MÉTODOS DE CONTROL EN SISTEMAS DOMÓTICOS: ÚLTIMAS TENDENCIAS EN SISTEMAS DISTRIBUIDOS

(CONTROL METHODS FOR INTELLIGENT HOMES SYSTEMS: NEW TRENDS IN DISTRIBUTED SYSTEMS)

Isaac Moreno Navarro, Emiliano Martín Candelario y Marina Álvarez Alonso
Facultad de Informática, Dpto. de Lenguajes y Sistemas (UPM)

ESPAÑA

Fecha de recepción: 30-XII-98
106-9

RESUMEN

Uno de los aspectos más delicados a la hora de diseñar un sistema domótico es el que se refiere al modelo de control que deseamos establecer.

Podemos considerar dos tendencias clásicas: la que trata de establecer un control centralizado del sistema y la que propone un control mucho más descentralizado, donde cada aparato sería responsable de cierta parte de la gestión del control ejercido sobre toda la red.

En este artículo tratamos de profundizar en las últimas novedades aparecidas en el campo de la programación de los controladores asociados a una red domótica, en particular Java y el más reciente Jini, que aportan una serie de soluciones que facilitan la compatibilidad entre dispositivos de distintos fabricantes y apuestan claramente por una arquitectura de tipo distribuido, dotando a las redes domóticas de una flexibilidad y un potencial de crecimiento inusitados.

SUMMARY

Control systems lie amongst the most difficult issues to be tackled when it comes to designing home automation systems.

Two classical tendencies may be considered to solve this problem. The first one consists of a centralised control system, whereas the second option advocates for a distributed network, where piece would take up certain responsibilities in the control of the whole system.

In this paper we analyse some of the latest developments as regards programming distributed networks, focusing on home automation, specially Java and the latest Jini environments have dramatically improved the potential of such kind of distributed systems, thanks to them, it is possible to build a heterogeneous network including devices from different brands which work in collaborative systems, thus avoiding complex configurations.

Introducción

En el momento del diseño de un sistema domótico existen una serie de elementos muy importantes que influirán decisivamente no sólo en la calidad, sino en el coste y en la eficacia futura del sistema que se está diseñando.

Entre estos elementos podemos citar, en primer lugar, al sistema de control, encargado de gestionar el correcto funcionamiento de los diferentes elementos que compondrán el sistema para proporcionar los resultados deseados por los usuarios.

En segundo lugar se encuentra el diseño de un elemento de intercomunicación adecuado entre los distintos módulos del sistema, para que el intercambio de información se realice de forma efectiva y las órdenes que deberán realizar los distintos actuadores del sistema lleguen de forma precisa para ser ejecutadas. También se deberá encargar de que la información de los distintos sensores sea recibida por los correspondientes destinatarios para que puedan tomar las decisiones de actuación correctas.

Podemos citar, en tercer lugar, el diseño de una serie de interfaces que resulten adecuados para la comunicación

Informes de la Construcción, Vol. 50 nº 459, enero/febrero 1999

entre el hombre, el usuario final del sistema y la máquina, que deberá obedecer sus órdenes.

También es importante definir las funciones que habrá de realizar el sistema, pues esto implicará el diseño de una serie de sensores y actuadores que deberán ser controlados por el propio sistema para que puedan ejercer su función dentro de la red domótica. Esto supone un importante esfuerzo de diseño previo, obviamente.

No obstante, en este artículo nos vamos a centrar en la parte que concierne al sistema de control, más concretamente en las arquitecturas entre las que podemos escoger y en particular cuáles son las últimas tecnologías con que contamos para poder abordar el desarrollo de un sistema distribuido. Así pues, supondremos que el problema del diseño de un sistema de comunicación entre los distintos módulos ya está resuelto en esta parte del diseño, así como las otras cuestiones que pudieran afectar a la funcionalidad del sistema.

De cualquier modo, un buen sistema de control creemos que debe estar diseñado de forma tal que permita su ampliación futura de forma fácil, de modo que la definición de las tareas que deba afrontar la red domótica no suponga un obstáculo a su desarrollo y pueda ser ampliada según las necesidades del consumidor o la disponibilidad de la tecnología.

Igualmente, un buen sistema domótico debe permitir que los avances tecnológicos sean fácilmente aplicables en lo que se refiere a la interacción con el usuario a través de interfaces cada vez más sencillos de manejar, ergonómicos e incluso vistosos y agradables. Sobre este asunto que nos ocupa volveremos en la parte dedicada a Java, ya que precisamente una de las claves de su éxito consiste en poseer un interfaz unificado para un gran número de aplicaciones con finalidades totalmente dispares, lo que facilita a los usuarios su interacción con la máquina, aunque no es éste el asunto sobre el que trata el presente escrito.

En cuanto a los objetivos de un sistema domótico, consideramos que es un tema lo suficientemente extenso como para abarcar un artículo por sí solo y, por lo tanto, no profundizaremos al respecto. Es más, queremos insistir en el concepto del diseño modular de los sistemas domóticos. Creemos que el éxito de una red domótica está estrechamente relacionado con la capacidad del sistema de variar su funcionalidad, incrementándola según las necesidades personales y según lo permita la tecnología, por lo que no podemos hablar de una funcionalidad determinada ni de unos objetivos concretos. De lo que sí podemos hablar es de que el sistema habrá de ser lo suficientemente versátil como para poder crecer y afrontar nuevos objetivos, siendo, en nuestra opinión un factor clave a tener en cuenta en el momento del diseño de un buen sistema domótico. Es por ello por lo que opinamos que en un futuro no lejano se impondrán las arquitecturas distribuidas frente a las que proponen sistemas de control centralizado, ya que en las arquitecturas distribuidas, como veremos aquí, es mucho más sencillo añadir nuevas funciones al sistema, variar los interfaces de acceso por parte del usuario final, ya que permiten una mayor compatibilidad entre componentes de distintos fabricantes, lo que a la larga incide en un abaratamiento del producto final y una mayor capacidad de elección por parte de los usuarios.

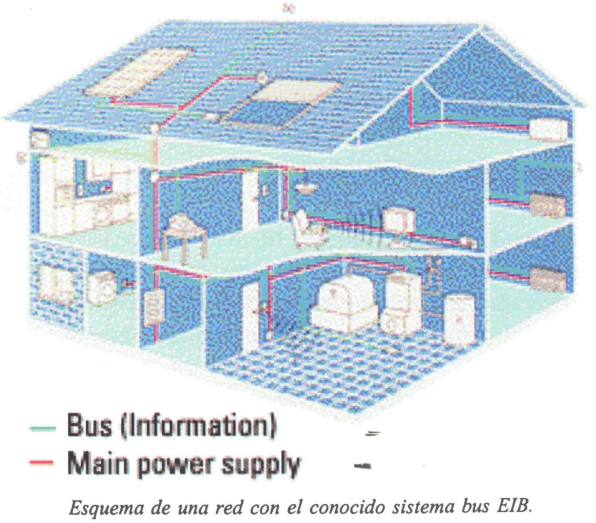
Breve definición del sistema de control

Repasemos, en primer lugar, las características que deben tenerse en cuenta al abordar el diseño de un sistema de control.

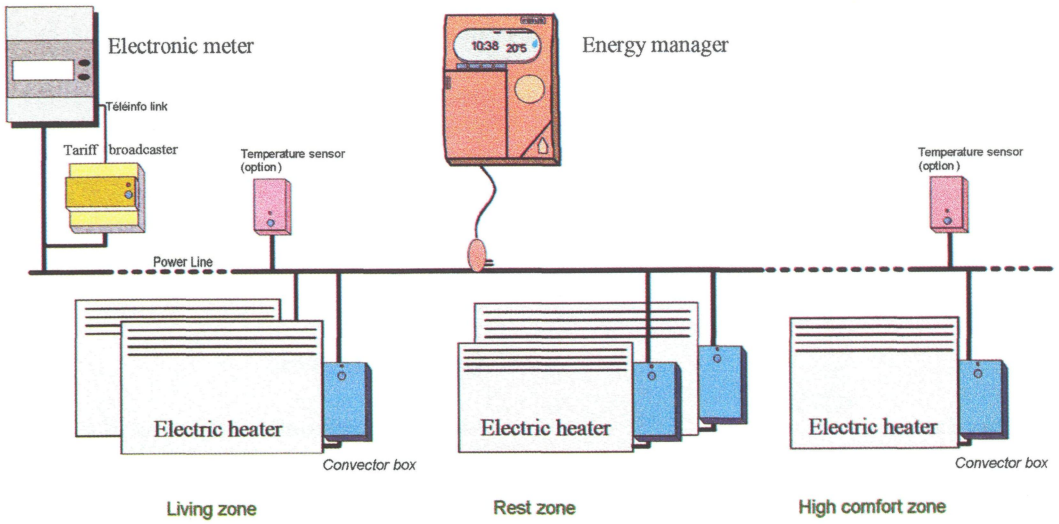
En primer lugar, el sistema de control debe encargarse de recoger la información del entorno mediante los distintos sensores que componen el sistema y, teniendo en cuenta los parámetros con los que haya sido programado, deberá decidir el modo de actuación más adecuado para satisfacer los objetivos o las necesidades de las personas que vivan en el hábitat controlado por el sistema domótico.

Tal y como hemos indicado previamente, en la actualidad existen dos tendencias a la hora de decidir la arquitectura del sistema de control de un sistema domótico, los sistemas distribuidos, en los cuales la inteligencia del sistema se reparte entre los diversos componentes que componen la red, actuando de forma cooperativa para lograr los objetivos del sistema, y los sistemas de control centralizado, donde un único dispositivo inteligente controla las acciones de los demás aparatos involucrados en la red, en base a la información que recibe de los sensores de la misma.

Es evidente que no siempre podemos efectuar una división tan radical en la clasificación de los sistemas domóticos,



Esquema de una red con el conocido sistema bus EIB.



Ejemplo de diseño basado en la especificación EHSA.

ya que en el momento de realizar aplicaciones prácticas se suele tender hacia modelos mixtos en los que el sistema de control centralizado no posee realmente el control absoluto sobre todos los elementos y en los que los sistemas de control distribuido cuentan con un pequeño módulo que centraliza, de algún modo, los parámetros que deberán seguir los componentes del sistema.

Sistema de control centralizado

Podemos considerar a este sistema de control como el más clásico y sencillo de instalar, al menos a priori.

Es el más sencillo de diseñar, ya que únicamente hay que tener en cuenta un módulo dotado de inteligencia en todo el sistema, que será el que controle al resto de los dispositivos instalados. En este caso, está claro que el nodo central será el más caro de todos los dispositivos instalados, por lo que los periféricos y demás dispositivos, sensores, termostatos, detectores de presencia, etc., serán más sencillos de diseñar y, en consecuencia, más baratos de fabricar.

A pesar de estas ventajas, inherentes a los sistemas centralizados, debemos considerar también una serie de desventajas importantes. En primer lugar, el escoger entre un sistema u otro es muy importante para el consumidor, ya que dependiendo del sistema que escoja es bastante probable que no tenga mucha libertad a la hora de escoger el fabricante de los dispositivos que compondrán su sistema.

Para solucionar este inconveniente, algunas asociaciones de fabricantes, como EHSA, o la propia Comisión Europea, han promovido la creación de una serie de estándares, sobre todo en lo que se refiere a la interconexión entre dispositivos y al tipo de bus a utilizar, tendentes a lograr una normalización en la construcción de sistemas domóticos, tratando de acabar con esta seria desventaja.

Otro inconveniente que presentan los sistemas centralizados, es la dependencia de todo el sistema del nodo central. En caso de fallo de este dispositivo, la red entera dejará de funcionar, por lo que el mantenimiento de dicho elemento es crucial para que la red domótica proporcione toda la funcionalidad que se espera de ella.

Sistema de control distribuido

Entre sus ventajas, la más importante es su gran escalabilidad, ya que pueden añadirse con relativa facilidad módulos que implementen funcionalidades que no existían en el momento de diseñarse el sistema o que no eran necesarias en el momento de su instalación. Además, suele ser más fácil construir un sistema basado en componentes de diversos fabricantes, lo cual redunda en una mayor libertad por parte del usuario en la elección de los dispositivos que desea y también incide en un coste final menor, ya que la competencia entre fabricantes es mayor.

Otra de las ventajas que aporta este tipo de arquitectura es que la integridad del conjunto no se ve afectada por la caída de un subsistema, ya que, al menos en teoría, al estar el control del sistema distribuido entre todos sus componentes, si uno de ellos deja de funcionar adecuadamente, los restantes deben ser capaces de cumplir con las tareas que tienen asignadas. Esto no es tan evidente, ya que el diseño de un sistema de estas características es relativamente complejo, sobre todo en el momento en el que se aborda un modelo mixto en el que existe, al menos, un elemento que, si no centraliza el control, sí centraliza, al menos, la información de control para el resto de los dispositivos (temperatura máxima, nivel de intensidad lumínica, etc.). Por otra parte, este concepto haría necesario diseñar un sistema que permitiera distribuir dicha información entre los demás nodos en caso de caída de un subsistema, para lograr que el resto de

Informes de la Construcción, Vol. 50 nº 459, enero/febrero 1999

la red siguiera funcionando normalmente, lo que puede encarecer notablemente el diseño.

Afortunadamente, tal y como veremos en el capítulo dedicado a Jini, actualmente se plantea un sistema muy sencillo y barato que permite superar este obstáculo y lograr un sistema domótico realmente distribuido y muy robusto frente a posibles fallos en subsistemas aislados.

La aproximación Java

A principios de la década de los noventa, Sun Microsystems comenzó a desarrollar un nuevo lenguaje de programación con un objetivo muy ambicioso. Su intención era la de crear un entorno de programación que fuese capaz de ser interpretado por casi cualquier tipo de plataforma y cuya utilidad no se viera restringida al mundo meramente informático de los ordenadores y algunas pequeñas aplicaciones muy especiales.

La intención final de Sun al desarrollar este lenguaje, era la de conseguir un lenguaje casi universal, de forma que pudiera ser utilizado tanto para programar complejas aplicaciones destinadas a funcionar a través de Internet, como para actividades cotidianas, aparentemente tan triviales como controlar la programación automática de la cafetera eléctrica. De hecho, el nombre que finalmente adoptó este lenguaje de programación proviene de uno de los mejores cafés del mundo, al menos desde el punto de vista de los norteamericanos: el café de Java.

Podemos decir que Java fue concebido en 1994, cuando Bill Joy presentó una propuesta en los Laboratorios Sun, en la que se proponían tres conceptos principales que debían ser considerados a la hora de abordar el desarrollo de este nuevo entorno:

Debía ser un lenguaje que pudiera ser ejecutado en cualquier plataforma o procesador.

Se debería construir una máquina virtual capaz de ejecutar dicho lenguaje.

El resultado final sería un sistema en red, capaz de distribuir las máquinas virtuales para trabajar como si se tratase de un sistema compuesto por una sola máquina.

En 1995, el lenguaje y la máquina virtual fueron lanzados al mercado con los nombres definitivos de Java y *máquina virtual Java* respectivamente. No obstante, el concepto final siguió siendo desarrollado por el equipo de Investigación y Desarrollo de los laboratorios Sun, dando lugar a un contexto del que hablaremos un poco más adelante, el Jini.

Sin embargo, al contrario de lo que en un principio vaticinaban en Sun, el verdadero éxito de Java se ha producido gracias a la tremenda expansión que ha sufrido Internet. Posiblemente, sin Internet, Java ya no existiría hoy en día.

Pero veamos con un poco de profundidad cuál es la arquitectura de Java y por qué ha tenido finalmente tanto éxito.

Arquitectura Java

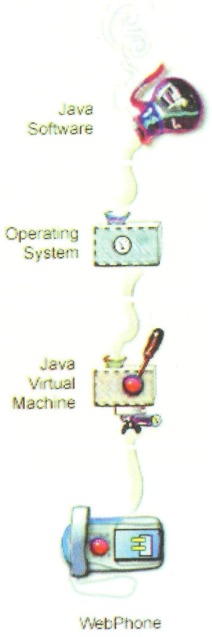
En el momento en que un desarrollador informático decide crear una aplicación, debe pensar detenidamente en qué entorno la va a ejecutar. Es decir, aunque sobre el papel es igual, en el momento del desarrollo no es lo mismo crear un programa para un ordenador personal que para un gran ordenador. El problema estriba en que ordenadores de distintas clases equipan procesadores diferentes y cada procesador posee un lenguaje propio que únicamente es conocido por sus *hermanos*.

Es evidente que esto plantea una problemática a la hora de crear aplicaciones, ya que aparte del coste del desarrollo que supone una aplicación, se debe sumar el coste adicional que supone hacer que dicho *software* funcione en otro ordenador, es lo que, en el argot informático, se llama *portabilidad* del *software*.

En primer lugar, cuando se *porta* una aplicación de una plataforma a otra distinta de aquélla en la que fue desarrollada, como mínimo se debe proceder a su *compilación*, es decir, se debe ejecutar el proceso de traducción que convierte el lenguaje de programación de alto nivel en el que se efectuó el desarrollo original en lenguaje propio de la máquina donde será ejecutado. Esto en el mejor de los casos, ya que en muchas ocasiones es necesario realizar otra serie de modificaciones adicionales que, en principio, no son tan evidentes.

La filosofía de Java rompe con esta norma, al menos en su concepto original. La gran baza de Java es su capacidad para ser integrado en cualquier plataforma sin necesidad de *recompilar* las aplicaciones desarrolladas.

Realmente los programas realizados en Java no se encuentran realmente *compilados*, sino que se trata de un código preprocesado listo para ser enviado a cualquier plataforma, independientemente del lenguaje nativo que ésta utilice para ejecutar el código máquina. Evidentemente existe un pequeño truco, no se trata de que el lenguaje Java sea entendido directamente por todos los procesadores del mundo, sino que se necesita instalar una *pieza* que haga de intérprete, en tiempo de ejecución, para que el procesador de la plataforma donde se ejecuta la aplicación entienda el código preprocesado Java. Es lo que se denomina *máquina virtual Java*. Se podría decir que el *truco*



En este ejemplo vemos la integración de sistema Java en un teléfono, de modo que este aparato puede aumentar considerablemente su funcionalidad.

En este caso, el programa Java puede ser tanto enviado a través de la línea telefónica como formar parte del sistema operativo incluido en el propio teléfono.

El sistema operativo contendrá en su interior una máquina virtual Java preparada para interpretar los programas Java que le llegan a través de la línea o ejecutar los que posea el teléfono previamente.

Finalmente es la máquina virtual Java la que se encarga de ejecutar el código Java que le hace llegar el sistema operativo.

El usuario solamente accede a un terminal gráfico de aspecto muy agradable y fácil de manejar. Entre las aplicaciones que se pueden incluir podemos citar las de compra telefónica, acceso a Internet, calendario-agenda, e incluso se puede utilizar como interfaz de control de un sistema domótico.

Ejemplo de arquitectura Java.

reside en crear un programa precompilado y realizar la compilación efectiva en tiempo de ejecución.

Es evidente que las ventajas de poder instalar una *máquina virtual Java* no se reducen únicamente al ámbito de la computación. En realidad, se puede dotar a cualquier microprocesador de una pequeña máquina virtual que lo capacite para la ejecución de programas realizados en Java. Por ejemplo, tal y como se ha citado anteriormente, se puede incluir un pequeño microchip en la cafetera que nos permita realizar su programación mediante unos sencillos menús, como los que se utilizan en cualquier página Web de las que estamos acostumbrados a ver en Internet.

Evidentemente, la utilización de Java se centra más en la creación de redes siguiendo una filosofía asociada con el Sistema de Control Distribuido, que en las redes con Sistema de Control Centralizado. La razón es obvia. Si implantamos un pequeño microchip en los dispositivos periféricos que compondrán nuestra red, resulta barato dotar a estos dispositivos de una inteligencia adicional que nos permita olvidarnos de una centralización excesiva y de un dispositivo de control centralizado que encarecería demasiado el coste final y que, además, nos obligue a escoger entre un reducido número de fabricantes que construyan dispositivos compatibles con dicho control centralizado.

Como ya hemos citado, finalmente Java no se ha utilizado como se pensó, en un principio, para un control masivo de aparatos domésticos, sino que su desarrollo se ha concretado en el mercado de Internet. Precisamente este auge en Internet es el que ha facilitado la aparición de aplicaciones destinadas al mercado de la domótica y la automatización de hogares.

El entorno Java no sólo nos permite realizar una gestión eficaz de los elementos instalados en nuestra casa, sino que permite que realicemos dicha gestión desde cualquier punto de la casa a través de navegadores e interfaces similares. Por ejemplo, un simple termostato puede ser gobernado a través de una página web, que se puede alojar en un dispositivo de control dedicado, como puede ser un pequeño teclado y una pantalla de cristal líquido o, incluso, un ordenador personal conectado a la red domótica de la casa para centralizar su control.

Aún más, con la corriente arrolladora de Internet se están poniendo de moda los llamados *web-tv*, o televisores con capacidad de navegar por Internet. Así, se podría llegar a gobernar la temperatura de la calefacción o el nivel de iluminación accediendo a un simple menú, con un único mando a distancia, el de la televisión, y sin dejar de ver nuestros programas favoritos.

Se ha dicho aquí que también es posible la gestión de este tipo de dispositivos desde un ordenador personal. Pero se puede ir mucho más allá. Puesto que estamos hablando de un entorno perfectamente integrado en las nuevas autopistas de la información, sobre todo lo que se refiere a Internet, nos encontramos con la posibilidad de poder gestionar un sistema domótico, no solamente desde una hipotética consola o centro de control, sino desde cualquier ordenador conectado a la red.

Son relativamente populares los dispositivos de control, sobre todo en el campo de las calefacciones en casas de campo, que pueden ser activados a través del teléfono. Suelen ser aparatos muy sencillos que únicamente reciben unas pocas órdenes, habitualmente el encendido o el apagado y, como mucho, la temperatura a la que debe llegar la calefacción o el aire acondicionado.

Gracias a Java, se pueden llevar a cabo estas tareas, tanto por teléfono como a través de Internet, pero ofreciendo una mayor gama de posibilidades ya que el interfaz que ofrece es muchísimo más rico en posibilidades. Y, sobre todo, que no existe limitación en cuanto al acceso geográfico al sistema.

Jini: la evolución de Java

De nuevo Sun Microsystems es la que ha lanzado un diseño revolucionario al mundo de la interconexión de aparatos heterogéneos a través de la red, con el lanzamiento de un nuevo entorno de trabajo: el Jini.

Jini es un proyecto de I+D inspirado por Bill Joy que expande considerablemente el poder de la tecnología Java. Gracias a Jini, una gran variedad de elementos tanto hardware como software son capaces de participar espontáneamente en un entorno de red.

El objetivo final de Jini es permitir que las personas sean capaces de utilizar los elementos conectados a la red de una forma tan sencilla como se utiliza un teléfono, que únicamente necesita ser enchufado a la red y funciona en el momento en el que comienza a enviar o recibir los tonos de marcado.

Arquitectura Jini

Veamos cuál es la arquitectura de alto nivel que nos presenta Jini y cuál es su filosofía de trabajo, destacando las nuevas características que lo definen frente al tradicional Java. Aunque en principio puede parecer una descripción puramente informática dirigida a ingenieros del *software*, enseguida veremos las aplicaciones prácticas que nos ofrece esta nueva herramienta en el campo concreto de la domótica.

Un sistema Jini es un sistema distribuido basado en la idea de *federaciones* de usuarios y de los recursos que necesitarán dichos usuarios. El objetivo principal es conseguir que la red se convierta en una herramienta flexible y fácil de administrar, en la cual los recursos ofrecidos sean fácilmente localizables, tanto por los usuarios humanos como por los posibles dispositivos que soliciten recursos a la red.

En este entorno consideraremos *recurso* tanto a los programas como a los dispositivos físicos que lo componen. Precisamente el enfoque que aborda esta arquitectura es el de hacer que la red refleje el enfoque dinámico de los entornos de trabajo, en los cuales es frecuente añadir o eliminar recursos con relativa asiduidad.

En el mundo de la domótica esto se traduce en una mayor flexibilidad a la hora de introducir nuevos dispositivos o de sustituir -e incluso eliminar- algunos de los existentes, sin que por ello la red domótica instalada deje de funcionar o pierda parte de su funcionalidad. Este aspecto es clave, ya que nos va a permitir una configuración totalmente a la medida del sistema domótico que queramos instalar, atendiendo a criterios de coste o funcionalidad, sin tener que preocuparnos de las posibles ampliaciones futuras, ya que éstas se podrán realizar sin ningún tipo de problema de compatibilidad.

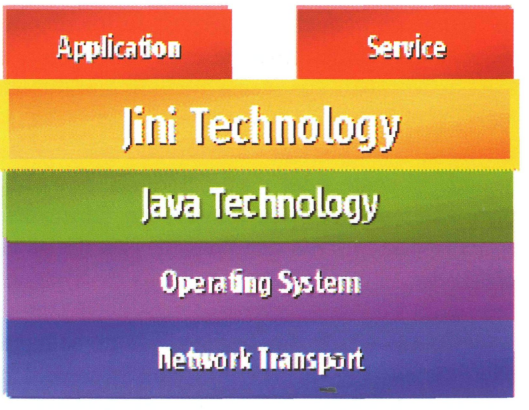
Un sistema Jini consta, principalmente, de tres partes:

. Un conjunto de componentes que proporcionan una infraestructura para poder establecer una *federación* de servicios en un entorno distribuido.

. Un modelo de programación que proporciona la posibilidad de desarrollar servicios distribuidos fiables.

. Una serie de servicios que pueden llegar a formar parte de dicha estructura y que ofrecerán servicios a los demás componentes del sistema (entendiendo que el sistema lo componen las máquinas y las personas).

El sistema Jini proporciona una extensión del ambiente Java desde una sola máquina virtual a una red de máquinas. El ambiente Java proporciona una plataforma adecuada para la computación distribuida gracias a su facilidad intrínseca para portar códigos y datos y permitir su ejecución en otros sistemas.



Descripción de la pila Jini.

La arquitectura Jini aprovecha al máximo las posibilidades ofrecidas por Java para simplificar notablemente la construcción de sistemas distribuidos, añadiendo mecanismos que permiten una fluidez hasta ahora impensable en la comunicación entre dispositivos, permitiendo que los objetos del sistema se puedan desplazar por toda la red.

La infraestructura que proporciona Jini nos da mecanismos que permiten que los diferentes servicios y mecanismos se puedan añadir o eliminar de la red de una manera muy sencilla y natural, normalmente de manera incluso automática. Se consigue un comportamiento mucho más dinámico que aquellos tipos de redes en los cuales toda la funcionalidad se encuentra centralizada y, por lo tanto, la inclusión o eliminación de dispositivos se debe realizar prácticamente a mano.

Como podemos apreciar en el esquema adjunto, frente a una arquitectura más o menos tradicional de acceso a la red, aparecen dos nueva capas, denominadas en la terminología de Sun como *Lookup* y *Discovery/Join*. Precisamente son estas nuevas capas las que van a facilitar una auténtica revolución en la forma de interconectar los dispositivos que componen una red, ya que se encargarán de hacer esto de manera totalmente automática, haciendo que, por primera, vez el concepto de *plug-and-play*, es decir, enchufar el dispositivo y comenzar a utilizarlo, se convierta en realidad en cualquier entorno en el que haya una red implicada.

Discovery and Join

La infraestructura Jini proporciona este primer concepto para resolver el difícil problema de hacer que un nuevo elemento se identifique en la red la primera vez que se conecta a ésta, sin que dicha red tenga conocimiento previo del elemento que se está añadiendo. Hay que destacar que cuando decimos elemento nos estamos refiriendo a cualquier dispositivo físico que conectemos a la red y a cualquier servicio ofrecido a través de software y que sea accesible por los distintos usuarios de la red.

Así pues, lo primero que hará un elemento dotado de la tecnología Jini al ser enchufado a una red será descubrir la red y notificar a dicha red que un nuevo elemento pasa a formar parte de la misma.

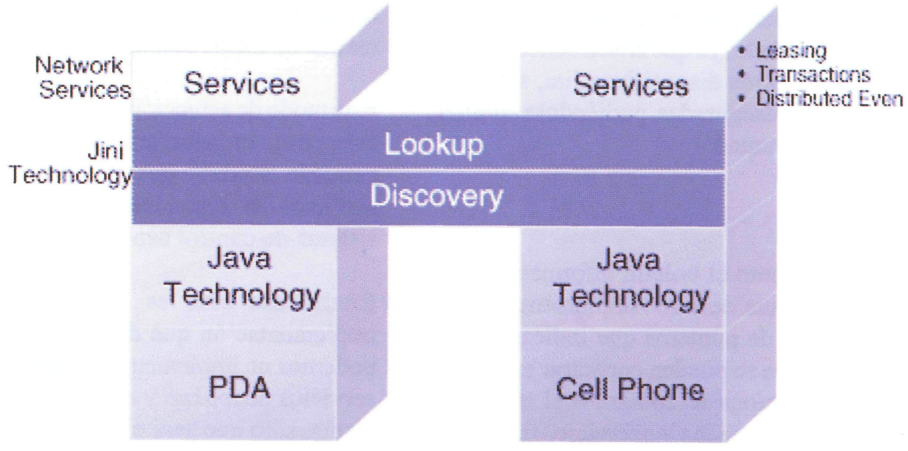
Esto se realiza, básicamente, en dos pasos:

1. El dispositivo Jini conectado a la red envía un paquete a la red a un puerto predeterminado por la arquitectura. Entre otra información, en dicho paquete se incluye una referencia para que la red sepa cómo acceder al nuevo dispositivo instalado.

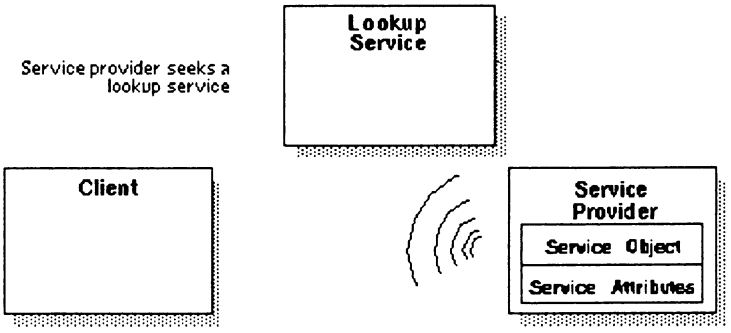
2. El sistema *Lookup* permanece a la escucha, en la dirección de puerto predeterminada por la arquitectura, a la espera de recibir el paquete enviado por el nuevo dispositivo. Cuando éste llega, *Lookup* utiliza el propio interface del dispositivo para completar la fase de reconocimiento inicial y admitir al nuevo dispositivo en la red.

En este momento se finaliza la fase de reconocimiento mutuo entre la red y el nuevo dispositivo. A continuación se procede a una segunda fase, en la cual, dicho dispositivo volverá a establecer una comunicación con el *Lookup* de Jini para añadir al sistema las nuevas funcionalidades proporcionadas por este dispositivo y que, a partir de ese momento, estarán disponibles y accesibles para todos los componentes de la red que las necesiten.

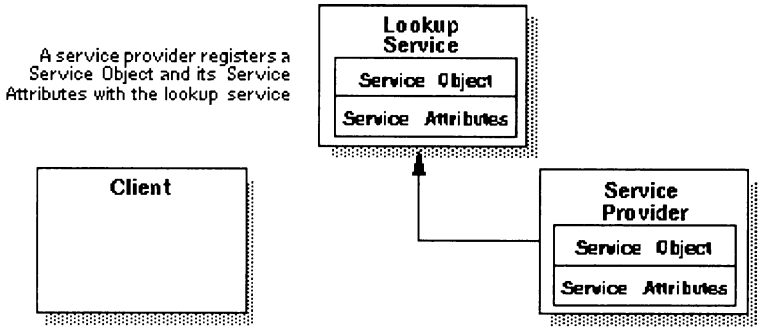
Básicamente, cuando un nuevo dispositivo se conecta a la red, se establece una fase de reconocimiento mutuo entre la red y el nuevo elemento físico añadido, de forma que el



Módulos de la arquitectura Jini.



Esquema de actuación del módulo Join.



Funcionamiento del reconocimiento inicial.

dispositivo indicará al controlador de la red cuáles son las nuevas características que aporta con su inclusión y de qué forma son accesibles. Es como si hubiera un diálogo en el que ambas partes se pusieran de acuerdo para trabajar de forma conjunta, sin necesidad de establecer ningún tipo de configuración adicional por parte del usuario.

Es evidente que las ventajas para el usuario son inmensas, ya que en el momento de instalar una red domótica en el hogar únicamente deberá pensar en el coste y en la funcionalidad que desea añadir, ya que la interconectividad y la compatibilidad con otros dispositivos Jini, sean del fabricante que sean, están aseguradas, pudiéndose ampliar la red prácticamente a voluntad.

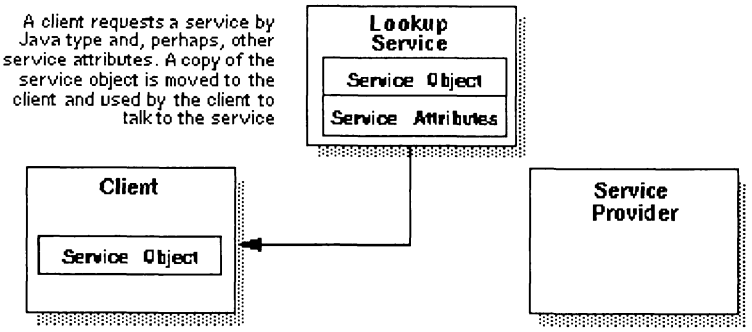
Lookup

Se puede decir que es como el boletín informativo que mantiene al día la estructura de la red implementada. No sólo almacena una serie de punteros que indican cómo acceder a los servicios que se pueden encontrar en la red, sino que además proporciona el código, o el modo de acceder al código que proporciona los servicios finales de los que es capaz la red.

Por poner un ejemplo, si se añade un dispositivo que regule la iluminación de una estancia y el usuario decide variar la intensidad de la luz desde cualquier interface, por ejemplo el teléfono-web empleado en el ejemplo de Java, el servicio *lookup* se encarga de averiguar donde se encuentran almacenados los manejadores que permitirán al teléfono variar los parámetros que puedan ser controlados y trasladarlos hasta el mismo. De esta forma, no es necesario que los manejadores, o *drivers*, sean cargados previamente a la utilización del periférico en cuestión, sino que se realiza de forma automática por el sistema.

Este modo de actuación nos puede llevar a pensar a que, en este caso, es necesaria la existencia de al menos un pequeño módulo que se encargue de mantener estos servicios de reconocimiento y que implica un cierto sistema de control centralizado.

Realmente esto es así, pero dependiendo de la implementación que deseemos llevar a cabo. Es decir, podemos no implementar un módulo centralizador de los servicios de *lookup* y el sistema funcionará de la misma forma, sólo que será el dispositivo de acceso al sistema por parte del usuario, el interface mencionado, el que



Funcionamiento del Lookup.

solicite la identificación de los demás dispositivos de la red cuando desee acceder a un determinado servicio, efectuando él mismo el servicio de *Lookup*.

HAVi y Jini

Qué es HAVi

En 1998 ocho de las más importantes compañías mundiales en el campo de los electrodomésticos de consumo, Grundig, Hitachi, Matsushita Electric, Philips, Sharp, Sony, Thomson y Toshiba, lanzaron una nueva especificación encaminada a definir un nuevo estándar en el campo de las aplicaciones audiovisuales y su interoperación, un protocolo que ha visto la luz como HAVi, de sus siglas en inglés: *Home Audio/Video Interoperability*.

El objetivo final de estos ocho grandes de la industria es convertir a HAVi en un estándar para el desarrollo de los próximos aparatos de consumo del mercado audiovisual.

El motivo que ha llevado a estas compañías a diseñar un protocolo de estas características se funda en el importante crecimiento que el mercado audiovisual ha sufrido en los últimos años, de forma que los usuarios finales pueden encontrar muy atractiva la posibilidad de interconectar los diferentes aparatos audiovisuales con que pueden contar en la actualidad, para formar redes de difusión de estos medios audiovisuales en el propio hogar.

Es evidente que si se desea que aparatos fabricados por distintas marcas sean capaces de actuar en un modo cooperativo entre sí, debe existir un elemento común, un software específico que aísle las diferentes arquitecturas internas de los distintos aparatos y que, siguiendo unas normas comunes, facilite una serie de servicios que, entre otras cosas, permita la interconexión entre los distintos aparatos implicados.

Objetivos de HAVi

Existen una serie de requisitos fundamentales que se han propuesto los creadores del HAVi.

Pasamos a enumerarlos a continuación:

. Conectividad *Plug and Play*. Debe ser posible para el usuario añadir y eliminar elementos de la red audiovisual sin necesidad de realizar complicadas configuraciones, sin más que enchufar o desenchufar los distintos aparatos.

. Interoperatividad de aplicaciones. Los distintos aparatos conectados a la red deben ser capaces de reconocer al resto de los elementos conectados a dicha red y compartir las funcionalidades que otros aparatos ofrecen.

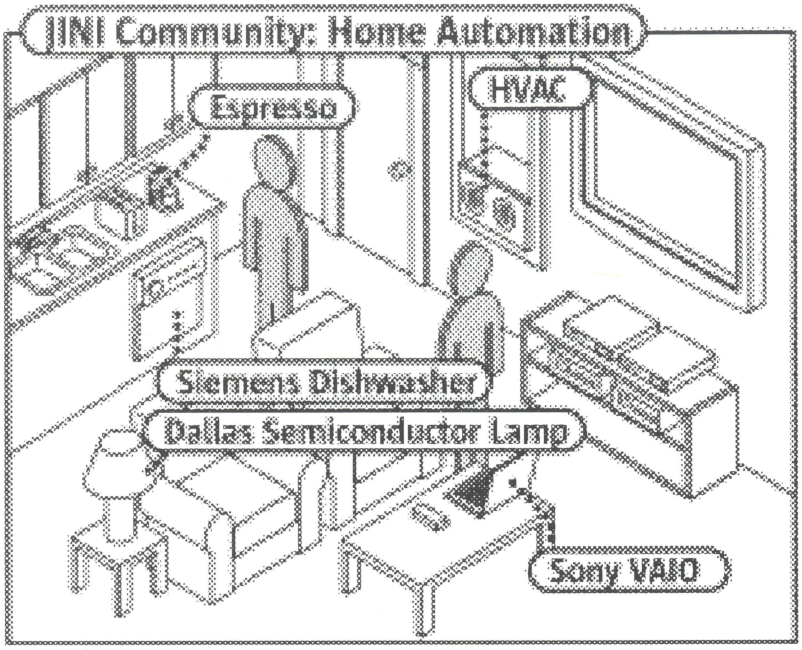
. Aparatos capaces de crecer en el futuro. Es decir, los dispositivos diseñados bajo esta filosofía deberán ser capaces de ofrecer nuevas funcionalidades mediante sencillas actualizaciones software proporcionadas a través de la red y asegurarán la compatibilidad total con otros aparatos que puedan ser diseñados en el futuro y que aporten nuevas funcionalidades.

HAVi y Jini unen sus fuerzas

Estudiando detalladamente las especificaciones del sistema HAVi podemos encontrar un gran número de similitudes con las especificaciones indicadas por la arquitectura Jini, lo que nos lleva a pensar que sería muy interesante elaborar una suerte de unión entre ambas filosofías de trabajo, ya que aparentemente persiguen un objetivo común.

Recientemente las compañías Philips, Sony y Sun han llegado a una serie de acuerdos encaminados al respecto. Dichas empresas consideran ambos entornos como complementarios y están trabajando en la consecución de un puente entre ambas tecnologías con el objetivo final de lograr una serie de electrodomésticos, principalmente encaminados al campo audiovisual, que cumplan con las especificaciones HAVi y que se interconecten a una red doméstica a través de los servicios proporcionados por la arquitectura Jini.

Según los responsables del proyecto, HAVi se centra en crear redes de funcionamiento para sistemas domésticos



Posible escenario Jini/HAVi.

de audio y vídeo, por lo que el puente entre esta tecnología y Jini supone una ampliación de este concepto, muy interesante, que permitiría acceder a la red doméstica desde cualquier lugar.

Conclusión

Indudablemente la tecnología Java/Jini cuenta con la suficiente madurez como para ser considerada un entorno de desarrollo de sistemas domóticos a tener muy en cuenta en el futuro. Esto, sumado al espaldarazo que supone el que ocho fabricantes de la talla de los involucrados en el sistema HAVi estén interesados en incorporar dicha tecnología, hace presagiar que en el futuro la práctica totalidad de los sistemas domóticos de nueva creación se desarrollarán en dicho entorno.

Esta aparente supremacía del sistema Jini no debe ser considerada como un intento monopolista de Sun, ya que no parece que sus intenciones se centren en acaparar el mercado. Precisamente una de las principales virtudes de este sistema es que se trata de un estándar abierto, en el que podrán participar todos aquellos fabricantes que lo deseen, por lo que el gran beneficiado será el consumidor final.

En un primer momento se puede pensar que fabricar elementos compatibles con redes de estas características encarecerá el coste final de dichos elementos, ya que deben añadir una cierta complejidad a su diseño para poder operar de forma cooperativa en los sistemas a los que se añadan. Sin embargo, la posibilidad de elección que tendrá el consumidor redundará finalmente en unos precios muchísimo más asequibles, gracias a la competencia entre fabricantes y, sobre todo, permitirá la construcción de redes totalmente a la medida, tanto en coste como en funcionalidad.

De cualquier modo, se ha demostrado que hoy en día es perfectamente factible construir casi cualquier tipo de dispositivo dotado de tecnología Java/Jini a un coste perfectamente asequible, ya que esta tecnología se está aplicando en los más diversos campos, puesto que ha demostrado una gran fiabilidad y una versatilidad muy grande.

En cuanto a la robustez de los sistemas diseñados frente a posibles fallos de algunos de sus subsistemas, se ha visto que la arquitectura Java/Jini consigue el objetivo de hacer que los distintos elementos de la red trabajen de forma realmente independiente, aunque en colaboración unos con otros. En caso de caída de alguno de los módulos de un sistema domótico construido con Jini, los propios componentes del sistema son capaces de comunicarse entre sí para averiguar cuál es la funcionalidad que puede seguir siendo utilizada por el sistema, y de qué modo se puede acceder a ella para seguir proporcionando servicio al usuario.

A todo esto hay que añadir la facilidad de manejo que este nuevo entorno facilita, ya que basta con emplear un simple navegador, como los que se utilizan habitualmente para acceder a Internet, para controlar todas las funciones que se deseen de nuestro entorno domótico. Además, esta unificación en cuanto al interfaz, facilitará una mayor disponibilidad respecto al número de aplicaciones disponibles, y hará más fácil la actualización de cualquier sistema con nuevas funciones.

A la hora de ampliar un sistema domótico en el futuro, no pensaremos únicamente en qué dispositivos queremos añadir, sino en qué funciones, qué programas nuevos, desearemos introducir para conseguir un control mucho más efectivo.

Otra cuestión que cobrará una gran importancia en el futuro, es el considerar el acceso a la red domótica desde cualquier parte del mundo. Java es un entorno que se ha desarrollado en Internet y para Internet y, por lo tanto, al construir un sistema basado en esta tecnología, resulta totalmente factible proporcionar un acceso al sistema domótico desde cualquier parte siempre que dispongamos de un acceso adecuado a la red. Ya no se trata de ejercer un control total de nuestro sistema, sino que se puede realizar desde cualquier parte, aunque no estemos físicamente presentes.

Esta extensión en las capacidades de la red domótica, puede parecerles ciencia ficción a los usuarios, que únicamente desean una serie de funciones relativamente limitadas y, ciertamente, pueden encontrar unos costes que superen ampliamente sus expectativas.

Es precisamente en este campo donde opinamos que seguirán instalándose sistemas de corte más clásico que lo que aquí se ha expuesto, aunque a la larga se nos antoja que la convergencia entre ambas opciones será inevitable.

BIBLIOGRAFÍA

(1) VAUGHN BRADSHAW y KENNETH E. MILLER: Building control systems. John Wiley & Sons, 1993.

(2) DAVID FLANNAGAN: Java in a Nutshell. A Desktop Quick Reference. O'Reilly & Associates, 1997.

(3) WASFI YOUSSEF: Building your own home: A step by step guide.

(4) PATRICIA MARTA ÁNGELy LILIANA BEATRIZ FRAIGI: Introd. a la Domótica. Escuela Brasileño-Argentina de Informática, 1993.

(5) Philips en la carrera hacia las redes domésticas. PC World España, Nº 152, 1999.

(6) Jini White Papers. Sun Microsystems Inc, 1998.

(7) JIM WALDO: Jini Arquitecture Overview. Sun Microsystems Inc., 1998.

(8) HAVi press release. Philips Electronics N.V., 1998.

(9) HAVi-Jini press release. Philips Electronics N.V., 1998.

* * *