmitirla a los usuarios en caso necesario. Este esquema es el que se sigue en la Notificación en Tiempo Real.

También es interesante introducir a los usuarios, no sólo como sumideros, sino también como fuentes de información. Para ello, los Servicios Remotos deben ser ubicuos, esto es, estar integrados en los

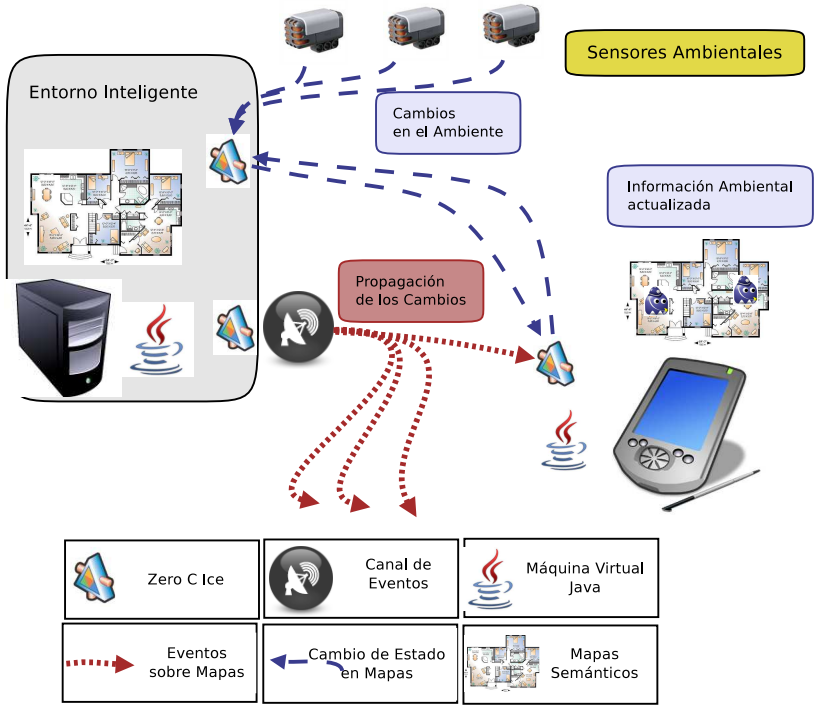

Figura 4.11: Esquema de Movilización con Entornos Inteligentes

dispositivos móviles y poder ser accedidos por el Entorno Inteligente, o incluso desde otros dispositivos móviles. Esta comunicación permite realizar una validación o retroalimentación de la información del sistema por parte del operador humano, con lo que es posible rectificar datos u obtener información adicional de los usuarios. Las peticiones pueden establecerse en varios sentidos. En primer lugar, clientes móviles que acceden a un Servicio Remoto centralizado del Entorno Inteligente, como por ejemplo, una petición para preguntar dónde se encuentra una habitación. Y viceversa, peticiones implícitas del Entorno Inteligente a un usuario para que devuelva la localización de un objeto. Esta filosofía se describe en la Localización en Tiempo Real.

Los modelos de comunicación propuestos pueden verse gráficamente en la figura 4.11.

4.2.1. Visualización y Localización de elementos en tiempo real

La detección de la localización junto al tratamiento y propagación de los datos, es un campo del que se ha escrito mucho en Computación Ubicua e Inteligencia Ambiental. En este apartado vamos a incluir Servicios Remotos que den soporte a la localización en interiores. Como los objetivos generales de la tesis se centran en la movilización de información, vamos a integrar las peticiones de localización en interiores desde los dispositivos móviles de los usuarios. Con esta herramienta el operador humano podrá resolver su situación pulsando en los Mapas Semánticos. Posteriormente estos datos son propagados en un Canal de Eventos donde los clientes móviles imprimen las notificaciones sobre los Mapas Semánticos mostrando la localización de los usuarios.

De forma más detallada hemos seleccionado los componentes que intervienen en la solución:

- Clientes Móviles que usan los Mapas Semánticos para que el operador humano determine su posición y visualice la del resto de objetos móviles. La solicitud de los datos de localización puede ser implícita o explícita. En el primer caso, es el propio operador humano quien notifica voluntariamente al Entorno Inteligente de su localización. Por contra, la solicitud explícita es pedida desde un servicio central a los dispositivos móviles.

  A su vez, los Clientes Móviles mostrarán los cambios de los objetos móviles en los Mapas Semánticos, respetando la escala y desplazamiento que el usuario haya aplicado al plano.

- Otras Fuentes de Localización, como sensores, también pueden alimentar al sistema con posicionamiento explícito. No hemos querido limitar la localización de interiores a las peticiones al operador humano. El uso de sensores de movimiento o la trazabilidad de los usuarios pueden permitirnos conocer la situación de los objetos y usuarios del edificio. Sin embargo, estas fuentes necesitan de un procesamiento inteligente de los datos

que permita identificar a los usuarios a partir de las muestras recogidas. Este proceso escapa a los objetivos de la tesis, de forma que nuestro propósito es mostrar simplemente el resultado de esta localización desde dispositivos móviles.

- Un Servicio de Localización Central donde se propaguen los datos de la localización de interiores. Este módulo puede formar parte del Entorno Inteligente del edificio. El Servicio de Localización se ha diseñado como un Canal de Eventos, donde el Servidor de Localización Central tiene dos funciones:
  - Recoger datos de la Localización que hayan sido procesados de cualquier fuente de posicionamiento, ya sea desde sensores o peticiones implícitas de un operador humano.
  - Procesar y limpiar los datos de la localización y propagarla por el Canal de Eventos. En esta etapa podría estar incluido el proceso de extracción inteligente comentado con anterioridad.
  - Mantenimiento de los usuarios móviles del canal. Los usuarios del Canal de Eventos se conectan o desconectan en tiempo de ejecución. Para ello, se usan funciones de suscripción y liberación respectivamente.

La implementación de los Servicios Remotos se ha realizado usando *ZeroC Ice*. Este middleware, como indicamos en 2.7, permite la novedosa capacidad de resolver Servicios desde el dispositivo móvil, de forma que ya no existen clientes y servidores, sino que todos los componentes implementan y solicitan servicios entre ellos. Gracias a esta filosofía, los servicios pueden ser resueltos y solicitados desde computadores estáticos o móviles.

La definición del servicio de localización ha sido resulta con dos operaciones básicas que permiten la solicitud e intercambio de datos.

- *Posicion PosicionExplicita()*. Este servicio representa una petición de localización sobre algún elemento que puede auto localizarse. En nuestro sistema, representa una petición explícita del

Servidor de Localización Central a un dispositivo móvil. La respuesta debe ser asíncrona o no bloqueante, ya que el operador humano puede tardar en ver la notificación y en dar la respuesta.

- *void PosicionImplicita(string id, Posicion p)*. Esta operación es usada para dos propósitos.

  En primer lugar, permite determinar la locación de objeto desde los clientes móviles al servidor. El dispositivo que genera el evento debe incluir la identificador y posición del recurso móvil, de forma que el receptor puede conocer el objeto al que se refiere la localización. Así, un usuario puede detallar la localización propia, de otros usuarios o de otros objetos; lo cual es muy valioso, por ejemplo, en labores de vigilancia. Por otra parte, también puede ser generado, cuando un sensor de identificación notifica al Servidor de Localización Central de la presencia de una persona.

  En segundo lugar, esta operación también es usada en el sentido de comunicación contrario, esto es, para propagar los cambios de localización de los objetos móviles. En este sentido, son los computadores de mano quienes reciben los datos de localización desde el Entorno Inteligente. Esta operación se difunde en el Canal de Eventos y permite mostrar la localización de un objeto móvil a todos los suscriptores del canal.

  Con ambas filosofías, convertimos a los computadores de mano a la vez en sumideros de localización y en generadores de ella. Esta retroalimentación es muy útil en los casos de vigilancia con incertidumbre, porque permite ajustar la localización entre el conocimiento parcial de los usuarios y del Entorno Inteligente, obteniendo una representación global de todos los elementos del sistema.

La organización y comunicación de los componentes puede detallarse de forma visual en la figura 4.12.

La propagación de la localización mediante Canales de Eventos requiere de un mantenimiento que debe efectuarse tanto en los clientes que se conectan, como en el administrador que controla el canal. A

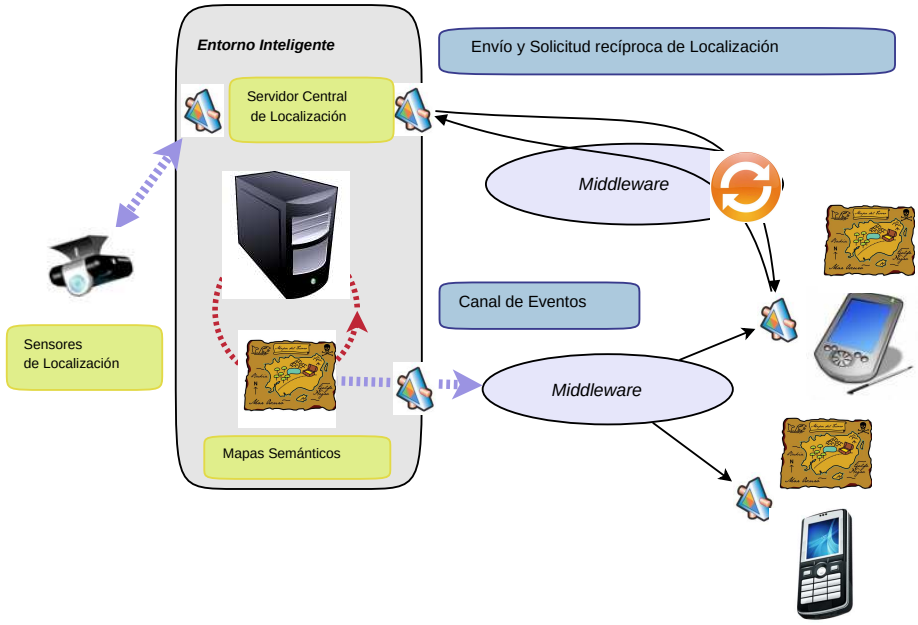

Figura 4.12: Arquitectura de los componentes de localización en tiempo real

continuación, vamos a describir el ciclo de vida de los clientes ubicuos dentro del Sistema de Localización.

- Inicialización. En esta etapa, es necesario crear los servicios móviles dentro de cada dispositivo, y posteriormente suscribirlos al servicio de localización central. Por tanto, al iniciar la aplicación se despliegan los servicios que implementará el dispositivo móvil:

  - *void PosicionImplicita(string id, Posicion p)*. Para conocer la posición del resto de elementos móviles del edificio y visualizarlos sobre los Mapas Semánticos.
  - *Posicion PosicionExplicita()*. Para recibir peticiones de localización que resolverá el operador humano usando los Mapas Semánticos.

  Posteriormente se notifica al administrador del canal, que un nuevo dispositivo móvil ha sido integrado en el canal, de forma que queda suscrito al mismo. Este paso puede realizarse usando *IceStorm*, una herramienta del ZeroC Ice que auto gestiona el Canal de Eventos, o bien administrar a los suscriptores con una función ad hoc. Sin embargo, si usamos una versión limitada de ZeroC Ice-E, en concreto, la versión para nuestra plataforma de desarrollo Java ME, no podemos usar la herramienta IceStorm y debemos gestionar la suscripción y propagación de cada uno de los clientes.

- Mantenimiento. Los clientes móviles recibirán notificaciones del Canal de Eventos de Localización en la función *void PosicionImplicita(string id, Posicion p)*, con lo que pueden saber quién (o qué) se ha desplazado y dónde se encuentra ahora. Si el identificador ya estaba pintado en el mapa, este es eliminado y actualizado con la nueva posición. De esta forma, el Mapa Semántico queda actualizado, mostrando la información por plantas de los elementos que allí se encuentra ubicados, véase la figura 4.13. Es importante detallar, que la posición recibida debe ser ponderada respecto al zoom y desplazamiento que el usuario

Figura 4.13: Ejemplo de localización de usuarios en tiempo real en un dispositivo móvil

ha aplicado al mapa, aplicando la transformación correspondiente a la escala y traslación horizontal y vertical del navegador de mapas (recuérdese la sección 4.1.2).

Desde la interfaz gráfica también deben solventarse las peticiones explícitas de localización que solicite el Servidor Central de Localización al operador humano. Cuando esto ocurre, aparece un mensaje en la pantalla del dispositivo móvil informando al operador humano de que su localización es solicitada. El operador puede rechazar o aceptar la petición, y en caso afirmativo, seleccionar un punto del Mapa Semántico para resolverla 4.14. Es apropiado que este tipo de solicitudes se declaren asíncronas porque el usuario puede tardar el tiempo que considere oportuno para resolver la petición, y mientras el solicitante no debería estar bloqueado por la espera.

Finalmente, el usuario puede pulsar la pantalla determinando su

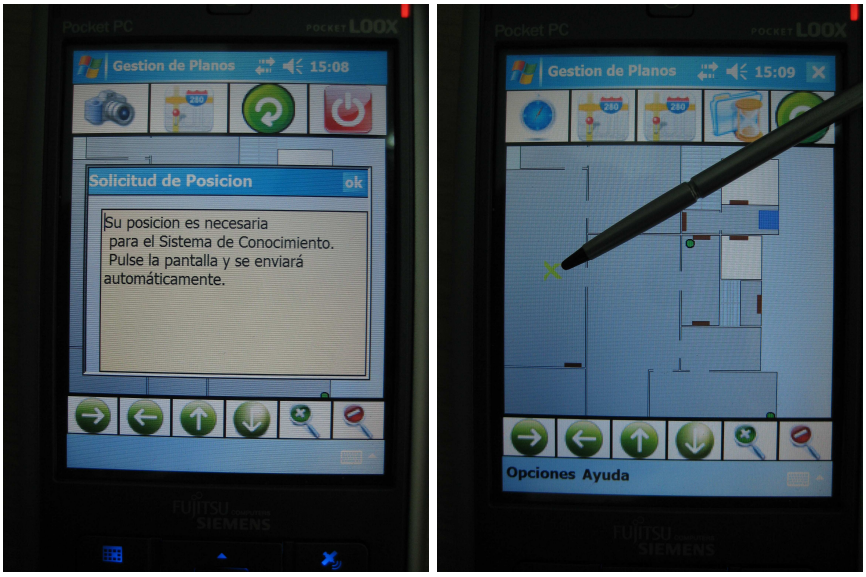

Figura 4.14: Solicitud asíncrona de Localización en un dispositivo móvil

posición en el Mapa Semántico, pero es importante que al igual que en la visualización de localizaciones, debemos realizar una transformación a las coordenadas. En este caso, la transformación representa la matriz inversa de la escala y traslación que el usuario ha aplicado en al plano. De forma análoga a las pruebas de representación del edificio, el sistema de coordenadas que se ha integrado es Universal Transversal de Mercator (UTM), de forma que los usuarios devuelven su localización en los Mapas Semánticos bajo este sistema.

- Cierre. Es muy apropiado, que al finalizar la aplicación móvil, liberemos a los clientes para que el Servidor Central de Localización no envíe peticiones e información a puntos desconectados. Aunque la herramienta *IceStorm*, por ejemplo, es capaz de eliminarlos del canal automáticamente cuando la conexión a un cliente lanza un error; si programamos nosotros mismos el canal debemos de ser conscientes de que puede haber errores en la propagación, y decidir en ese caso si eliminarlos o no.

4.2.2. Envío de Notificaciones a los Dispositivos Móviles

Finalmente, en este apartado presentamos el envío de notificaciones que el Entorno Inteligente puede enviar a los usuarios que se encuentran conectados al sistema. El objetivo es proporcionar un mecanismo de propagación de información hacia los Mapas Semánticos que se muestran en los dispositivos móviles. Las notificaciones representan cualquier análisis que efectúa el Entorno Inteligente a partir del ambiente que monitoriza. Aquí presentaremos las técnicas para administrar esta comunicación, sin adentrarnos en ningún proceso de detección concreto.

En particular, hemos usado el término notificaciones porque especifica un amplio rango de usos, mientras que las alertas tienen una connotación peligrosa. De esta forma, las notificaciones pueden usarse para informar o sugerir y no siempre están asociadas con alarmas.

La comunicación de notificaciones es más sencilla que la notificación en tiempo real de localizaciones, ya que los dispositivos móviles no efectúan el doble papel de generadores y receptores de información. Aquí, los usuarios son vistos simplemente como sumideros de eventos. Al igual que en el apartado anterior, para propagar la información se ha hecho uso de un Canal de Eventos que transmite las notificaciones que el Entorno Inteligente detecta en tiempo real.

El esquema y organización de los componentes que intervienen en modelo se presenta en la figura 4.15.

Uno de las aportaciones de las notificaciones es su integración con los Mapas Semánticos. La estructura de una notificación contiene:

- Tiempo de creación. Este campo especifica la marca de tiempo en el que la notificación ha sido detectada. El formato del tiempo se corresponde con el estándar Universal Time Coordinated (UTC) que mantiene su precisión usando relojes atómicos extremadamente precisos.

- Texto con la explicación. Este parámetro determina en lenguaje

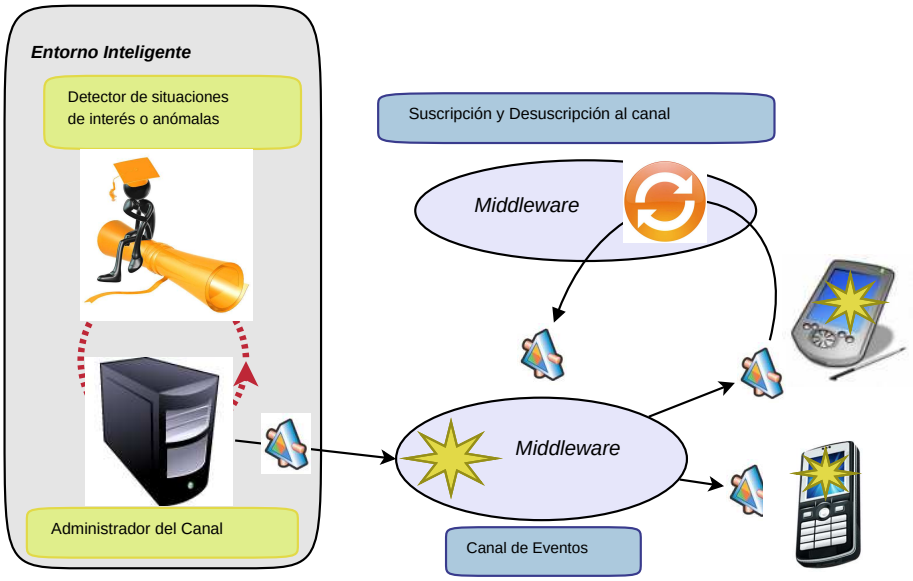

Figura 4.15: Arquitectura de los componentes de notificaciones en dispositivos móviles

natural una explicación de la notificación.

- Ruta sugerida. Representada por un vector de puntos que determinan la ruta que se sugiere al usuario.
- Objeto destino. Si la notificación tiene como objetivo alcanzar u operar con un objeto del entorno, éste puede ser identificado en este campo.

Estos dos últimos parámetros integran la visualización dentro de los Mapas Semánticos y son optativos, pero su uso es recomendable porque ofrecen un mayor potencial en la visualización, imprimiendo la ruta sobre el Plano Semántico y coloreando el destino para resaltar su importancia.

Como vemos, el servicio que representa la recepción de notificaciones es unidireccional de forma que el Entorno Inteligente envía la notificación y no espera la recepción implícita del usuario. Esto es recomendable cuando las notificaciones son informativas, pero en caso de otros contextos, se podría requerir una respuesta de aceptación del operador humano, devolviendo un parámetro de respuesta, aunque fuera asíncronamente para no bloquear el comportamiento del Entorno Inteligente.

Además, si varias notificaciones se acumulan en el dispositivo, se almacenan en un pequeño visualizador de notificaciones que permite navegar y visualizar de forma individual cada una de ellas. La funcionalidad de este elemento, junto a la visualización de notificaciones con rutas asociadas puede verse en la figura 4.16.

4.3. Síntesis

Los Mapas Semánticos han surgido como herramienta para visualizar los edificios y describir el entorno que rodea a los usuarios. En este capítulo se ha expuesto un modelo gráfico que permite definir los elementos de los edificios y mostrarlos en dispositivos móviles. Para dotar de mayor potencialidad de definición, se han incluido

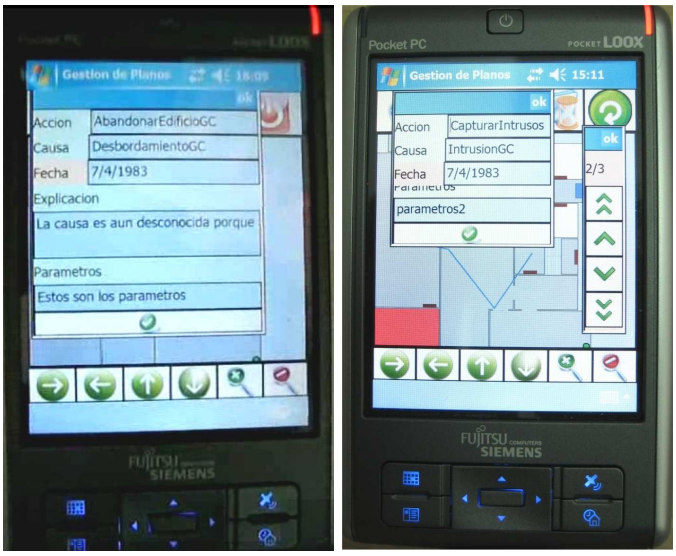

Figura 4.16: Notificaciones en el dispositivo móvil sobre los Mapas Semánticos

transformaciones geométricas para modificar los elementos de los mapas.

Posteriormente, se ha descrito la necesidad de aportar a los mapas más información de la meramente geométrica. La *semántica* nos permite etiquetar los elementos descritos en una clase de pertenencia, agruparlos y especificar la movilidad de los usuarios por los mismos. Todos estos conceptos gráfico-semánticos han sido definidos en una Ontología de Edificios que es procesada desde los dispositivos móviles. Esta aportación permite que los usuarios puedan seleccionar elementos del mapa y operar con ellos, visualizar grupos de elementos como las plantas de los edificios, o realizar un Cálculo de Rutas.

El Cálculo de Rutas en Mapas Semánticos, ha sido ampliamente detallado en el capítulo, donde se ha integrado una versión reducida del Algoritmo A* dentro de los dispositivos móviles generando una ruta entre cualquier elemento del mapa de forma automática. A su vez, hemos estudiado el uso de las heurísticas para guiar la búsqueda basándonos en las distancias, que se corresponde con Dijkstra, y en una propuesta personal basada en la minimización de ángulos.

El segundo bloque del capítulo muestra los desarrollos sobre Servicios Remotos que se han incluido sobre los Mapas Semánticos. Estos servicios nos permiten comunicar a los usuarios con el Entorno Inteligente de forma amigable e intuitiva. Al igual que en el capítulo anterior, los dispositivos móviles han sido integrados en el sistema como sumideros y generadores de información. Esta filosofía se ha seguido en la Localización en Tiempo Real, que permite propagar los cambios de localización sobre dispositivos móviles conectados al sistema y también, que el Entorno Inteligente solicite la posición de los usuarios mediante los Mapas Semánticos.

Finalmente, hemos desarrollado el envío Notificaciones procedentes del Entorno Inteligente. Dicha información es descrita sobre los Mapas Semánticos, y en esta ocasión, los dispositivos sólo han sido evaluados como sumideros de datos. En ambos Servicios Remotos, debemos destacar, con especial énfasis, la organización basada en Canales de Eventos, que permiten conocer la suscripción y salida de los dispositivos móviles en el sistema. Gracias a ellos, el Entorno Inteligente monitoriza de forma dinámica los clientes conectados, pudiendo propagar y solicitar datos de forma apropiada.

Capítulo 5

Conclusiones y Trabajo Futuro

Sólo dos cosas contribuyen a avanzar; ir más aprisa que los demás, y seguir el buen camino.

René Descartes.

Los resultados expuestos en la tesis han sido fruto de varios años de trabajo en el campo de la Computación Ubicua e Inteligencia Ambiental. En la actualidad, la tendencia a la movilización de la información ha supuesto una revolución que incrementa las posibilidades de comunicación de muchos sistemas. En algunos casos, sirve para trasladar aplicaciones estáticas que habían sido pensadas para equipos de mesa, y cuya funcionalidad puede ser adaptada a los dispositivos móviles. Otros sistemas surgen desde cero, gracias al cambio de perspectiva para comunicar a las personas y al entorno en el que habitan.

Los desarrollos del Sistema obtenidos en este trabajo pueden describirse bajo estas dos filosofías de movilización. Por ejemplo, inicialmente, se planteó el acceso a recursos multimedia, como cámaras ambientales desde dispositivos de mano. Este tipo de aplicaciones existían previamente en equipos de escritorio, y los trabajos empezaron

por integrar y adaptar las fuentes de vídeo y audio a la recepción móvil. Una vez resuelta esta problemática, se introdujo a los dispositivos como elementos multimedia del sistema, de la misma forma que los computadores de sobremesa nos permiten ofrecer transmisiones desde el micrófono o la cámara web.

Sin embargo, la solicitud de localización de los usuarios sólo ha sido posible desarrollarla gracias a la deslocalización de las aplicaciones. Gracias a la integración de los Servicios Remotos dentro de los dispositivos, los usuarios pueden observar el recorrido de los objetos móviles del entorno, y a su vez resolver su localización personal desde un dispositivo de mano a los Entorno Inteligentes. Este tipo de servicios no podrían resolverse desde las arquitecturas estáticas de los equipos de escritorio.

Uno de las constantes del trabajo, ha sido el profundo estudio de las posibilidades multimedia de los dispositivos móviles empleando Máquinas Virtuales, en concreto Java Micro Edition. Éstas nos permiten trasladar gran parte de los resultados a otros dispositivos o incluso otras plataformas, ya sea parcialmente o totalmente. La problemática asociada al uso de Máquinas Virtuales se produce por la complejidad en tiempo de ejecución, mayor que la de aplicaciones nativas programadas ad hoc el Sistema Operativo. Realizar un proceso tan complejo como la reproducción de flujos multimedia en tiempo real, suponía una prueba del rendimiento de las mismas, más aún cuando uno de los objetivos es que la decodificación se realice sin delegarla en terceras aplicaciones.

Del mismo modo, se ha estudiado la integración de los recursos multimedia de los computadores de mano, para generar los flujos desde los mismos en tiempo real. Esto ha ido unido unívocamente a las capacidades de las propias Máquina Virtuales para acceder al micrófono o a la cámara de una forma eficiente, y la forma de resolver las limitaciones que se han encontrado en computadores ligeros y de mano.

Por otra parte, hemos realizado un estudio de la conectividad con Entornos Inteligentes, donde los Servicios Remotos son resueltos tanto

por servidores centrales que monitorizan muchos dispositivos, como por los propios computadores de mano. La creación y administración de los mismos ha sido realizada bajo el Middleware ZeroC Ice, por su facilidad para trasladar las funcionales a dispositivos móviles de forma más o menos transparente.

Para establecer un contexto de comunicación común entre el operador humano y la información procedente del Entorno Inteligente, se ha propuesto el uso de Mapas Semánticos. Éstos nos han permitido representar el espacio físico y los elementos de las instalaciones de forma gráfica: podemos efectuar zoom, navegar por los mismos, seleccionar elementos con una pulsación o visualizar los mimos por plantas. Pero a su vez, también incorporan semántica, etiquetando los elementos y las propiedades de los objetos. El análisis de estas propiedades nos ha permitido automatizar el proceso de Cálculo de Rutas, incorporando la heurística clásica de las distancias (Dijkstra) y una propuesta personal basada en una heurística de minimización de ángulos.

Además, se ha mostrado el valor de actualizar los Mapas Semánticos en tiempo real usando los Servicios Remotos. Este tipo de Servicios pueden ser usados para muchos propósitos y adaptados en multitud de escenarios. Como ejemplo, en el capítulo 4 de la tesis, se analizan de forma detallada dos aplicaciones genéricas: la localización en tiempo real y la transmisión de eventos en tiempo real. Para ambos procedimientos se ha propuesto una arquitectura que comunica el Entorno Inteligente y los dispositivos móviles usando los Canales de Eventos que nos proporciona el Middleware.

Como podemos apreciar, los campos que se han visto involucrados durante el análisis y desarrollo del Sistema han sido extensos y diversos. Esto ha hecho que los resultados sean atractivos al mezclar técnicas multimedia con los dispositivos móviles, o al incorporar conceptos tan cercanos como los planos en la palma de la mano de los usuarios. Sin embargo, casi todo el trabajo realizado ha sido desarrollado desde cero, porque pocas herramientas han sido compatibles bajo los requerimientos expuestos, o bien ha sido necesario una ardua adaptación de estas herramientas para hacerlas útiles en la

movilización.

5.1. Conclusiones

En esta sección vamos a detallar las conclusiones derivadas de los desarrollos de la tesis, y comentaremos los problemas que han surgido en su desarrollo y las decisiones para poder solventarlos. Para ello, es conveniente revisar los objetivos iniciales que se propusieron. En esta sección vamos a describir las soluciones desarrolladas, junto con los puntos fuertes e inconvenientes que se encontraron a priori y posteriori.

5.1.1. Servidor Multimedia para Fuentes Ambientales

El primer objetivo de la tesis fue integrar diferentes recursos multimedia situados en el ambiente y poder transmitir un flujo en tiempo real de forma homogénea hacia los dispositivos móviles. La decisión más importante fue la de elegir una codificación de vídeo y de audio acorde con las restricciones del tiempo real. No todos los protocolos multimedia incluyen la sensibilidad al tiempo real, y un claro ejemplo podemos verlo en la propias cámaras IP, cuya transmisión viene determinada por un protocolo orientado a conexión como TCP, sobre el que construye la conectividad HTTP.

La elección de UDP permite **reducir los retardos en la transferencia de datos**, a expensas de no asegurar la recepción o pérdida de los mismos. Evidentemente, la urgencia de la transmisión puede ser relativa. Si los datos de una cámara llegan unos segundos después al dispositivo móvil usando un protocolo u otro, puede ser trivial o dramático, según el caso práctico. Por ejemplo, si tenemos que detectar una situación en tiempo real, la urgencia y prioridad para minimizar el retardo es alta; por el contrario, si evaluamos una visualización para comprobar el estado de un recurso remoto, como el jardín de una casa, la espera de unos segundos no implica grandes desventajas.

En nuestro caso, hemos apostado por **la comunicación más apropiada para el tiempo real**, esto es, una transferencia bajo UDP no orientada a conexión. Esta filosofía no es trivial, porque la codificación de los datos debe ser compatible con la pérdida de algunos fragmentos, debiendo recomponerse ante esta situación. Por tanto, la elección del tipo de comunicación afecta a su vez a las capas superiores.

En cuanto a la codificación de los datos, la elección del formato de audio fue sencilla, en contraposición a la del vídeo. El formato G.711 es un estándar de compresión de datos que cuenta con una aceptación y difusión dentro de los estándares de voz IP. La dos implementaciones más conocidas basadas en el G.711 son los algoritmos logarítmicos Mu-law y A-law. En esta tesis, hemos trabajado con el primero porque ofrece unos resultados ligeramente mejores para el cómputo en ordenadores. Además es libre desde 1972. Tanto la liberación como la complejidad del algoritmo han sido positivos. En primer lugar, porque hemos encontrado implementaciones del mismo para integrarlo en dispositivos de mano ligeros. Y por otra parte, porque la sencillez del mismo, lo hacen apto para ser ejecutado en dispositivos móviles.

Como hemos comentado, la codificación del vídeo ha resultado más compleja. Respecto a los novedosos formatos para codificar vídeo que han surgido en la actualidad, en este trabajo se ha integrado un formato primario: **Motion - Jpeg Compressed Video** (M-JPEG). Esta codificación realiza un proceso de compresión JPEG para cada frame de forma independiente a sus predecesores, de forma que no existe una compresión en tiempo de la secuencia de imágenes. El resultado es un formato anticuado respecto a los recientes MP4 y V8, comentados en el Estado del Arte, que reducen el tamaño de la secuencia usando técnicas de predicción temporal y espacial, así como compensaciones de movimiento.

Sin embargo, el uso de M-JPEG para el vídeo fue fundamental por su baja complejidad respecto los otros formatos. La realización del proceso de descompresión dentro de las Máquinas Virtuales desaconsejaba la implementación de los nuevos formatos, que junto a la integración del protocolo real, sólo se han llevado a cabo en aplicaciones nativas móviles. Además, otro de los objetivos funcionales incluía a los

dispositivos móviles como fuentes de vídeo del sistema, debiendo de incluir el proceso de codificación en tiempo real. En definitiva, sólo un formato sencillo para el vídeo nos permitía adaptar la codificación y decodificación del flujo de vídeo dentro de las Máquinas Virtuales de los computadores de mano.

Como requisito no funcional, además, el proceso de descompresión y compresión debía realizarse íntegramente en las Máquinas Virtuales de los dispositivos móviles, ya que un requerimiento de la movilización era efectuar el proceso desde las propias aplicaciones sin delegar funcionalidad en terceros. Una vez desarrolladas y testadas las aplicaciones móviles, podemos enumerar las ventajas de M-JPEG como formato de vídeo para recursos ambientales:

- Pueden perderse fragmentos sin afectar a varias secuencias, sólo al frame que se vea involucrado.
- No es necesario acumular varios frames para transmitir el flujo de vídeo, así como no es necesario esperar a que varias secuencias de vídeo estén disponibles para empezar a decodificar. Esto se traduce en una reducción de los tiempos de espera, tanto en el Servidor Multimedia, como en los dispositivos móviles.
- La complejidad de la decodificación es menor, que con los algoritmos de vídeo entrelazados, reduciendo el tiempo de decodificación y presentación. Gracias a ello, este codificador ha podido ser portado con éxito en teléfonos ligeros.

Sin embargo, también en la independencia de la codificación temporal radican su problemas:

- Consumo de ancho de banda mayor que en los formatos con compresión en tiempo, ya que en lugar de transmitir los cambios respecto al frame anterior, se transmite toda la imagen. En este caso, aumenta el número de paquetes que navegan por la red.
- Es necesario repintar toda la pantalla del dispositivo móvil en cada frame, en lugar de solicitar la sobre escritura de zonas concretas.

Dejando a un lado los protocolos y codificación elegidos, otro objetivo fundamental que se ha conseguido ha sido proteger las fuentes multimedia de los accesos de los clientes móviles. Los modelos de conexión más básicos establecerían una comunicación directa desde los dispositivos móviles con la fuente de vídeo o audio, por ejemplo, mediante la conexión a la cámara IP. Esta filosofía no controla los accesos y los recursos del sistema, desprotegiendo los dispositivos que se conectan al entorno.

Para poder integrar una política de control sobre el acceso a los recursos multimedia, se han definido unos Servicios Remotos que permiten aceptar o rechazar la solicitud de un cliente para visualizar u oír el ambiente. Además, **el cliente solo reproduce el flujo multimedia, sin tener acceso directo a la fuente que lo genera**. Otra ventaja que ofrece el control de sesiones multimedia, es la posibilidad de realizar multicast, de forma que una sola conexión a la fuente puede propagar el flujo a varios clientes.

5.1.2. Fuentes Multimedia Ubicuas

El uso de las Máquinas Virtuales para el desarrollo de dispositivos móviles permite la portabilidad de los resultados de la tesis a distintas plataformas. Además, para no depender de librerías nativas que efectúen algunas tareas de la visualización multimedia móvil, se han implementado todas las fases: recepción en tiempo real, decodificación y visualización en código puro de Java ME. Este esfuerzo en el desarrollo, aumenta la portabilidad, ya que sólo son necesarios los requisitos mínimos de la Máquina Virtual para lanzar la aplicación.

Otra ventaja de la implementación de todas las fases de la reproducción multimedia en tiempo real, es la posibilidad de **trasladar los resultados a otras plataformas de desarrollo con un esfuerzo mínimo**. Esto nos permite no depender de un reproductor externo que funcione en unos dispositivos concretos. Por ejemplo, la aplicación para dispositivos ligeros, implementada en Java ME CLDC, podría ser portada a la plataforma Android de forma sencilla. La recepción y envío de los paquetes en tiempo real se realizaría usando

la librería *jLibRTP*, que nosotros hemos adaptado a Java ME. La decodificación y decodificación también seria automática, ya que las el formato de descompresión de sonido y audio se ha implementado en código Java ME y es compatible con Android. Por tanto, la única especificación a realizar en este dispositivo es cómo presentar las imágenes dentro de la interfaz de usuario o cómo escribir las muestras de sonido lineal en el altavoz; lo cual son procesos básicos dentro de cualquier plataforma.

Por otra parte, como se ha detallado en la tesis, **se han realizado diferentes desarrollos para dispositivos móviles ligeros y computadores de mano**. Los motivos para desarrollar aplicaciones en ambas versiones radican en adaptar las aplicaciones móviles a la potencialidad de cada dispositivo. Evidentemente, los computadores de mano, tales como PDAs, ofrecen una mejor capacidad de procesamiento que los teléfonos ligeros. En los resultados de la tesis se ha puesto en relieve hasta dónde pueden llegar los desarrollos multimedia para cada dispositivo. Por ejemplo, la reproducción de sonido en los teléfonos ligeros no ha permitido abrir el altavoz y reproducir los datos en vivo, sino que ha sido necesario finalizar el flujo del mismo para que el sonido sea escrito en el altavoz. Esta limitación intrínseca puede ser solventada parcialmente con estructuras circulares y varios reproductores concurrentes, pero sirve para poner en contexto las limitaciones que ofrecen en un dispositivo ligero las Máquinas Virtuales, y a las reducidas plataformas, respecto a un computador de mano.

Además, gracias a la integración de un Sistema Operativo en los dispositivos de mano, es posible solventar las limitaciones de las Máquinas Virtuales. Como hemos visto, hemos encontrado carencias en el acceso a las fuentes multimedia móviles, por ejemplo, en la reproducción de sonido o en la captura de imágenes desde la cámara. En nuestro caso, se han resuelto estas limitaciones para el Sistema Operativo Windows Mobile creando **librerías nativas que resuelven las carencias de la Máquina Virtual**. La portabilidad de nuestros desarrollos a otros Sistemas Operativos pasa por resolver estas limitaciones sobre reproducción y captura de la Máquina Virtual para

la nueva plataforma, siendo el resto de la aplicación portable. En este sentido, no sería necesario si quiera volver a compilar la aplicación, sólo debería de implementarse la funcionalidad de la interfaz JNI en el nuevo dispositivo de forma nativa.

Como vemos, las diferencias entre las Máquinas Virtuales de los dispositivos móviles ligeros y los computadores de mano radican, entre otras cosas, en la accesibilidad al micrófono, altavoz o cámara de los mismos. En el caso de los teléfonos ligeros, no es posible suplir las carencias creando librerías nativas, sino que hemos de confiar en las capacidades de la Máquina Virtual. Por ejemplo, si el dispositivo no puede acceder a la cámara, no podemos generar un flujo de vídeo desde los dispositivos ligeros.

En la tesis, se han mostrado los resultados de la recepción y transmisión para los computadores de mano y dispositivos ligeros. De ellos se deriva, de forma evidente, que el comportamiento es directamente proporcional a la capacidad de cómputo, la Máquina Virtual del dispositivo y el tiempo de respuesta para acceder a los dispositivos multimedia (cámaras, micrófonos o altavoces). Los protocolos y formatos usados han sido elegidos para minimizar los retardos de respuesta del sistema y poder ofrecer un flujo lo más cercano al tiempo real que se ha generado.

Ya hemos destacado que el retardo en las transferencias puede ser insignificante o crítico según el caso. Por ejemplo, reproducir desde un dispositivo móvil una fuente de sonido con un retardo de un segundo no es relevante, pero en el caso de las conversaciones con voz IP sí lo es. Respecto a la voz IP usando Máquinas Virtuales, la calidad de los resultados depende del dispositivo. En los modelos en que se han testado los desarrollos, sólo los computadores de mano ofrecen un comportamiento que permita una conversación sensata. En estos casos, hay que sumar el tiempo de recepción y transmisión que se han obtenido en los resultados experimentales, obteniendo un retardo de menos de un segundo. Para mejorar estos tiempos, siempre es posible adaptar estos resultados a una plataforma más potente como Android, implementaciones nativas, o simplemente usar un dispositivo con mayor capacidad.

5.1.3. Mapas Semánticos

El uso de mapas como herramienta de representación ha sido empleada por la sociedad desde los orígenes de la civilización. Establecer un **modelo para describir el espacio que nos rodea**, nos ha permitido compartir la localización y forma del entorno con otros humanos. En la actualidad, los mapas nos ayudan, por ejemplo, a establecer una ruta para el próximo destino de viaje desde un computador de sobremesa. La introducción de los dispositivos móviles ha servido para poder manejar los mapas desde la palma de la mano. Esto ha supuesto una importante revolución porque las rutas y entorno han podido integrarse, junto a la localización, gracias a los populares navegadores GPS.

En este trabajo se ha introducido una definición sobre el interior de los edificios que nos rodean, de forma que sirva como modelo para poder comunicar la información del Entorno Inteligente al operador humano. Esta visión es muy apropiada para la Inteligencia Ambiental o la *domótica*, que recordemos, tiene como finalidad la de comunicar e integrar los dispositivos de nuestro alrededor para ayudarnos en nuestras labores cotidianas. Como gran parte de nuestro tiempo y nuestras labores transcurren en edificios, nuestro objetivo es representarlos junto a los dispositivos que existan en el mismo.

En torno a esta temática surgen los Mapas Semánticos. Una primera parte de los mismos, ha integrado la representación gráfica de los objetos. En ella, hemos apostado por un modelo 2D cenital del entorno y que, a su vez, está dividido según la altura que representan las diferentes plantas del edificio. Una representación 3D hubiera dotado de mayor atractivo al sistema, pero su complejidad a la hora de representar los datos, y el uso de las Máquinas Virtuales como motor gráfico, desestimaron el uso del mismo.

La representación gráfica que hemos adoptado incluye una geometría euclídea. En concreto, se han definido las formas geométricas tales como polígonos y elipses, así como algunos casos específicos de éstos, como cuadrados o círculos. Aunque parezcan pocos elementos, con ellos se pueden definir prácticamente cualquier representación. El

uso de otras formas, permitiría ampliar las posibilidades gráficas, pero para los elementos básicos de un edificio, los que se han definido en la tesis, son suficientes. A su vez, la modificación de los elementos gráficos puede realizarse gracias a la inclusión de las transformaciones geométricas (escala, translación y rotación). Esto nos permite, operar con grupos de elementos de forma conjunta, como puede comprobarse en el uso del navegador que se ha incluido en la aplicación móvil.

Una carencia de los modelos completamente gráficos, es que aunque se incluyan métodos de selección como en este trabajo, es necesario conocer más propiedades de los objetos que están incluidos en el mapa. Para resolver esta limitación, se ha incluido una Ontología que etiqueta los elementos y añade propiedades en función de la clase a la que pertenecen. La definición semántica está definida como una Ontología en el sentido genérico de la misma, esto es, no se hace uso de formatos como OWL porque el tratamiento de la información no se realiza con razonadores, que son difícilmente mantenibles en los dispositivos móviles. Nuestra Ontología quiere establecer un marco de relación y propiedades entre las entidades comunes de los edificios, tales como habitaciones, puertas, plantas o escaleras. Para definir la instancia de cada edificio, el usuario puede introducir la información gráfica y semántica en el formato XML que se ha comentado en el capítulo anterior.

Durante el desarrollo de los traductores la Ontología de Edificios, que permiten leer una instancia del edificio y operar con ella en el dispositivo móvil, se reflexionó sobre la posibilidad de que el usuario introdujera etiquetas e información que no hubiese sido contemplada en la Ontología, de forma que en cada contexto se pudiera ampliar el modelo original y operar con nuevas propiedades. Sin embargo, la modificación de los objetos en tiempo de ejecución no es compatible con las Máquinas Virtuales de Java ME CLDC, lo que imposibilita introducir nuevas etiquetas en tiempo de ejecución. Además, introducir la reflexión en los Mapas Semánticos se aleja de los propósitos del trabajo, pero puede ser un trabajo futuro interesante en computadores de mano.

Aunque el razonamiento de la Ontología con motores lógicos

no puede realizarse en los dispositivos de mano debido a la falta de razonadores móviles, en este trabajo se ha introducido una operación que permite calcular la ruta entre los elementos de la Ontología basándose en el **Algoritmo A***. El Cálculo de Rutas permite a los usuarios conocer el camino desde un punto del Mapa Semántico a otro, respetando los enlaces entre puertas, habitaciones y elementos comunicantes. Esto es muy útil para ayudar al operador humano a moverse por espacios que no conoce y ayudarle a encontrar sitios de interés, destacando sus aplicaciones para turismo y comercio.

El estudio de A* se ha realizado en profundidad y se ha vinculado al algoritmo de Dijkstra como un caso concreto del mismo. A su vez, se ha introducido una **nueva heurística que reduce los tiempos de búsqueda y que está basada en la distribución geográfica y la minimización de los ángulos que comunican la linea recta entre el origen y el destino**. El uso de A* en ambas heurísticas determina que tanto las distancias como los ángulos responden a la misma complejidad en el peor de los casos, aunque los resultados en grafos distribuidos homogéneamente destaca el mejor comportamiento de los ángulos.

Además de un modelo para el tratamiento de información del espacio dentro de los dispositivos móviles, **los mapas pueden representar un marco para comunicar cambios en el ambiente**. En este sentido, hemos propuesto los Mapas Semánticos como estructura de intercambio de datos entre el operador humano y el Entorno Inteligente. Si hablamos de intercambio de información desde los dispositivos móviles, es interesante plasmar la comunicación con Servicios Remotos que puedan ser accedidos o desplegados en los computadores de mano, de forma que la interacción entre el Entorno Inteligente y los usuarios sea en tiempo real, ubicua y bidireccional.

La creación de Servicios Remotos que actualicen y distribuyan la información desde los Mapas Semánticos puede ser innumerable. Esto significa que, depende del contexto donde se instale el sistema pueden desarrollarse unos servicios u otros, por ejemplo, mostrar la localización de personal de seguridad o notificar la ruta hasta el restaurante que acabamos de contratar. Sin embargo, hemos

seleccionado un par de casos de uso genéricos que se han desarrollado y que sirven para ilustrar la interacción entre el Entorno Inteligente y los dispositivos móviles: la Localización y la Notificación en Tiempo Real.

En la Localización se describe a los dispositivos como receptores de los cambios de localización detectados por el Entorno Inteligente. A su vez, **se incluye a los dispositivos como fuentes de datos, permitiendo al usuario enviar su localización explícitamente o responder a la petición de localización por parte del sistema**. Así, los mapas sirven para mostrar los diferentes usuarios o para resolver la localización de los mismos con una pulsación en la pantalla. La distribución de los cambios sobre localización se realiza mediante Canales de Eventos, que permiten al Entorno Inteligente acceder a los Servicios Remotos de los dispositivos móviles de los usuarios. De esta forma, el sistema puede propagar las nuevas localizaciones o solicitar la localización a uno o varios usuarios del canal.

Por otro lado, la Notificación se ha desarrollado como un mecanismo asíncrono de eventos que son mostrados en los Mapas Semánticos. Este tipo de arquitectura es más sencilla que la Localización, porque, como podemos observar, los clientes móviles sólo son vistos como sumideros de información. Las notificaciones son descritas en el Mapas Semántico mediante una ruta impresa en el mismo y enfatizando los objetos destino a donde acudir, además de incluir otros parámetros para describir el evento. Las Notificaciones son útiles en sistemas de vigilancia o para publicitar el espacio que nos rodea.

5.2. Trabajo Futuro

La línea de trabajo desarrollada en esta tesis pretende establecer un marco donde los dispositivos móviles se conectan con el ambiente y entre sí. En concreto, la comunicación transmite recursos multimedia, pudiendo establecer conexiones de vídeo o sonido ubicuas. De forma paralela, otro tipo de información, como la espacial es integrada como eje de comunicación entre el Entorno Inteligente y los usuarios;

creando un sistema de interacción entre los dispositivos ambientas y móviles.

Estos objetivos abstractos se han concretado en una propuesta técnica que vino determinada por otros requisitos no funcionales, como la portabilidad de la solución y el desarrollo íntegro de las capas de comunicación. La elección de la plataforma Java ME cuya ejecución recae en las Máquinas Virtuales de los dispositivos, limita los resultados y potencia del sistema respecto a otras soluciones, pero otorga la posibilidad de ejecutar la aplicación en más dispositivos.

La mejora de los resultados multimedia expuestos en la tesis pasa por dos vertientes: la codificación de audio/vídeo y la plataforma de desarrollo. Los protocolos a bajo nivel no pueden ser más óptimos para el tiempo real (UDP y RTP), por lo que las mejoras deben referirse a la codificación y decodificación multimedia. A su vez, **el protocolo RTP podría ampliarse a RTCP**, ampliando notificaciones de control y ajustar el flujo multimedia durante la transmisión. Este control podría realizarse usando el propio Middleware y dotaría de una mayor calidad del servicio y de dinamismo a la transmisión del flujo en tiempo real.

En esta tesis la constante ha sido mantener una implementación ajustable a las limitaciones computaciones de las Máquinas Virtuales, pero en otros contextos futuros, donde los desarrollos pueden realizarse de forma nativa, deberían de estudiarse otras codificaciones que redujeran el ancho de banda consumido o los tiempos de respuesta del sistema. La codificación de vídeo debe ser más estudiada ya que el formato M-JPEG es muy básica, pero requerirán de mayor potencia para traducir las codificaciones. Serán los trabajos futuros quienes evalúen las capacidades de los últimos formatos, como VP8, dentro de las reproducciones en tiempo real y al vuelo en los dispositivos móviles.

Además, en la tesis se ha hecho énfasis en la **homogeneidad de los flujos multimedia respecto los dispositivos que los generan, tanto ambientales como móviles**. Gracias a esta homogeneidad un cliente móvil puede reproducir una cámara estática o la cámara de un móvil de forma transparente. Para ello, los dispositivos móviles deben

generar la misma codificación como emisores, que la que reciben como receptores. Es importante, por tanto, evaluar si los formatos modernos sirven para ofrecer un flujo multimedia desde los dispositivos móviles en tiempo real, sobre todo los asociados al vídeo que necesitan de mayor capacidad de cómputo que los referentes al audio.

Los tiempos de respuesta y retardos son críticos en algunos casos. Algunas aplicaciones móviles recientes permiten la transmisión de vídeo desde los dispositivos en tiempo real; sin embargo los retardos producidos entre la emisión y la recepción responden a varios segundos. Este comportamiento es inaceptable si pensamos en esta tecnología para vídeo conferencias. Es importante por tanto, estudiar los retardos de los formatos respecto a las transmisiones en tiempo real y en *streaming*, y no sólo como reproductores estáticos de archivos.

Por otro lado, como hemos comentado, la recepción y decodificación de los flujos multimedia podría haberse realizado usando terceras aplicaciones, que delegaran la responsabilidad en soluciones nativas que no pudieran extrapolarse a otras plataformas. El desarrollo de estas capas de comunicación fue evaluado positivamente como un requisito no funcional porque facilita el trabajo futuro para **portar la aplicación a otros Sistemas Operativos o lenguajes**, como C++ o Android. Es aquí donde pueden seguir realizándose mejoras en los trabajos futuros. De hecho, gracias a contar con la implementación en una plataforma limitada como Java ME, la portabilidad podría realizarse con una traducción del código, que a priori debe ser más sencilla que incluso que en Java ME debido a la restricción de herramientas respecto a otras plataformas. Además, existen herramientas que permiten realizar este proceso de forma semiautomática, [xmlvm, 2003].

La interacción con los Entornos Inteligentes y los dispositivos móviles representa un marco de trabajo cuya evolución es muy amplia. A diferencia de los desarrollos en flujos multimedia ubicuos, donde las fases están bien estudiadas y pueden desarrollarse mejoras concretas, la Inteligencia Ambiental y la Computación Ubicua presentan un campo de trabajo extenso donde se realizarán trabajos futuros en sectores muy diferentes. Los perfiles más relacionados con los trabajos de esta tesis hacen referencia a sistema de integración de dispositivos

multimedia y la comunicación entre el sistema y los computadores de mano.

La comunicación en la Inteligencia Ambiental deberá converger hacia una filosofía orientada a componentes, de forma que se puedan instalar y conectarse de forma automática dispositivos electrónicos. El Middleware usado en esta tesis ofrece un marco inmejorable para conectar diferentes plataformas y dispositivos empotrados, incluso para recursos con capacidades mínimas a diferencia de otras soluciones como OSGi [Marples and Kriens, 2001]. Si bien es cierto, que deberían de respetarse los **formatos abiertos y estandarizados**, al estilo de *Universal Plug and Play* (UPnP) o *Jini*, que permitieran a los Middlewares integrar diferentes componentes heterogéneos de forma automática.

Dejando a un lado las capas de comunicación, será crucial la integración de **inteligencia** en la domótica. La inclusión de inteligencia debe realizarse ahora que los dispositivos electrónicos empiezan a ser comunes en nuestro día a día. De ello depende que el término Inteligencia Ambiental no se degrade y acabe convirtiéndose en una quimera. Por tanto, los avances tecnológicos deberían estar acompañados por avances inteligentes.

En el trabajo expuesto sobre localización de interiores en esta tesis, los esfuerzos se han centrado en la movilización de la información. En concreto, se ha estudiado la propagación de las localizaciones, así como la solicitud y envío de nuevas posiciones por parte de dispositivos y usuarios. Por ejemplo, un tema a resolver en muchos escenarios de localización hace referencia a la identificación de los usuarios. Muchas veces, los sensores de localización son disparados concurrentemente y provocan falsos positivos que deben procesarse. Estos datos deberían preprocesarse con un modulo inteligente que tuviera en cuenta, por ejemplo, las trayectorias de los usuarios y los eventos que sean imposibles de haberse producido.

La inteligencia puede aplicarse de forma muy potente en escenarios locales, pudiendo estudiarse situaciones de complejidad importante. Por ejemplo, para **detectar situaciones de peligro** como

atropellos o intrusiones [Castro et al., 2010]. Los resultados de los análisis que evalúa el Entorno Inteligente deben ser comunicados a los usuarios, para ello, hemos visto en la tesis cómo enviar notificaciones integradas en los Mapas Semánticos, de forma que el operador humano pueda conocer visualmente las zonas afectadas y descripción de los eventos.

En este sentido, sería importante trasladar los módulos inteligentes a los ambientes domésticos de forma popular. *El verdadero progreso es el que pone la tecnología al alcance de todos*, [Henry, 1947]. El desarrollo de módulos electrónicos e inteligentes, que incorporen hardware y software, potenciarían el interés comercial para integrar estas tecnologías en nuestro día a día. Particularmente, creemos que los campos más atractivos a nivel particular serían la seguridad y la visualización de recursos multimedia dentro del hogar. Por otro lado, los comercios podrían ofrecer Entornos Inteligentes destinados a comunicarse con los usuarios y ofrecer publicidad orientada.

La tecnología descrita en las Fuentes Multimedia Ubicuas podría estar fuertemente vinculada a las **Redes Sociales** en un futuro. El concepto de Redes Sociales surgió para definir la estructura de personas que están relacionadas entre sí mediante un vínculo como la amistad, el trabajo o cualquier otro vínculo. Hoy día, las Redes Sociales han disfrutado de un importante auge a partir de aplicaciones informáticas que permiten compartir información con un círculo de personas vinculadas entre sí. Las críticas más importante provienen sobre el control de la privacidad a la hora de propagar esa información sobre las personas de cada círculo, de forma que pueden establecerse niveles de visibilidad según los grados de separación entre unas personas y otras.

Además, la ubicuidad en las Redes Sociales está muy valorada, ya que permite compartir contenido desde cualquier localización y cualquier momento. Recientemente han surgido novedosas aplicaciones móviles para publicar información desde computadores de mano en las redes. Desde nuestro punto de vista, la integración de las Fuentes Multimedia Ubicuas en las Redes Sociales permitiría comunicar a los usuarios, no sólo mediante mensajes e imágenes estáticas,

sino mediante voz IP o vídeo conferencias, a priori gratuitas.

En este sentido, parece sensato que todas las Redes Sociales que hemos creado con diferentes herramientas converjan en servicios unificados. En un futuro, podríamos integrar desde la agenda de contactos del móvil a los remitentes y destinatarios de nuestro correo electrónico dentro de Redes Sociales, de forma que pudiéramos enviarles mensajes de correo o móvil, publicar una fotografía o realizar una vídeo llamada a un grupo de nuestra red.

Dentro de este mundo virtual integrado, pensamos que sería útil incluir otros elementos, no humanos pero sociales, dentro de la red. Hoy día, éstas hacen referencia a las relaciones persona-persona, pero también estamos vinculados a otros elementos, como el entorno. Por ejemplo, **la integración de los espacios públicos o privados dentro de las Redes Sociales** pueden dar a conocer los edificios a los usuarios de forma remota. Por ejemplo, haciendo referencia a los trabajos de esta tesis, podrían publicarse los Mapas Semánticos y recursos multimedia ambientales de los edificios o a las personas que trabajen en los mismos. También, en el ámbito particular doméstico, podrían incluirse otros aparatos domésticos dentro de los recursos multimedia, tales como televisiones o cámaras, de forma que la red de dispositivos multimedia de un hogar sirviera para comunicar a las personas que los habitan con otros usuarios.

En referencia a las posibilidades de comunicación, tanto los recursos multimedia ambientales como los móviles, nos permitirán comunicar a las personas con los edificios y con otras personas. Es evidente que esto genera un problema de privacidad y visibilidad como el que hemos comentado en las Redes Sociales, porque por un lado es necesario publicar estos Servicios, de forma que otros usuarios puedan comunicarse con nosotros, pero por otro lado no deben estar disponibles para cualquier persona. Quizá las propias Redes Sociales, aunque han introducido los peligros de la privacidad, también puedan ayudarnos a resolverlos. Por ejemplo, podrían usarse los grados de separación para restringir el acceso a las cámaras de nuestro hogar. Así, según los casos, las personas en función de su vínculo, esto es, cercanía en grado de separación, podrían o no realizar vídeo llamadas usando

los dispositivos móviles y ambientales. Incluso esta restricción podría granularizarse por habitaciones, de forma que algunas zonas fuesen menos restrictivas que otras.

Glosario

A

Aibo Robot con forma de perro dotado de sensores y actuadores, así como módulos que facilitan la comunicación para solicitarle tareas., p. 55.

Algoritmo A* Algoritmo de búsqueda en grafos que permite encontrar un camino entre entre un nodo origen y destino en función de una heurística que guía y minimiza la búsqueda., p. 156.

Android Sistema Operativo móvil desarrollado por Google., p. 17.

AR Augmented Reality o Realidad Aumentada. Tecnología capaz de recoger la visualización procedente del mundo físico, y sobreponer sobre ella meta datos informáticos asociados a los objetos reconocidos en tiempo real, p. 29.

B

Bluetooth Tecnología de comunicación inalámbrica que permite el intercambio de datos entre dispositivos a corta distancia., p. 19.

bytecode Código binario que codifica las operaciones y argumentos de una aplicación que será ejecutada sobre una Máquina Virtual., p. 41.

C

CAD Herramientas que facilitan el dibujo y diseño asistido por computadora., p. 137.

Canal de Eventos Comunicación entre un administrador y varios clientes. Está basada en la propagación de información a varios clientes (multicast) o a todos los clientes (broadcast), en función del criterio del administrador. Los clientes pueden suscribirse y desuscribirse a la comunicación notificando al administrador en tiempo de ejecución., p. 69.

Computación Ubicua Modelo que traslada los procesos de computación en los dispositivos de nuestro día a día., p. 14.

computadores de mano Dispositivo móvil orientado a ofrecer una amplia conectividad y potentes prestaciones más allá de la telefonía., p. 37.

Cálculo de Rutas Proceso que resuelve el camino entre dos elementos que están comunicados entre sí dentro de una malla de nodos conectados., p. 6.

códec Especificación que transforma un flujo multimedia para obtener datos primitivos de audio o vídeo. Se basan en la pérdida de información para reducir el tamaño resultante. En general, podemos hablar de "decodificación" para recuperar los datos primitivos o "codificación" para obtener los comprimidos., p. 62.

D

dispositivos ligeros Dispositivo móvil con poca capacidad de cómputo y prestaciones limitadas., p. 37.

Domótica Sistemas para la automatización de viviendas capaz de comunicarse con los sensores y actuadores del ambiente para facilitar las labores humanas., p. 18.

E

Entorno Inteligente Sistema de monitorización que evalúa el estado del espacio a partir de sensores ambientales para guiar e informar a los usuarios que habitan en el entorno en la realización de sus labores cotidianas., p. 6.

F

flujo multimedia Flujo de datos multimedia. En esta tesis, hace referencia al envío de datos, de vídeo o audio, bajo protocolos ligeros y sensibles al tiempo real., p. 5.

formato En este trabajo es usado como sinónimo de códec. Véase códec. En algunos casos, en la literatura, hace referencia a un conjunto de códec, por ejemplo de audio y vídeo, para realizar la compresión de sonido e imágenes conjuntamente., p. 36.

formato lineal Codificación del sonido que representa una onda sonora analógica mediante una cuantificación digital. Corresponde a un muestreo de los datos a una frecuencia , tamaño y canal de la muestras determinados. Este formato no comprime los datos, a diferencia de G.711., p. 62.

G

G.711 Estándar libre para la codificación de audio., p. 64.

GNU GPL Licencia destinada a proteger la libre distribución, modificación y uso de software., p. 43.

GPS Global Positioning System. Sistema que permite obtener la posición (longitud y latitud) mediante un dispositivo que recibe la señal de unos satélites que orbitan para este fin., p. 29.

H

H.264 Norma que define el códec de vídeo de alta compresión conocido más popularmente como MP4., p. 26.

I

Inteligencia Ambiental Paradigma en el que los objetos de nuestro alrededor cooperan para ayudar a realizar las labores cotidianas humanas., p. 18.

IPhone Dispositivo y Sistema Operativo móvil desarrollado por Apple., p. 17.

J

Java ME CDC Configuración de Java ME orientada al desarrollo de computadores de mano sustentada por una Máquina Virtual con mayor potencialidad., p. 49.

Java ME CLDC Configuración de Java ME orientada al desarrollo de dispositivos ligeros y con poca capacidad de cómputo., p. 47.

Java Micro Edition Plataforma basada en Java para el desarrollo de aplicaciones para dispositivos móviles y empotrados., p. 8.

Java Specification Requests Especificaciones de algunas características y paquetes, que pueden ser opcionales, dentro de una Máquina Virtual Java., p. 43.

jLibRTP Java Library RTP. Librería libre en Java para el manejo de sesiones en tiempo real bajo el protocolo RTP., p. 106.

JMF Java Media Framework. Librería que permite el tratamiento de formatos, protocolos y dispositivos multimedia bajo la plataforma Java. La versión *cross-all* es independiente del Sistema Operativo., p. 92.

JNI Java Native Interface. Componente de una Máquina Virtual que permite realizar llamadas a librerías nativas desde la aplicación Java. La especificación de las operaciones nativas se definen como métodos y funciones., p. 113.

K

Kilo Virtual Machine KVM. Máquina Virtual compacta para dispositivos ligeros cuya funcionalidad no puede ampliarse con librerías nativas., p. 52.

L

lenguaje compilado Lenguaje de programación donde las instrucciones se traducen a código máquina para una ejecución óptima., p. 41.

librería nativa Componente software que desarrolla una funcionalidad concreta y que ha sido desarrollada específicamente para un Sistema Operativo., p. 49.

M

M-JPEG Motion- JPEG Compressed Video. Codificación donde cada frame es codificado de forma independiente en formato JPEG., p. 63.

Mapas Semánticos Mapas que incluyen a la definición gráfica la estructuración de los elementos que vienen determinados en ellos., p. 7.

Middleware Software que abstrae el despliegue y solicitud de servicios distribuidos con independencia de la plataforma de desarrollo y protocolo de comunicación., p. 8.

MIDP Perfil de Java ME CLDC que incluye el soporte de interfaz de usuario MIDlet para dispositivos móviles ligeros., p. 48.

MMAPI Mobile Media API. Librería que permite reproducir flujos de vídeo y audio en JavaME CLDC y cuya potencialidad depende de la Máquina Virtual., p. 20.

multicast Multidifusión. Envío de un paquete de datos a varios destinatarios.

Máquina Virtual Software que representa a una máquina física, de forma que puede ejecutar aplicaciones como si fuese una computadora real., p. 7.

P

Personal Profile Perfil de Java ME CDC que incluye el soporte de interfaz de usuario AWT y varios paquetes opcionales para computadores de mano., p. 50.

PhoneME Proyecto libre para expandir el uso de Java ME en dispositivos móviles y empotrados., p. 44.

R

RTP Protocolo situado en la capa de sesión que incluye una huella de tiempo y otras características que lo hacen adecuado a las transmisiones en tiempo real., p. 57.

RTSP Protocolo que permite iniciar, configurar o parar una transmisión RTP (véase RTP), incluyendo el recurso y formatos con que generar el flujo multimedia., p. 83.

S

Secure Sockets Layer Interfaz implementada por un objeto que puede ser accedida de forma remota por otros objetos., p. 69.

Servicios Remotos Interfaz implementada por un objeto que puede ser accedida de forma remota por otros objetos., p. 66.

SIP Servicio remoto que permite iniciar, modificar o finalizar sesiones de voz en tiempo real., p. 23.

streaming Tratamiento de flujos multimedia en continuo, de forma que la recepción y tratamiento de los datos es concurrente y no es necesario a que finalice la transmisión para comenzar la reproducción., p. 12.

T

transformaciones geométricas Operaciones que permiten obtener una nueva figura a partir de una previamente dada. Las más importantes dentro la geometría son: la Translación, la Rotación y la Escala, p. 140.

U

UTM Coordenadas Universal Transversal de Mercator. Sistema de coordenadas basado en la proyección del globo terrestre y que incluye unas divisiones para minimizar las deformaciones alejadas del ecuador., p. 146.

V

voz ip Sistema para la señal de la voz que la digitaliza y envía por la red empleando un protocolo IP., p. 23.

VP8 Códec de vídeo libre y de alta compresión., p. 26.

W

WAV WAVEform audio format. Formato de sonido que incluye una pequeña cabera que definen las muestras de sonido, y que concluye con el propio sonido en formato lineal (véase formato lineal)., p. 118.

Windows Mobile Sistema Operativo móvil desarrollado por Microsoft, ahora llamado Windows Phone., p. 17.

Bibliografía

[Ahonen, 2010] Ahonen, T. (2010). Mobile phone market shares for year of 2009 and last quarter 2009. *Communities Dominate Brands*.

[Amini et al., 1998] Amini, P., Weeks, J., and Raemy, F. (1998). Pure-java jpeg-encoder for limited java apis (e.g. j2me). *BioElectroMech*.

[Baratz, 1996] Baratz, A. (1996). Javasoft ships java 1.0. *Sun Microsystems*.

[Barfield and Caudell, 2001] Barfield, W. and Caudell, T. (2001). Fundamentos de informática usable y realidad aumentada. *Mahwah, NJ: Lawrence Erlbaum*.

[Berc et al., 1996] Berc, L., Fenner, W., Frederick, R., and McCanne, S. (1996). Rfc2035: Rtp payload format for jpeg-compressed video. *RFC from Internet Society*.

[Bobillo et al., 2008] Bobillo, F., Delgado, M., and Gomez-Romero, J. (2008). Representation of context-dependant knowledge in ontologies: A model and an application. *EXPERT SYSTEMS WITH APPLICATIONS*.

[Bruck et al., 2009] Bruck, J., Gao, J., and Jiang, A. (2009). Localization and routing in sensor networks by local angle information. *ACM Transactions on Sensor Networks*, 5.

[Caizzone et al., 2008] Caizzone, G., Corghi1, A., Giacomazzi, P., and Nonnoi1, M. (2008). Analysis of the scalability of the overlay skype system. *IEEE Communications Society*.

[Castro et al., 2010] Castro, J., Delgado, M., Medina, J., and Ruiz-Lozano, M. (2010). Expert fuzzy system for prediction of pedestrian accidents. *III Simposio sobre Lógica Fuzzy y Soft Computing, LFSC2010*.

[Cept, 2002] Cept (2002). Global system for mobile communications. *European Conference of Postal and Telecommunications Administrations*.

[Cha et al., 2003] Cha, H., Lee, J., Nang, J., Park, S., Jeong, J., Yoo, C., and Choi, J. (2003). A video streaming system for mobile phones: Practice and experience. *WIRELESS NETWORKS. 5th Conference on European Personal Mobile Communications. GLASGOW, SCOTLAND.*

[Cheng et al., 2006] Cheng, R., Zhang, Y., Bertino, E., and Prabhakar, E. (2006). Preserving user location privacy in mobile data management infrastructures. *Privacy enhancing technologies. 6th International workshop.*

[CMT, 2009] CMT (2009). Comision del mercado de las telecomunicaciones.

[Community, 2008] Community, S. F. (2008). Symbian. *http://www.symbian.org/*.

[Corp., 2002] Corp., S. (2002). Superwaba. *http://www.superwaba.com/*.

[Delgado et al., 2007] Delgado, M., Medina, J., Ruiz, M., and Vila, A. (2007). Integrating signals from different cameras for video-surveillance mobilization. *2nd International Symposium on Ubiquitous Computing and Ambient Intelligence*.

[Delgado et al., 2009] Delgado, M., Ros, M., and Vila, M. (2009). Correct behavior identification system in a tagged world. *EXPERT SYSTEMS WITH APPLICATIONS*.

[Dersch, 2009] Dersch, H. (2009). Pure-java jpeg-decoder for limited java apis (e.g. j2me). *HFU Furtwangenl*.

[Dijkstra, 1959] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *In Numerische Mathematik, S. 269-271*.

[Even, 2006] Even, R. (2006). Rfc4587 - rtp payload format for h.261 video streams. *RFC from Internet Society*.

[Feiner et al., 1993] Feiner, S., MacIntyre, B., and Sellgmann, D. (1993). Knowledge-based augmented reality. *Communications of the ACM 36(7)*.

[Fraile et al., 2008] Fraile, J., Bajo, J., Abraham, A., and Corchado, J. (2008). Hoca home care multi-agent architecture. *Advances in Soft Computing*.

[Gehlen et al., 2006] Gehlen, G., Aijaz, F., and Walke, B. (2006). Mobile web service communication over udp. *IEEE*.

[Google, 2005] Google (2005). Google talk. *http://www.google.com/talk/*.

[Google, 2009] Google (2009). Latitude. *www.google.com/latitude*.

[Gordon and Talley, 1999] Gordon, R. and Talley, S. (1999). Essential jmf - java media framework. *Prentice Hall*.

[Greenfield, 2004] Greenfield, A. (2004). Everyware: The dawning age of ubiquitous computing.

[Gregory, 2002] Gregory, C. (2002). Streaming media: Building and implementing a complete streaming system. *Edit. Wiley*.

[Gualdi et al., 2008] Gualdi, G., Prati, A., and Cucchiara, R. (2008). Video streaming for mobile video surveillance. *IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 10, NO. 6*.

[H. et al., 2000] H., H. C., Lin, Y., and Cho, S. (2000). igsm: Voip service for mobile networks. *IEEE Communications Magazine*.

[Ham and Feher, 2007] Ham, J. Hunstable, B. and Feher, G. (2007). Ustream. *http://www.ustream.tv/*.

[Handley and Jacobson, 1998] Handley, M. and Jacobson, V. (1998). Rfc2327: Session description protocol. *RFC from Internet Society*.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*.

[Henning, 2004] Henning, M. (2004). A new approach to object-oriented middleware. *IEEE Internet Computing, 8, 6675.*

[Henry, 1947] Henry, F. (1947). El verdadero progreso es el que pone la tecnología al alcance de todos. *Cita*.

[Hiwasaki and Ohmuro, 2009] Hiwasaki, Y. and Ohmuro, H. (2009). Itu-t g.711.1: Extending g.711 to higher-quality wideband speech. *IEEE Communications Magazine*.

[Hiwasaki et al., 2006] Hiwasaki, Y., Ohmuro, H., Mori, T., Kurihara, S., and Kataoka, A. (2006). A g.711 embedded wideband speech coding for voip conferences. *IEICE TRANSACTIONS ON INFORMATION AND SYSTEMS*.

[Jiménez, 2003] Jiménez, A. (2003). ¿por qué pagamos tanto si podemos hablar gratis? *El Mundo*.

[Lantau, 2000] Lantau, G. (2000). ffmpeg. *http://ffmpeg.org/*.

[Lee et al., 2008a] Lee, R.-G., Chen, K.-C., Hsiao, C.-C., and Tseng, C.-L. (2008a). A mobile care system with alert mechanism. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE*.

[Lee et al., 2008b] Lee, R.-G., Hsiao, C.-C., Chen, K.-C., and Liu, M.-H. (2008b). An intelligent diabetes mobile care system with alert mechanism. *BIOMEDICAL ENGINEERING-APPLICATIONS, BASIS AND COMMUNICATIONS*.

[Li et al., 2002] Li, S., Hsieh, H.-C., L.S., S., and W.S., C. (2002). Pda watch for mobile surveillance services. *IEEE Workshop on Knowledge Media Networking*.

[M et al., 2006] M, D., Gherbi, A., Mourad, A., and Yahyaoui, H. (2006). A selective dynamic compiler for embedded java virtual machines targeting arm processors. *Science of Computer Programming*.

[Marples and Kriens, 2001] Marples, D. and Kriens, P. (2001). The open services gateway initiative: An introductory overview. *Ieee Communications Magazine*.

[Medina and Ruiz, 2007] Medina, J. and Ruiz, M. (2007). Architecture for databases access and consultation through handheld devices. *2nd International Symposium on Ubiquitous Computing and Ambient Intelligence*.

[Menkens et al., 2007] Menkens, C., Kjellin, N., and Davoust, A. (2007). Ims social network application with j2me compatible push-to-talk service. *International Conference on Next Generation Mobile Applications*.

[Mobilizy, 2009] Mobilizy (2009). Wikitude ar travel guide. *http://www.wikitude.org/*.

[Montoya and Piedrahita, 2005] Montoya, E. and Piedrahita, T. (2005). Performance analysis of jxta/jxme applications in hybrid fixed/mobile environments. *EAFIT University, Medellin, Colombia*.

[Moya et al., 2009] Moya, F., Villa, D., Villanueva, F., Rincon, F., Barba, J., and JC., L. (2009). Embedding standard distributed object-oriented middlewares in wireless sensor networks. *Journal on Wireless Communications and Mobile Computing*.

[Muchow, 2002] Muchow, J. W. (2002). *Core J2ME Technology and MIDP*. Prentice Hall.

[Muftah and Mustafa, 2006] Muftah, A. and Mustafa, B. (2006). Smart surveillance using pda. *Masters thesis, Universiti Teknologi Malaysia*.

[OpenWengo, 2005] OpenWengo (2005). Qutecom. *http://www.qutecom.org/*.

[Oracle and Java, 2007] Oracle and Java (2007). phoneme project. *https://phoneme.dev.java.net/*.

[Orfali, 2002] Orfali, R. (2002). Cliente/servidor. guia de supervivencia. *Edt. McGraw-Hill*.

[Ovum, 2005] Ovum (2005). Estudio de j2me. *Datamonitor Group*.

[Peshkin and Sanderson, 1985] Peshkin, M. A. and Sanderson, A. C. (1985). Reachable grasps on a polygon: The convex rope algorithm. *tech. report CMU-RI-TR-85-06, Robotics Institute*.

[Piontek et al., 2007] Piontek, H., Seyffer, M., and Kaiser, J. (2007). Improving the accuracy of ultrasound-based localisation systems. *PERSONAL AND UBIQUITOUS COMPUTING*.

[Postel, 1980] Postel, J. (1980). User datagram protocol.rfc 768. *RFC from Internet Society*.

[Pous et al., 2005] Pous, M., Foster, G., and Pesch, D. (2005). Performance evaluation of sip-based multimedia services in umts. *IEEE Wireless Communications*.

[Ramdoyal, 2003] Ramdoyal, R. (2003). Generic pda based museum guide with sound enhancement. *Masters thesis, FacultÃ©s Universitaires Notre-Dame de la Paix*.

[Reventos, 2009] Reventos, L. (2009). La regulación española impide el desarrollo de la vozip. *El Pais*.

[Rosenberg, 2002] Rosenberg, J. e. a. (2002). Rfc3261: Session initiation protocol. *RFC from Internet Society*.

[Russell and Norvig, 2003] Russell, S. and Norvig, P. (2003). Inteligencia artificial: un enfoque moderno. *Ed. Prentice Hall*.

[Schulzrinne et al., 2003] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (2003). Rtp: A transport protocol for real-time applications. rfc 3550. *RFC from Internet Society*.

[Serpa and Rodriguez, 2007] Serpa, F. and Rodriguez, L. (2007). Streaming de audio a traves de dispositivos moviles utilizando j2me. *Universidad El Bosque*.

[Shechter, 2006] Shechter, A. (2006). Fring. *fringland Ltd.*

[Slama et al., 1999] Slama, D., Garbis, J., and Russel, P. (1999). Enterprise corba. *Edt. McGraw-Hill*.

[Spencer, 2002] Spencer, M. (2002). Asterisk. *http://www.asterisk.org/*.

[Theora, 2004] Theora (2004). Vp3. *On2 Technologies*.

[Tierno and Campo, 2005] Tierno, J. and Campo, C. (2005). Smart camera phones:limits and applications. *IEEE PERVASIVE COMPUTING*.

[Trias, 2003] Trias, J. (2003). Geometria para la informática gráfica y cad. capítulo *Matrices ampliadas*. *Edicions UPC*.

[Waba, 1999] Waba (1999). Wabasoft inc. *http://www.wabasoft.com/*.

[Wagner et al., 2005] Wagner, D., Pintaric, T., Ledermann, F., and Schmalstieg, D. (2005). Towards massively multi-user augmented reality on handheld devices. *PERVASIVE COMPUTING, PROCEEDINGS*.

[Weiser, 1991] Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265:94–104.

[Wenger et al., 2005] Wenger, S., Hannuksela, M., Stockhammer, T., Westerlund, M., and Singer, D. (2005). Rfc3984 - rtp payload format for h.264 video. *RFC from Internet Society*.

[Wu et al., 2005] Wu, B., Peng, H., Chen, C., and Chan, Y. (2005). An encypted mobile embedded surveillance system. *IEEE*.

[xmlvm, 2003] xmlvm (2003). xmlvm. *http://xmlvm.org/*.

[Yamada and Kamioka, 2005] Yamada, S. and Kamioka, E. (2005). Access control for security and privacy in ubiquitous computing environments. *IEICE Transactions On Communications*, E88B:846–856.

[yCad, 2005] yCad (2005). Java cad library. *http://sourceforge.net/projects/ycad/*.

[Yoo et al., 2008] Yoo, H., Yoon, J., Lee, S., Park, H., Go, K., Kim, M., and Lee, K. (2008). A mobile reservation protocol for video streaming. *IEEE*.

[Zelkha and Epstein, 1998] Zelkha, E. and Epstein, B. (1998). From devices to ambient intelligence. *Digital Living Room Conference*.

[Zhang et al., 2008] Zhang, H., Wang, X., Guo, L., Zou, C., Lv, Y., and Xiao, B. (2008). Data receiving method for mobile multimedia system, involves receiving multiplexing frames, decoding stream media packets in multiplexing frames, and decoding packets by decoding module according to time information of packets. *IEEE*.

[Zhao, 2002] Zhao, Y. (2002). Standardization of mobile phone positioning for 3g systems. *IEEE Communications Magazine*.

[Zopf, 2002] Zopf, R. (2002). Rfc3389: Real-time transport protocol (rtp) payload for comfort noise (cn). *RFC from Internet Society*.