Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de Sistemas de Información

Profesores Guías:
Sr. Patricio Inostroza (U. de Chile)
Sr. Pedro Rodríguez (U. del Bío-Bío)

Descubrimiento de Servicios Computacionales en un Ambiente Distribuido

Trabajo de Titulación presentado en conformidad a los requisitos para obtener el título de Ingeniero Civil en Informática

1 de Marzo de 2004

Sr. Jorge Andrés García Ruiz

DEDICATORIA

A Jorge, Liliana, Feliciano y Gerardo,
cuyas almas forjaron, forjan y seguirán forjando la mía.

Especiales gracias a:

DIOS.
Toda mi gran y única familia llena de maravillosas personas.
Juan Ruiz A. por su ejemplo de vida.
Osvaldo, Patricia y Claudia por su amistad, compañerismo y
apoyo durante mis años en la Universidad.
A mis profesores guías por su confianza y ayuda en este proyecto.

Resumen

PROBLEMÁTICA ABORDADA

Implementar periféricos, dispositivos o aplicaciones como un Servicio Distribuido requiere una comprensión acabada del tema. Así entender ¿Que es? ¿Cómo funciona? Y ¿Cuáles son sus ventajas?, es necesario.

Múltiples áreas tecnológicas, de hardware y software, están involucradas en el desarrollo de un producto de este tipo. Varias arquitecturas de software se muestran como alternativa de desarrollo. Conocer y evaluar sus características para realizar una comparación objetiva es útil.

Finalmente, diseñar y construir un prototipo de algún servicio es el mayor desafío para evaluar la correcta comprensión de la temática en estudio.

OBJETIVOS

Objetivos Generales:

- Diseñar y construir un prototipo que permita publicar un "Servicio Distribuido" particular.
- Diseñar y construir un cliente que permita acceder al anterior "Servicio Distribuido".

Objetivos Específicos:

1. Definición clara de descubrimiento de Servicios Distribuidos y sus implicancias.
2. Conocer el estado actual de las distintas tecnologías participes del tema.
3. Evaluar las alternativas de arquitecturas y/o propuestas software para desarrollar servicios de este tipo (UPnP, Jini, etc.)
4. Plantear un listado de productos posibles a desarrollar y seleccionar el más adecuado.
5. Realizar un diseño viable para el servidor y cliente propuesto.
6. Utilizar la arquitectura y/o propuestas software elegida para construir el prototipo y evaluar los resultados.

APORTE

Sin duda los Sistemas Distribuidos se masificarán en el corto plazo y sus aplicaciones como los Servicios Distribuidos serán algo cotidiano en nuestra relación con el entorno. Por ser tecnología poco difundida en nuestro país, es obligatorio introducir conceptos.

A su vez, conocer el estado actual del desarrollo en esta área entrega un marco de trabajo más objetivo, a la hora de diseñar proyectos de este tipo.

Así también el prototipo de Servicio Distribuido genera una base de conocimiento y experiencia que es un primer paso en la construcción de un dispositivo (hardware y software) capaz de satisfacer los anhelos de "conectar y usar" que simplifican al usuario común muchas labores de configuración previa.

LÍMITES

Entendiendo que es un área aún en pleno desarrollo las definiciones conceptuales podrían estar sesgadas por el avance que hasta el momento se tiene del tema, en buenas cuentas, se realiza un esfuerzo por entregar definiciones que clarifiquen la intención antes que lo logrado.

En cuanto al estudio de las distintas tecnologías del tema. Se realiza hasta el punto en que las preguntas ¿Qué es?, ¿Cuáles son sus componentes principales?, ¿Para qué sirve?, ¿Cuáles son sus ventajas y desventajas?, ¿Cuáles son sus requerimientos? son contestadas.

La evaluación de las arquitecturas y/o propuestas software está bajo el marco de trabajo pensado para las opciones de prototipo.

El listado tentativo de productos a desarrollar está limitado a recursos disponibles en nuestra facultad, a una problemática práctica y acotado al tiempo determinado para la Habilitación Profesional, es decir un semestre.

El prototipo no estará exento de mejoras. Es un primer paso al diseño y utilización de esta tecnología, así futuros proyectos podrán utilizarlo como base.

METODOLOGÍA

Esencialmente el estudio se realiza a través de investigación en distintas fuentes (Internet, libros, revistas especializadas, documentos técnicos, etc.) para luego redactar en forma ordenada y clara lo aprendido. Para el estudio de los protocolos de descubrimiento de servicios se tradujeron documentos técnicos oficiales y rescataron las ideas esenciales. Una vez terminados los capítulos 1 al 3 se tuvo una visión más realista y clara de las posibilidades de esta tecnología. Así la evaluación, en el capítulo 4, en conjunto con los profesores guías, de una "lluvia de ideas" de prototipos, pudo ser hecha de mejor manera. El prototipo fue construido con una metodología incremental e iterativa, ya que la programación fue aumentando su complejidad a medida que el proceso de aprendizaje, de la tecnología elegida, entregó nuevas herramientas.

ESTRUCTURA DEL INFORME

Los capítulos han sido organizados para lograr una comprensión progresiva del tema. Así en el primero se entregan los fundamentos para entender ¿Que es? y ¿Dónde se sitúa? el Descubrimiento de Servicios Computacionales. El segundo entrega una visón general de las industrias y tecnologías involucradas en el tema. El tercero describe específicamente los protocolos de descubrimiento de servicio más conocidos y su funcionamiento. En el cuarto, se describen y evalúan las propuestas de prototipo, seleccionando una. Finalmente, la experiencia del prototipo es documentada en el quinto capítulo, haciendo un esfuerzo por plasmar cada una de las tareas realizadas, sean aciertos o errores, sólo así se garantiza un traspaso realista de conocimiento a futuros desarrolladores.

CONCLUSIONES MÁS RELEVANTES

- Los Servicios Distribuidos y su forma de descubrimiento, son el siguiente paso lógico de tecnologías actuales, tales como procesamiento distribuido y servicios Web.

- Actualmente existe la capacidad y tecnología para desarrollar completamente un dispositivo que satisfaga los requerimientos ideales de un Servicio Computacional Distribuido.

- Las propuestas Software para el anuncio, descubrimiento y uso de los servicio tienen muchas más coincidencias que diferencias. La elección respecto a cual se masificará, probablemente, será una decisión estratégica y comercial.

- La dificultad de programación dependerá de la arquitectura elegida, sin embargo todas ellas se basan en similares conceptos básicos, lo que garantiza un apropiado aprendizaje.

Índice

Capítulo 1. ¿Qué es el Descubrimiento de Servicios Computacionales Distribuidos?

Introducción

Gerardo va conduciendo por la nueva autopista que une el centro de la ciudad con la periferia, debe elegir alguna de las entradas para llegar al centro, pero su computador personal a recopilado información comunicándose con los demás automóviles que vienen en sentido contrario. Así es capaz de discriminar el acceso más expedito a la calle destino tomando como criterio el promedio de desplazamiento de los autos que ya la han recorrido. Todo, por supuesto, sucede sin que él siquiera se de cuenta, limitándose a seguir la recomendación de su computador portátil.

Después de un largo viaje, el descanso es necesario así que Gerardo mira el listado de hoteles cercanos que su PDA (Asistente Digital Personal) le provee. Luego de meditarlo, selecciona aquel que tiene la mayor cantidad de huéspedes extranjeros, ya que a su parecer será el mejor. Justo antes de dormir una siesta recuerda que debe imprimir un informe para su jefe zonal. Sin mayor preocupación da el mandato de impresión desde su computador portátil. Instantes después el auxiliar del hotel trae a su habitación el documento impreso en la recepción.

El relato anterior es un borrador de algunas iniciativas tecnológicas que se pretenden masificar. La tecnología inalámbrica debería llegar a un punto en que su uso sea común, la conectividad entre un computador y su entorno se convertirá en un estándar, pudiendo comunicarse con: autos, electrodomésticos y muchos otros servicios.

Sin embargo el relato no tendría mucho de especial, si no fuese por la despreocupación del usuario en la localización, configuración y petición de los servicios (análisis de tráfico, listado de hoteles, reserva, impresión).

Podemos entender un servicio como facilidades que están disponibles para un computador. La “transparencia” en el proceso de comunicación y auto configuración de estos, son un factor crítico de éxito.

La masificación de redes computacionales y el exponencial crecimiento de Internet [Sitio Web Crecimiento de Internet Hosts a nivel mundial] a cambiado la forma en que los

usuarios comparten, distribuyen y analizan información. Las ventajas de estar "en línea, en cualquier lugar y momento" a llevado a desarrollar tecnologías para conectar a redes no sólo computadores, que es nuestra concepción general, sino múltiples dispositivos (periféricos, electrodomésticos, etc.) sin la intervención de un computador personal.

Esta innovación rompe con el paradigma de considerar las redes como meros canales de comunicación que unen a múltiples tipos de computadores, realizando labores similares. Por el contrario visualiza una "red" como coordinadora de servicios que entran y salen brindando a usuarios una absoluta libertad respecto al uso de ellos.

Por ejemplo, podemos imaginar una impresora conectada directamente a una red, sin un computador de por medio, enviando un aviso ("lista para imprimir") a una comunidad predeterminada de usuarios. Además sin mayor esfuerzo comunicar a la comunidad el hecho de su desconexión. Este concepto de enchufar y utilizar, antes propuesto a nivel de computadores personales como Plug and Play [Sitio Web Microsoft's Universal Plug and Play], está siendo mejorado y ampliado para su utilización en conceptos como "Computación Ubicua" [Sitio Web Curso Computación ubicua y Ambiente inteligente], "Conectividad Total", "Entorno Inteligente" y muchas otras líneas de investigación y desarrollo. La última propuesta de protocolo para Internet IPv6, justifica su capacidad de entregar una cantidad prácticamente infinita de direcciones IP a la futura necesidad de poner "en línea" muchos más aparatos de los que hoy podemos imaginar.

La masificación de Servicios Distribuidos esta ligado al desarrollo del Hardware y Software apropiado.

Los dispositivos deben ser capaces de conectarse a alguna red sin necesitar la intervención de un computador personal y cumplir con los requisitos mínimos de software apropiado. Los avances en redes inalámbricas [Sitio Web Intel] deben llegar a convertirse en un estándar.

Por su parte la industria del Software entrega propuestas (plataformas, arquitecturas, lenguajes de programación y protocolos) tendientes a satisfacer los requerimientos que los Servicios Distribuidos necesitan.

En la búsqueda de una definición clara de "Descubrimiento de Servicios Computacionales Distribuidos" es necesario entender algunos conceptos fundamentales para luego explicar como se integran los "Servicios" en este escenario. A continuación se describen con este fin, las materias que permitirán dar una mejor comprensión del objeto de este proyecto.

1.1 ¿Qué es un Sistema Computacional Distribuido?

Los servicios computacionales y su descubrimiento se basan esencialmente en el concepto de Sistema Distribuido. En los objetivos perseguidos y hasta la tecnología utilizada su influencia a resultado fundamental para plantear, siquiera, la posibilidad de una comunidad de servicios computacionales con los protocolos de descubrimiento asociados. Así al entender la esencia y funcionamiento general de los Sistemas distribuidos se estará también clarificando la definición de Descubrimiento de Servicio Computacional.

1.1.1 Necesidad

El vertiginoso avance de las tecnologías computacionales siempre ha superado las expectativas de los tecnólogos que se arriesgan a entregar una visión futura del mercado, esto principalmente porque la rapidez de los cambios y la invención de nuevos conceptos muchas veces entrega un enfoque totalmente distinto al pensado sin ellas. Así, por ejemplo, muy pocos fueron capaces de prever el potencial de la computación personal y el profundo cambio cultural que significa hoy Internet.

Sin lugar a dudas, uno de los avances que contribuyó enormemente a la masificación de los computadores fueron las redes computacionales. Estas, a través de líneas de comunicación y equipamiento, permiten que dos o más computadores intercambien datos. La más utilizada aplicación de este concepto es conocida como Internet, donde millones de computadores, líneas de comunicación, equipos de ruteo y muchos otros componentes forman una "telaraña" de interconexión computacional internacional. La utilidad que da a los millones de usuario es

primordialmente el intercambio de mensajes (e-mail) y publicación o búsqueda de información (World Wide Web) [Sitio Web List-NEWS.com].

Sin embargo y a pesar de este escenario, existe un potencial desperdiciado y que cada vez atrae más la atención. Si nos detenemos a observar la configuración actual de las redes computacionales veremos que cada computador conectado es una "isla" que tiene sus propias tareas que realizar, así el intercambio de mensajes con otras "islas" es sólo en pos del objetivo particular, en ningún momento han llegado a un consenso para perseguir o realizar una labor en común. Aquí es cuando la idea de visualizar a una red computacional bajo el prisma de una comunidad donde existen entidades que interactúan para realizar objetivos comunes y colectivos al mismo tiempo, toma un mayor sentido.

¿Por que utilizar la Computación Distribuida? Existen esencialmente tres líneas de justificación (Rodríguez, 2003):

- **Compartir recursos**, de una manera más natural, estos pueden ser de variados tipos: capacidad de procesamiento, periféricos, información, accesibilidad, etc.

- **Distribución de carga**, los procesos pueden ser delegados según criterios (posición geográfica, capacidad de procesamiento, seguridad, etc.) para optimizar su ejecución y distribución de resultados.

- **Ejecución de Aplicaciones en ambientes más adecuados**

Algunas definiciones más formales del concepto, contribuirán a entenderlo de una mejor manera:

> "Un sistema distribuido es aquél al que sus usuarios ven como un ordinario sistema operativo centralizado; sin embargo, se ejecuta en diferentes e independientes CPUs. El concepto clave aquí es la transparencia; en otras palabras, el uso de diversos procesadores deberá ser invisible (transparente) al usuario. Otra forma de expresar esta

misma idea es diciendo que el usuario verá al sistema como un uniprocesador virtual y no como una colección de máquinas diferentes" (Tanembaum *et al*, 1985)

"Sistema en el cual componentes de hardware y software, localizados en computadores en red, se comunican y coordinan sus acciones sólo por paso de mensajes" (Colouris, 2001)

"Conjunto de computadores independientes que se muestran al usuario como un sistema único y coherente" (Van Steen, 2002)

1.1.2 Elementos básicos de un Sistema Distribuido

La mayoría de los autores de sistemas distribuidos coinciden en una característica principal: transparencia. Transparencia no sólo aplicada al usuario final, sino a cada persona que se involucre, desde el desarrollo hasta la aplicación final de un sistema de este tipo.

Según el Manual de Referencia de la ANSA (1987) existen ocho áreas de desarrollo en las que se debe aplicar la transparencia en un sistema distribuido:

- Transparencia en el acceso: Se refiere a la capacidad de poder obtener recursos de una misma manera, sin importar que estos sean remotos o locales.
- Transparencia en respuesta: Esto permite que se puedan utilizar diversas instancias de los objetos para incrementar la confiabilidad del sistema.
- Transparencia en ubicación: El sistema se ve como una entidad y cualquiera de los recursos existentes puede utilizarse de igual forma sin importar su localización geográfica.
- Transparencia en concurrencia: Tanto usuarios como aplicaciones debe operar al mismo tiempo, sin que el trabajo de uno interfiera con el otro.

- Transparencia en fallas: Evita que existan pérdidas en las tareas de los usuarios, a pesar de que ocurra alguna falla en el hardware o en el software.

- Transparencia en migración: Permite que existan movimientos en los objetos del sistema sin que esto repercuta en las aplicaciones o afecte a los usuarios.

- Transparencia en rendimiento: El sistema debe ofrecer flexibilidad en cuanto a la configuración se refiere; esto quiere decir, puede ser reconfigurado para incrementar su rendimiento sin que esto afecte a las aplicaciones o a los usuarios.

- Transparencia en escalabilidad: permite que el sistema pueda incrementar o disminuir su tamaño según se requiera. Esto debe llevarse a cabo sin necesidad de cambiar la estructura del sistema. Es decir, se puede agregar o quitar componentes.

1.1.3 Ventajas de los Sistemas Distribuidos

Por su estructura y sus características, los sistemas distribuidos ofrecen grandes ventajas a los usuarios. La primera es que se adecuan a un concepto común en la vida de los seres humanos: la distribución. En efecto, el hombre comparte información y recursos sin necesidad de complejos protocolos de comunicación u otros requerimientos, simplemente se comunica. Así que los sistemas distribuidos se acercan más al concepto que tiene el hombre de compartir recursos.

El costo y el rendimiento de los sistemas son reducidos, gracias a que cada componente del sistema se desenvuelve como una entidad. Es programado por separado y no dependiendo de otros componentes. No se requieren costosos sistemas de redes que requieran de una configuración exhaustiva. Además, los sistemas de comunicación han mejorado su rendimiento, implementando protocolos que permiten la rápida y efectiva transmisión de datos, incluyendo multimedia.

Otra ventaja importante es la modularidad, ya que un sistema distribuido deja de ser centralizado y cada una de las entidades que lo componen tienen integrado su propio sistema de control. Es decir, cada entidad es independiente y es programada cuidadosamente para que tenga un óptimo desempeño dentro de una comunidad de servicios.

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

En cuanto al concepto de expansión, se pueden agregar procesadores y servidores que incrementen la capacidad de almacenamiento y procesamiento del sistema como también nuevos componentes, dispositivos o periféricos.

La disponibilidad es siempre una ventaja primordial de los sistemas Distribuidos y se engloba bajo el concepto de redundancia, es decir los servicios deben permanecen al alcance de quien los solicite a pesar de que ocurra alguna falla en ellos, por supuesto siempre y cuando esta no sea esencial.

Por último, dentro de las ventajas de un sistema distribuido existe la confiabilidad. Esta cualidad se debe a que engloba todos los conceptos anteriores, en especial la disponibilidad, ya que si un componente está disponible, a pesar de que exista alguna falla, el usuario tiene como garantía que podrá seguir usándolo.

Así, una aplicación dentro de un sistema distribuido debe ser capaz de localizar entidades remotas, comunicarse con ellas y por último obtener de ellas los procesos que requiera. Existen

tecnologías que ofrecen esta funcionalidad y que permiten un manejo dinámico de entidades, específicamente para todo tipo de dispositivos y periféricos conectados a de una red.

Tecnologías tales como Jini [Sitio Web Jini] de Sun Microsystems, Universal Plug and Play [Sitio Web Microsoft's Universal Plug and Play] de Microsoft y Salutation [Sitio Web Salutation] de Salutation Consortium, son capaces de manipular dispositivos implementando el concepto de sistema distribuido.

1.2 La Implementación del Concepto

El concepto de Sistema Distribuido a sido interpretado de distintas formas en la industria tecnológica, generando así muchos productos diferentes. En esencia todos buscan cumplir con los elementos básicos de un sistema de este tipo, pero son las diferencias quienes nos ayudaran a estructurar el concepto de Descubrimiento de Servicios. A continuación se describen las distintas interpretaciones que ayudarán en esta tarea.

1.2.1 Interpretación Inicial: Software Distribuido

Se puede afirmar que esta interpretación forma parte de otras, ya que se constituye muchas veces en la base de comunicación de cualquier sistema distribuido.

1.2.1.1 Modelo Cliente / servidor

La arquitectura cliente / servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes.

En este modelo las aplicaciones se dividen en tal forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario.

Los clientes realizan generalmente funciones como:

- Manejo de la interfaz de usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.
- Por su parte los servidores realizan, entre otras, las siguientes funciones:
    - Gestión de periféricos compartidos.
    - Control de accesos concurrentes a bases de datos compartidas.
    - Enlaces de comunicaciones con otras redes de área local o extensa.

Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en computadores personales y/o estaciones de trabajo y los servidores en computadores departamentales o de grupo.

Entre las principales características de la arquitectura cliente / servidor están las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

1.2.1.2 Arquitecturas de Componentes Cliente / servidor

El modelo Cliente / servidor ha sido implementado de distintas formas mediante el avance conceptual en el desarrollo del software (Rodríguez, 2003), así es posible listar *Modelos de Clientes Servidor Orientados* a:

- Funciones: Socket
    - Implementación costosa
    - Orientado a funciones
- Remote Procedure Calls (RPC) (Van Steen, 2002)
    - No soporta objetos explícitamente.
- Objetos (Componentes Distribuidos) (Coulouris,2001)
    - Microsoft Distributed Component Object Model (DCOM)
        - Menos maduro, menos potable y además propietario.
    - Java Remote Method Invocation (RMI) (Rodriguez, 2003)
        - Sólo para Java.
    - Common Object Request Broker Architecture (CORBA) de la OMG. (Rodriguez 2003)
        - Multiplataforma, multilenguaje

A medida que la Arquitectura Cliente / servidor evoluciona, se acerca a la definición correcta de un Sistema Distribuido, no es justo realizar una comparación de las orientaciones anteriores, ya que cada una en su momento fue una mejora al desarrollo de software que a su vez constituyó la base para plasmar el siguiente avance.

1.2.2 Interpretación de la Industria Electrónica : La interoperabilidad de Dispositivos.

La masificación de redes computacionales y el exponencial crecimiento de Internet [Sitio WEB Crecimiento de Internet Hosts a nivel mundial] ha cambiado la forma en que los usuarios comparten, distribuyen y analizan información. Las ventajas de estar en línea en "cualquier lugar y momento" ha llevado a desarrollar tecnologías (Vinton, 2002) para conectar a redes no sólo computadores (que es nuestra concepción general) sino múltiples dispositivos (periféricos, electrodomésticos, etc.), sin la intervención de un computador personal.

Esta innovación rompe con el paradigma de considerar las redes como meros canales de comunicación que unen a múltiples tipos de computadores, realizando labores similares. Por el contrario visualiza una "red" como coordinadora de servicios, que entran y salen, brindando a usuarios una absoluta libertad respecto al uso de ellos.

Los dispositivos deben ser capaces de conectarse a alguna red sin necesitar la intervención de un computador personal y cumplir con los requisitos mínimos de software apropiado. Los avances en redes inalámbricas [Sitio Web Intel] deben llegar a convertirse en un estándar.

1.2.2.1 Comunicación Machine to Machine

El concepto Machine to Machine no es nuevo, hace ya tiempo que las máquinas electrónicas de muchas grandes empresas intercambian información, ahorrando tiempo, dinero y errores humanos en el trabajo cotidiano de muchas oficinas y fábricas. Las peticiones de repuestos, el aviso de sistemas dañados o la autogestión de energía son sólo algunas de sus aplicaciones.

Podemos pensar, por ejemplo, en cómo que sería pasar los peajes en las autopistas sin tener que parar el automóvil. En lugar de una señorita que nos indique la tarifa y nos cobre, el sistema informático carga a nuestra tarjeta telefónica el cobro, sin que nosotros tengamos que preocuparnos siquiera. Parece demasiado bueno para ser posible pero en algunos peajes de los Estados Unidos se hace de esta forma y aún más, en Chile está en proceso de implementación [Sitio Web Proyecto Autopista Central]. Y es sólo un ejemplo.

1.2.2.2 Conectividad Total

Este concepto, basado prácticamente en un sueño emblemático de la industria electrónica, se podría definir como la capacidad de interoperabilidad de múltiples dispositivos de una manera fácil para cumplir un objetivo. El siguiente ejemplo muestra claramente lo que se persigue:

Un Verdadero Control Remoto Universal

Te apresuras para llegar a casa desde la tienda de electrodomésticos cercana, ansioso de probar tu nuevo control remoto universal. Convencido de que este no es un control remoto común de Televisión o reproductor DVD. Este es realmente “universal” con este controlas y facilitas todos los aspectos cotidianamente hogareños (Ver Figura 1.1). En realidad este parece mas una PDA que un control remoto, aunque es mucho más que ambos.

Lo primero que haces al llegar a casa es probar si este realmente puede controlar tu televisor y reproductor DVD. Clic. La TV enciende. Clic, el DVD comienza a sonar. Clic, la función de picture and picture de tu TV aparece. ¡Genial! No fue necesario ingresar algún código especial de la TV o reproductor.

Estas verificando la habilidad para controlar tu Sistema de Sonido cuando el teléfono suena. Bien, ahora es cuando se presenta la gran prueba. Clic. La TV en mute y dices "hola" en un receptor remoto. Es tu Jefe diciendo que necesita el último informe de ventas. Clic. El informe se muestra en una pantalla remota. Tu Jefe dice que necesita una copia impresa. Clic . Y dices a tu Jefe que vaya a buscar la copia impresa a la sala de impresión fuera de su oficina. "Grandioso" dice tu Jefe. Clic, y cuelgas, las persianas de tu ventana se cierran, el volumen de la TV vuelve y te acomodas en el sillón para ver una de tus películas favoritas.

¿Qué permite a este verdadero control "universal" integrarse tan fácilmente y trabajar fantásticamente con tu red casera? La respuesta, una plataforma independiente y que soporta el cruce de protocolos [Sitio Web Salutation].

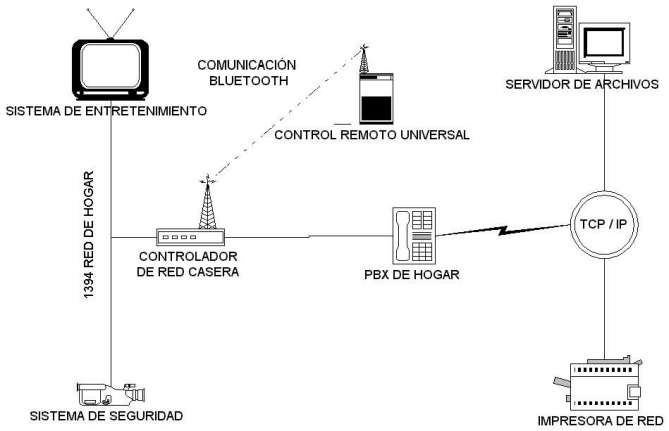


Figura Nº 1.1 Funcionamiento Control Remoto Universal

1.2.2.3 Entorno Inteligente

Cuando comenzaron con sus primeros ensayos con electrodomésticos de avanzada y dispositivos automáticos para el hogar, los franceses bautizaron domótica a una nueva disciplina arquitectónica, señalada por el espíritu de investigación y la búsqueda de la novedad que la técnica hacía posible. Para los diccionarios franceses de 1988, el término domotique era de uso aceptado, entendido como "el concepto de vivienda que integra todos los automatismos en materia de seguridad, gestión de la energía, comunicaciones y otros servicios". Para desentrañar su etimología, no hay más que combinar dos términos procedentes de distintas épocas y disciplinas: domus (casa, en latín) y telemática. Debido a la proliferación de términos nuevos que la informática provoca, el concepto también se asocia hoy al de tecnología del hogar inteligente [Sitio Web de Entorno Inteligente].

La domótica abre nuevas posibilidades en relación con la integración del hogar, como también constituye un medio para que el usuario pueda controlar y ¨gerenciar¨ su espacio cotidiano.

Por ejemplo, en un día típico, el dueño de una casa inteligente no deberá preocuparse por el estado del tiempo: su sistema despertador le informará, a través de la pantalla del velador, cómo se encuentra todo allí afuera, e incluso se encargará de despertarlo antes si el reporte de transito señala que se encontrará con un taco de tránsito insalvable de camino a la oficina. El sistema también estará programado para levantar de manera automática las persianas, regular la calefacción, encender la cafetera para que el desayuno esté listo, en el momento de salir de la ducha. Cuando comiencen los movimientos diurnos en la casa, las alarmas se desactivarán automáticamente, también el sistema de riego se pondrá en marcha durante la noche para aprovechar las horas de menor consumo de energía, se detendrá, y las luces del parque y el interior se apagarán una a una.

Al salir rumbo a la oficina, no hará falta revisar las puertas y ventanas: el sistema se encargará de informar si alguna está mal cerrada. Y luego, de manera remota, podrá conectarse con su casa para verificar que todo esté en orden, e incluso dar ordenes al sistema.

Básicamente se puede decir que la domótica se encarga del control de la vivienda, mediante hardware y software.

Algunas de las acciones que puede realizar son:

- Controlar la temperatura de los diversos recintos independientemente o en conjunto.
- Controlar la iluminación, tanto externa como interna y regularla según la presencia del individuo, bien mediante la regulación de las persianas, lámparas, tubos fluorescentes, etc.
- Regular el sistema de riego de plantas y jardines captando la humedad del terreno.
- Detectar inundaciones cortando el suministro de agua automáticamente.
- Detectar humo y/o gases activando la alarma y avisando al usuario.
- Detectar la presencia de intrusos, mediante sensores u otras técnicas, dando aviso y realizando llamadas telefónicas correspondientes.

1.2.2.4 ¿Qué relación existe entre esta "interpretación" y los Sistemas Distribuidos?

Uno de los grandes problemas de los tecnólogos es lograr separar el concepto original "puro" y definido, de los productos generados por él.

La interoperabilidad de Dispositivos cumple con la mayoría de las características de un Sistema Distribuido transparencia de acceso, respuesta, ubicación, concurrencia, etc. Así se podría afirmar que la interpretación más usada de un sistema Distribuido "Colaboración de elementos software" puede ser expandida también a la interacción entre dispositivos y periféricos.

1.2.3 Interpretación del área WEB: Servicios Web.

Los Servicios Web constituyen el siguiente paso en la evolución de la tecnología orientada a objetos, y representan una revolución al cambiar arquitecturas tradicionales tipo cliente-servidor a distribuidas o a tipo igual-a-igual (peer-to-peer). Su importancia y distinción de otras propuestas Software, justifica mencionarla como un tópico independiente a pesar de que clasifica dentro de la Interpretación de Software.

Estos servicios se sustentan por un conjunto de estándares que permiten a los desarrolladores implementar aplicaciones distribuidas. Utilizando herramientas muy distintas se crean aplicaciones que utilizan una combinación de módulos de software que pueden ser llamados desde diversos sistemas distribuidos en regiones geográficas distintas. La arquitectura de los servicios Web es una meta-arquitectura. Permite que ciertos servicios de red sean dinámicamente descritos, publicados, descubiertos e invocados en un ambiente de cómputo distribuido.

Los servicios Web son aplicaciones auto-contenidas y modulares que pueden ser:

- Descritas mediante un lenguaje de descripción de servicio, como el lenguaje WSDL (Web Service Description Language)
- Publicadas al someter las descripciones y políticas de uso en algún Registro bien conocido, utilizando el método de registro UDDI (Universal Description, Discovery and Integration) u otro.
- Encontradas al enviar peticiones al Registro y recibir detalles de uso del servicio que se ajusta a los parámetros de la búsqueda.
- Asociadas al utilizar la información contenida en la descripción del servicio para crear una instancia de servicio disponible o proxy.

- Invocadas sobre la red al utilizar la información contenida en los detalles de uso de la descripción del servicio.

- Compuestas con otros servicios para integrar servicios y aplicaciones nuevas.

Operaciones de Servicios Web:

- *Publicar / cancelar.* Los proveedores de servicios publican (publicitan) la disponibilidad de su servicio a uno o más Registros de servicios, o cancelan la publicación de su servicio.

- Buscar. Los solicitantes de servicios interactúan con uno o más Registros de servicios para descubrir un conjunto de servicios comerciales con los que pueden interactuar para encontrar una solución.

- Ligar, Unir (Bind). Los solicitantes de servicios negocian con los proveedores de servicios para acceder e invocar servicios .

Entre las razones por las cuales los servicios Web jugarán un rol principal en la siguiente generación de sistemas distribuidos, están:

- Interoperabilidad. Cualquier servicio Web puede interactuar con cualquier otro servicio Web. El protocolo estándar SOAP permite que cualquier servicio pueda ser ofrecido o utilizado independientemente del lenguaje o ambiente en que se haya desarrollado.

- Omnipresencia. Los servicios Web se comunican utilizando HTTP y XML. Cualquier dispositivo que trabaje con éstas tecnologías puede tanto ser huésped y acceder a los servicios Web. Por ejemplo, pronto serán utilizados en teléfonos, automóviles o aún en máquinas vendedoras de refrescos. Una máquina de venta de refrescos puede comunicarse vía inalámbrica con el servicio Web de un proveedor local y ordenar un pedido de suministro.

- Barrera mínima de participación. Los conceptos detrás de los servicios de Web son fáciles de comprender y se ofrecen Herramientas de Desarrollo (ToolKits) por IBM, Sun Microsystems, la Organización de Apache y muchas otras. Estas permiten a los desarrolladores crear e implementar rápidamente servicios de Web.

- Apoyo de las Industrias. La gran mayoría de las compañías apoyan el protocolo SOAP y la tecnología derivada de los servicios Web.

1.2.4 Otras Interpretaciones.

Aunque están bajo la categoría de Software Distribuido, su propuesta innovadora pero aún inmadura obliga a considerarlas como interpretaciones independiente.

1.2.4.1 Aplicaciones P2P (peer to peer)

Las redes P2P convierten el PC del usuario, usado principalmente para recibir información de Internet, en un elemento activo que permite a los navegantes intercambiar información entre ellos y agrupar capacidad de procesamiento. Haciéndolo además al margen de la World Wide Web.

Se podría incluso hablar de aplicaciones de proceso distribuido (compartir labores de procesamiento), lo que les permite colaborar en la resolución de tareas muy complejas y costosas, especialmente de carácter científico. Otra aplicación es el intercambio libre de archivos. Las industrias tradicionales ven impotentes como las obras de sus artistas, protegidas por las leyes de la propiedad intelectual, desfilan incontroladas ante sus ojos.

Este tipo de aplicaciones está aún en una etapa de desarrollo, hay muchos problemas a superar para considerarse una alternativa de producto eficiente. La mayoría de ellos funcionan bajo una administración centralizada y aquellas que funcionan independientes son muy inestables y lentas ya que depende de los atributos de cada usuario (capacidad de procesamiento, ancho de banda, configuración de software, etc.) para que el sistema como un todo funcione correctamente.

1.2.4.2 Aplicaciones de Mensajería Instantánea (MI)

La Mensajería Instantánea es un punto intermedio entre los sistemas de chat y los mensajes de corre electrónico. Las herramientas de mensajería instantánea son programas regularmente gratuitos y versátiles, que residen en el escritorio y, mientras hay una conexión a Internet, siempre están activos.

El servicio de mensajería instantánea ofrece una ventana donde se escribe el mensaje, en texto plano o acompañado de iconos o "emoticons" (figura que representan estados de ánimo), y se envían a uno o varios destinatarios quienes reciben los mensajes en tiempo real. El receptor lee y puede contestar en el acto.

Últimas versiones han añadido una serie de aplicaciones extra como la posibilidad de entablar conversaciones telefónicas por Internet, Sistemas de información financiera en tiempo real, compartir diferentes tipos de archivos y programas incluidos juegos en línea.

Algunas implementaciones de mercado entregan funcionalidad que satisfacen postulados esenciales que se pueden asimilan con los sistemas distribuidos, por ejemplo: permitir a los usuarios “Compartir Recursos” con una comunidad deseada e incluso “Interactuar” a través de medios: escritos, visuales y multimediales en general para algún objetivo común.

1.3 El concepto de "Descubrimiento Servicio Computacional Distribuido"

El diccionario de La Real Academia Española [Sitio Web Diccionario De La Lengua Española] entrega algunas definiciones de "servicio" :

1. Acción y efecto de servir.
2. Favor que se hace a alguien.
3. Organización y personal destinados a cuidar intereses o satisfacer necesidades del público o de alguna entidad oficial o privada.
4. Prestación humana que satisface alguna necesidad social y que no consiste en la producción de bienes materiales.
5. Actividad llevada a cabo por la administración o, bajo un cierto control y regulación de esta, por una organización, especializada o no, y destinada a satisfacer necesidades de la colectividad.

Todas estas tienen en común esencialmente tres componentes:

- **Ser de utilidad**: La etimología da un claro indicio de "servir", es decir debe satisfacer un objetivo claro de la mejor manera posible.
- **Tener destinatario**: Esto no significa necesariamente que debe ser acotado a un receptor, más bien implica clarificar cual es el ámbito de acción, esto para listar rangos de características de los posibles destinatarios y adaptarse a ellos para responder no solo en forma eficaz sino también eficiente.
- **Sentido dinámico**: Acción, favor, organización, prestación o actividad. Todas y cada una de ellas entregan un sentido sistémico a la definición, mostrando de esta forma que un servicio no debe ser algo estático que sea incapaz de responder a cambios en su entorno sino capaz de retroalimentarse y adaptarse a ellos.

Superponiendo las distintas Interpretaciones de Sistema Distribuido, descritas anteriormente, y las características de un "Servicio" (en el sentido amplio de su significado) se puede entender que todas las propuestas entregan las herramientas suficientes para considerarlos gestores de un "Servicio Computacional" que además responde a las características de Distribuido ver Figura 2.

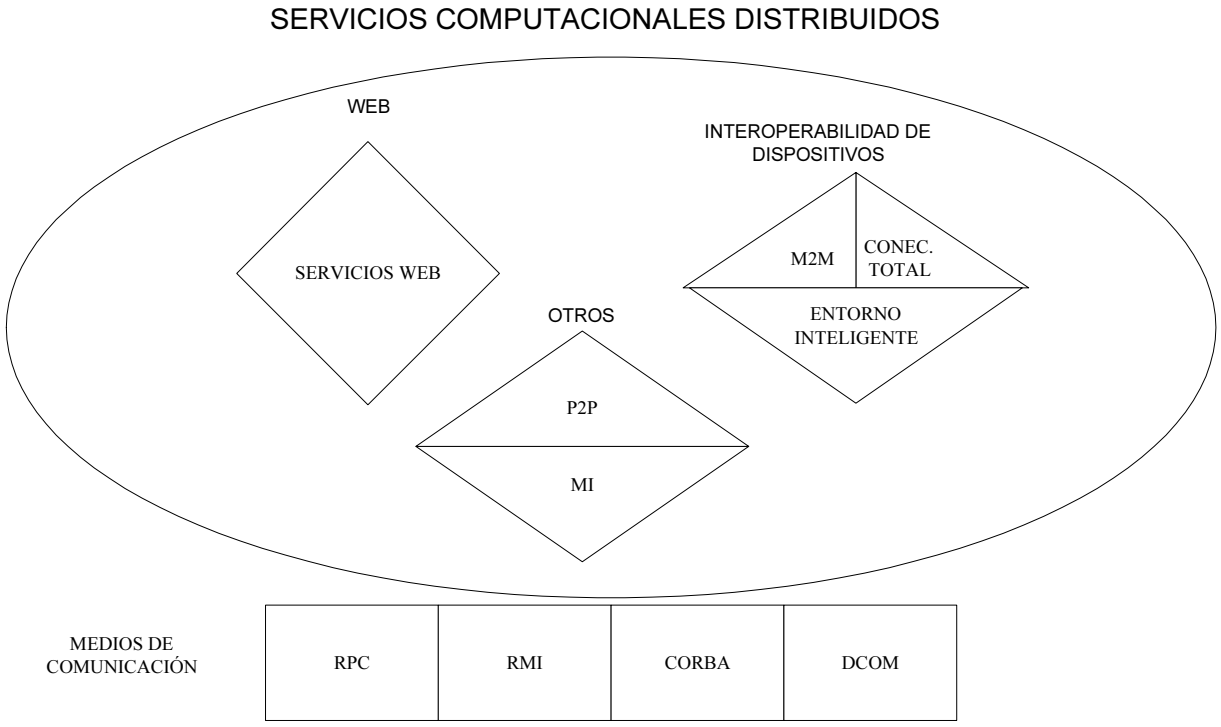


Figura Nº 1.2 Servicios Computacionales Distribuidos

Comprendiendo entonces que existen múltiples productos construidos con distintas tecnologías influidas por estas cuatro interpretaciones básicas de Sistemas Distribuidos, es posible plantearse un desafío fundamental en la publicación de estos "Servicios Computacionales Distribuidos":

- ¿Cómo soy capaz de enterarme de su existencia?
- ¿De que manera se establece la comunicación?
- ¿Cómo logro responder a las preguntas anteriores en escenarios muy disímiles?

Las distintas propuestas, dentro de cada interpretación, enfrentan este desafío de la mejor manera posible. Las Arquitecturas de Objetos Distribuidos introducen una nueva capa en las comunicaciones llamada "Middleware". Esta se encarga de transparentar el proceso de descubrimiento de los servicios. Por su parte, los Servicios Web utilizan estándares en la descripción, descubrimiento y comunicación de ellos (WSDL, UDDI, SOAP). Las aplicaciones P2P y de Mensajería Instantánea son las más retrasadas en este sentido, ya que comúnmente requieren saber de ante mano la ubicación del servicio al que desean acceder (IP, puerto, etc.), aunque se han implementado mejoras como directorios de usuarios o simplemente, tal como la mayoría de los programas de Mensajería Instantánea más populares, coordinados por un servidor central.

La interoperabilidad entre dispositivos, a diferencia de los anteriores, es aún una lucha por la estandarización. En la mayoría de los foros de discusión del tema y algunas revistas especializadas destacan la importancia de definir una utilización universal, para evitar futuros problemas de incompatibilidad que no serán tan fáciles de solucionar como en las aplicaciones computacionales, ya que aquí existe un componente de restricción, la estrecha relación del hardware y software en ellos. Actualmente existen propuestas y cada una apoyada por actores del mercado que aseguran tener la solución definitiva. Por ejemplo: JINI, Salutation, UPnP y SLP. Cada uno de estas será objeto de estudio más adelante.

Quizás todo lo anterior presenta una escena difícil de superar, pero de hecho existen sistemas que han resuelto este tipo de problemática y muchas más. Dos ejemplos de ello son, la sociedad humana y los organismos biológicos.

Sociedad [Sitio Web Diccionario De La Lengua Española]: Agrupación natural o pactada de personas, que constituyen unidad distinta de cada uno de sus individuos, con el fin de cumplir, mediante la mutua cooperación, todos o alguno de los fines de la vida.

"Es evidente que la organización de los sistemas biológicos no es consecuencia de una evolución hacia el desorden molecular. El orden biológico es arquitectónico, funcional y cognitivo, además, en el nivel celular y supra-celular, se manifiesta por una serie de estructuras y funciones acopladas de creciente Complejidad y carácter jerárquico." (Teilhard *et al*, 1963)

Ambos sistemas poseen las siguientes propiedades:

- Interconectados
- Dinámicos.
- Tolerantes a fallas.
- Distribuidos.
- Bien organizados.
- Auto-reparables
- Diseñado en capas.
- Diseñado a partir de componentes simples.

Lo cual plantea, sin lugar a dudas, un gran desafío.

Capítulo 2. Tecnologías Involucradas

El descubrimiento de servicios computacionales distribuidos, nace de la necesidad de integrar dispositivos a través de una red en forma trasparente al usuario, publicando estos como servicios a disposición de quien desee utilizarlos. Así los protocolos de descubrimiento de servicio pueden ser implementados en microprocesadores presentes en celulares, impresoras, scanner y toda clase de aparatos.

Aunque los objetivos generales de este proyecto son diseñar y construir un prototipo que permita publicar y utilizar un Servicio Computacional Distribuido, limitado a la tecnología hardware presente en la facultad, comprender todas las tecnologías involucradas en el tema contribuirá a evaluar los protocolos de descubrimiento en un escenario real.

Fundamentalmente son dos las tecnologías que deben interactuar para crear un dispositivo de las características de un servicio computacional distribuido:

- Hardware
    - Tecnologías de comunicación.
    - Microprocesadores.
- Software
    - Protocolos de descubrimiento de servicios.

2.1 Hardware

2.1.2 Tecnologías de comunicación Móvil y de Hogar

Existe un crecimiento masivo en el uso de tecnologías inalámbricas y comunicaciones móviles, así como la demanda de ancho de banda debido a la masificación de Internet y nuevos servicios multimedia.

La poca regulación inicial de las telecomunicaciones ha fragmentado el mercado, de manera que hoy se necesita promover estándares que garanticen la Interoperabilidad entre dispositivos de distintos operadores y/o fabricantes.

La Figura Nº 2.1 representa las tecnologías de comunicaciones actuales, en función del ancho de banda que ofrecen frente al tipo de movilidad que soportan.

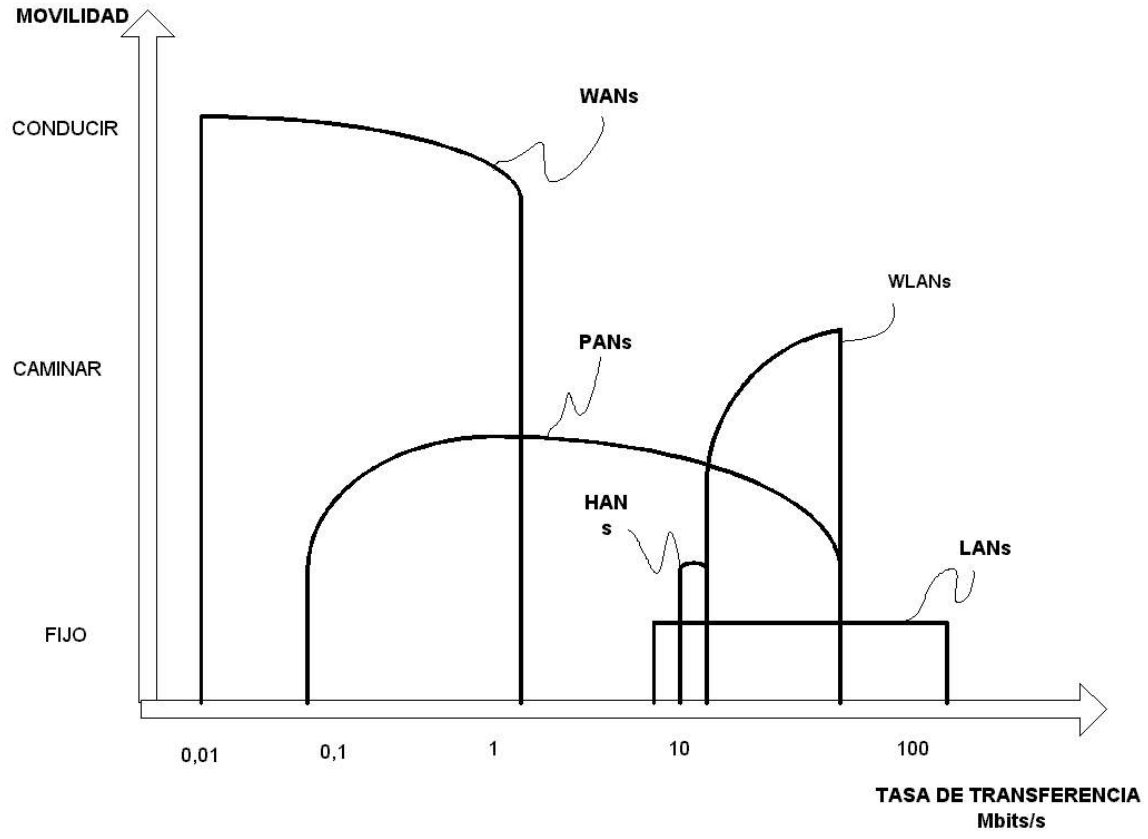


Figura Nº 2.1 Tecnologías de Comunicación Actuales

A continuación una descripción de las tecnologías de comunicación clasificadas según su aplicación.

2.1.2.1 Redes de Área Extensa (WANs)

Una Red de Área Extensa es aquella que cubre una amplia superficie o área geográfica (desde un edificio o campus hasta miles de kilómetros).

Tradicionalmente, las redes WAN funcionaban a menor velocidad que las LAN, y por ello, su implantación era menor. Sin embargo debido a las mejoras continuas en este tipo de redes la diferencia entre los conceptos de "local" y "remoto" tiene una frontera progresivamente menos definida (Ver Tabla 2.1).

Tabla 2.1 Tecnología de redes inalámbricas de área extensa

Tipo	**Tecnología**	**Banda de Frecuencia**	**Alcance**	**Tasa de datos**
2G	GSM	900 / 1800 / 1900 MHz	*	9,6 kbps
2.5G	GPRS	900 / 1800 / 1900 MHz	*	≤ 171 kbps
	EDGE	900 / 1800 / 1900 MHz	*	≤ 384 kbps
3G	UMTS	2 GHZ	*	≤ 2 Mbps

* El alcance dependerá de la infraestructura disponible, pero normalmente será toda una ciudad.

2.1.2.2 Redes de Área Local Inalámbricas (WLANs)

La Tecnología Wireless Local Area Network (WLAN) complementa tecnologías de acceso para la Tercera Generación de las redes celulares. Los estándares WLAN, HIPERLAN/2 e IEEE 802.11, permiten incluso rangos de datos más altos que UMTS (por arriba de los 54 Mbps) para cubrir áreas conflictivas y en la ciudad (Ver Tabla 2.2).

Tabla 2.2 Tecnologías de redes inalámbricas de área local

Tecnología	**Banda de frecuencia**	**Tasa de datos**	**Alcance**
IEEE 802.11b,g	2,4 GHz	11 Mbps	100 m
IEEE 802.11g	2,4 GHz	54 Mbps	100 m
IEEE 802.11ª	5 GHz	54 Mbps	100 m
Hiperlan /2	5 GHz	54 Mbps	150 m

2.1.2.3 Redes de Área Personal (PANs)

Son todos estándares de comunicación inalámbricos personales, usados para comunicar un equipo portátil o de escritorio, con toda una serie de periféricos: impresora, módem, teléfono móvil, teclado, etc. todo ello sin cables, y a una distancia reducida, normalmente circunscrita a una sala o habitación (Ver Tabla 2.3).

Tabla 2.3 Tecnologías de redes de área personal

Tecnología	**Banda de frecuencia**	**Tasa de datos**	**Alcance**
IrDA	Infrarrojos	115 kbps 1.152 Mbps 4 Mbps	2 m
Bluetooth v1.1	2,4 GHz	1 Mbps	10 / 100 m
IEEE 802.15.3	2,4 GHz	55 Mbps	10 m
IEEE 802.15.4	868 MHz 915 MHz 2.4 GHz	20 kbps 40 kbps 250 kbps	10–20 m

2.1.2.4 Redes del Hogar (HANs)

Actualmente en algunos hogares existe más de un computador personal, y esto se generalizará en cuestión de tiempo, así la necesidad de conectarlos para compartir recursos se vuelve una necesidad. Junto con esta iniciativa nace la de interconectar juntamente electrodomésticos y aparatos en general para tener un mayor control a través de la red. A continuación se describen algunas iniciativas tecnológicas relacionadas con lo que se conoce como redes del Hogar.

Tabla Nº 2.4 Tecnología de redes del Hogar

Tecnología	**Banda de frecuencia**	**Tasa de datos**	**Alcance**	**Medio físico**
HomeRF	2,4 GHz	20 Mbps	50 m	Inalámbrico
HomePNA	4-10 MHz	10 Mbps	-	Línea telefónica
HomePlug	4,3-20,9 MHz	14 Mbps	-	Red eléctrica

2.1.3 Microprocesador

La electrónica ha sido causante y soporte de esta verdadera revolución en nuestra sociedad, pero dentro del campo de la electrónica el microprocesador es con frecuencia el protagonista. Este hecho no es ajeno para la mayoría de las personas que hablan sin problemas de su nuevo y flamante Pentium IV, o de su Macintosh con procesador PowerPC. No obstante, la mayor parte de los microprocesadores no residen en los computadores, sino ocultos en cualquier dispositivo electrónico, que el usuario utiliza sin estar consciente de ello.

El microprocesador es un producto que surge del "matrimonio" de las tecnologías microelectrónicas y de la arquitectura de computadores, se podría definir entonces como un componente electrónico que contiene, en un único encapsulado, el procesador (CPU) de una máquina programable de tratamiento de información.

La clave del éxito de los microprocesadores como componente electrónico reside en que modificando el programa almacenado en memoria pueden adaptarse a numerosas y diferentes aplicaciones. De este modo los microprocesadores no solo se utilizan para construir potentes computadores, quizás su aplicación más espectacular y conocida, sino también para electrodomésticos, dispositivos de comunicación, sistemas de control, periféricos específicos, etc.

Los Microcontroladores son Microprocesadores especiales generalmente de menores prestaciones de cálculo, pues están optimizados para tareas específicas, con numerosos dispositivos integrados en un único encapsulado, como memoria, puertos de entrada / salida o diversos periféricos. Estos se emplean en el desarrollo de sistemas industriales (por ejemplo, robótica, autómatas programables o tarjetas de control industrial), aplicaciones de consumo (como lavadoras, hornos microondas, audio y video, cámaras fotográficas o videojuegos), telecomunicaciones (teléfonos móviles, PDAs), automoción (frenos ABS, inyección

electrónica, computador de abordo, etc.). La sustitución de lógica convencional por microcontroladores permite reducir el tamaño de diseño, tiempo de desarrollo y así el costo final del producto.

Clasificación de los microprocesadores

Según su propósito, podemos distinguir:

- Microprocesador de propósito general. Válidos para múltiples tareas diferentes de tratamiento de la información. Son los que se emplean fundamentalmente para construir computadores o máquinas de propósito general (editar, realizar cálculos, diseñar, etc.).

- Microprocesador de propósito específico. Son los que se programan para llevar a cabo una tarea o aplicación específica, es decir, se programan una sola vez y después siempre ejecutan el mismo programa, que se guarda en ROM

Los microprocesadores también pueden clasificarse por el número de instrucciones que incorporan en su repertorio:

- CISC (Complex Instruction Set Computer) Computadores de extenso conjunto de instrucciones, es decir, aquellos que incluyen muchas instrucciones diferentes (del orden de cien). El ejemplo clásico es la familia Intel 80x86.

- RISC (Reduced Instruction Set Computer) Son las máquinas de conjunto de instrucciones reducido, es decir, incluyen relativamente pocas instrucciones en su repertorio (del orden de treinta). Ejemplos de ellos son la familia Alpha, SPARC y PowerPC.

La distinción únicamente en cuanto al número de instrucciones que incorpora no permite apreciar a primera vista la gran trascendencia que conlleva, que ha sido causa de una dura batalla en el estilo de diseño de microprocesador en los últimos años

Ventajas del uso de microprocesadores

Las ventajas del uso de microprocesadores en el diseño de sistemas digitales sustituyendo lógica convencional, son las siguientes:

a) Reducción del número de componentes, lo que conlleva la miniaturización del sistema, reducción de consumo y aumento de fiabilidad.

b) Programación, permite la simplificación del diseño y reducción del tiempo de desarrollo.

c) Disminución de costos, causado por los aspectos descritos en los puntos a) y b) y por la fabricación en serie de muchas unidades de un mismo microprocesador, pues podrá aplicarse a muchos diferentes problemas solo reprogramándolo.

2.1.3.1 Microprocesadores Actuales

El sector industrial de los Microprocesadores y específicamente de los microcontroladores, es bastante dinámico por lo que sus productos mejoran constantemente en características, lanzando nuevas versiones al mercado. Las principales empresas fabricantes de Microcontroladores en el mundo son [Sitio Web The Embedded Systems Sourcing]: Dallas Semiconductor, National Semiconductor, Zilog, Analog Devices, Atmel Corporation, Cygnal Integrated Products, Hitachi Semiconductor, Infineon technologies, Axis, Intel, Texas Instruments. Este amplio mercado garantiza la posibilidad de encontrar el microcontrolar adecuado casi para cualquier tipo de aplicación y a un precio asequible.

2.2 Software

El campo del descubrimiento de servicios, está aún en pleno desarrollo. Es por esto que agrupaciones de compañías, universidades y personas interesadas en el tema forman un consorcio para desarrollarlo. Esto genera varios problemas con los protocolos de descubrimiento propuestos por ellos:

- Nadie garantiza compatibilidad entre ellos.
- Uso de terminología diferente.
- Se dificulta la comparación, ya que cada cual interpreta las necesidades de diferente manera.
- Poco compromiso de las compañías desarrolladoras de dispositivos. Varias de ellas pertenecen a muchos consorcios, lo cual es un impedimento al momento de decidir que protocolo soportará determinado dispositivo.

A pesar de estos contratiempos, la interacción e investigación generada por estos consorcios ha entregado al campo del descubrimiento de servicios computacionales un desarrollo continuo. Alguien (mercado, compañías, entidades de regulación) deberá tomar la decisión respecto a cual de los protocolos se generalizará, o bien enfocar a cada uno un nicho de mercado diferente.

Una variedad de protocolos de descubrimiento de servicios están actualmente en desarrollo. Los más conocidos, analizados en profundidad en el siguiente capítulo, son:

- Service Location Protocol (SLP), desarrollado por IETF.
- JINI, un acercamiento basado en Java de Sun.
- Salutation.
- Universal Plug and Play (UpnP) liderado por Microsoft.

La figura Nº 2.2 muestra las compañías que contribuyen activamente al desarrollo de estos protocolos.

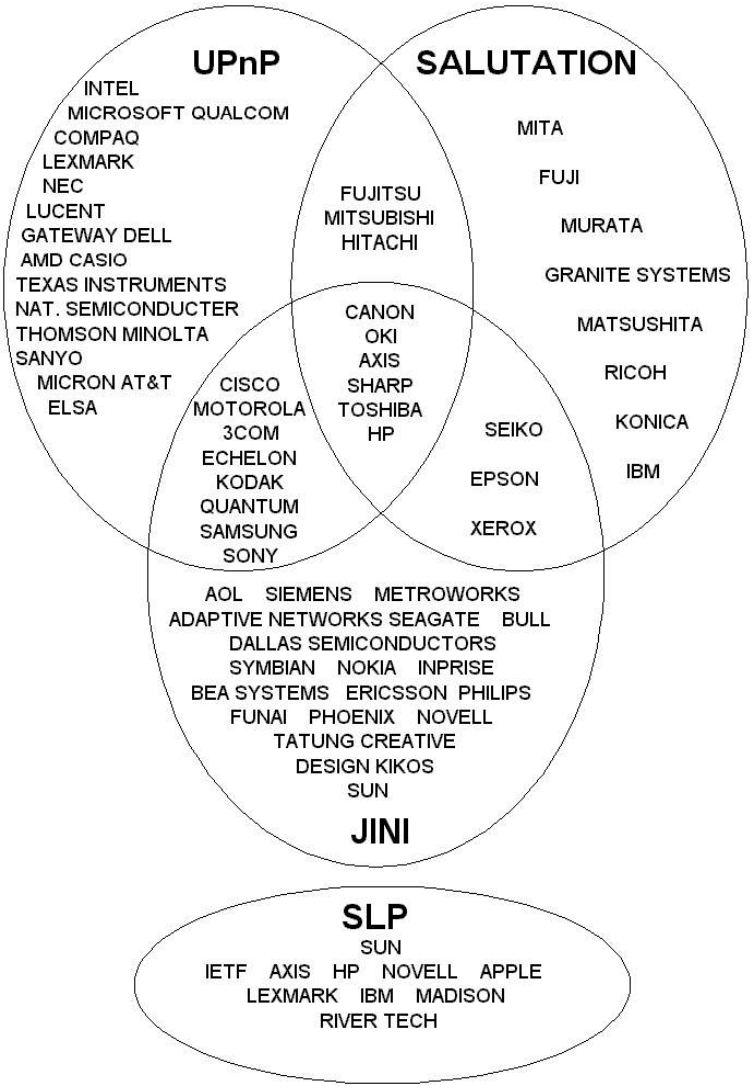


Figura Nº 2.2 Compañías asociadas a los protocolos de descubrimiento de servicios

2.3 Posibilidad Real de Integración

Tecnologías Inalámbricas

Wi-Fi Certified, es una certificación que concede la Wi-Fi Alliance [Sitio Web Wi-Fi Alliance] (asociación de más de 130 fabricantes y proveedores de dispositivos) para garantizar que un producto es ínter operable con aparatos de otros fabricantes en una red sin cables. Actualmente existen alrededor de 450 aparatos que cuentan con este certificado, lo cual habla de un gran paso hacia la integración.

Por otro lado este tipo de redes no tienen ningún problema de compatibilidad con los diferentes protocolos de descubrimiento de servicio ya que la mayoría no son dependientes de la capa física e incluso algunos se adaptan a variados protocolos de comunicación y esta tendencia se generaliza cada vez más. No sucede así con la integración en microprocesadores, ya que para optimizar su comunicación estos últimos deben poseer el controlador necesario para que la comunicación inalámbrica sea fácilmente implementada en el dispositivo.

Microprocesadores

Actualmente existen circuitos integrados con las características más que suficientes para que un dispositivo funcione como un Servicio Distribuido, la dificultad es seleccionar el adecuado para cada caso, considerando los requerimientos del protocolo y muchas otras características (por Ejemplo, si es Jini idealmente el microprocesador debe soportar o tener embebida una JVM). En cuanto a su integración con las tecnologías de comunicación, sólo dependerá de la dominante en el mercado, ya que las placas que integran estos microprocesadores normalmente traen algún chip de comunicación. Así en la medida que una tecnología inalámbrica se masifique este tipo de integración también lo hará.

Puentes entre Protocolos de Descubrimiento de Servicios

La variedad de diferentes protocolos de descubrimiento de servicios hacen necesario tener puentes entre los protocolos para habilitar también descubrimiento entre dispositivos que no utilicen el mismo. Por ejemplo, descubrir impresoras SLP con dispositivos Salutation.

La real posibilidad de integración, entre estas fundamentales líneas tecnológicas, es muy alta ya que cada cual se ha preocupado de este tópico. Hoy es inviable desarrollar un producto que adolezca de incompatibilidad, ya que el mercado en general trabaja con una infinidad de plataformas y soluciones a las que debe ser integrado.

Capítulo 3. Protocolos de descubrimiento de servicios

Este capítulo entrega un resumen de las características de algunos de los protocolos de descubrimiento de servicio más importantes. La metodología utilizada fue la recolección de documentos técnicos oficiales del consorcio, su correspondiente traducción y resumen.

3.1 JINI (Para mayor detalle ver Anexo A)

Sun Microsystems introduce Jini, basado en tecnología Java, en 1998. El corazón de Jini es un trío de protocolos: discovery, join y lookup. Un par de estos protocolos, discovery y join, ocurren cuando se conecta un dispositivo a la red; discovery ocurre cuando un servicio busca un servicio lookup con quien pueda registrase, y join ocurre cuando un servicio localiza un servicio lookup y desea suscribirse a este. Lookup ocurre cuando un cliente o usuario localiza e invoca un servicio descrito por su tipo de interfaz (escrita en el lenguaje de programación Java) y posiblemente otros atributos. Para que un cliente en una comunidad Jini use un servicio:

- El proveedor de servicio debe localizar un servicio lookup por una petición multicast en la red local o por conocimiento a priori de un servicio lookup remoto.
- El proveedor de servicio debe registrar un objeto servicio y sus atributos en un servicio lookup. Este objeto servicio contiene una interfaz Java para el servicio, que incluye los métodos que usuarios y aplicaciones invocarán para ejecutar el servicio (Ver figura 3.1a).
- El cliente entonces solicita un servicio por el tipo y quizás por atributos. El servidor lookup envía una copia del objeto servicio a través de la red al cliente, quien usa este para conversar con el servicio (Ver figura 3.1b).
- El cliente interactúa directamente con el servicio vía el objeto servicio. (ver figura 3.1b).

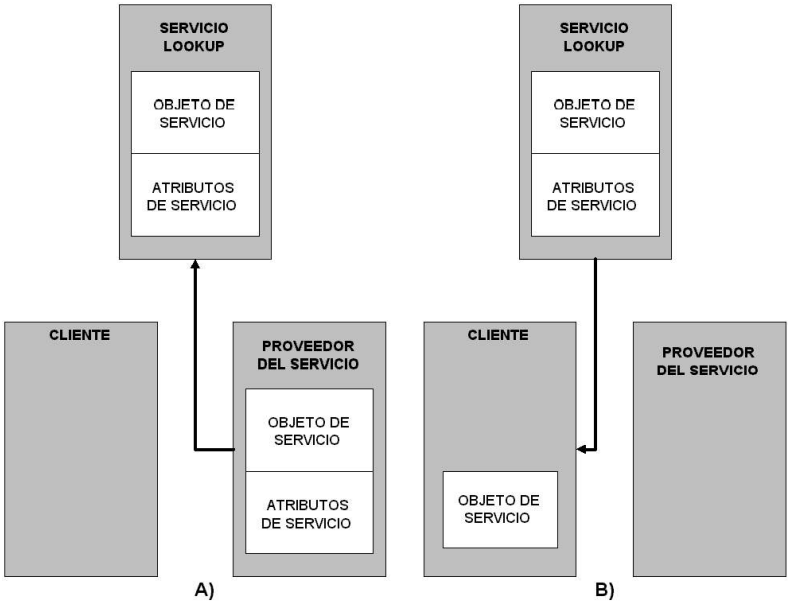


Figura 3.1 Funcionamiento de Jini

La tecnología Jini se compone de una infraestructura y un modelo de programación que determina como dispositivos se conectan con otros para formar una improvisada comunidad. Jini usa el método de invocación remota de Java (RMI) para mover código por la red.

Se puede visualizar el servicio Jini lookup como un servicio de directorio. Jini usa tres protocolos de descubrimiento relacionados. Cuando una aplicación o servicio es recién activada, el protocolo de petición multicast busca un servicio lookup en la vecindad. Los servicios lookup utilizan el protocolo de anuncio multicast para anunciar su presencia a servicios en la comunidad que pudieran estar interesados. El protocolo de descubrimiento unicast entonces establece comunicación con un específico servicio lookup ya conocido.

Sin embargo, un servicio Jini lookup no es solamente un servidor de nombres. Este traza las interfazs con que los clientes ven a los objetos proxy servicio. Este también mantiene atributos de servicio y procesa consultas. Los clientes descargan el proxy servicio que es usualmente un stub RMI que puede comunicarse con el servidor. Este proxy objeto permite a clientes usar el servicio sin saber nada acerca de este. Aquí, no son necesarios los drivers en caso de que el servicio proveído sea un periférico (por ejemplo impresora). El objeto servicio

descargado puede ser el servicio mismo o un objeto inteligente capaz de conversar en un protocolo de comunicación privado.

3.1.1 Arrendamiento en Jini

Jini concede acceso a sus servicios en base al arrendamiento. Un cliente pude solicitar un servicio por un determinado periodo de tiempo, y Jini dará un arriendo negociado por ese periodo. Este arriendo debe ser renovado antes que este expire; sino Jini soltará los recursos asociados con el servicio. El arrendamiento permite a Jini ser robusto cuando se enfrenta a fallas abruptas o desconexión de dispositivos y servicios.

3.1.2 Programación distribuida en Jini

Además del servicio básico de descubrimiento y mecanismos join y lookup, Jini soporta eventos remotos y transacciones que ayudan a programadores a escribir programas distribuidos en una forma fiable y escalable. Los eventos remotos notifican un objeto cuando un cambio deseado ocurre en el sistema. Nuevos servicios publicados o algunos cambios de estado en servicios registrados pueden activar estos eventos. Por ejemplo, el servicio lookup puede notificar a una PDA Jini, que tiene registrado su interés en impresoras, cuando una de ellas esté disponible. También, Jini soporta la noción de transacciones y compromiso atómico de ellas.

3.2 Universal Plug and Play

UpnP es una evolución del estándar inicial de Microsoft que extiende el modelo de periféricos Plug-and-Play. Esto apunta a permitir el anuncio, descubrimiento y control de dispositivos en red, servicios y electrodomésticos. En UPnP, un dispositivo puede dinámicamente unirse a una red, obtener una dirección IP, transmitir sus capacidades por

demanda, y conocer la presencia y capacidades de otros dispositivos. Un dispositivo también puede abandonar una red fácilmente, y automáticamente abandonar el estado de disponibilidad anterior. UpnP está influenciado por TCP/IP y tecnologías Web, incluyendo IP, TCP, UDP, HTTP, y XML. Este usa la pila de protocolos mostrada en la figura 3.2 para el descubrimiento, anuncio, descripción y eventos.

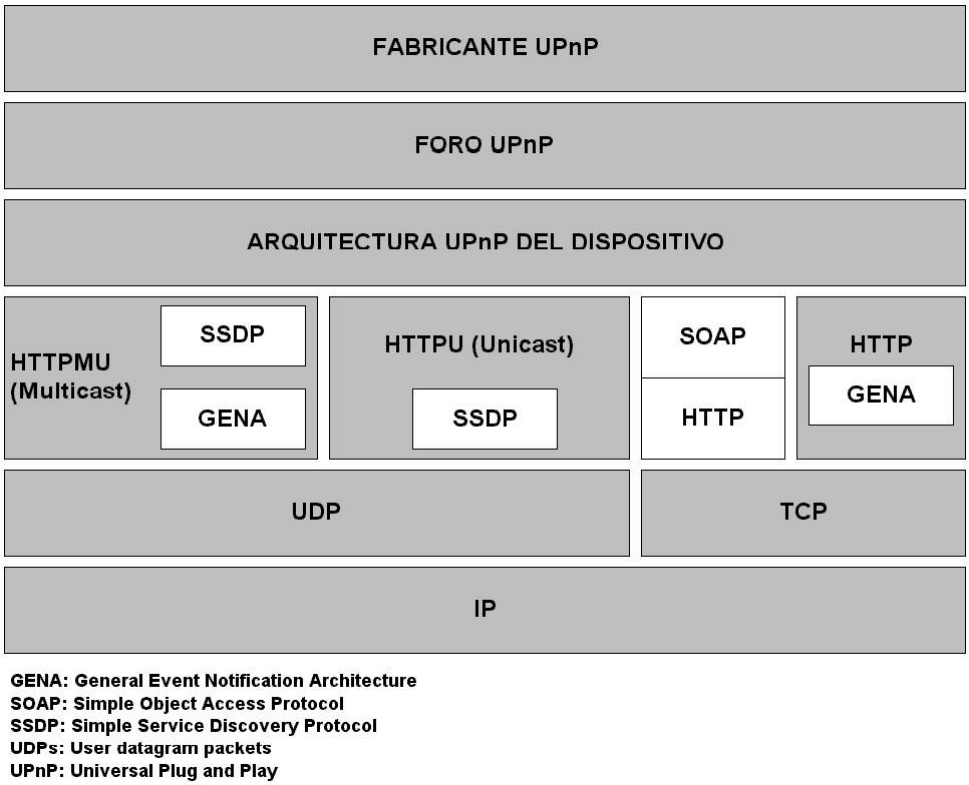


Figura 3.2 Protocolos utilizados por UPnP

3.2.1 Suscripción y descubrimiento en UpnP

UpnP usa Simple Service Discovery Protocol (SSDP) para el descubrimiento de servicios. Este protocolo anuncia la presencia de un dispositivo a otros y descubre otros dispositivos o servicios. Por consiguiente, SSDP es análogo al trío de protocolos en Jini. SSDP usa http sobre multicast y unicast UDP, conocidos como HTTPMU y HTTPU, respectivamente.

Un dispositivo acoplado envía un mensaje de anuncio (ssdp:alive) multicast para anunciar sus servicios a puntos de control. Los puntos de control funcionan de forma similar que los servicios lookup de Jini. Un punto de control, si está presente, puede guardar el anuncio, u otros dispositivos pueden también ver directamente este mensaje multicast. UpnP puede trabajar con o sin los puntos de control (servicio lookup). Este envía un mensaje multicast de búsqueda (ssdp:discover) cuando un nuevo punto de control es agregado a la red. Cualquier dispositivo que escuche este multicast responderá con un mensaje unicast de respuesta.

UpnP usa XML para describir las capacidades y características de los dispositivos. El mencionado mensaje de anuncio contiene una URL que apunta a un archivo XML en la red que describe las capacidades del dispositivo. Recuperado este archivo XML, otros dispositivos pueden inspeccionar la información de las características del dispositivo y decidir si es importante o relevante para ellos. XML permite una compleja y poderosa descripción de dispositivos y capacidades de servicios a diferencia de una simple descripción de atributos en Jini.

3.2.2 Descripción de servicio UPnP

Luego de que un punto de control descubre un dispositivo, este aprende mas acerca de cómo usarlo, controlarlo, coordinarse con él por medio de la descripción del archivo XML. El control es expresado como una colección de objetos de Simple Object Access Protocol (SOAP) y sus URLs en el archivo XML. Para usar un control específico, un mensaje SOAP es enviado a un objeto de control SOAP a la URL especifica. El dispositivo o servicio retorna valores específicos de la acción.

Una descripción UPnP para un servicio incluye una lista de acciones a las que el servicio responderá y una lista de variables que muestran el estado del servicio en tiempo de ejecución. El servicios publica actualizaciones cuando estas variables cambian, un punto de control puede suscribirse para recibir esta información. Las actualizaciones son publicadas enviando mensajes de evento que contienen los nombres y valores de una o mas variables de

estado. Estos mensajes también son expresados en XML y formateados usando la Arquitectura General de Notificación de Eventos.

UPnP ofrece agregar una descripción de alto nivel de servicios en la forma de una interfaz de usuario. Esta característica permite que los usuarios controlen directamente el servicio. Si un dispositivo o servicio tiene una URL de presentación, entonces el punto de control puede recuperar una página desde esta URL, cargar la página en un navegador web y (dependiendo de las capacidades de la página) permite al usuario controlar el dispositivo o ver el estado de él.

3.2.3 Configuración automática de IP

Otra importante característica de UPnP es la configuración automática de direcciones IP. AutoIP permite a un dispositivo unirse a la red sin una administración explícita. Cuando un dispositivo se conecta a la red, intenta adquirir una dirección IP desde un servidor Dynamic Host Configuration Protocol (DHCP). Sin embargo, en la ausencia de un servidor DHCP, es reclamada automáticamente una dirección IP desde un rango de direcciones reservadas de la red local en uso. El dispositivo pide una dirección aleatoriamente elegida y hace una petición ARP para comprobar si alguien más reclama la dirección.

3.3 SALUTATION

El consorcio Salutation está desarrollando otro estándar homónimo, para el descubrimiento y uso de servicios, especialmente entre dispositivos de capacidades disímiles. La arquitectura Salutation provee un método estándar para aplicaciones, servicios, y dispositivos para que describan y anuncien sus capacidades a otras aplicaciones, servicios, y dispositivos. La arquitectura permite la búsqueda y descubrimiento basada en capacidades particulares.

La arquitectura está compuesta por dos grandes componentes: El administrador Salutation y administrador transporte. El administrador Salutation es el centro de la arquitectura y es similar a el servicio lookup de Jini o punto de control en UPnP. Este es mas bien definido como un corredor de servicios. Un proveedor de servicio registra su capacidad con un administrador Salutation. Cuando un cliente consulta su administrador Salutation local para realizar una búsqueda, todos los administradores Salutation se coordinan para realizar la búsqueda. Entonces, el cliente puede usar el servicio retornado. Un administrador Salutation se apoya en los administradores de transporte que proveen canales de comunicación fiables, independientemente de la capa de transporte de red inferior.

El administrador Salutation provee una independiente interfaz de transporte para servidores y aplicaciones clientes. Esta interfaz (SLM-API) incluye servicio de registro, servicio de descubrimiento, y servicio de acceso a funciones. La interfaz entre el administrador Salutation y administrador de transporte (llamada SLM-TI) logra un protocolo de comunicación independiente en la arquitectura Salutation. El administrador de transporte es una entidad, dependiente del transporte de red que lo soporta. Un administrador Salutation puede tener más de un administrador transporte, en caso de que este sea unido a múltiples redes físicas diferentes. Sin embargo, el administrador Salutation ve su transporte inferior a través de la SLM-TI y realiza las tareas siguientes.

3.3.1 Registro de servicio

El administrador Salutation contiene un registro para mantener información de los servicios, y un cliente puede registrarse o cancelar su registro. Todos los registros son efectuados con el administrador Salutation local o uno cercano conectado al cliente.

3.3.2 Descubrimiento de servicio

El administrador Salutation descubre otros administradores Salutation y registra servicios. Este descubre servicio remotos que satisfacen los tipos y conjunto de atributos

especificados por el administrador Salutation local. Esta única característica llamada capacidad de intercambio, es necesaria porque los servicios son básicamente registrados con el administrador Salutation local en el mismo equipo. Esta cooperación de los administradores Salutation forma un concepto similar al servicio lookup de Jini. Sin embargo, una diferencia es que este está distribuido a través de la red.

3.3.3 Disponibilidad del servicio

Una aplicación cliente puede solicitar al administrador Salutation periódicamente chequear la disponibilidad de servicios. Este procedimiento es efectuado entre el administrador local y el correspondiente administrador. Esta es una débil versión del servicio de eventos de Jini y UpnP.

3.3.4 Administración de sesión de servicio

La sesión de servicio es manejada en uno de tres modos: nativo, emulado, o salutation. El administrador Salutation puede no involucrase en el intercambio de mensajes en la sesión de servicio, dependiendo de la modalidad. El modo nativo intercambia a través de un protocolo nativo, el administrador Salutation nunca está involucrado en el intercambio de mensajes. En el modo emulado, el protocolo administrador Salutation transporta mensajes entre el cliente y servicio pero no inspecciona el contenido, y en el modo salutation, los administradores Salutation no sólo transportan mensajes sino también definen los formatos a ser usados en la sesión.

Una unidad funcional es un bloque básico construido en el arquitectura Salutation. Esta es la mínima función que constituye un cliente o servicio. Una colección de unidades funcionales definen un registro de servicio. Por ejemplo, las unidades funcionales [Imprimir], [Scanear], [Enviar fax] pueden definir un servicio fax. Cada unidad funcional está compuesta de un descriptivo registro atributo, especificado en ISO 8824 ASN.1.

3.3.5 Salutation-Lite

Salutation-Lite es una versión menor de la arquitectura Salutation dirigido a pequeños dispositivos. Este es obviamente más aplicable a pequeños aparatos de información como una PDA y computadores manuales. Salutation-lite también se presta bien para redes de bajo ancho de banda como IR y Bluetooth. El estándar es vigilado por el Consorcio Salutation, que provee cinco niveles de membresía.

3.4 SERVICE LOCATION PROTOCOL SLP

Service Location Protocol (SLP) es un estándar descentralizado, liviano y extensible para el descubrimiento de servicios de Internet Engineering Task Force (IETF). Este emplea URLs servicios, que definen el tipo de servicio y dirigen a un servicio particular. Por ejemplo "service:printer:lpr://hostname" es la URL para una línea de servicio de impresión disponible en hostname. Basado en el servicio URL, usuarios (o aplicaciones) pueden examinar servicios disponibles en su dominio, seleccionar y usar el que deseen.

Hay tres agentes en SLP: el usuario, servicio y directorio. El UA (agente usuario) es una entidad software que envía peticiones de descubrimiento de servicio en nombre de alguna aplicación del usuario. Como un repositorio de información centralizado, el DA (agente directorio) guarda anuncios de SAs (agentes servicios) y procesa consultas de descubrimiento de los UAs. Un SA se anuncia por sí mismo registrándose con un DA. El mensaje de registro contiene la URL para el anuncio del servicio y para la vida del servicio, además un conjunto de atributos para el servicio. El SA periódicamente renueva su registro con el DA, que almacena el registro y envía una confirmación de recepción a el SA. Un UA envía un mensaje de petición de servicio al DA para pedir la ubicación del servicio. El DA responde con un mensaje de respuesta que incluye las URLs de todos los servicios que coinciden con la petición del UA. Ahora, el UA puede acceder a uno de los servicios apuntados por la URL retornada. En SLP, el

DA es opcional. Un DA podría no existir en una red pequeña, en tal caso los mensajes de petición se servicio de los UAs son directamente enviados a los SAs.

En SLP se puede examinar servicios y encontrarlos basados en una búsqueda de palabras a través de sus atributos seleccionando el más apropiado entre todos los servicios disponibles en la red. SLP permite a los UAs usar operadores de consulta tal como AND, OR, comparadores, y coincidencias de sub-palabras. Este es más poderoso que el servicio de coincidencia de atributos de Jini y UpnP, que pueden buscar solo igualdades.

3.5 BLUETOOTH SDP

Al contrario de Jini, UpnP, Salutation, o SLP, Bluetooth SDP está especificado sólo para los dispositivos Bluetooth. Está dirigido principalmente al problema de descubrimiento de servicios. Este no provee acceso a servicios, anuncio de servicios, o registro de servicios y no tiene notificación de eventos cuando un servicio ya no está disponible. SDP soporta búsqueda por; clases, atributos y también permite examinar servicios. Este último es usado cuando un cliente Bluestooth no conoce los servicios disponibles en su vecindad. SDP está estructurado como un contorno Bluetooth y se ejecuta en un canal, orientado a la conexión, predeterminado de el L2CAP (Logical Link Layer). Salutation ha propuesto un puente entre su descubrimiento de servicios y Bluetooth SDP. Este puente es sinérgico porque complementa a Bluetooth agregando anuncio, administración y eventos. Bluetooth, devuelve la mano, sirviendo a Salutation como un transporte (Salutation es independiente del transporte) en el corazón de los dispositivos.

3.6 ACEPTACIÓN DEL MERCADO

No existen muchos productos Jini disponibles en el mercado hoy en día. En 1999, un año después de la introducción de Jini, compañias como Epson, Canon, Seagate, y Quantum acordaron insertar Jini en algunas de sus líneas de productos. Sin embargo después, en el mismo año, estas compañías advirtieron que esto podría tomar más de dos años para

completarse. La gente entonces predijo que los dispositivos Jini serían los primeros en el mercado, pero los planes para ellos son aún inciertos. Por otro lado, PsiNaptic un líder en computación ubicua, liberó un producto Jini muy importante, Jmatos. Jmatos soporta un semiconductor Dallas Tiny (Internet Interfaz) , que tiene una máquina virtual embebida. Tini es un microcontrolador con un rico conjunto de interfazs.

A diferencia de Jini, muchos productos UPnP están disponibles en el mercado hoy además de los cientos de miles de computadores con Windows XP (que vienen con soporte UPnP).

Algunos productos Salutation están disponibles, pero la mayoría son productos de automatización de oficina: maquinas de fax, impresoras, copiadoras y scaners. NuOffice de IBM un sistema de red para oficina basado en Lotus Notes, permite a los usuarios importar y exportar datos a cualquier dispositivo Salutation.

Apoyado por IETF y alineado con otros protocolos establecidos (incluidos Lightweight Directory Access Protocol, Domain Naming System, and DHCP), los desarrolladores han aceptado ampliamente SLP como un protocolo de descubrimiento de servicio simple y con mínimos requerimientos. Otra razón de su aceptación es su alcance, ya que este intenta sólo localizar (no acceder o entregar) el servicio. SLP es usado por la tecnología JetSend de Hewlett Packard que sostiene equipamiento de oficina HP y electrónica de consumo. Otros vendedores con impresoras y otros productos de red con SLP incluyen Axis, Lexmark, Xerox, Minolta, IBM y Novel. Además, para equipamiento de oficina y red, varias plataformas soportan SLP, incluidos Sun, Caldera, Novel y Apple.

El descubrimiento de servicios ha recorrido un largo camino para llegar a tener una mayor estandarización y desarrollo, pero el cuadro aún no está claro si se considera la aceptabilidad y disponibilidad del mercado. Además en su actual forma muchos de los estándares de descubrimiento de servicios no están dirigidos a las necesidades y requerimientos de los dispositivos móviles. Su uso potencial en aplicaciones móviles y ambientes inteligentes es incierto.

Los actuales protocolos de descubrimiento de servicios están diseñados para usarse en redes de área local (LANs y WLANs). El rango de IP multicast, por ejemplo, limita el descubrimiento en Jini. Esto es inadecuado para clientes móviles que requieren acceso a servicios desde una red de área ancha (WAN). Otro problema es la carencia de soporte para dispositivos móviles. Por ejemplo, Jini requiere JVM y capacidades RMI por parte del cliente, que ha impedido su masivo uso en dispositivos móviles. Una solución a este problema es introducir una arquitectura sustituto de Jini. Usando sustitutos, un dispositivo no debe tener o entender JVM o RMI. Este sólo debe recordar código Jini que usa RMI y estar habilitado para enviar código a un proxy (el sustituto) en la red local que actuará en su nombre. Desafortunadamente, sustitutos son más bien una solución para dispositivos estacionales que móviles.

Una limitación de los actuales marcos de trabajo en el descubrimiento de servicios es que ellos no consideran un importante contexto de información. Por ejemplo, ellos no soportan el servicio de ruteo y selección basado en la ubicación de los clientes. Otro inexplorado contexto de información incluye; distancia a los servicios, tiempo, carga del servicio, calidad de los servicios.

3.7 TABLA COMPARATIVA PROTOCOLOS

	Jini	**Salutation**	**SLP**	**UpnP**
Desarrollador	Sun Microsystems	ConsorcioSalutation	IETF	Microsoft
Licencia	Open, pero pagada para uso comercial	Open (sólo para miembros)	Open source	Open source
Versión	2.0	2.1	2.0	1.1
Anuncio	Si	no	Si	si
Registro	Si	si	Si	no
Seguridad	Basada en Java	Autentificación	Autentificación	IP dependiente
Transporte de red	Independiente (TCP/IP impl.)	independiente	TCP/IP (predominante)	TCP/IP
Lenguaje de programación	Java	Independiente	Independiente	Independiente
SO y plataforma	Java	Independiente	Dependiente	Dependiente
Movilidad de código	si (Java RMI)	no	No	no
Repositorio central	opcional	opcional	Opcional	no
Servicio Repositorio	Servicio Lookup	Conjunto de SLMs	DA	No
Eventos	Si	si	No	si
Protocolo de acceso	RMI	RPC	No	SOAP
Formato descripción	Objetos Java	Registros	Texto	XML
Entidades principales	Servicio Lookup, Cliente, Servicio	Administrador Salutation, Administrador Transporte, Cliente Servicio	DA, SA, UA	Punto Control, Dispositivo (servicio)
Servicio Anuncio	Protocolos descubrimiento/ suscripción	Registro con SLM local	Servicio de registro	Anuncio
Servicio Descubrimiento	Consultando al servicio lookup	Consultando al SLM local en cooperación con otros	Activo, Pasivo, Estático	Contactando al punto de control / escuchando los anuncios.
Acceso a servicio	Objeto de servicio proxy basado en RMI	Administrador de sesión de servicio.	Protocolo de servicio para descubrir servicio.	Invocando la acción al servicio (SOAP), consultando por variable estado.
Tiempo de vida del servicio registro.	Arrendamiento	no	Tiempo de vida en el registro.	Encabezado de Control en mensaje “alive”.
Servicio de grupo	Grupo	no	Alcance	no
Tipo de búsqueda	Tipo de interfaz y coincidencia de atributo.	Unidades funcionales y atrib. dentro de él.	Tipo de servicio y coincidencia de atributo	Descripción en XML

Capítulo 4. Elección de un Prototipo

4.1 Objetivos y Requisitos del prototipo

Para realizar un análisis objetivo de las propuestas de prototipo a desarrollar, se debe considerar la meta perseguida y limitaciones correspondientes.

Los objetivos Generales De la Habilitación Profesional son:

- Diseñar y construir un prototipo que permita publicar un Servicio particular.
- Diseñar y construir un cliente que permita acceder al anterior Servicio.

Y específicamente

- Realizar un diseño viable para el servidor y cliente propuesto.
- Utilizar la arquitectura y / o propuesta software elegida para construir el prototipo y evaluar los resultados.

Por otro lado, es necesario ajustarse a los límites enunciados en la presentación del Proyecto "El listado tentativo de productos a desarrollar estará limitado a recursos disponibles en nuestra facultad, a una problemática práctica y acotado al tiempo determinado para la Habilitación Profesional".

Derivado de lo anterior, se pueden enumerar requisitos mínimos que debería satisfacer el proyecto elegido:

1. *Cumplir con el rol de servicio*: Tal como se describió en el capítulo 1, debe cumplir con las tres máximas de servicio: ser de utilidad, tener destinatario y sentido dinámico. Es decir,

debe entregar una funcionalidad que satisfaga una necesidad de un grupo determinado siendo capaz de interactuar con su entorno (nº usuarios, peticiones, otros servicios, etc.).

2. *Utilizar los protocolos de descubrimiento de servicios*: El sentido dinámico del servicio debe solucionarse con la utilización de algún protocolo de descubrimiento de servicio, tema central del proyecto.

3. *Resolver una problemática práctica*: El enfoque que se ha pretendido dar a este trabajo es la utilización de los protocolos de descubrimiento a dispositivos (electrodomésticos, computadores móviles, periféricos, transductores en el hogar, etc.) por lo que la utilización de estos en los servicios web y agentes de software no sería una categoría válida a desarrollar.

4. *Alcanzable en el tiempo*: Las limitaciones de tiempo también son una variable a considerar a la hora de plantear y evaluar propuestas de prototipazo. Así la complejidad debe poder abordarse durante 4 a 5 meses.

5. *Alcanzable con los recursos disponibles*: Según se mencionó en el capítulo 2, un dispositivo que desee ser parte de una comunidad de servicios, con la capacidad de descubrimiento, debe cumplir requerimientos mínimos de procesamiento y memoria para sustentar el software necesario, sin embargo implementar esto de manera óptima requiere una mayor investigación y recursos. Así la propuesta para realizar este prototipo es delegar los requerimientos a un Pc conectado al dispositivo. Entonces la interfaz PC/dispositivo requerirá un mayor desarrollo y recursos a medida que la funcionalidad esperada aumente. Limitar esta funcionalidad a los recursos disponibles es una decisión que contribuirá al éxito de los objetivos.

6. *Entregar experiencia para futuros desarrollos*: Los servicios que se pueden entregar a través de dispositivos conectados a una red son muy variados y amplios. Así desarrollar un proyecto en el cual estén presente todos los componentes de un servicio será una preocupación. El prototipo deberá ser un primer paso para la creación de servicios de mayor complejidad y elaboración ya que entregará una guía práctica de las problemáticas que se enfrentaron y como se solucionaron.

4.2 Descripción de Propuestas

Las propuestas detalladas a continuación son un primer acercamiento ("Lluvia de ideas") a la definición de una idea para desarrollarla en profundidad. Quizás alguna de ellas no cumpla a cabalidad con alguno de los requisitos antes enumerados, pero un análisis posterior develará esto.

Proyector Distribuido

Por lo general, en una presentación (reunión, seminario, etc.) se utiliza un proyector de video para desplegar la información del computador. En el estado actual, esto demanda que el proyector debe ser conectado al computador de cada presentador, lo que presenta diversas desventajas: desplazamientos, tiempo de espera, imposibilidad de poder compartir ("switchear") el proyector entre dos o más participantes, etc.

Se propone diseñar y construir un prototipo que permita publicar el proyector como un servicio. Diseñar y construir un cliente que permita a uno o más presentadores, conectados en red, utilizar el servicio de proyección.

Servicio de almacenaje

Hace un tiempo se realizó un cambio de equipamiento del laboratorio principal de computación a máquinas SUN, el proceso de adaptación no fue fácil y una de las mayores molestias enunciadas por los usuarios era la necesidad de utilizar los computadores del laboratorio adjunto para almacenar o sacar archivos desde su cuenta, ya el equipamiento SUN no dispone de disqueteras.

Una posible solución es conectar una disquetera a la red y publicarla como un servicio. Así quien la necesite puede solicitar el servicio y utilizarla, es más se podría pensar en varias de ellas para responder a la demanda.

Se propone diseñar y construir un prototipo que permita publicar y utilizar una disquetera en el laboratorio de máquinas SUN.

Servicio de impresión

Conectar una impresora en red y permitir a un grupo de usuarios su uso requiere de: configuración de la impresora, configuración de cada equipo. Y si esta es retirada del equipo conectado a la red todos los usuarios del grupo no se enterarán del suceso, produciendo problemas al tratar de imprimir. De aquí la necesidad de adaptar la impresora para que se publique como un servicio que sea capaz de dar aviso de su disponibilidad y también de ausencia en la red, dando así dinamismo al cambio de lugar y utilización en cualquier punto de la red.

Se propone diseñar y construir un prototipo que permita publicar y utilizar una impresora como un servicio.

I-radio

En entornos de trabajo donde la herramienta principal es el computador, la utilización de una radio para distender el ambiente es muy común. Sin embargo cada vez que algún usuario desea regular alguna función de esta, debe acercarse o solicitar el control remoto. Buena solución para esto resultaría habilitar tantos controles remotos como usuarios hayan. Una propuesta de solución es diseñar y construir un prototipo que permita publicar y utilizar una radio conectada a la red, dando la posibilidad a los usuario de regular algunas variables (volumen, banda, sintonía, etc.) sin moverse de su estación de trabajo. Incluso podría garantizarse un uso más igualitario en la elección de la música ya que el servicio podría "arrendarse" por un determinado tiempo a cada usuario.

Fax en línea

Una máquina de fax tradicional, permite a un usuario enviar un documento en papel a otra máquina de similares características. Necesariamente el usuario debe buscar una o solicitar a su secretaria enviar el documento. Así normalmente este aparato se mantiene en un lugar fijo, impidiendo su traslado al lugar de mayor utilización por un determinado periodo, ya que los

usuarios no se enterarían del cambio. Implementar la máquina de fax como un servicio, que fuese capaz de avisar a todos los usuarios su disponibilidad, no importando el lugar físico de su ubicación sería apropiado. Además cada uno desde su propio Pc podría enviar el documento, sin la necesidad de un intermediario. Esta independencia del lugar físico permitiría cambiar la máquina de fax de lugar, según la demanda lo amerite, sin desmedro del servicio para los usuarios. Así se propone diseñar y construir un prototipo que permita publicar y utilizar una máquina de fax..

Calentador Eléctrico

En muchas oficinas de todo tipo, la utilización de un calentador eléctrico de agua, para tomar café, es utilizado. Sin embargo al existir varios de ellos en un sector físico pequeño se malgastan algunos recursos, ya que nadie consulta a una oficina cercana si acaba de hervir agua o si está apunto de hervir, simplemente activa el propio generando un doble gasto. Además desplazarse para encenderlo y luego desplazarse para buscar el agua podría llegar a ser una molestia si no está cercano al hervidor. Una solución sería publicar un hervidor capaz de indicar su estado (encendido, apagado, etc.) y permitir cambiarlo. Así todos los usuarios de la comunidad podrían aprovechar la iniciativa de alguno de ellos y utilizar el agua cuando el sistema indique que esté lista. Se propone diseñar y construir un prototipo que permita publicar y utilizar una calentador eléctrico, dando la posibilidad de activarlo cuando un usuario desee y avisando cuando el agua esté lista.

Servicio de Alto parlante

La instalación de una red de alto parlantes en la facultad para diversos usos: entrega de avisos (comienzo de charlas, suspensión de talleres, noticias del departamento, indicaciones de emergencia, etc) y quizás música en momentos apropiados, requeriría de un cableado y un lugar central de transmisión, además se complicaría aún más si deseáramos poder seleccionar cuales altoparlantes habilitar para determinado aviso. Una solución sería publicar un altoparlante como un servicio en que los usuarios puedan ejecutar archivos de sonido. La gran ventaja de publicarlo como un servicio de red, es su movilidad, ya que podría cambiarse de laboratorio buscando un nuevo punto de red según la necesidad del aviso, sin alterar

configuraciones en los clientes. La propuesta es diseñar y construir un prototipo que permita publicar y utilizar un altoparlante que pueda conectarse en cualquier punto de la red y transmitir un archivo de sonido (aviso).

Simulación de control domestico

Los sistemas de seguridad en el hogar se componen normalmente de un conjunto de sensores conectados a un punto central que los supervisa y dependiendo de cada tipo emite alguna acción al momento de encontrar alguna anomalía (sonido intenso, aviso a una central de seguridad, etc.). Fusionar esta idea con la de los servicios podría resultar interesante, ya que cada uno de los tipos de sensores sería un servicio a publicar, así a medida que se necesite podrían agregarse más de ellos sin necesidad de configuración, ya que el sistema los entenderá como nuevos servicios. Es decir, se logra un sistema de seguridad ampliable, auto configurable y adaptable a las necesidades de cada entorno. Se propone diseñar y construir un prototipo que permita publicar y utilizar algún sistema de seguridad para una casa con sensores de varios tipos.

Pantalla de avisos

La idea nace de tener un módulo visual que entregue avisos y que tenga la suficiente independencia para cambiar de lugar sin mayores inconvenientes. Este podría ser utilizado para: Anunciar suspensión de certámenes, promocionar algún evento, indicar horarios de revisión de certámenes, y cualquier otro tipo de avisos. Todos los usuarios autorizados pueden conocer la disponibilidad de la pantalla al momento de conectarse a un punto de red y publicar en ella algún texto. En definitiva se propone diseñar y construir un prototipo que permita publicar y utilizar un display de leds (parecido al que traen algunas pesas digitales) para mostrar avisos (suspensión certámenes, certámenes, noticias importantes, etc.) publicados por profesores o secretarias, con la capacidad de cambiarlo a cualquier punto de la red.

4.3 Ventajas y Desventajas de cada Propuesta.

En esta sección se intenta expresar las mayores ventajas y desventajas de cada propuesta para tener una visión más clara de cuales pueden ser o no realizables y convenientes para los objetivos planteados.

Proyector Distribuido

Ventajas	**Desventajas**
Producto muy interesante.	Dificultad para independizar el servicio de un PC.
Aplicable a distintos entornos (académico, empresarial, etc.)	La tasa de transferencia de la red influye en la tasa de actualización de la imagen.
La movilidad de los clientes justifica perfectamente el uso del descubrimiento de servicio.	Proceso de captura de la imagen.

Servicio de Almacenaje

Ventajas	**Desventajas**
Resuelve una necesidad de la Facultad.	Quizás es necesario transferir los archivos desde el cliente.
Conectado a un PC el control de la disquetera lo realiza el S.O.	Si la demanda por su uso es demasiada, deberá generarse una cola de espera.
Se pueden agregar disqueteras para ampliar el servicio.	Necesidad de capturar el servicio y luego colocar el disquete.

Servicio de impresión

Ventajas	**Desventajas**
Aplicable a muchos entornos (empresarial, domestico, académico, etc.)	En el caso de no usar drivers, será necesario conocer muy bien el funcionamiento de la impresora.
La experiencia podría ser replicable a varios modelos de impresora realizando los ajustes necesarios.	Este tipo de servicio, ya está bastante desarrollado por algunos fabricantes de impresoras.
Existen experiencias de desarrollo anteriores.	Debido a la gran cantidad de impresoras USB, quizás sea necesario trabajar con él.

I-radio

Ventajas	**Desventajas**
Llamativo.	No entrega gran utilidad.
El servicio tiene varios eventos y acciones.	Requiere una interfaz con el PC.
Adaptable a distintos modelos de radio digital.	La posibilidad para todos de controlar la radio, podría convertirse en un caos.

Fax en línea

Ventajas	**Desventajas**
Utilidad clara.	Semejante a la utilidad entregada por un servidor de fax.
Muchos más usuarios tendrían acceso directo al fax.	El requisito de enviar un archivo a diferencia de un papel, puede convertirse en un gran obstáculo.
Podría usarse como punto de partida un servicio de impresión.	Construcción y diseño de interfaz con el PC.

Calentador Eléctrico

Ventajas	**Desventajas**
Requiere de todos los conceptos de un protocolo de descubrimiento (eventos, acciones, descubrimiento, etc.)	Necesidad de modificar el calentador para que refleje los estados y quizás la temperatura.
Resultado práctico y llamativo.	Construcción y diseño de interfaz con el PC.
Podría ampliarse los estados (temperatura del agua) si el tiempo lo permite.	Puede ser riesgoso si el calentador no tiene agua.

Servicio de alto parlante

Ventajas	**Desventajas**
Resultado práctico y llamativo.	Necesidad de conversión digital / análogo en el servicio.
Ampliable a transmisión en tiempo real.	Conectado a un Pc pierde su independencia.
Adaptable a distintas aplicaciones (publicidad, noticiosas, etc.)	Construcción y diseño de interfaz con el PC.

Simulación de control doméstico

Ventajas	**Desventajas**
De interesante proyección.	Se necesitan recursos para adquirir sensores y actuadores para simular la seguridad de la casa.
La seguridad de la casa puede ser mejorada por etapas, adquiriendo nuevos sensores y "enchufándolos" al sistema.	Se necesita un Pc por cada tipo de sensor, ya que no se dispone de microcontroladores.
Cada elemento de seguridad es independiente, todo el sistema está destruido sólo cuando todos los sensores han sido manipulados.	Construcción y diseño de interfaz con el PC.

Pantalla de avisos

Ventajas	**Desventajas**
Llamativo.	Construcción o compra de la pantalla (leds, matricial o cualquier tipo de pantalla).
Útil y adaptable a distintas aplicaciones (publicidad, noticioso, señalización, etc)	Es necesario programar la conversión de texto a letras de la pantalla.
La independencia que daría el protocolo de descubrimiento, permitiría su fácil cambio de lugar físico.	Construcción y diseño de interfaz con el PC.

4.4 Decisión final y Resultados esperados.

Luego de un análisis en conjunto con los profesores guías, se decide realizar durante la Habilitación Profesional el prototipo del "Proyector de video Distribuido" bautizado como:

Esencialmente motivaron a esta decisión las siguientes razones:

- Características del servicio justifican plenamente la utilización de un protocolo de descubrimiento de servicio.
- Desafió que implica la dificultad del proyecto en sí.
- Utilidad en la facultad y otros entornos.
- Es la propuesta que originalmente generó la investigación del Proyecto.

Aún cuando los objetivos específicos del proyecto han sido ya enunciados, es conveniente plantear los resultados esperados del prototipo que se construirá.

El prototipo inicial de un cliente capturará el contenido de la pantalla y lo enviará al servicio de proyección (ejemplo una imagen jpg). Además, el contenido deberá ser recapturado y reenviado periódicamente al servicio de proyección. El servicio, por su lado, proyectará el contenido (jpg) que recibe.

De lo anterior se puede enlistar algunas de las característica que se esperan del prototipo :

Proyección:

El servicio deberá ser capaz de proyectar una imagen entregada por algún cliente, con las restricciones que el servicio indique con anterioridad (resolución, tamaño, etc.).

Tasa de actualización:

Se espera que la tasa de actualización de la imagen proyectada, sea suficiente para reflejar de buena forma los tiempos de cambio promedio entre diapositiva de un expositor común. Estará limitado a la capacidad de la red y a los computadores utilizados.

Múltiple acceso al servicio:

Los usuarios deberán poder compartir el servicio. Es decir, el servicio de proyección deberá poder atender a múltiples clientes de manera simultánea o acordada por ellos mismos.

Características propias del descubrimiento:

El prototipo tendrá como base las tres características que entregan los protocolos de descubrimiento de servicio: anuncio, descubrimiento y suscripción.

Capítulo 5. Desarrollo del Prototipo

5.1 Elección del protocolo

Como ya se ha descrito en capítulos anteriores, las coincidencias entre las alternativas de descubrimiento de servicios son mayores a las diferencias. Es mas la opción respecto a cual será universalmente utilizada, seguramente pasará por una decisión estratégica y comercial.

Así decidir que plataforma utilizar para Vistazo, fue difícil. El esfuerzo para utilizar cualquiera de ellas era similar, ya que no existía experiencia previa de desarrollo. Sin embargo la propuesta inicial del servicio de proyección, por parte del profesor Inostroza implicaba el uso de Jini.

Considerar algunas características específicas de Vistazo contribuyó también para tomar la decisión final:

1. *Manejo de imágenes:* Se esperaba poder manipular imágenes de fácil manera, con el lenguaje de programación que utilizara la tecnología elegida

2. *Interfaz de Usuario:* Era importante considerar la posibilidad de programar una interfaz de usuario apropiada para el cliente del servicio.

3. *Múltiples Sistemas Operativos:* Si el proyector se publica como un servicio, no es posible asegurar que todos los clientes utilizarán el mismo Sistema Operativo. Considerar esta posibilidad es importante.

Así Jini se impuso como la elección más viable, ya que al funcionar sobre Java se obtiene una API muy variada para el manejo de imágenes, construcción de interfazs y muchas otras. Las limitaciones para ejecución en variados ambientes son aminoradas, ya que cualquier arquitectura que ejecute una JVM (java virtual machine) habilitaría el uso de Vistazo.

Además, características propias de Jini contribuyeron a aceptarla definitivamente como la tecnología con la cual se desarrollaría Vistazo.

1. *Movilidad de código:* La posibilidad de transferir objetos desde una máquina virtual a otra, serializándolos, entrega una libertad de programación muy valorada.

2. *Uso RMI:* La característica anterior se logra por el uso de RMI, más aún, Jini implementa su propia propuesta llamada JERI, que mantiene todas las ventajas de RMI pero eliminando la complejidad asociada.

3. *Multi-plataforma:* La portabilidad no requiere un esfuerzo adicional de programación, ya que al ser Java un lenguaje semi-interpretado logra un código universalmente ejecutable por cualquier JVM, no importando la arquitectura o sistema operativo utilizado.

4. *Documentación:* Siguiendo la propuesta de Java, Jini entrega una completa documentación de su API. Además la posibilidad de obtener en Internet un libro guía del profesor Jan Newmarch convierte a Jini en una tecnología bien documentada, aunque en inglés.

Así considerando esencialmente; preferencia inicial para el proyecto, características específicas esperadas para Vistazo y ventajas de la utilización de Jini, la decisión fue tomada. Jini sería la tecnología utilizada.

5.2 Plataforma de Desarrollo

Computador:

Modelo : IBM 300PL

Procesador : Pentium 3 550 Mhz.

Memoria Ram : 128 MB.

Disco Duro : 13 GB.

Tarjeta video : S3 virge integrada.

Monitor : Compaq V55.

Sistema Operativo:

Kernel : Linux 2.4.18-bf2.4

Distribución : Debian Unstable.

Entorno gráfico: Gnome 2.4.

Software Desarrollo:

Java : JDK 1.4.2-02 para linux.

Jini : versión 2.0.001.

Proteus Technologie (única versión disponible).

Entorno Integrado de Desarrollo: Anjuta 1.2.0.

5.3 Etapas del Prototipo.

5.3.1 Aprendizaje

Antes de comenzar la programación, fue necesario entender el funcionamiento y utilización de la API de Jini entregada por Sun Microsystems. Para este fin se utilizó esencialmente cuatro fuentes de información:

Documentación oficial de la API de Jini: Mantiene la misma rigurosidad y formato que la documentación de Java, pero muy poco explicativa para entender como funciona, más bien se utilizó como una referencia de programación.

Jan Newmarch's Guide to Jini Technologies: Libro escrito por el Sr. Newmarch, profesor asociado de School of Network Computing y Monash University. Gran guía de aprendizaje y referencia que abarca toda la complejidad de Jini. La disponibilidad en Internet de la versión 3.0 actualizada a Jini 2.0 y sus archivos de ejemplos hicieron de esta, la fuente primaria de aprendizaje.

Especificación de la arquitectura Jini: Traducción no oficial de "The Jini Architecture Specification" realizada en el Proyecto de Título y entregada como Anexo. Este documento es una descripción general del funcionamiento de la Tecnología Jini.

Internet: Información y ayuda fue encontrada aquí, esencialmente a través de listas de correo como jini-users@sun.com , javaSpain@yahoogroups.com y múltiples sitios Web.

A medida que el estudio de estas fuentes entregaba las bases suficientes, se programaban pequeños ejemplos que incrementaban su complejidad y acercaban al prototipo propuesto.

5.3.2 Ejemplos

Una vez entendido que el proceso de búsqueda de un servicio Lookup, similar a un directorio de servicios, es el mismo tanto para cliente como servidor, se programa un pequeño ejemplo (ver anexo Ejemplo Unicast Registrar) que a través de una llamada unicast se

comunique con él y solicite el objeto llamado “registrar”. El objeto “registrar” es usado de diferentes formas por clientes y servicios: los servicios usan este para registrarse y los clientes para localizar estos servicios. A continuación una figura que grafica la secuencia del programa.

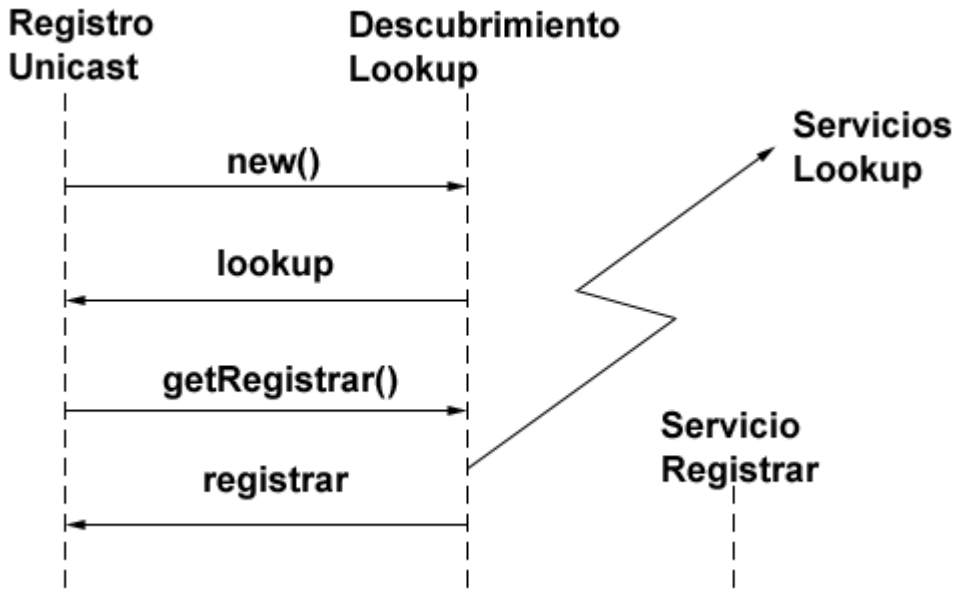


Figura Nº 5.1 Diagrama de secuencia UML para lookup.

Este programa supone el conocimiento, por parte del cliente, de la dirección IP del lookup, sin embargo Jini esta habilitado para realizar una llamada Multicast UDP en la red local y esperar por respuestas de los lookup habilitados con la ayuda de un evento llamado “discovery”. Así sin conocer de ante mano su localización puede también comunicarse y solicitar el objeto registrar. También fue programado un ejemplo de este tipo (ver anexo Ejemplo Multicast Registrar). A continuación se grafica el proceso multicast.

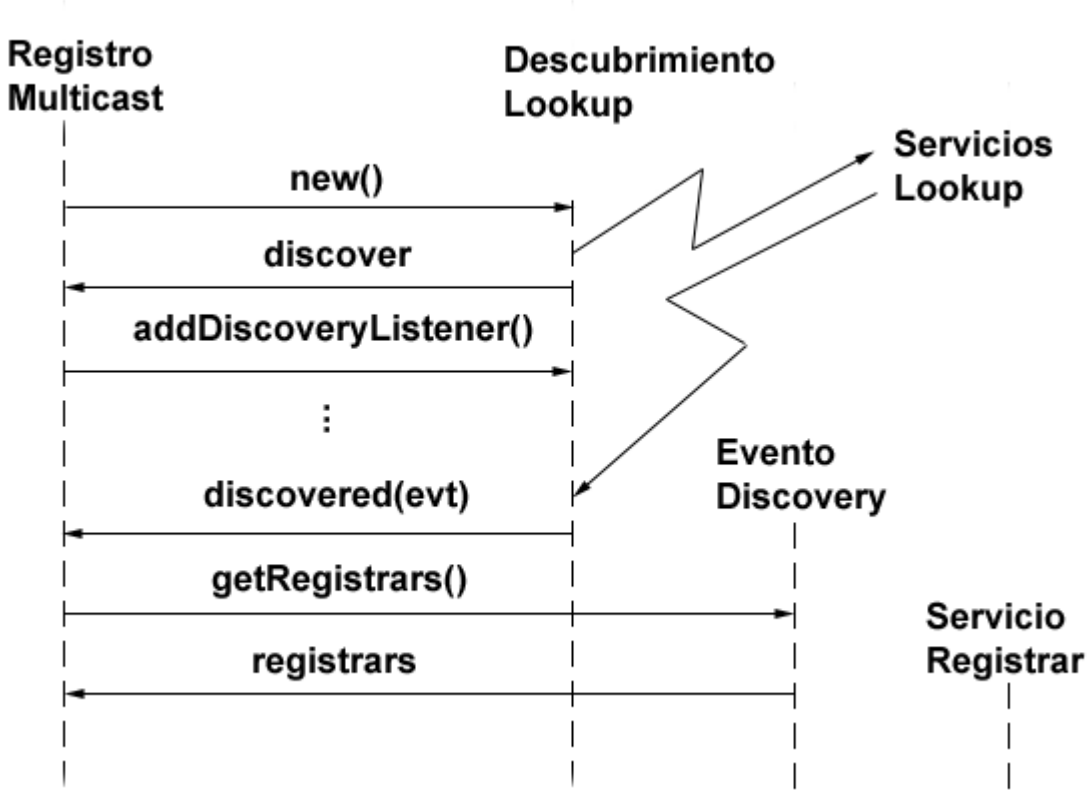


Figura Nº 5.2 Diagrama de secuencia UML para discovery.

5.3.2.1 Proveedor del servicio.

Un proveedor de un servicio busca un servicio lookup usando unicast a través de un objeto llamado LookupLocator o bien por medio del objeto LookupDiscovery que realiza una llamada multicast. En ambos casos, un objeto ServiceRegistrar es retornado para actuar como un Proxy para el servicio lookup. El proveedor registra entonces el servicio con el servicio lookup usando el objeto ServiceRegistrar y su método register( ) :

```
package net.jini.core.lookup;
public Class ServiceRegistrar {
   public ServiceRegistration register(ServiceItem item, long leaseDuration)
                    throws java.rmi.RemoteException;
}
```

El segundo parámetro es el tiempo, en milisegundos, que el lookup mantendrá el servicio registrado. Este tiempo no es obligatorio para el lookup, él puede rechazarlo o bien determinar otro que estime conveniente. El primer parámetro es del tipo:

```
package net.jini.core.lookup;
public Class ServiceItem {
   public ServiceID serviceID;
   public java.lang.Object service;
   public Entry[] attributeSets;

   public ServiceItem(ServiceID serviceID,
            java.lang.Object service,
            Entry[] attrSets);
}
```

El proveedor del servicio creará un objeto ServiceItem usando el constructor y pasará este a register( ). El parámetro ServiceID es valorado nulo cuando es registrado por primera vez. El servicio lookup configurará un valor no nulo cuando este registre el servicio. El segundo parámetro es el objeto servicio que se desea registrar. Este objeto debe ser serializado y enviado al lookup. Así cuando un cliente solicite el servicio, este objeto será entregado. El tercer parámetro es un conjunto de entidades que dan información adicional acerca del servicio. Sino existe esta información, debe ser declarado como nulo.

El servicio se registra llamando al método register( ). El valor retornado es del tipo ServiceRegistration, este objeto es creado por el servicio lookup y es retornado para ejecutar en el proveedor del servicio. El objeto registration actúa como un Proxy para controlar el estado del objeto servicio exportado y almacenado en el lookup. Estos objetos son mostrados en la siguiente figura.

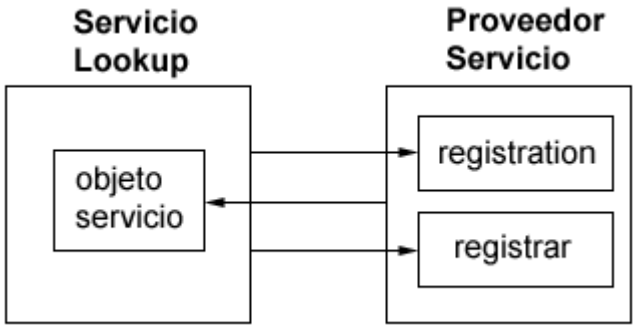


Figura Nº 5.3 Objetos en el proceso de registro.

La comprensión del proceso de exportación del objeto servicio, permitió plantear programar un ejemplo de servidor unicast que exportara este mismo servicio, que funcionalmente no realiza ninguna labor (ver anexo Ejemplo Servicio Simple).

5.3.2.2 Cliente del servicio.

Un cliente obtiene un objeto ServiceRegistrar del servicio lookup y usa este para buscar un servicio almacenado en lookup usando el método lookup( ):

```
public Class ServiceRegistrar {
   public java.lang.Object lookup(ServiceTemplate tmpl) throws
                java.rmi.RemoteException;
   public ServiceMatches lookup(ServiceTemplate tmpl,int maxMatches)  throws
                        java.rmi.RemoteException;
}
```

El primero de estos métodos busca solo un servicio que coincida con la petición. El segundo busca un conjunto (hasta maxMatches) de servicios coincidentes.

El método lookup usa una clase de tipo ServiceTemplate para especificar el servicio buscado:

```
package net.jini.core.lookup;

public Class ServiceTemplate {
   public ServiceID serviceID;
   public java.lang.Class[] serviceTypes;
```

```
   public Entry[] attributeSetTemplates;

   ServiceTemplate(ServiceID serviceID,
                   java.lang.Class[] serviceTypes,
                   Entry[] attrSetTemplates);
}
```

A pesar que cada servicio debe tener asignado un serviceID por un servicio lookup, un cliente podría no conocerlo, en este caso es establecido nulo. El attributeSetTemplates es un conjunto de elementos Entry para realizar la comparación de atributos. El parámetro más importante es una lista de serviceTypes. Los servicios exportan instancias de una clase. ¿Como hace el cliente para obtener una apropiada instancia desde un servicio lookup?. Aunque el servicio lookup mantiene instancias de objeto para el servicio, el cliente sólo conocerá acerca de un servicio por esta especificación (a menos que conozca el ID del servicio). La especificación es ciertamente una interfaz Java, así el cliente necesita preguntar usando esta interfaz. Una interfaz puede tener un objeto clase tal como las clases comunes, así la lista de servicesType es comúnmente una lista de objetos para la interfaz del servicio. El cliente usualmente solicita un objeto interfaz. Para ejemplificar, un servicio que solo devuelva “Hola Mundo” sería definido por una interfaz como esta:

```
public interfaz MyServerInterfaz {
   public String sayHello() throws java.rmi.RemoteException;
}
```

5.3.3 Servicio “Hola Mundo”.

Así bajo la certeza de poder localizar un servicio lookup y obtener su objeto registrar. Y utilizar este de distintas maneras por el proveedor y el cliente, se avocó a la tarea de programar un proveedor para registrar un objeto servicio. Este objeto, en un principio, sólo devolvería un mensaje “Hola Mundo”. La interfaz que implementa el servicio es la antes mostrada, así el objeto servicio a exportar sólo responde con un mensaje:

```
public class MyServerInterfazImpl implements MyServerInterfaz, java.io.Serializable {
        public String sayHello(){
                return ("Hola mundo desde Myserver!!");
        }
        public MyServerInterfazImpl() {  } }
```

La clase que representa al proveedor tiene además la capacidad de almacenar en un archivo el ID de servicio (ver anexo MyHelloServer). Se programó dos versiones de Cliente; unicast, multicast (ver anexos ClienteHelloUnicast y ClienteHelloMulticast).

De esta forma los clientes sólo necesitan la interfaz para realizar la búsqueda del servicio en el lookup que encontraron por método unicast o multicast, según corresponda. Este fue el primer prototipo de servicio que se programó con Jini, aunque aún muy lejano al prototipo esperado.

5.3.4 Servicio "Hola Mundo" con Jeri.

La implementación anterior es solo una de las múltiples opciones de arquitectura de un servicio. A continuación se explicarán algunas de ellas y se mejorará la programada.

En definitiva, un cliente busca una implementación de una interfaz, la cual puede ser hecha de diferentes maneras.

5.3.4.1 Proxy es el servicio.

Un alternativa es que el Proxy realice toda la tarea del servicio no necesitando algún procesamiento por parte del servidor, es llamado un Proxy ¨pesado¨. El rol del servidor es registrar el Proxy con los servicios lookup y mantenerlo vivo (renovando el arrendamiento). El servicio por sí mismo se ejecuta completamente en el lado del cliente. Un diagrama de clase para el servicio "Hola Mundo" usando este método (implementación anterior) es mostrado a continuación:

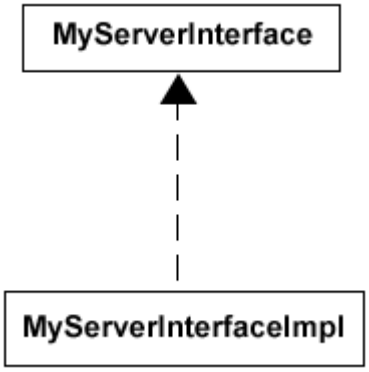


Figura Nº 5.4 Diagrama de clase para “Hola Mundo”

Si sólo se aprecia estas clases, la transferencia desde Maquinas Virtuales Java (JVMs) es de la siguiente manera:

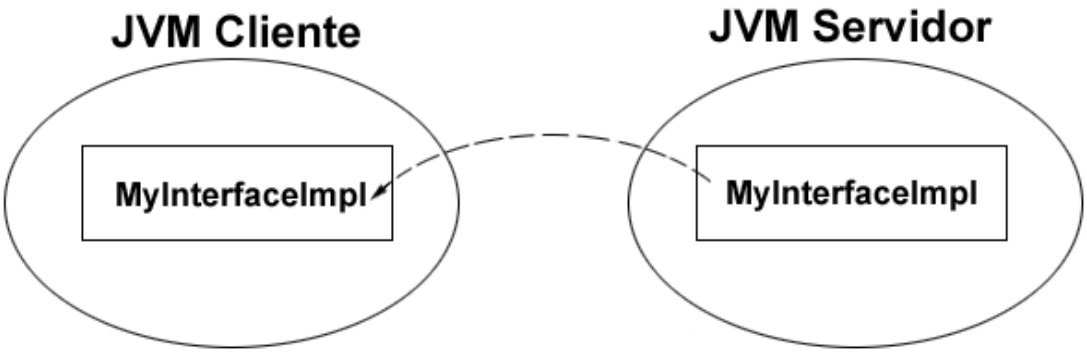


Figura Nº 5.5 Objetos en las JVMs.

El cliente pregunta por un MyServerInterfaz, que fue entregado al servicio lookup por el servidor, así lo que el cliente obtiene es un MySerInterfazImpl. Esta, se ejecuta completamente en el cliente, sin comunicarse con el servidor. Esta opción es útil cuando el servicio es puramente software y necesita ningún tipo de comunicación con el servidor. Algo así como un calendario o una agenda que usa archivos locales en el cliente y no en el servidor.

5.3.4.2 Proxy RMI

El extremo opuesto a la propuesta anterior es donde todo el procesamiento es realizado en el lado del servidor. El Proxy solo existe en el cliente para tomar las llamadas e invocar a métodos del servicio alojado en el servidor, retornando el resultado al cliente. RMI hace esto de forma transparente. La figura muestra una implementación de "Hola mundo" usando este mecanismo.

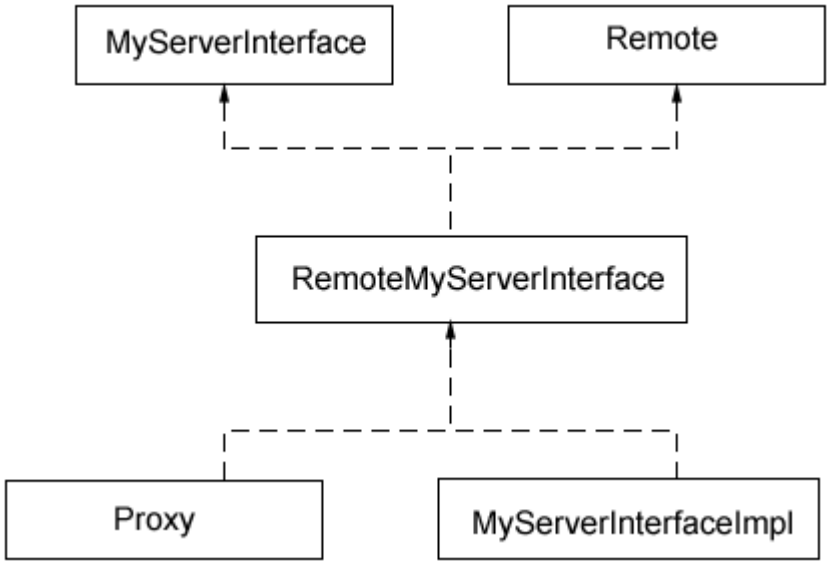


Figura Nº 5.6 Diagrama de Clase para Proxy RMI.

Los objetos en las JVMs son los siguientes:

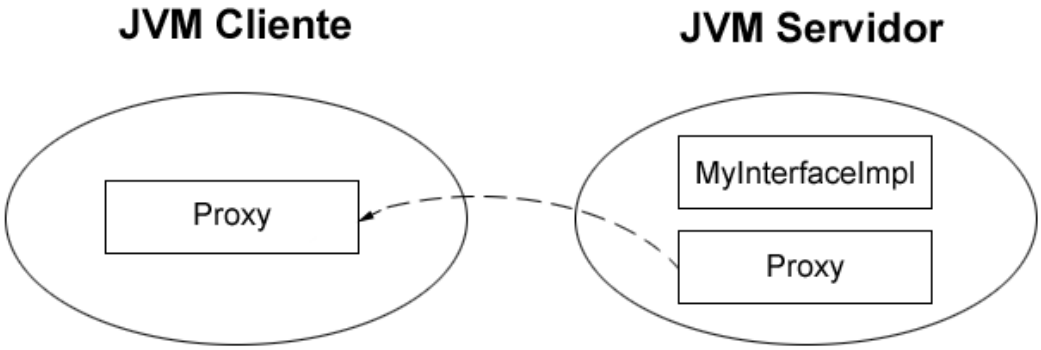


Figura Nº 5.7 Objetos en las JVMs para Proxy RMI.

La estructura de clases es un poco más compleja para lograr funcionar con RMI. La interfaz RemoteMyServerInterfaz es definida por conveniencia (MyServerInterfazImpl puede implementar MySeverInterfaz y Remote directamente).

Jini es una capa de abstracción para procesamiento distribuido. Como tal, confía en un número de mecanismos para procesamiento distribuido. Uno de estos es Invocación de Métodos Remotos (RMI). En RMI, un objeto permanece en el servidor y un Proxy "liviano" es enviado al cliente. El cliente hace llamadas a métodos en este Proxy. El Proxy transmite estas llamadas a través de la red al servidor, quien llama al objeto servicio original. El resultado de la llamada a este método es enviado de vuelta al Proxy, el cual retorna el resultado al cliente. Esta es una versión orientada a objetos de las llamadas a procedimientos remotos.

La implementación original de RMI usaba JRMP (Java Remote Method Protocol). Este es un protocolo particular construido directamente sobre TCP. Desde entonces muchas otras implementaciones de RMI han surgido, como RMI sobre HTTP (protocolo de transporte Web), RMI sobre IIOP (protocolo de transporte Corba) y RMI sobre SSL (capa socket segura).

Todas estas implementaciones tienen su propia interfaz de programación, con clases especializadas. Esto, sin embargo, debería ser abstracto: un mecanismo con archivos de configuración para seleccionar el protocolo de implementación a usar. Por ejemplo, un archivo de configuración puede especificar usar RMI sobre TCP o RMI sobre IIOP. Entonces las

aplicaciones pueden ser escritas independientes de la implementación, eligiendo el protocolo en tiempo de ejecución a través de la información de configuración. Esto es precisamente lo que hace Jini al desarrollar su nueva implementación llamada Jini ERI. Jeri resuelve esta y muchas otras problemáticas haciendo la programación más simple y ordenada.

Implementar la interfaz remota permite al Proxy generado, realizar llamadas a métodos del objeto remoto MyServerInterfazImpl.

Esta estructura de Proxy RMI es usada cuando el servicio no necesita realizar procesamiento en el lado del cliente, pero sí en el servidor. Por ejemplo, una agenda que almacena toda la información en el servidor. También cuando los servicios están estrechamente ligados a una pieza de hardware del servidor u otros casos.

Existen otras dos tipos de implementaciones de servicios: Proxy No-RMI, RMI y Proxy No-RMI. El primero es la utilización de un Proxy pero sin el mecanismo de RMI para comunicación con el servidor, sino cualquier protocolo que convengan ambos (cliente y servicio). La segunda opción es una mezcla de las dos anteriores.

Considerando las características del proyecto Vistazo es evidente que el servicio realiza procesamiento en el servidor, ya que la imagen debe ser proyectada en él. Así se decidió realizar una variante del servicio “Hola Mundo” implementando un Proxy Jeri. Fué necesario entonces, construir una nueva clase remota (RemoteMyServerInterfaz) que extienda MyServerInterfaz y la clase Remote. Reemplazando así al objeto servicio en el lookup y encargándose de la comunicación con él desde el cliente:

```
import java.rmi.Remote;
public interfaz RemoteMyServerInterfaz extends MyServerInterfaz, Remote {}
```

El proveedor del servicio, se encarga de exportar este Proxy con la ayuda del objeto Exporter (ver anexo MyServerRMI). En los clientes no es necesario realizar algún tipo de cambios, para ellos el Proxy se comporta de la misma forma como si fuese un objeto local. Sin embargo la interfaz debe implementar este nuevo Proxy:

```
public class MyServerInterfazImpl implements RemoteMyServerInterfaz {
        public String sayHello(){
                return ("Hola mundo desde Myserver!!");
        }
public MyServerInterfazImpl() throws java.rmi.RemoteException {    }
}
```

Esta implementación aseguró la viabilidad de ejecutar alguna acción desde el cliente en el servidor, específicamente para Vistazo solicitar desplegar una imagen en pantalla.

Una modificación al objeto servicio, creando un objeto Runtime que permite ejecutar un comando de sistema desde Java permitió ejecutar la utilidad QIV (Quick Image Viewer) y desplegar una imagen almacenada en el servidor a la orden de un cliente:

```
public class MyServerInterfazImpl implements RemoteMyServerInterfaz {
        public String sayHello(){
                 Runtime runtime = Runtime.getRuntime();
                try {runtime.exec("qiv -f -i /home/jorge/trivia/Pantallazo.gif");}
                        catch (Exception e) {}
                return ("Viewer ejecutado?");
        }
        public MyServerInterfazImpl() throws java.rmi.RemoteException {   }
}
```

5.3.5 Envío y recepción de archivo.

Siendo el servicio capaz de desplegar una imagen almacenada en el servidor que lo aloja, por orden del cliente que utiliza un Proxy para comunicarse con él, el paso lógico fue intentar transferir un archivo de imagen desde el cliente al objeto servicio, para que este se desplegara en el servidor. La primera alternativa fue almacenar la imagen en un objeto BufferImage y pasar este como parámetro en un método del servicio, luego de programarlo y realizar algunas pruebas se clarificó la imposibilidad de usarlo con Jeri, ya que no es un objeto serializable, atributo primordial para ser utilizado en una llamada remota. La implementación fallida de este servicio es la siguiente:

```
public class MyServerInterfazImpl implements RemoteMyServerInterfaz {
        public String sayHello(BufferedImage input){
                 Runtime runtime = Runtime.getRuntime();
                try {
```

```
                File outputFile = new File("screen.png");
                ImageIO.write(input, "PNG", outputFile);
            }catch (Exception e) {System.out.print(e.getMessage());}
            try { runtime.exec("qiv -f -i screen.png");} catch (Exception e) {}
            return ("Viewer ejecutado?");
        }
public MyServerInterfazImpl() throws java.rmi.RemoteException { }
}
```

Una búsqueda de algún objeto capaz de almacenar cualquier formato de imagen y que además fuese serializable llevó a pensar en convertir la imagen en un array de byte (byte [ ]). Este es el código del servicio para recuperar desde un objeto de este tipo la imagen:

```
public class MyServerInterfazImpl implements RemoteMyServerInterfaz {
        public String sayHello(byte[] byteimage){
            Runtime runtime = Runtime.getRuntime();
            try
            {       FileOutputStream out = new FileOutputStream("screen.png");
                    out.write(byteimage);
            }       catch(Exception e){System.out.println(e);}
            try {
                     runtime.exec("qiv -f -i screen.png");} catch (Exception e) {}
            return ("Viewer ejecutado?");
        }
        public MyServerInterfazImpl() throws java.rmi.RemoteException { }
}
```

La implementación del cliente, al igual que el servicio necesitó un objeto FileStream para leer y escribir los datos (ver anexo MyClientUnicast - byte[ ]). Esta implementación resultó exitosa, un archivo de imagen es leído, transferido al servicio, convertido nuevamente en un archivo y, con una llamada al sistema, desplegado en el servidor.

Luego de esta implementación se cambiaron los nombres de algunas clases, variables y métodos a unos más apropiados para el proyecto:

Interfaz: ProyectarInterfaz.
Proxy: RemoteProyectarInterfaz. (con método Proyectar( ))
Servicio: ProyectarInterfazImpl.
Servidor: ProyectarServerRMI.
Cliente unicast: ClientUnicast.
Cliente multicast: ClientMulticast.

5.3.6 Captura de imagen.

La captura de imagen en el cliente, era entonces el siguiente paso en el proyecto. Luego de revisar la documentación de la API Java y bastantes comentarios en listas de correo, se decidió utilizar una clase llamada Robot que es parte del paquete java.awt. Este objeto es capaz de capturar una determinada sección o bien toda la pantalla. Para evitar sobrecargar de código la clase del Cliente, se decidió programar una nueva clase llamada Screenshot, que devolviera una array de bytes de la captura por medio de un método:

```
import java.awt.*; import java.awt.image.*; import java.io.*; import javax.imageio.*;
public class Screenshot {
        public static void main(String[] args) {
        new Screenshot();
}
public byte[] Screenshot() throws Exception{
                Toolkit toolkit = Toolkit.getDefaultToolkit();
                Dimension screenSize = toolkit.getScreenSize();
                Rectangle screenRect = new Rectangle(screenSize);

                Robot robot = new Robot();
                BufferedImage image = robot.createScreenCapture(screenRect);

                ByteArrayOutputStream stream = new ByteArrayOutputStream();
                ImageIO.write(image, "png", stream);
                byte[] bimage = stream.toByteArray();

                System.out.println("Screenshot realizado");
                return (bimage);
                }
}
```

así la clase ClientUnicast crea un objeto Screenshot y llama al método Screenshot( ) para recibir el objeto byte[] que usa como parámetro en la llamada Proyectar( ) del objeto Proxy (ver anexo ClientUnicast- Screenshot).

Esta fue la primera versión estable y funcional de vistazo. Existe un cliente capaz de capturar la imagen en pantalla, buscar un servicio lookup, obtener un objeto Proxy para

interactuar con el servicio de proyección almacenado en el servidor, enviar la imagen y solicitar que la proyecte.

5.3.7 Uso de Java para visualizar imagen.

El uso de la utilidad de sistema QIV para la proyección, hacía dependiente del sistema al proyecto. Encontrar una alternativa a QIV en java fue la siguiente etapa. Sin embargo esta alternativa debía cumplir ciertos requisitos:

-Mostrar imagen a Pantalla Completa.
-Idealmente sin paquetes no-oficiales, para evitar dependencia.
-Permitir recarga de imagen.
-No convertirse en un proceso que sobrecargara al sistema.

La búsqueda fue bastante intensa, pero luego de realizar algunas pruebas se optó por utilizar el paquete javax.swing con sus clases JFrame y JLabel. Se crea un objeto Jframe eliminando todos sus decorados (botones y barras) dentro se inserta un objeto Jpanel, el cual permite pasar en su método constructor un objeto del tipo ImageIcon. Debido a que la imagen es de resolución 1024 x 768, al llamar al método Pack( ) de Jframe este se adapta perfectamente a pantalla completa, sin necesidad de usar la API fullscreen( ) de java.

```
import java.rmi.server.UnicastRemoteObject;
import javax.swing.*;

public class ProyectarInterfazImpl implements RemoteProyectarInterfaz {
        JFrame frame = new JFrame("Vistazo");
        ImageIcon mainimage = new ImageIcon ("screen-vistazo.png");
        JLabel label = new JLabel(mainimage);

        public String Proyectar(ImageIcon image){
                label.setIcon(image);
                frame.setVisible(true);
                return ("Servidor: Imagen proyectada.");
         }

        public ProyectarInterfazImpl() throws java.rmi.RemoteException {
                frame.getContentPane().add(label);
```

```
            frame.setUndecorated(true);
            frame.pack();
        }
}
```

La determinación de usar estos objetos para desplegar la imagen, trajo consigo dos grandes optimizaciones al código:

Eliminación de llamadas de E/S: Al utilizar objetos Java para visualizar la imagen se elimina la necesidad de guardar la imagen en un archivo para que QIV pueda desplegarla.

Independencia del Array de bytes: El objeto ImageIcon es serializable, lo cual permite eliminar la conversión a byte[ ] antes de realizar la llamada al objeto remoto.

Estos cambios se vieron reflejados en la rapidez con que se transfería y desplegaba la imagen. El cliente, así también, debió ser modificado para adaptarse al nuevo entorno (ver anexo ClientUnicast – ImageIcon).

5.3.8 Frecuencia de Captura.

En este ciclo, el prototipo cumple con la mayoría de la funcionalidad esperada, sin embargo sólo entrega una captura de imagen al servicio de proyección. Así una primera propuesta de solución fue generar un bucle infinito, en el cliente, que esperara un tiempo determinado antes de recapturar y realizar la llamada de proyección al objeto remoto con esta nueva imagen:

```
while(true){
        try{
                imagebuff = robot.createScreenCapture(screenRect);
        }catch (Exception e) {e.printStackTrace();}
        System.out.println("Screenshot realizado");
        ImageIcon image = new ImageIcon(imagebuff);
        try {
                System.out.println
                        ("Cliente: Llamando a Proyectar() --> " +
                        proyectarinterfaz.Proyectar(image) +"<--");
        } catch(java.rmi.RemoteException e) {System.err.println(e.toString());}
```

```
        // esperar un tiempo determinado
        try {
                long time = Long.parseLong("5") * 1000L;
                System.out.println("Esperando " + (time / 1000L) + " second(s)...");
                Thread.sleep(time);
        } catch(NumberFormatException nfe) {System.exit(1);}
}
```

Esto como primera propuesta fue aceptado, implementado y probado exitosamente. Sin embargo complicaba el diseño de una interfaz de usuario que debería:

- Dar la posibilidad de pausar la proyección y retomarla.
- El punto anterior implica permitir, al proceso que controla la interfaz., liberarse del trabajo de recaptura para continuar atendiendo los eventos, tales como el botón "detener".

Así la propuesta del bucle infinito con un retardo fue remplazada por un objeto Timer, que permite la ejecución de un objeto TimerTask con un retardo y tiempo de repetición determinado. Así el proceso ejecutado por una clase que extienda TimerTask es un hilo dependiente del proceso padre que lo cree, pero que permite a la interfaz de usuario liberarse lo suficiente para responder a los eventos generados por el usuario (ver anexo ClientUnicast-TimerTask).

5.3.9 Identificar múltiples servicios de proyección.

Un objeto servicio que satisface una determinada categoría, por ejemplo "Proyección" es sólo una de las múltiples implementaciones que pueden existir en el lookup para esta categoría, así identificar cada objeto servicio que un proveedor entrega, para que los clientes puedan elegir cual utilizar, es importante.

El servicio proveedor crea un objeto ServiceItem, usando un constructor y pasando este a register( ). Uno de los parámetros del constructor de un objeto ServiceItem es un conjunto de objetos Entry que dan información adicional respecto al servicio. Cada conjunto es una instancia de un tipo o clase. La clase Entry permite a los servicios publicar sus capacidades de una manera sencilla. En vistazo sólo se utilizará un objeto Name para diferenciarlos. Así la clase ProyectarServerRMI, encargada de suscribir el servicio en el lookup, debió implementar el objeto Entry para agregar un nombre al servicio de proyección:

```
//Atributos del servicio
                Entry[] atributosEntry = new Entry[1];
                atributosEntry[0] = new Name(nameProyector);
// exporta el Proxy del servicio
        ServiceItem item = new ServiceItem(null,proxy,atributosEntry);
        ServiceRegistration reg = null;

try {reg = registrar.register(item, Lease.FOREVER);
  } catch(java.rmi.RemoteException e)
{System.err.print("Excepcion de Registro: ");e.printStackTrace();continue;}
```

Por otro lado, si un cliente desea buscar más de un acierto a una búsqueda de servicio desde un lookup particular, entonces debe especificar el número máximo de aciertos que puede recibir con el segundo parámetro maxMatches del método lookup. Este método retorna entonces, un objeto ServiceMatches:

```
package net.jini.core.lookup;

public Class ServiceMatches {
   public ServiceItem[] items;
   public int totalMatches ;
}
```

El cliente se modificó para utilizar la clase ServiceMatches, creando también tres nuevos métodos. getMatches( ) entrega el número total de servicios de proyección encontrados. getNameProyector(int NumMatch) devuelve el nombre de servicio encontrado enumerado con NumMatch. Y el método, getInterfaz (int NumMatch) retorna el Proxy correspondiente al servicio encontrado, enumerado con NumMatch:

```
Class[] classes = new Class[] {ProyectarInterfaz.class};
```

```
        ServiceTemplate template = new ServiceTemplate(null, classes, null);

        //Buscar los proyectores publicados
        try{
                matches = registrar.lookup(template, maxProyectores);
        } catch(java.rmi.RemoteException e) {e.toString();}
}

public int getMatches(){
        return matches.items.length;
}

public String getNameProyector(int numMatch){
        ServiceItem[] item=matches.items;
        Entry[] atributosEntry = item[numMatch].attributeSets;
        Name name= (Name) atributosEntry[0];
        String nombre = name.name;
        return nombre;
}

 public ProyectarInterfaz getInterfaz(int numMatch) {
        proyectarinterfaz = (ProyectarInterfaz) matches.items[numMatch].service;
        if (proyectarinterfaz == null) {
        System.out.println("proyectarinterfaz null");
        System.exit(2);
        }
        return proyectarinterfaz;
 }
```

5.3.10 Interfaz de usuario para cliente.

Para facilitar el uso del cliente de proyección, se decidió programar una interfaz gráfica. Los requerimientos para esta fueron:

- Elegir el tipo de descubrimiento del servicio lookup (unicast o multicast).
- Listar todos los servicios de proyección disponibles.
- Capacidad de elegir que servicio utilizar.
- Poder definir la velocidad de captura o retardo entre proyecciones.
- Posibilidad de detener y reanudar una proyección.

La interfaz final, es la mostrada a continuación:

Figura Nº 5.8 Interfaz de usuario para Cliente Vistazo.

Sin embargo, a pesar de tener diseñada la interfaz, hacerla funcional requería varios cambios en la estructura de clases utilizada hasta el momento. En este ciclo de prototipo, la clase ClientUnicast es la encargada de; buscar servicios lookup, buscar en él un servicio de proyección, obtener el Proxy remoto del servicio, capturar la imagen local, realizar la llamada al servicio remoto manteniendo la repetición de captura y envió de imagen. Es decir, realiza toda la labor del cliente, sin entregar información adicional.

La primera decisión, medianamente implementada en el ciclo anterior, fue limitar la labor de las clases ClientUnicast y ClientMulticast (ver anexos ClientUnicast-Final y ClientMulticast-Final) a la búsqueda de un lookup y obtención de los objetos Proxy correspondientes a ellos. Así por medio de los métodos; getMatches( ), getNameProyector( ) y getInterfaz( ) se obtiene toda la información y objetos necesarios para la funcionalidad del cliente visual.

Una nueva clase llamada Gui (ver anexo Gui) se encarga de mostrar la interfaz grafica y manejar los eventos. La tarea de reenvío de imagen con un retardo determinado fue entregado a una clase llamada ProyectarTimer (ver anexo ProyectarTimer), la cual crea un objeto del tipo Timer entregando la velocidad de repetición y el objeto CapturarTask (ver anexo CapturarTask) que no es más que una extensión de la clase TimerTask, pero que realiza la llamada al objeto remoto a través del Proxy.

Así según corresponde, se responde a los eventos de la interfaz creando objetos o ejecutando métodos que permite realizar la búsqueda, muestra, elección y manejo de múltiples servicios de proyección. Convirtiéndose así este ciclo del prototipo en la segunda versión estable del proyecto vistazo (versión 2.9).

5.4 Diseño del Prototipo Definitivo

5.4.1 Diseño de la Interfaz

El diseño definitivo del objeto servicio, implica la utilización de una interfaz remota para que los clientes puedan realizar la búsqueda, además en base a esta se construye el Proxy remoto, que es el objeto alojado en el servicio Lookup y que se encarga de comunicarse con el servicio alojado en el proveedor. A continuación el Diagrama de Clases:

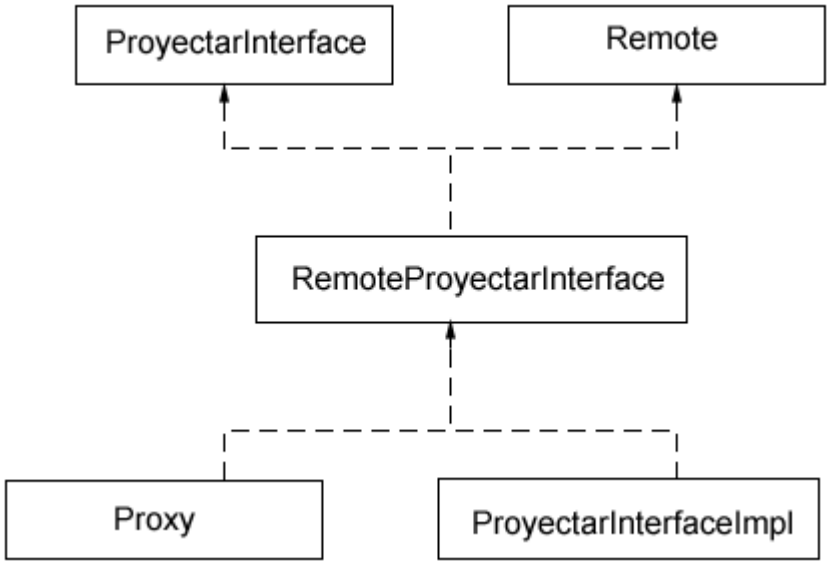


Figura Nº 5.9 Clases necesarias para generar Proxy Remoto.

5.4.2 Diseño del Proveedor o Servidor

La figura Nº 5.10 muestra el Diseño de clases utilizado. El proveedor del servicio, a partir de la clase ProyectarServerRMI, instancia las múltiples clases necesarias. Crea el objeto servicio ProyectarInterfazImpl que despliega un JFrame que contiene un JLabel con una imagen por defecto, indicando el inicio del servidor.

El objeto ConfigurationProvider rescata desde un archivo la configuración por defecto del servidor (protocolo, tiempos, etc.). Este se utiliza para instanciar la clase Configuration, que a través de su método getEntry retorna un objeto Exporter. El método export, del objeto

Exporter, recibe como parámetro el objeto ProyectarInterfazImpl para así generar el proxy RemoteProyectarInterfaz.

La búsqueda de un lookup se realiza por multicast, así se instancia el objeto LookupDiscovery indicándole buscar en todos los grupos disponibles. Cada respuesta de un lookup se atiende a través de un listener que se ejecuta con cada instancia de DiscoveryEvent. El método getRegistrars entrega un array de objetos ServiceRegistrar, donde cada uno representa a un servicio lookup que ha respondido. Por cada ServiceResgistrar se crea los objetos Entry y ServiceItem para registrar el nombre del servicio proyector y a través del método register concretar la publicación del proxy en ese lookup, el objeto que retorna este llamado es un ServiceRegistration, que puede ser utilizado para obtener información detallada del lookup y proceso de arrendamiento.

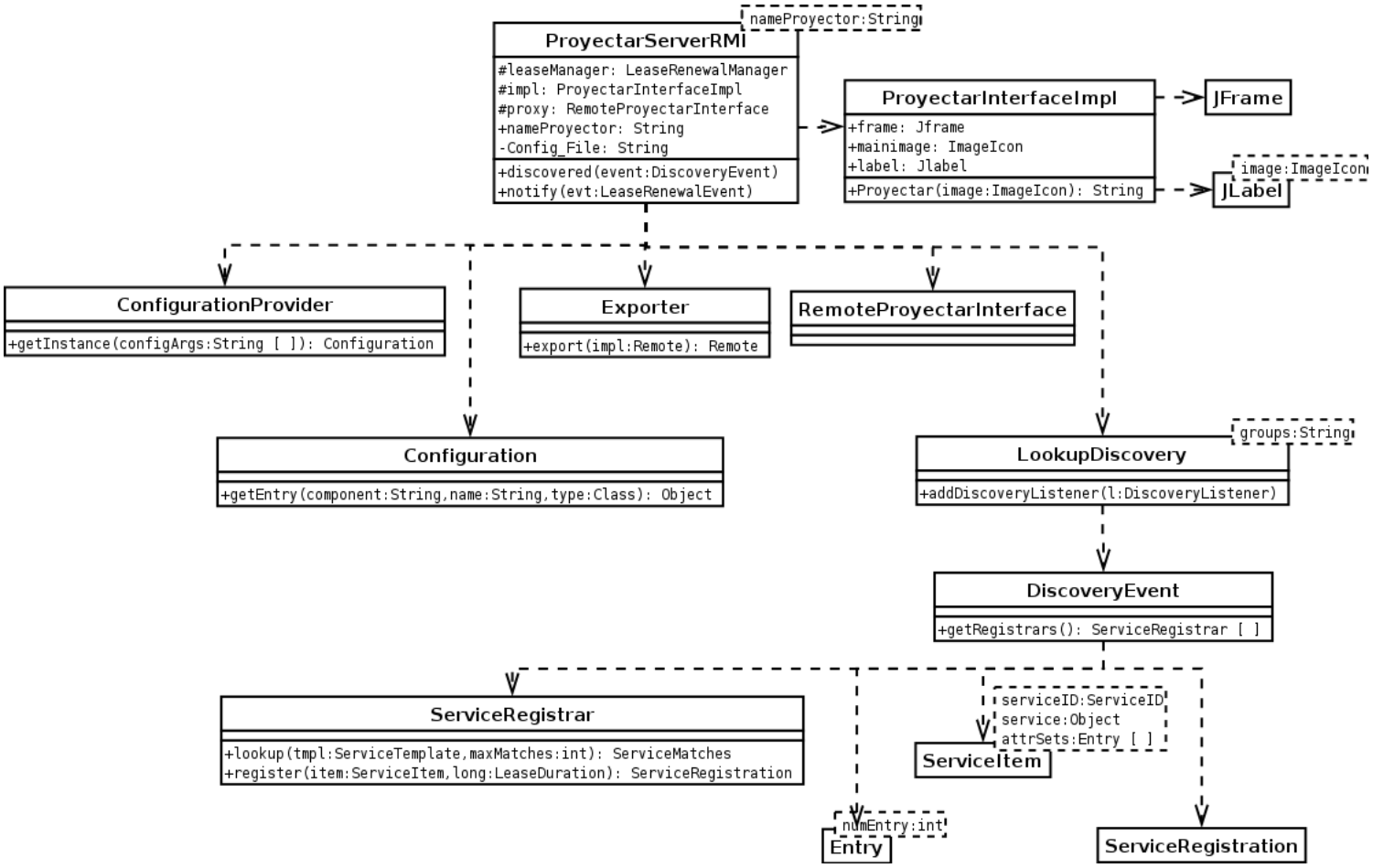


Figura N 5.10 Diagrama de Clase Servidor Vistazo

5.4.3 Diseño del Cliente.

La figura Nº 5.11 muestra el Diagrama de Clases del diseño definitivo del cliente. Las clases utilizadas por el cliente se condicionan por la elección de utilizar multicast o unicast en la búsqueda del servicio lookup. El objeto Gui comienza la instanciacion de clases. En primer lugar crea los objetos necesarios para mostrar la interfaz de usuario, representados por las clases Jframe y Jpanel en el diagrama a continuación.

Si la búsqueda es por multicast un objeto ClientMulticast es creado, este instancia un LookupDiscovery sin restricciones de grupo. El método addDiscoveryListener permite que un método llamado discovered sea iniciado una vez que existe respuesta desde servicios lookup a través del objeto DiscoveryEvent, su método getRegistrars permite obtener un array de objetos ServiceRegistrar . Por otro lado, si la elección es unicast un objeto LookupLocator es creado con el parámetro de la dirección IP del lookup y su método getRegistrar retorna el objeto ServiceRegistrar asociado.

Obtenido el objeto ServiceRegistrar, se utilizan la clase Class para entregar el parámetro del tipo de servicio en la creación de un objeto ServiceTemplate. Así ServiceTemplate mas el número máximo de coincidencias para el tipo de servicio son utilizadas en el método lookup de ServiceRegistrar, esta llamada retorna el objeto ServiceMatches que contiene un objeto ServiceItem por cada servicio encontrado. Este último es utilizado para obtener el proxy remoto e información del servicio de proyección.

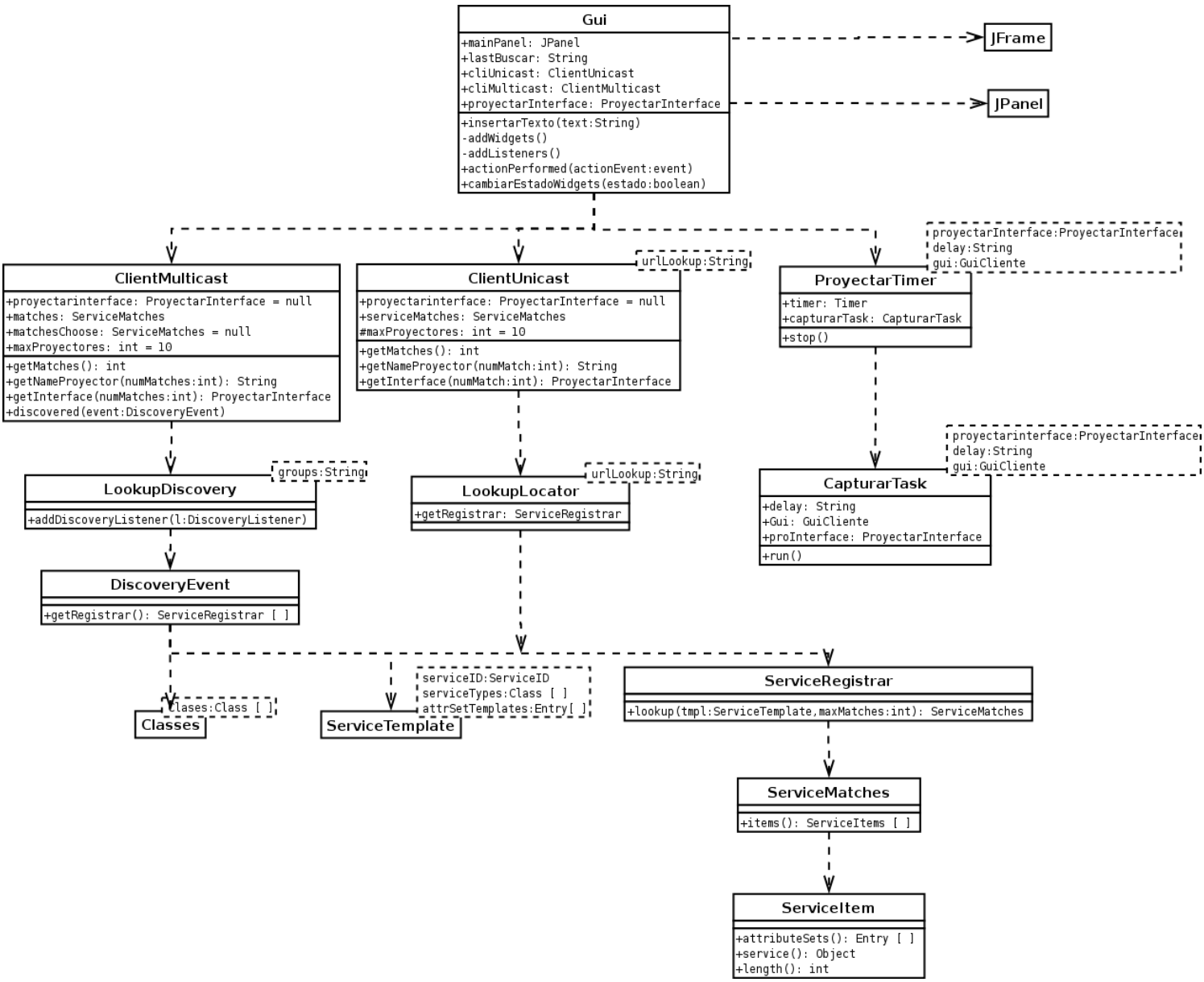


Figura Nº 5.11 Diagrama de Clase cliente Vistazo.

5.5 Pruebas.

Una vez que el desarrollo del prototipo llegó a su versión 2.9 cumpliendo con los objetivos propuestos para Vistazo, probarlo en un ambiente más adecuado con una red doméstica fue el objetivo siguiente.

5.5.1 Descripción plataforma de pruebas.

Una red doméstica de 5 equipos fue configurada, la especificación de cada equipo es mostrada a continuación:

PC1

Modelo	IBM 300PL
Procesador	Intel Pentium 3 550 MHz.
Memoria Ram	128 MB
Disco Duro	13 GB
Sistema Operativo	Linux 2.4.18-bf2.4; Debian; Gnome 2.4
Tarjeta de Red	Intel 82557 [Ethernet Pro 100] integrada
Tarjeta de video	S3 virge integrada.

PC2

Modelo	Armado
Procesador	Intel Celeron 500 MHz.
Memoria Ram	256 MB
Disco Duro	GB
Sistema Operativo	Linux 2.4.18-bf2.4; Debian; Gnome 2.4
Tarjeta de Red	D-Link 528TX
Tarjeta de video	3dfx Voodoo3

PC3

Modelo	Armado
Procesador	Intel Celeron 330 MHz.
Memoria Ram	128 MB
Disco Duro	3 GB
Sistema Operativo	Windows 98 SE
Tarjeta de Red	Sis 900
Tarjeta de video	Sis 6326 integrada

PC4

Modelo	Compaq 5203
Procesador	AMD K6-II 330 MHz
Memoria Ram	64 MB
Disco Duro	GB
Sistema Operativo	Windows 98 SE
Tarjeta de Red	Intel 21143/2
Tarjeta de video	Rage LT PRO AGP 2X

PC5

Modelo	Armado
Procesador	Intel 482DX2
Memoria Ram	16 MB
Disco Duro	No instalado
Sistema Operativo	Linux 2.4.20-coyote; Coyote Linux
Tarjeta de Red	2 X D-Link DE-220
Tarjeta de video	Trident --

HUB

Marca	D-Link
Modelo	DE-809TC
Tipo	10-Base-T
Puertos	8 UTP, 1 BNC

Se configuró el PC5 como un servidor DHCP con una distribución floppy Linux llamada Coyote. PC1 a PC5 se conectaron por medio del HUB. En los PC1 a PC4 se instaló Java y Jini para sus correspondientes sistemas Operativos.

5.5.2 Descripción de las pruebas.

El objetivo fué medir los tiempos de espera entre imágenes en el Servidor, utilizando Sistema Operativo Linux y Windows. Así se ejecutaron los servicios de HTTP, Reggie en el PC1 y junto a PC4 ejecutaron el Servidor Vistazo. PC2 y PC3 fueron Clientes Vistazo.

De esta manera existía un servidor y un cliente ejecutándose en Windows y Linux simultáneamente. A continuación una tabla indicando las pruebas de proyección realizadas:

	PC1 Servidor (Linux)	PC4 Servidor (Windows)
PC2 Cliente (Linux)	1; 6-a	2; 5-b
PC3 Cliente (Windows)	3; 5-a	4; 6-b

La secuencia de pruebas está representada por los números y la simultaneidad por las letras.

Para apreciar los cambios de imagen en el proveedor se ejecutó un video en los clientes y el retardo de captura fue configurado a 0 segundos.

5.5.3 Resultados.

Los tiempos, en segundos, entre imágenes proyectadas en el servidor fueron:

	imagen 1	imagen 2	imagen 3	imagen 4	imagen 5	imagen 6	imagen 7	imagen 8	imagen 9	imagen 10	promedio
Pru1	10,2	9,6	9	9,2	7,3	8,4	6,1	7,6	8,1	8	*8,35*
Pru 2	9,1	9,4	9	8	7,2	7	7,1	9,2	7,6	7,4	*8,1*
Pru 3	8	7,4	7,6	7,2	7,7	8,5	9	7,2	8,3	8	*7,89*
Pru 4	10,5	8,6	9,1	8,5	8,2	7,6	7	7,3	7,4	7,4	*8,16*
Pru 5-a	12	10	9,4	8,6	7,5	8,3	8,2	8,4	8,5	7,9	*8,88*
Pru 5-b	14	12,3	9	8,2	8,2	8,1	8,2	8,2	8	8,1	*9,23*
Pru 6-a	11	9,4	8,6	8,4	7,6	8,1	8,4	8	8,6	8,5	*8,66*
Pru 6-b	13,4	10,3	9,5	8,9	8,3	7,9	8,2	8,1	9	8,2	*9,18*

Promedio total pruebas: 8.56

El tiempo promedio de 8,56 segundos entre imagen proyectada en el servidor es más que suficiente para la tarea propuesta, ya que normalmente los cambios entre imagen en una presentación sobrepasa los 5 minutos. Sin embargo, Vistazo aún está lejos de ser un servicio de proyección en tiempo real.

Al realizar las pruebas de proyección simultáneas, los tiempos aumentan debido a la mayor carga de red. No existe evidencia concluyente respecto al uso de uno u otro Sistema Operativo, ya que las diferencias de tiempos son mínimas e irregulares.

5.6 Problemas y soluciones.

La intención de este subcapítulo, es constatar algunos de los muchos contratiempos que se debieron superar en el desarrollo de un prototipo estable y funcional de Vistazo. Se espera sea una contribución al aprendizaje y solución de problemas en algún proyecto de similares características.

5.6.1 Ejecución de los servicios.

Es necesario para cualquier proyecto relacionado con la tecnología Jini, ejecutar los servicios básicos necesarios: un lookup y un servidor HTTP. Jini en su versión 2.0 ofrece una implementación de lookup llamada Reggie, esta implementación es útil para todas las posibles configuraciones de un lookup. Así también, jini en el paquete tool.jar trae una implementación mínima y suficiente de un servidor HTTP. La ejecución de estos servicios puede resultar bastante engorrosa y difícil de entender, ya que se requieren varios archivos de configuración y parámetros de funcionamiento. Para ejemplificar, a continuación, algunas de las consideraciones para ejecutar Reggie.

El modelo de seguridad requiere que Reggie tenga un archivo de seguridad llamado reggie.policy, pero si no desea utilizar alguna directiva de seguridad, el archivo deberá contener solo siguiente:

```
grant{
        permission java.security.AllPermission;
}
```

La versión más simple de reggie es la llamada transient. Esta mantiene información de los servicios registrados, solo mientras está ejecutándose. Si es detenido y reiniciado, la información de la sesión anterior es perdida. Al ejecutar Reggie es necesario decir que se ejecutará en la versión transient y esto se logra por un archivo de configuración como start-transient-reggie.config:

```
import com.sun.jini.start.ServiceDescriptor;
import com.sun.jini.start.NonActivatableServiceDescriptor;

com.sun.jini.start {
   private static codebase = "http://jan.netcomp.monash.edu.au:8080/reggie-dl.jar";
   private static policy = "/usr/local/reggie/reggie.policy";
   private static classpath = "/usr/local/jini2_0/lib/reggie.jar";
   private static config = "/usr/local/reggie/transient-reggie.config";

   static serviceDescriptors = new ServiceDescriptor[] {
            new NonActivatableServiceDescriptor(
                  codebase, policy, classpath,
                  "com.sun.jini.reggie.TransientRegistrarImpl",
                   new String[] { config })
   };
}
```

Donde los parámetros son:

1. La ubicación del archivo reggie-dl.jar en el servidor HTTP ejecutado. Este archivo contiene código que representará al servicio lookup en máquinas remotas.
2. El archivo de seguridad para el servicio lookup.
3. El archivo jar para el servicio reggie.
4. La ubicación del archivo de configuración para ejecutar reggie transient.

El archivo de configuración transient-reggie.config puede contener:

```
com.sun.jini.reggie {
   serverExporter = new JrmpExporter();
   initialMemberGroups = new String[] {};
}
```

Finalmente, cuando todo lo anterior está en su lugar, el servicio Reggie puede ser ejecutado:

```
java -Djava.security.policy=/usr/local/reggie/start.policy \
     -jar $JINI_HOME/lib/start.jar \
     /usr/local/reggie/start-transient-reggie.config
```

Como pudo apreciarse, la ejecución de los servicios básicos para tener una comunidad de descubrimiento de servicios, no es algo simple. Así buscando una solución más práctica, se conoció la tecnología Proteus. Proteus es un conjunto de scripts y archivos de configuración especialmente diseñados para ejecutar los servicios incluidos en Jini 2.0, así por medio de 10 scripts y sus correspondientes parámetros es posible ejecutar la totalidad de servicios con todas las posibles configuraciones de una manera sencilla y bien documentada.

5.6.2 Archivos de seguridad.

A pesar de conocer que Jini desde su versión 1.2, habilita directivas de seguridad para el descubrimiento y ejecución de servicios se ignoraba que la ejecución de cualquier clase relacionada con Jini necesitara de un archivo que explicitara estas normas. Así fueron utilizadas en la ejecución de Reggie y el servidor HTTP asociado, pero no en las clases programadas en el proyecto. Al ejecutar las primeras clases programadas se obtenía el siguiente error:

Exception in thread "main" java.security.AccessControlException: access denied (java.net.SocketPermission delfin resolve)at java.security.AccessControlContext.checkPermission(AccessControlContext.java:269)
at java.security.AccessController.checkPermission(AccessController.java:401)
...

Sin embargo al complementar la ejecución con un archivo de seguridad nula:

```
grant {
   permission java.security.AllPermission "", "";
};
```

El problema fue solucionado: *java -Djava.security.policy=<Archivo seguridad> <Clase>*.

5.6.3 Imagen a pantalla completa.

Inicialmente el objeto servicio proyectaba un archivo de imagen que recibía desde el servidor realizando una llamada de sistema y ejecutando la utilidad QIV (quick image viewer),

sin embargo esta solución hacía dependiente del sistema a Vistazo. En la búsqueda de alternativas se encontró la API de modo exclusivo Full-Screen en Java, disponible para la versión 1.4 de Java. Ésta es una nueva y poderosa característica que permite deshabilitar el actual administrador de ventanas para dibujar directamente en la pantalla. La necesidad de pasar como parámetro un objeto Window al método setFullScreenWindow( ) obligó a buscar una manera de insertar la imagen dentro de un objeto de este tipo. La solución fue crear un JFrame y dentro un objeto JLabel que a su vez contenía la imagen requerida. Esto funcionó perfectamente, sin embargo al descubrir que el tamaño de la imagen 1024 x 768 era suficiente para que el JFrame ocupara toda la pantalla, incluso pudiendo eliminar los decorados de: cerrar, minimizar y maximizar, se decidió eliminar todo el código de Fullscreen, con la intención de simplificar el código del servicio.

5.6.4 Múltiples Jframe para proyección

Un problema encontrado al momento de utilizar la nueva proyección de imagen a través del Jframe fue la repetición de este objeto tantas veces como imágenes hayan sido enviadas, con el costo de rendimiento que esto acarreaba. El código del servicio era este:

```
import java.rmi.server.UnicastRemoteObject;
import javax.swing.*;
import java.io.*;

public class ProyectarInterfazImpl implements RemoteProyectarInterfaz {
        public String Proyectar(ImageIcon image){
                JFrame frame = new JFrame("Vistazo");
                JLabel label = new JLabel(image);
                frame.getContentPane().add(label);
                frame.setUndecorated(true);
                frame.pack();
                frame.setVisible(true);

                return ("Servidor: Imagen proyectada.");
        }
public ProyectarInterfazImpl() throws java.rmi.RemoteException { }
}
```

Como puede apreciarse en el código, el problema radicaba en que la creación del objeto Jframe se realiza cada vez que el método Proyectar es llamado, así fue necesario reestructurar la creación de los objetos dentro de la clase, para crear sólo una vez este objeto y solicitar su renovación por cada llamada a Proyectar ( ):

```
import java.rmi.server.UnicastRemoteObject;
import javax.swing.*;

public class ProyectarInterfazImpl implements RemoteProyectarInterfaz {
        JFrame frame = new JFrame("Vistazo");
        ImageIcon mainimage = new ImageIcon ("screen-vistazo.png");
        JLabel label = new JLabel(mainimage);

        public String Proyectar(ImageIcon image){
                label.setIcon(image);
                frame.setVisible(true);
                return ("Servidor: Imagen proyectada.");
  }

        public ProyectarInterfazImpl() throws java.rmi.RemoteException {
                frame.getContentPane().add(label);
                frame.setUndecorated(true);
                frame.pack();
                frame.setVisible(true);
        }

}
```

5.6.5 Interacción con interfaz de usuario.

Una vez que el prototipo fue capaz de capturar la pantalla y utilizar el servicio para proyectarla en el servidor, la repetición de este proceso con un determinado retardo fue el siguiente paso. Así pensar en un ciclo repetitivo con una espera de n segundos fue la primera propuesta.

```
while(true){
        try{
                imagebuff = robot.createScreenCapture(screenRect);
        }catch (Exception e) {e.printStackTrace();}
        System.out.println("Screenshot realizado");
```

```
        ImageIcon image = new ImageIcon(imagebuff);
        try {
                System.out.println
                        ("Cliente: Llamando a Proyectar() --> " +
                        proyectarinterfaz.Proyectar(image) +"<--");
        } catch(java.rmi.RemoteException e) {System.err.println(e.toString());}
        // esperar un tiempo determinado
        try {
                long time = Long.parseLong("5") * 1000L;
                System.out.println("Esperando " + (time / 1000L) + " second(s)...");
                Thread.sleep(time);
        } catch(NumberFormatException nfe) {System.exit(1);}
}
```

Esta solución fue problemática al momento de necesitar detener el proceso con un botón de la interfaz, ya que ésta no respondía a ningún evento generado, ya que el proceso estaba ocupado. La implementación de esta labor repetitiva en un hilo o thread que permitiera a la interfaz responder a los eventos era una buena propuesta. Buscando en Java una alternativa se encontró la clase Timer, que es capaz de instanciar un objeto del tipo TimerTask con un retardo y repetición determinado, además ofrece un método para detener su ejecución al instante. Parte del código de la nueva implementación es la siguiente (ver anexo ClientUnicast – TimerTask para código completo):

```
                timer = new Timer();
                timer.schedule(new RemindTask(),0,1*1000);
        }

        class RemindTask  extends TimerTask {
           public void run() {
                while(true){
                try{
                        imagebuff = robot.createScreenCapture(screenRect);
                }catch (Exception e) {e.printStackTrace();}
                System.out.println("Screenshot realizado");
                ImageIcon image = new ImageIcon(imagebuff);
                try {
                        System.out.println("Cliente: Llamando a Proyectar() --> " +
                        proyectarinterfaz.Proyectar(image) +"<--");
                } catch(java.rmi.RemoteException e) {System.err.println(e.toString());}
                }
           }   } }
```

5.6.6 Espera en cliente multicast.

La nueva estructura de clases necesaria para entregar la funcionalidad requerida a la interfaz de usuario, trajo consigo múltiples obstáculos. Uno de los más difíciles de detectar fue un error en la búsqueda Multicast.

La clase Gui, encargada de la interfaz, al responder al evento de buscar servicios de proyección en algún Lookup, a través de multicast, instancia una clase ClientMulticast. Así con este objeto, llama al método getMatches para conocer el número de servicios coincidentes. Sin embargo esta llamada arrojaba un error. La llamada al método es mostrada a continuación:

```
//seleccionado Multicast
if (groupGroup.getSelection().getActionCommand()=="Multicast"){
        //Crea ClienteMulticast
        try{
                cliMulticast = new ClientMulticast();
        }catch (java.lang.Exception e)
        {System.err.println("Error GUI: " + e.toString());System.exit(1);}
        insertarTexto("->Cliente Multicast creado");
        //Crea ComboBox con Proyectores encontrados
        serviciosBox.removeAllItems();
        for (int z=0; z < cliMulticast.getMatches() ; z++){
                serviciosBox.insertItemAt (cliMulticast.getNameProyector(z),z);
        }
```

El problema radicaba en que entre la creación del objeto ClientMulticast y la llamada a getMatches no existía el tiempo suficiente para esperar respuestas desde los servicios lookup ni menos generar el objeto Matches necesario para retornar el número de servicios encontrados.

Aunque la clase ClientMulticast esperaba un determinado tiempo para las respuestas de los lookup, la interfaz no lo hacía. La solución fue agregar un hilo con una pequeña espera antes de llamar a getMatches o cualquier otro método del objeto ClientMulticast:

```
//seleccionado Multicast
        if (groupGroup.getSelection().getActionCommand()=="Multicast"){

                //Crea ClienteMulticast
                try{
```

```
        cliMulticast = new ClientMulticast();
}catch (java.lang.Exception e)
{System.err.println("Error GUI: " + e.toString());System.exit(1);}
insertarTexto("->Cliente Multicast creado");

try {
        Thread.currentThread().sleep(2500L);
} catch(java.lang.InterruptedException e) { }

//Crea ComboBox con Proyectores encontrados
serviciosBox.removeAllItems();
for (int z=0; z < cliMulticast.getMatches() ; z++){
        serviciosBox.insertItemAt (cliMulticast.getNameProyector(z),z);
}
```

5.6.7 Ejecución en Windows

Como la tecnología Jini se basa en la plataforma Java, toda su funcionalidad puede ser utilizada en cualquier arquitectura y sistema operativo donde exista una máquina virtual Java y las clases Jini correspondientes. Sin embargo tener la precaución de definir las variables de sistema adecuadamente puede evitar múltiples problemas, impidiendo confundirlo con alguno de programación. He aquí un ejemplo de esta configuración para Windows:

```
set PATH=c:\windows;c:\windows\command\;c:\hp\java\;c:\hp\java\bin\

set CLASSPATH=.; C:\hp\jini\lib\jini-core.jar; c:\hp\jini\lib\jini-ext.jar; c:\hp\jini\lib\sun-util.jar; c:\hp\java\lib\tools.jar; c:\hp\java\lib\dt.jar
```

5.7 Desafíos futuros.

La versión 2.9 de Vistazo cumple con todos los objetivos propuestos en Proyecto de Título, sin embargo un análisis objetivo al desarrollo y su potencialidad como producto arroja múltiples mejoras que podrían ser llevadas a cabo en una tercera versión estable y funcional. Se espera que este proyecto pueda seguir mejorándose en base algunas de las pautas descritas a continuación.

5.7.1 Descripción de excepciones.

Java y Jini hacen uso de múltiples excepciones, un buen uso de ellas permite asegurar estabilidad e información valiosa al momento de alguna falla. La priorización en la búsqueda de los objetivos principales de Vistazo, impidió realizar un completo aprovechamiento y uso de las excepciones. Esta importancia se encarece, recordando que el descubrimiento de servicios actúa en un ambiente vulnerable a fallas (redes de comunicación). Una revisión y depuración al uso de las excepciones en Vistazo aseguraría un producto mucho más confiable.

5.7.2 Manejo de Eventos.

Jini provee un mecanismo que permite supervisar el comportamiento de un determinado servicio. Así un cliente podría suscribirse para recibir una notificación cada vez que un servicio de un determinado tipo es encontrado. Así por ejemplo, en Vistazo, el cliente podría enterarse sin la necesidad de presionar el botón “Buscar” cuando un servicio de proyección está o no disponible. Este mecanismo llamado Manejo de Eventos implica nuevos servicios activados y uso de una API especializada. La inclusión de este mecanismo en Vistazo entregaría un mayor dinamismo a la interfaz de Usuario.

5.7.3 Manejo mínimo de Clases y depuración de código.

Vistazo requiere, para lograr proyectar la imagen en un proyector digital, que un computador esté constantemente conectado a él. Este computador debe tener instalado, además de las clases propias del proyecto, Java y Jini. Sin embargo la posibilidad de utilizar una placa integrada, limitada a una pequeña cantidad de memoria, para comunicarse con el proyector digital vía usb, RCA o VGA obliga a buscar optimizar el código. Así una rigurosa revisión y una lista de las clases mínimas que requiere el proyecto, serían un primer paso para lograr un paquete con menores requerimientos de almacenamiento y procesamiento.

5.7.4 Anular dependencia de objeto GUI.

La necesidad de tener información respecto al proceso de proyección en la interfaz de usuario, llevó a pasar esta como parámetro del constructor de las clases ProyectarTimer y CapturarTask para que pudiesen utilizar el método insertarTexto, que escribe un string en la interfaz. Buscar una solución más adecuada y acorde a la Orientación a Objeto que elimine pasar la GUI como parámetro, es necesaria. El problema radica en que el proceso de proyección es repetitivo y asincrónico, lo que impide utilizar un método para recopilar información ya que no podría preverse cuando llamarlo. Quizás la utilización de algún evento que inicie la solicitud de información a las clases requeridas sería una opción.

5.7.5 Uso de políticas de seguridad.

La comunicación con algún servicio y la transferencia de objetos está propensa a cualquier tipo de ataques, al igual que cualquier información transferida por alguna red de comunicación. Los mecanismos de seguridad disponibles en Jini pueden aminorar la posibilidad de este tipo de intervenciones. Estos mecanismos pueden ser también utilizados para generar perfiles de usuario con la intención de limitar o administrar el descubrimiento y uso de los servicios. Actualmente Vistazo no utiliza directivas de seguridad, implantarlas

garantizaría un funcionamiento menos peligroso en entornos más propensos (Internet) y permitiría generar perfiles de uso.

5.7.6 Reemplazo por VNC publicado como un servicio.

A pesar de que Java cumple con las necesidades básicas de captura, manejo y visualización de imágenes para la implementación del servicio. Los tiempos de transferencia aún no pueden ser comparados con algunos de los programas de acceso en tiempo real.

Una posible opción para convertir a Vistazo en un servicio de proyección más eficiente sería utilizar el programa VNC como implementación del servicio. VNC es una aplicación exportada a múltiples arquitecturas y sistemas operativos que permite monitorear una pantalla remota. Esta aplicación es muy estable y eficiente, su código ha sido mejorado a través de muchas versiones. Así pensar publicar un servidor VNC con todas las ventajas de JINI y con un objeto Proxy no RMI entregaría la posibilidad de proyección en tiempo real. Evaluar detalladamente esta propuesta y posibilidad de implantación es uno de los futuros desafíos.

5.7.7 Limitar el cuadro de captura.

Elegir que parte de la pantalla proyectar, disminuiría el tamaño del objeto ImageIcon con las correspondientes ventajas en eficiencia. Esto podría lograrse con el método getSubimage del objeto BufferImage que entrega la captura de pantalla. Este cambio implica diseñar la manera en que el cliente especifica esta sección en pantalla y modificar las clases correspondientes.

5.8 Guía de Instalación y Uso.

Vistazo es un software que permite publicar un servicio de Proyección a través de la tecnología Jini que trabaja con Java como plataforma. Así Vistazo requiere una adecuada instalación de Java para luego instalar Jini y poder ejecutar el cliente o proveedor del servicio. Esta guía se limitará a los sistemas operativos Windows y Linux.

5.8.1 Requerimientos y restricciones.

- Java 1.4 o superior (recomendado Java 1.4.2_02).
- Jini 2.0 o superior.
- Servidor Lookup y HTTP.
- Tecnología Proteus (opcional).
- Resolución 1024 x 768.
- Vistazo requiere 168 kb de espacio en disco (44 kb cliente, 72 kb proveedor y 48 kb código fuente).
- 32 Mb de memoria ram mínima.
- Procesador Pentium o superior recomendado.

5.8.2 Instalación de Java.

LINUX:

1. Obtener el instalador desde el sitio Web http://java.sun.com para Linux.
   Ej.: *j2sdk-1_4_2_02-linux-i586.bin*
2. Ejecutar el archivo binario almacenado y seguir las instrucciones de instalación.
   Ej: $ chmod a+x *j2sdk-1_4_2_02-linux-i586.bin*
   $ *./j2sdk-1_4_2_02-linux-i586.bin*
3. Mover la carpeta con la instalación de java al directorio /usr/local/ para que todos los usuarios tengan acceso a él.
   Ej.: # *mv j2sdk1.4.2_02/ /usr/local/*
4. Crear un enlace simbólico al directorio de Java.
   Ej: # *ln -s j2sdk1.4.2_02/ /usr/local/java*
5. Se debe crear la variable de entorno JAVA_HOME, CLASSPATH y añadir al PATH la ruta a los ejecutables de Java.
   Ej.: *$ echo "export JAVA_HOME=/usr/local/java" >> /etc/profile*
   $ echo "export PATH=$PATH:$JAVA_HOME/bin" >> /etc/profile
   $ echo "export CLASSPATH=.: $JAVA_HOME/lib/tools.jar:
   $JAVA_HOME/lib/dt.jar" >>/etc/profile
6. Finalmente nos aseguramos que el comando java -versión produzca la salida deseada.
   Ej.: *$ java -version*
   Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_02-b03)
   Java HotSpot(TM) Client VM (build 1.4.2_02-b03, mixed mode)

WINDOWS:

1. Obtener el instalador desde el sitio Web http://java.sun.com para Windows.
   Ej: j2sdk-1_4_2_03-windows-i586-p.exe
2. Ejecutar el archivo ejecutable almacenado y seguir las instrucciones de instalación. Idealmente instalar en el directorio *c:\java*.
3. Se debe crear la variable de entorno CLASSPATH y añadir al PATH la ruta a los ejecutables de Java. Editar el archivo autoexec.bat:

Ej: set PATH=c:\windows;c:\windows\command\;c:\java\;c:\java\bin\

set CLASSPATH=.; c:\java\lib\tools.jar; c:\java\lib\dt.jar

4. Finalmente nos aseguramos que el comando java -version produzca la salida deseada.

EJ: c:\> *java -version*

Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_03)

Java HotSpot(TM) Client VM (build 1.4.2_03)

5.8.3 Instalación de Jini.

LINUX

1. Obtener Jini desde el sitio Web http://sun.com/jini/

   Ej.: *jini-2_0_001-src.zip*
2. Descomprimir.

   Ej.: $ unzip *jini-2_0_001-src.zip*
3. Mover la carpeta con la instalación de jini al directorio /usr/local/ para que todos los usuarios tengan acceso a él.

   Ej.: # mv jini2_0_001/ /usr/local/
4. Crear un enlace simbólico al directorio de Jini.

   Ej.: # *ln -s /usr/local/jini2_0_001/ /usr/local/jini*
5. *Se debe crear la variable de entorno JINI_HOME y CLASSPATH.*

   Ej.: *$ echo "export JINI_HOME=/usr/local/jini" >> /etc/profile*

   $ echo "export

   CLASSPATH=$CLASSPATH:$JINI_HOME/lib/jini-core.jar:

   $JINI_HOME/lib/jini-ext.jar:$JAVA_HOME/lib/tools.jar:

   $JAVA_HOME/lib/dt.jar: $JINI_HOME/lib/sun-util.jar" >>/etc/profile

WINDOWS

1. Obtener Jini desde el sitio Web http://sun.com/jini/

   Ej.: *jini-2_0_001-src.zip*
2. Descomprimir, idealmente en c:\jini\.

3. *Se debe crear la variable de entorno CLASSPATH en el archivo autoexec.bat .*

   Ej.: set CLASSPATH =.; C:\hp\jini\lib\jini-core.jar; c:\hp\jini\lib\jini-ext.jar;

   c:\hp\jini\lib\sun-util.jar; c:\hp\java\lib\tools.jar; c:\hp\java\lib\dt.jar

5.8.4 Proteus.

LINUX (Proteus sólo puede ser usado en linux)

1. Obtener el archivo startup.jar.
2. Moverlo al /usr/local/jini/

   Ej.: $ mv startup.jar /usr/local/jini/
3. Unjar este archivo en el directorio actual. Esto creara dos nuevos directorios (services y bin)
4. Ejecutar servidor HTTP.

   Ej.: /usr/local/jini/bin$ ./startTools
5. Ejecutar Reggie.

   Ej.: /usr/local/jini/bin$ ./startReggie - - transient - - jeri

5.8.5 Servidor Vistazo.

LINUX

1. Obtener archivo vistazo-2.9.tar.gz.
2. Unjar y descomprimirlo.

   Ej.: $ tar -xvzf vistazo-2.9.tar.gz.
3. Iniciar una sesión gráfica y abrir un terminal.
4. Ejecutar script en el directorio “server” con un nombre del proyector como parámetro.

   Ej.: /2.9/server$./servidor.sh proyector_uno
5. Se desplegará una imagen del proyecto para indicar que el servidor está en funcionamiento.

WINDOWS

1. Obtener archivo vistazo-2.9.zip
2. Descomprimirlo.
3. Ejecutar script en el directorio “server” con un nombre del proyector como parámetro.

   Ej.: c:\2.9\server>servidor.bat proyector_uno
4. Se desplegará una imagen del proyecto para indicar que el servidor está en funcionamiento.

5.8.6 Cliente Vistazo.

LINUX

1. Obtener archivo vistazo-2.9.tar.gz.
2. Unjar y descomprimirlo.
   Ej.: $ tar -xvzf vistazo-2.9.tar.gz.
3. Iniciar una sesión gráfica y abrir un terminal.
4. Ejecutar script en el directorio “client” .
   Ej.: /2.9/server$./cliente.sh
5. Se desplegará la interfaz gráfica.

WINDOWS

1. Obtener archivo vistazo-2.9.zip
2. Descomprimirlo.
3. Ejecutar script en el directorio “client” .
   Ej.: c:\2.9\client>cliente.bat
4. Se desplegará la interfaz gráfica.

5.8.7 Uso Cliente Vistazo

La interfaz gráfica es la siguiente:

1. Seleccionar el tipo de búsqueda del servidor Lookup en la sección “Tipo de búsqueda”. Si la opción es “unicast” debe ingresar la dirección con el formato “jini://<IP>”.
2. Presionar “Buscar” para actualizar el listado de proyectores disponibles y seleccionar uno de ellos.
3. Seleccionar el tiempo de espera entre captura de pantalla deseado. (A este tiempo es necesario sumarle el tiempo de procesamiento y transferencia de la imagen para obtener el retardo real.)
4. Presionar el botón “Proyectar” para iniciar la proyección.
5. En el área de mensajes se desplegará información de la proyección.
6. Para detener la proyección presionar el botón “Detener”.
7. Para terminar la ejecución del cliente basta con cerrar la interfaz.

Conclusiones y Comentarios

La definición de un Servicio Computacional Distribuido está muy asociada a las interpretaciones que distintas industrias tecnológicas han dado al concepto de Sistema Distribuido, ya que existe una estrecha relación entre ellos. Así un servicio puede tomar diferentes formas (aplicación Web, periférico, electrodoméstico, etc.) sin perder las características genéricas y esenciales de un buen servicio: *ser de utilidad, tener destinatario y sentido dinámico*.

Las tecnologías de descubrimiento de servicios dan la posibilidad de replantear las redes de comunicación, ya no sólo como meros canales de datos sino más bien como el centro de una comunidad de servicios que entran y salen potenciando la integración con gran independencia y simplicidad.

Para la fabricación de dispositivos con las características esperadas, son necesarios dos componentes Hardware y Software.

En el Hardware existen dos grandes líneas de trabajo que interesan a los desarrolladores: tecnologías de comunicación (especialmente inalámbricas) y el desarrollo de microprocesadores capaces de satisfacer los requerimientos de procesamiento y memoria. Este ámbito está muy avanzado y cada día se supera aún más en rendimiento e integración.

El Software está presente a través de los Protocolos de Descubrimiento de Servicios. Actualmente existen varios de ellos, aunque la mayoría en etapa de desarrollo. Todos son producto de Alianzas o Consorcios de: Compañías, Universidades, Investigadores, etc. El gran problema lo generan estos mismos, ya que cada uno pretende ser el estándar del mercado.

Así el desarrollo de dispositivos con las características del Descubrimiento de Servicios Computacionales Distribuidos no se ha limitado por el Hardware, sino más bien por la integración o estandarización en los Protocolos de Descubrimiento.

A pesar de no existir un estándar para diseñar un Protocolo de Descubrimiento, muchas características comunes son encontradas en ellos, dificultando las comparaciones. La opción respecto a cual de ellos se masificará, probablemente, pasará por una decisión estratégica y comercial.

La elección de una tecnología de descubrimiento, para un determinado proyecto, dependerá del tipo de servicio, características esperadas y plataforma o arquitectura del dispositivo en que se implantará.

El desarrollo del proyecto Vistazo ayudo a generar una visón realista de esta tecnología y documentación práctica para futuros desarrolladores.

Una clara definición de conceptos e investigación previa del funcionamiento del protocolo a utilizar, en un proyecto de Servicios Computacionales, ayuda en la programación.

El descubrimiento instantáneo de Servicios aún es dependiente de requerimientos en el cliente. Sin embargo la masificación de una tecnología de descubrimiento, incentivará a que los Sistemas Operativos cumplan de ante mano con ellos, logrando así satisfacer plenamente la ¨transparencia¨ al usuario.

Jini lleva consigo todas las grandes ventajas de un lenguaje estable, maduro y multifuncional como Java. Sin embargo sus múltiples características son dependientes de una correcta configuración de servicios que normalmente centralizan el sistema, existen alternativas de descentralización pero no son suficientemente difundidas.

Jini, es una propuesta con múltiples alternativas y recursos que hacen difícil la programación inicialmente, pero superada esa etapa entrega una gran cantidad de características adicionales aplicables a múltiples escenarios.

Es recomendable en un primer acercamiento a los protocolos de descubrimiento, utilizar una implementación, en un lenguaje de programación familiar para el desarrollador. Luego utilizar esa experiencia para aprender una arquitectura más amplia, como lo es Jini.

Vistazo, en su versión 2.9, es funcional y muy estable. Sin embargo, su utilización para proyección en tiempo real requiere la integración con algún software de proyección específico.

Todos los esfuerzos por conocer, entender y utilizar las tecnologías de Descubrimiento de Servicios, no importando cual de ellas, son una excelente preparación para enfrentar la pronta masificación de ellos, ya que la migración de muchos servicios como: aplicaciones, dispositivos y periféricos requerirán un conocimiento y experiencia acabado.

En definitiva, la implementación de Servicios Distribuidos es el siguiente paso lógico de la tecnología por integrarse de manera más natural y sin necesitar intervención del usuario, por medio de las tecnologías de descubrimiento, al quehacer diario de las personas.

Bibliografía

COMER D. (1996). Redes globales de información con INTERNET y TCP/IP : principios básicos, protocolos y arquitectura, Prentice-Hall Hispanoamericana (ed), México, 1996.

COULOURIS G., DOLLIMIRE J.,KINDBERG (2001). Distributed Sistems: Concepts and Desing, Addison-Wesley (ed), 3sd Edition 2001.

GONZALES J.A., CABEZA L., MARTINEZ J. (1994). Introducción a los Microcontroladores, McGraw-Hill (ed), 1994.

JAWORSKI J. (1999). Java 1.2 Al Descubierto, Prentice Hall (ed), Madrid 1999.

NEWMARCH J. (2000). A Programmer's Guide for JINI Technologies, APress (ed) , November 2000.

PASCOE B. (1999). Salutation Architectures and the newly defined service discovery protocols from Microsoft and Sun, Salutation Consortium (ed), 1999.

RODRIGUEZ P. (2003) Apuntes Curso Arquitectura Distribuida, Departamento Sistemas de Información, Universidad del Bío-Bío, 2003.

TANEMBAUM, VAN RENESSE (1985). Distributed Systems, Mullender (ed), 1985.

TEILHARD C., PIERRE T (1963). El Fenómeno Humano, Colección Ensayistas de hoy (ed), Madrid, 1963.

VAN STEEN M., TANENBAUM (2002). Distributed Systems: Principles and Paradigms, Prentice (ed), 2002.

VINTON G (2002). Entreviendo el Futuro de Internet, Novatica (ed), 2002.

Referencias WEB

Sitio Web Aula Datos Telefónica
http://www.auladatos.movistar.com
Fecha última visita: 17/06/03

Sitio Web Axis Communications
http://developer.axis.com/products/etrax100lx/index.html
Fecha última visita: 05/05/03

Sitio Web Compañía MAXIM
http://www.maxim-ic.com/quick_view2.cfm/qv_pk/3609
Fecha última visita: 06/05/03

Sitio Web Compañía Vicomsoft
http://www.vicomsoft.com
Fecha última visita: 28/04/03

Sitio Web Crecimiento de Internet Hosts a nivel mundial
http://www.redhucyt.oas.org/webesp/estesp01.htm
Fecha última visita: 22/04/03

Sitio Web Curso Computación ubicua y Ambiente inteligente: Tecnologías para el futuro
http://www.it.uc3m.es/cdk/curs/ami
Fecha última visita: 23/04/03

Sitio Web Diccionario De La Lengua Española.
http://buscon.rae.es/diccionario/drae.htm
Fecha última visita: 22/04/03

Sitio Web Domotica.net
http://www.domotica.net
Fecha última visita: 14/06/03

Sitio Web Entorno Inteligente
http://elhogarinteligente.8m.com
Fecha última visita: 23/04/03

Sitio Web Grupo JavaSpain
http://es.groups.yahoo.com/group/javaSpain/
Fecha última visita: 23/02/04

Sitio Web Homeplug powerline alliance
http://www.homeplug.org
Fecha última visita: 14/06/03

Sitio Web IEEE 802.11 DSSS: The Path To High Speed Wireless Data Networking
http://www.ydi.com/deployinfo/wp-80211-dsss.php
Fecha última visita: 08/05/03

Sitio Web IEEE Standars Associations
http://standards.ieee.org/getieee802/802.11.html
Fecha última visita: 08/05/03

Sitio Web Intel
"Acceso a redes continuo e ininterrumpido"
http://www.intel.com/es/home/trends/future/intelligentroaming.htm
Fecha última visita: 23/04/03

Sitio Web Jan Newmarch
http://jan.netcomp.monash.edu.au/
Fecha última visita: 23/02/04

Sitio Web Jini
http://www.jini.org
Fecha última visita: 25/04/03

Sitio Web Joan Arnedo Moreno
http://uoc.terra.es/art/uoc/arnedo0202/arnedo0202.html#links
Fecha última visita: 20/06/03

Sitio Web Lesson: Full-Screen Exclusive Mode API
http://java.sun.com/docs/books/tutorial/extra/fullscreen/index.html
Fecha última visita: 9/02/04

Sitio Web Lesson: Tutorial de UML
http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html
Fecha última visita: 20/02/04

Sitio Web Lesson: User Interfazs that Swing: A Quick Start Guide
http://java.sun.com/docs/books/tutorial/uiswing/mini/index.html
Fecha última visita: 11/02/04

Sitio Web List-NEWS.com
http://list-news.com/articles/99october/100599.html
Fecha última visita: 22/04/03

Sitio Web Microsoft's Universal Plug and Play (UPnP).
http://upnp.org
Fecha última visita: 25/04/03

Sitio Web Módulo VII - Ambiente Cliente/Servidor
http://www.unice.br/~anaclara/amb_cliserv.doc
Fecha última visita: 28/04/03

Sitio Web Plug and Play Technology
http://www.microsoft.com/hwdev/tech/pnp/default.asp
Fecha última visita: 22/04/03

Sitio Web Primera Conferencia P2P
http://www.openp2p.com/pub/a/p2p/2001/02/20/oram.html
Fecha última visita: 23/04/03

Sitio Web Proteus Technologies
http://www.proteus-technologies.com/jini/
Fecha última visita: 20/01/04

Sitio Web Proyecto Autopista Central
http://www.autopistacentral.cl/funcionamiento/formas_pago.php
Fecha última visita: 14/05/03

Sitio Web Salutation
htttp://www.salutation.org
Fecha última visita: 23/04/03

Sitio Web Service Location Protocol Project
http://www.srvloc.org
Fecha última visita: 23/04/03

Sitio Web Simple Object Access Protocol (SOAP)
http://www.w3.org/TR/SOAP/
Fecha última visita: 01/05/03

Sitio Web The Embedded Systems Sourcing
http://microcontroller.com
Fecha última visita: 01/05/03

Sitio Web The Java Web Services Tutorial
http://java.sun.com/j2ee/webservices/
Fecha última visita: 05/05/03

Sitio Web Transmeta
http://www.transmeta.com/why/index.html
Fecha última visita: 06/05/03

Sitio Web UDDI
http://www.uddi.org
Fecha última visita: 05/05/03

Sitio Web Web Services Description Language (WDSL)
http://www.w3.org/TR/wsdl/
Fecha última visita: 01/05/03

Sitio Web Wi-Fi Alliance
http://www.wi-fi.org
Fecha última visita: 07/05/03

Sitio Web Wireless Ready Aliance
http://www.wirelessready.org
Fecha última visita: 28/04/03

Sitio Web Wikipedia
http://es.wikipedia.org
Fecha última visita: 14/06/03

Sitio Web XML
http://www.xml.org
http://www.xml.com
Fecha última visita: 28/04/03

ANEXOS

Anexo A

Especificación de la Arquitectura JINI

Traducción del original "The Jini™ Architecture Specification" por Jorge García Ruiz.

Objetivos del sistema

Un sistema Jini es un sistema distribuido basado en la idea de grupos federados de usuarios y recursos requeridos por esos usuarios. El objetivo primordial es convertir la red en una herramienta flexible, fácil de administrar donde los recursos pueden ser encontrados por clientes humanos y computacionales. Los recursos pueden ser cualquier dispositivo hardware, programas software, o una combinación de ambos. El enfoque del sistema es para hacer a la red una entidad más dinámica que refleje de mejor manera la naturaleza dinámica de los grupos de trabajo habilitando la capacidad de agregar y eliminar servicios de forma flexible.

Un sistema Jini consiste en las siguientes partes:

- Un conjunto de componentes que provee una infraestructura para servicios federados en un sistema distribuido.
- Un modelo de programación que soporta y alienta la producción de servicios distribuidos fiables.
- Los servicios pueden hacerse parte de una federación del sistema Jini y ofrecer funcionalidad a cualquier otro miembro de la federación.

Aunque estas piezas son separables y distinguibles, están interrelacionadas, esto puede impedir distinguirlas en la práctica. Los componentes que constituyen la infraestructura tecnológica Jini hacen uso del modelo de programación tecnológica Jini, los servicios que residen dentro de la infraestructura también usan este modelo el cual está muy bien soportado por los componentes en la infraestructura.

Los objetivos finales del sistema abarcan un número de diferentes audiencias, esos objetivos incluyen lo siguiente:

- Habilitar usuarios para compartir servicios y recursos sobre una red.
- Proveer un fácil acceso a los recursos en cualquier lugar de la red permitiendo al usuario cambiarse de ubicación.
- Simplificar las tareas de construcción, mantenimiento y modificación de una red con dispositivos, software y usuarios.

El sistema Jini extiende el ambiente de aplicaciones Java desde una única maquina virtual a una red de máquinas. El ambiente de aplicación Java provee una buena plataforma computacional para la computación distribuida, ya que código y datos pueden moverse de una máquina a otra. El ambiente ha sido construido para ser seguro, permitiendo confidencialidad para ejecutar código obtenido desde otra máquina.

La marcada clasificación en Java permite identificar clases de un objeto a ser ejecutado en una máquina virtual incluso cuando el objeto no es originario de esa máquina. El resultado es un sistema en que la red soporta una configuración fluida de objetos que pueden moverse de un lugar a otro según se necesite y llamar a cualquier parte de la red para ejecutar su función.

La arquitectura Jini aprovecha esas características del ambiente de aplicación Java para simplificar la construcción de un sistema distribuido. La arquitectura Jini agrega mecanismos que permiten fluidez de todos los componentes en un sistema distribuido, extendiendo, al completo sistema de red, la facilidad de movimiento de los objetos.

La infraestructura tecnológica Jini provee mecanismos para unir o separar de una red dispositivos, servicios y usuarios. El Acoplamiento o separación de un sistema Jini son sucesos fáciles y naturales, muchas veces automáticos. Los sistemas Jini son lejos más dinámicos que los actuales grupos de red donde la configuración de una red es una función centralizada y realizada manualmente.

Adopción Ambiental

El sistema Jini une computadores y dispositivos computacionales en algo que al usuario parece un solo sistema. Esto supone la existencia de una red de velocidad de conexión razonable entre estos computadores y dispositivos. Algunos dispositivos requieren mayor ancho de banda y otros menor, monitores e impresoras son un ejemplo de estos extremos. Se asume que la latencia de la red es razonable. Se asume también que cada dispositivo habilitado para tecnología Jini tiene suficiente memoria y poder de procesamiento. Los dispositivos que no cumplen con los requerimientos de memoria y/o procesamiento pueden ser conectados a un sistema Jini, pero son controlados por otra pieza de hardware y/o software que presenta el dispositivo al sistema Jini y él contiene ambos, procesamiento y memoria. La arquitectura para dispositivos no equipados con una maquina virtual Java (JVM) está explicada en forma completa en la Especificación de la Arquitectura de dispositivos Jini.

La infraestructura tecnológica Jini es una tecnología centrada en Java. La arquitectura Jini obtiene mucha de esta simplicidad asumiendo que el lenguaje de programación Java es una implementación de lenguaje por componentes. La posibilidad de dinámicamente descargar y ejecutar código es central para un número de características de la arquitectura Jini. Sin embargo, el carácter de tecnología centrada en Java depende del ambiente de aplicación Java en lugar de el lenguaje de programación Java. Cualquier lenguaje de programación puede ser soportado por un sistema Jini, si este tiene un compilador que produzca bytecodes.

Conceptos claves

El propósito de la arquitectura Jini es aliar grupos de dispositivos y componentes software en un solo sistema dinámico distribuido. La alianza resultante provee simplicidad

de acceso, facilidad de administración y soporte distribuido que es proveído por un gran sistema monolítico pero manteniendo la flexibilidad, uniformidad de respuesta y control al igual que un computador personal o estación de trabajo.

La arquitectura de un único sistema Jini es designado a un grupo de trabajo. Los miembros de esta alianza aceptan concordar en nociones básicas de confianza, administración, identificación, y política. Esto es posible para aliar sistemas Jini en grandes organizaciones.

Servicios

El concepto más importante dentro de la arquitectura Jini es por supuesto un servicio. Un servicio es una entidad que puede ser usada por una persona, un programa, u otro servicio. Un servicio puede ser computacional, de almacenamiento, un canal de comunicación a otro usuario, un filtro de software, un dispositivo hardware, u otro usuario. Dos ejemplos de servicios son impresión de un documento y conversión de un formato de procesador de texto a algún otro.

Los miembros de un sistema Jini crean federaciones para distribuir acceso a servicios. Un sistema Jini no debería entenderse como un conjunto de clientes y servidores, usuarios y programas, o incluso programas y archivos. Más bien, un sistema Jini consiste de servicios que son unidos para realizar una tarea particular. Los servicios pueden hacer uso de otros servicios, y un cliente de un servicio puede ser el mismo un servicio con sus propios clientes. La naturaleza dinámica de un sistema Jini permite agregar o quitar servicios a una federación en cualquier momento según la demanda, necesidad o los cambios en los requerimientos de un grupo de trabajo que usa el sistema.

Jini provee mecanismos para la construcción de servicios, lookup, comunicación, y uso en un sistema distribuido. Ejemplos de servicios incluyen: dispositivos como

impresoras, monitores, o discos; software como aplicaciones o utilidades, información como base de datos y archivos; y usuarios de un sistema.

Los servicios en un sistema Jini se comunican usando un protocolo de servicio, que es un conjunto de interfazs escritas en el lenguaje de programación Java. El conjunto de protocolos es abierto. La base del sistema Jini especifica un pequeño número de protocolos que definen la interacción crítica de servicio.

Servicio Lookup

Los servicios son encontrados y resueltos por un servicio lookup. El servicio lookup es el mecanismo de unión principal entre el sistema y los proveedores, el mayor punto de contacto entre el sistema y usuarios de él. En términos precisos, un servicio lookup traza interfazs indicando la funcionalidad de un servicio para un conjunto de objetos que implementan el servicio. Además describe entradas asociadas a un servicio permitiendo una selección más detallada de servicios basados en propiedades comprensibles por la gente.

Los objetos en un servicio lookup pueden incluir otros servicios lookup; esto provee una jerarquía de lookup. Más allá, un servicio lookup puede contener objetos que encapsulan otros directorios de nombres o servicios, siendo una especie de puente entre un lookup Jini y otras formas de servicios lookup. De seguro, pueden ser encontradas referencias a servicios Jini en otros directorios de nombres y servicios, proveyendo un medio para que clientes de esos servicios obtengan acceso a los sistemas Jini.

Un servicio es agregado a un servicio lookup por intermedio de un par de protocolos llamados descubrimiento y suscripción, primero el servicio localiza un apropiado servicio lookup (usando el protocolo de descubrimiento), y entonces este se suscribe a él (usando el protocolo de suscripción).

Método de Invocación Remota Java (RMI)

La comunicación entre servicios se logra usando método invocación remota Java (RMI). La infraestructura que sustenta la comunicación entre servicios no es en sí misma un servicio que es descubierto y usado por estos, más bien, una parte de la infraestructura tecnológica Jini. RMI provee mecanismos para buscar y activar grupos de objetos.

Fundamentalmente, RMI es un lenguaje de programación Java que ha extendido los mecanismos de llamada a procedimientos remotos. RMI no solo permite pasar datos de un objetos a otro a través de una red, sino también objetos completos incluyendo código. Mucha de la simplicidad del sistema Jini se hace posible por la habilidad de mover código a través de la red encapsulado como un objeto.

Seguridad

El diseño del modelo de seguridad para la tecnología Jini es sustentado por dos conceptos un director y una lista de control de acceso. Los servicios Jini son accesados de parte de alguna entidad, el director, que generalmente rastrea los pasos de un usuario particular del sistema. Los servicios, por si mismos, pueden solicitar acceso a otros servicios basados en la identidad de el objeto que implementa el servicio. El acceso a un servicio depende del contenido de acceso de una lista de control asociada con el objeto.

Arriendo

El acceso a muchos de los servicios en el ambiente de sistema Jini está basado en arriendo. Un arriendo es una concesión de una garantía de acceso por un periodo de tiempo. Cada arriendo es negociado entre el usuario del servicio y el proveedor de el servicio como parte del protocolo servicio: Un servicio es pedido por algún periodo; el acceso es garantizado por algún periodo, presumiblemente tomando el periodo de la petición en

cuenta. Si el arriendo no es renovado antes, el servicio es liberado, cualquiera sea la causa del porque el recurso ya no se necesita, el cliente o la red falla, o el arriendo no permite renovación, entonces ambos el usuario y proveedor de el recurso pueden concluir que puede ser liberado.

Los arriendos son exclusivos o no exclusivos. Los arriendos exclusivos aseguran que otro no puede tomar un arriendo de un recurso durante el periodo pactado. Los arriendos no exclusivos permiten que múltiples usuarios compartan un recurso.

Transacciones

Una serie de operaciones, relacionadas a uno o múltiples servicios, pueden ser encapsuladas en una transacción. La interfaz de transacción Jini provee un protocolo de servicio necesario para coordinar un compromiso de dos fases. La misma noción de semántica de transacciones se usa esas interfazs.

Eventos

La arquitectura Jini apoya eventos distribuidos. Un objeto puede permitir a objetos registrar interés en eventos del objeto y recibir una notificación de la ocurrencia de tal evento. Esto permite a los programas basados en eventos distribuidos ser escritos con una variedad de garantías de fiabilidad y escalabilidad.

Apreciación de Componentes

Los componentes de un sistema Jini pueden ser segmentados en tres categorías: infraestructura, modelo de programación, y servicios. La infraestructura es un conjunto de componentes que permiten construir una federación de sistema Jini, mientras que los

servicios son las entidades dentro de la federación. El modelo de programación es un conjunto de interfazs que permiten la construcción de servicios fiables, incluyendo aquellos que son parte de la infraestructura y aquellos que se unen en una federación.

Estas tres categorías, sin embargo distintas y separables, están enlazadas a tal grado que la distinción entre ellas puede ser borrosa. Es más, esto hace posible construir sistemas que tienen alguna de las funcionalidades de un sistema Jini con variantes en las categorías o sin los tres componentes. Pero un sistema Jini gana este completo poder porque un sistema construido con la infraestructura particular y modelos de programación descritos están basados en una noción de servicio. Separar los segmentos dentro de la arquitectura permite que el código original pueda ser cambiado en forma mínima para tomar parte en un sistema Jini. No obstante, el completo poder de un sistema Jini estará disponible sólo para nuevos servicios que son construidos usando el modelo integrado.

Un sistema Jini puede ser visto como una extensión de la infraestructura de red, modelo de programación, y servicios que hacen un exitoso uso de la tecnología Java. Estas categorías en los correspondientes componentes de la familia del ambiente de Aplicación Java son mostrados en la siguiente tabla:

	Infraestructura	Modelo de programación	Servicios
Base Java	Java VM RMI Seguridad Java	Java APIs JavaBeans ...	JNDI Enterprise Beans JTS ...
Java + Jini	Discovery/Join Seguridad Distribuida Lookup	Arrendamiento Transacciones Eventos	Impresión Administrador de transacción JavaSpaces Service ...

Tabla: Segmentación de la arquitectura Jini

Infraestructura

La infraestructura de la tecnología Jini define la mínima esencia de la tecnología Jini. La infraestructura incluye lo siguiente:

- Un sistema de seguridad distribuido, integrado en RMI, que extiende el modelo de seguridad de la plataforma Java al mundo de los sistemas distribuidos.
- Los protocolos de descubrimiento y suscripción, los protocolos de servicios (hardware y software) para descubrir, volverse parte de, y anunciar servicios proporcionados a otros miembros de la federación.
- El servicio lookup, que sirve como un repositorio de servicios. Las entradas en el servicio lookup son objetos escritos en el lenguaje de programación Java; estos objetos pueden ser descargados como parte de una operación lookup y actuar como un proxy local para el servicio que coloca el código dentro en un servicio lookup.

Los protocolos de descubrimiento y suscripción definen la manera en que un servicio de cualquier tipo puede llegar a ser parte de un sistema Jini; RMI define el lenguaje base que permite a la tecnología Jini comunicar servicios; el modelo de seguridad distribuido y estas implementaciones definen como son identificadas las entidades y como ellas obtienen derechos para realizar acciones en su propio nombre y en nombre de otros; y el servicio lookup refleja los miembros actuales de la federación y actúa como un mercado central ofreciendo y buscando servicios para miembros de la federación.

Modelo de programación

La infraestructura principal habilita el modelo de programación y hace uso de este. Las entradas en el servicio lookup son arrendadas, permitiendo al servicio lookup reflejar perfectamente el conjunto de servicios actualmente disponibles. Cuando los servicios se

suscriben o abandonan un servicio lookup, eventos son señalizados, y los objetos que han registrado su interés en esos eventos reciben notificaciones cuando un nuevo servicio se habilita o antiguos servicios dejan de estar activados. El modelo de programación descansa en la capacidad de mover código que es soportado por la infraestructura base.

Ambos la infraestructura y los servicios que usan esa infraestructura son entidades computacionales que existen en el ambiente físico de un sistema Jini. Sin embargo, los servicios también constituyen un conjunto de interfazs que definen protocolos de comunicación que pueden ser usados por los servicios y la infraestructura para comunicarse entre ellos mismos.

Estas interfazs, tomadas juntas, constituyen la extensión distribuida del estándar modelo de programación Java que constituye el modelo de programación Jini. Entre las interfazs que constituyen el modelo de programación Jini están los siguientes:

- La interfaz de arrendamiento, que define la manera de asignar y liberar recursos usando un modelo de duración renovable.
- Interfazs de evento y notificación, que son una extensión del modelo de eventos usado por componentes JavaBeans para el ambiente distribuido, permite una comunicación basada en eventos entre servicios de tecnología Jini.
- Interfazs de transacción, que permiten a entidades cooperar de tal manera que todos los cambios que hicieron al grupo ocurren atómicamente o ninguno de ellos ocurre.

La interfaz de arriendo extiende el modelo del lenguaje de programación Java agregando tiempo a la noción de pertenencia referida a un recurso, permitiendo reclamar referencias sin riesgo en caso de una falla de red.

Las interfazs de evento y notificación extienden los modelos estándares de eventos usados por componentes JavaBeans y el ambiente de aplicación Java para casos distribuidos, permitiendo ser manejados por terceros objetos mientras realizan varias entregas. El modelo también reconoce que la entrega de un notificación distribuida puede demorarse.

La interfaz transacción introduce un protocolo orientado a objeto que permite a aplicaciones que usan tecnología Jini coordinar cambios de estado. El protocolo de transacción provee dos pasos para coordinar las acciones de un grupo distribuido de objetos. El primer paso es llamado la fase de votación, en la que cada objeto "vota" si ha completado esta porción de la tarea y está listo para comprometer cualquier cambio hecho. En el segundo paso, un coordinador emite una petición "commit" para cada objeto.

El protocolo de transacción Jini difiere de otras interfazs de transacción en que este no asume que las transacciones ocurren en un sistema de procesamiento de transacciones. Tales sistemas definen mecanismos y requerimientos de programación que garantizan la correcta implementación de una semántica transaccional particular. El protocolo de transacción Jini toma una mirada tradicional de la orientación a objeto, dejando la correcta implementación de la deseada semántica transaccional a la implementación de los objetos particulares que son envueltos en una transacción. La meta del protocolo de transacción es definir las interacciones que tales objetos deben coordinar con tales grupos de operaciones.

Las interfazs definidas en el modelo de programación Jini son usadas por la infraestructura de componentes en el lugar apropiado y por los servicios iniciales de tecnología Jini. Por ejemplo, el servicio lookup hace uso del arrendamiento e interfaz de eventos. El arrendamiento asegura que los servicios registrados continúan estando disponibles, y los eventos ayudan a los administradores a descubrir problemas y dispositivos que necesitan configuración. El servicio JavaSpaces, un ejemplo de servicio de tecnología Jini, utiliza arrendamiento y eventos, y también soporta protocolo de transacción Jini. El administrador de transacciones puede ser usado para coordinar la fase de votación de una transacción para aquellos objetos que soportan protocolo transacción.

La implementación de un servicio no requiere usar el modelo de programación Jini, pero tales servicios necesitan usar el modelo para interactuar con la infraestructura tecnológica Jini. Por ejemplo, cada servicio interactúa con el servicio lookup Jini usando el modelo de programación; y si un servicio ofrece recursos en base al arrendamiento o no, el registro del servicio con el servicio lookup estará arrendado y necesitará una periódica renovación.

La unión del modelo de programación a los servicios y la infraestructura está hecha como una federación de un sistema Jini no es solo una colección de servicios y protocolos. La combinación de infraestructura, servicio, y modelo de programación, todo diseñado para trabajar junto y construido para usar cada otro, simplifica el conjunto del sistema y unifica este de manera que lo hace fácil de entender.

Servicios

La infraestructura tecnológica de Jini y el modelo de programación están diseñadas para permitir ofrecer y encontrar servicios en una federación de red. Estos servicios hacen uso de la infraestructura para hacer llamadas entre sí, para descubrirse, y para anunciar su presencia a otros servicios y usuarios.

Los servicios aparecen programados como objetos escritos en lenguaje de programación Java, quizás compuesto por otros objetos. Un servicio tiene una interfaz que define las operaciones que pueden ser solicitadas. Algunas de estas interfazs están pensadas para ser usadas por programas, mientras otras están pensadas para ser ejecutadas por el receptor para que el servicio pueda actuar recíprocamente con un usuario. El tipo de servicio determina las interfazs y también define el conjunto de métodos que pueden ser usados para acceder al servicio. Un servicio puede ser implementado usando otros servicios.

Ejemplos de Servicios disponibles con tecnología Jini :

- Un servicio de impresión, que puede imprimir desde aplicaciones escritas en Java así como “legacy applications” .

- Un servicio JavaSpaces, que puede ser usado para comunicación simple y para almacenar grupos de objetos relacionados escritos en Java.

- Administrador de transacciones, permite a grupos de objetos participar en el protocolo de transacción definido por el modelo de programación.

Arquitectura de Servicio

La servicios forman la base interactiva de un sistema Jini, ambos a la programación y los niveles de interfaz usuario. Los detalles de la arquitectura de servicio son lo mejor para entender cuando los protocolos de descubrimiento y lookup de Jini están presentes.

Protocolos de Descubrimiento y Lookup

El corazón de un sistema Jini es un trio de protocolos llamados descubrimiento, suscripción y lookup. Un par de estos protocolos, descubrimiento y suscripción, ocurren cuando un dispositivo es conectado. Descubrimiento ocurre cuando un servicio está buscando un servicio para registrarse. Suscripción ocurre cuando un servicio localizó un servicio lookup y desea suscribirse en este. Lookup ocurre cuando un cliente o usuario necesita localizar e invocar un servicio descrito por su interfaz (escrita en Java) y posiblemente otros atributos. Vea Figura AR.22 bosquejo del proceso de descubrimiento.

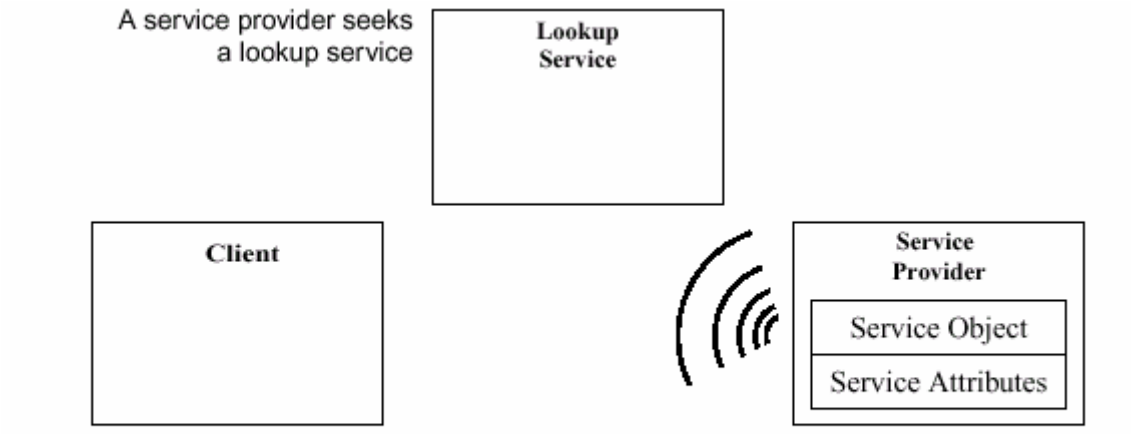


FIGURE AR.2.2: ***Discovery***

Descubrimiento / Suscripción Jini es el proceso de agregar un servicio a un sistema Jini. Un servicio proveedor es quien origina el servicio, un dispositivo o software, por ejemplo.

Primero, el servicio proveedor localiza un servicio lookup a través de una petición multicast en la red local para cualquier servicio lookup en que pueda identificarse (figura AR.2.2). Entonces, un objeto servicio es cargado en el servicio lookup (figura AR.2.3). Este objeto servicio contiene la interfaz para el servicio, incluidos los métodos que usuarios y aplicaciones invocarán para ejecutar el servicio junto con cualquier otro atributo descriptivo.

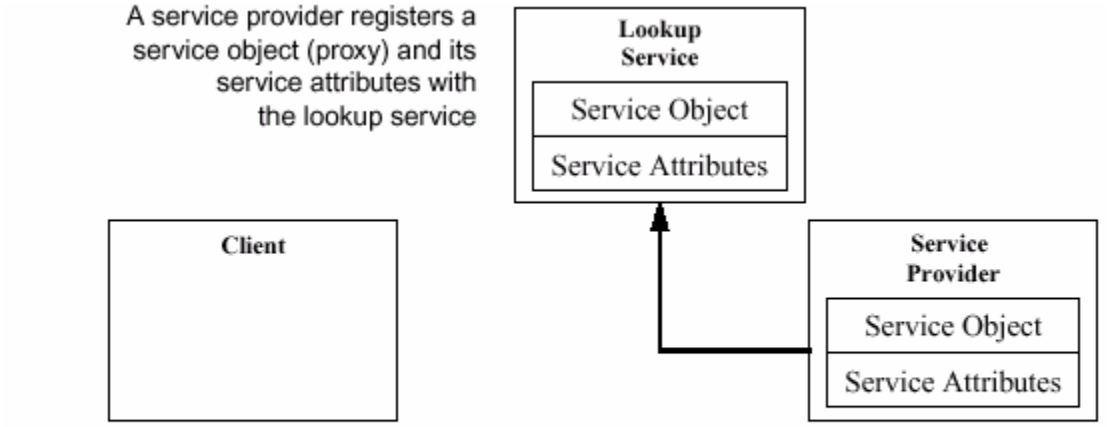


FIGURE AR.2.3: ***Join***

Los servicios deben ser capaces de buscar un servicio lookup; sin embargo, un servicio puede delegar las tareas de buscar un servicio lookup a una tercera parte. El servicio ahora está listo para ser visto y usado, tal como lo muestra el siguiente diagrama (Figura AR.2.4).

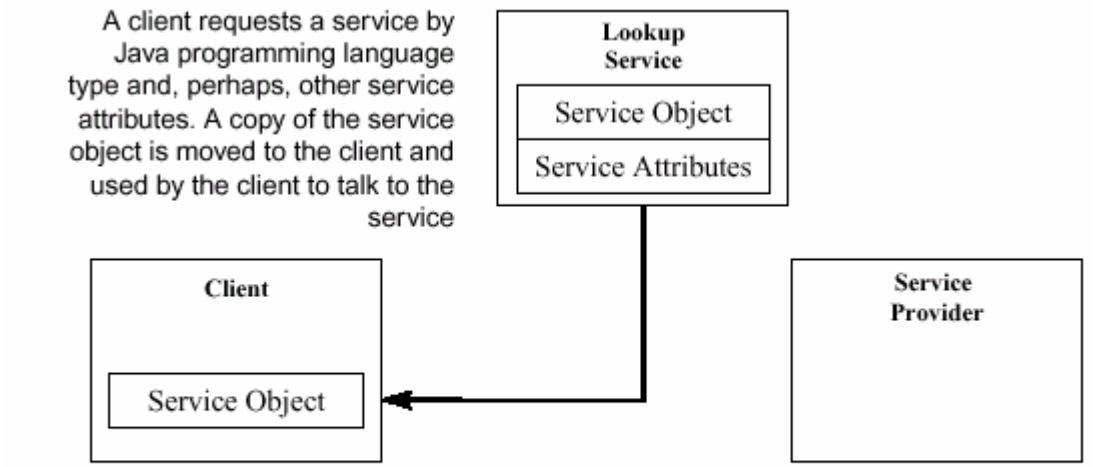


FIGURE AR.2.4: ***Lookup***

Un cliente localiza un apropiado servicio para este tipo, es decir por la interfaz escrita en Java, con los atributos descriptivos que son usados en una interfaz de usuario para el servicio lookup. El objeto servicio es cargado en el cliente.

La etapa final es invocar el servicio, como lo muestra el siguiente diagrama (figura AR.2.5):

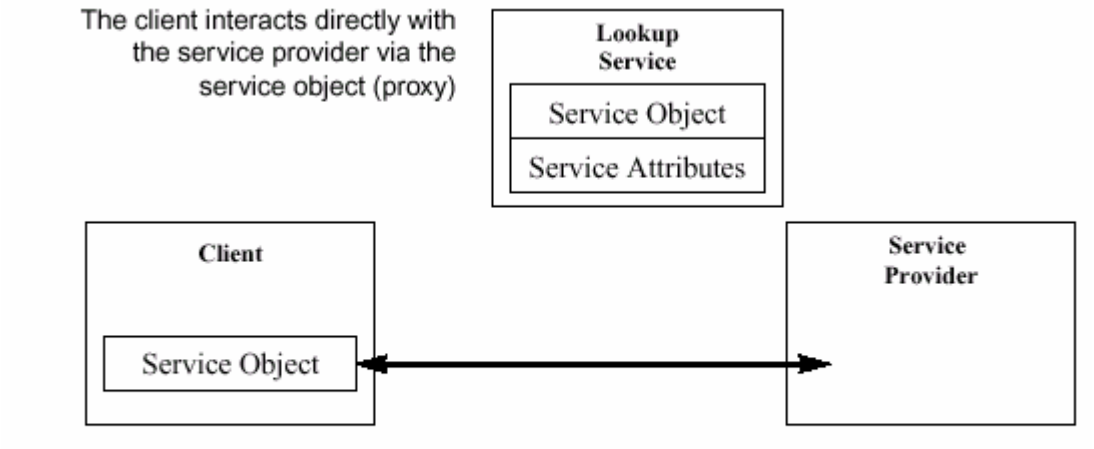


FIGURE AR.2.5: ***Client Uses Service***

Los métodos del objeto servicio pueden implementar un protocolo privado entre este y el servicio original proveído. Diferentes implementaciones de la misma interfaz de servicio pueden usar protocolos de interacción completamente diferentes.

La habilidad de mover objetos y código desde el servicio proveedor a el servicio lookup y desde este al cliente de el servicio da al servicio proveedor mucha libertad en el patrón de comunicación entre el servicio y estos clientes. Este movimiento de código también asegura que el objeto servicio sostenido por el cliente es siempre un proxy sincronizado porque el objeto servicio es proporcionado por el mismo servicio. El cliente solo sabe que este es distribuido con una implementación de una interfaz escrita en el leguaje de programación Java, así el código que implementa la interfaz puede hacer lo que

sea necesario para proveer el servicio. Porque este código viene originalmente del mismo servicio, el código puede tomar ventaja de los detalles de implementación del servicio que solo es conocido en el código.

El cliente interactúa con un servicio vía conjunto de interfazs escritas en Java. Estas interfazs definen un conjunto de métodos que pueden ser usadas para interactuar con el servicio. Interfazs programáticas son identificadas por el tipo de sistema, y servicios pueden ser encontradas en un servicio lookup, preguntando por aquellos que soportan una interfaz particular. Buscando un servicio de esta manera, asegura que el programa buscado para el servicio sabrá como usarlo, porque este uso es definido por el conjunto de métodos que están escritos para el tipo.

Interfazs programáticas pueden ser implementadas como referencias RMI a el objeto remoto que implementa el servicio o como computación local esto provee todos los servicios localmente, o como una combinación. Tales combinaciones, llamadas proxies inteligentes, implementan algunas de las funciones de un servicio localmente y el resto a través de llamadas remotas a una implementación centralizada del servicio.

Una interfaz de usuario puede también ser almacenada en un servicio lookup como un atributo de un servicio registrado. Una interfaz de usuario almacenada en un servicio lookup es una implementación que permite manipular directamente el servicio por un usuario del sistema.

En efecto, una interfaz de usuario para un servicio es una forma especializada de la interfaz servicio que permite a un programa, como un navegador, una manera opcional permitiendo al usuario humano interactuar directamente con un servicio.

En situaciones que el servicio lookup no es encontrado, un cliente puede usar en cambio una técnica llamada peer lookup. En estas situaciones, el cliente puede enviar el mismo paquete de identificación que es usado por un servicio lookup para pedir proveedores de servicio a registrar. Los servicios proveedores intentará entonces registrarse con el cliente como si este fuera un servicio lookup. El cliente puede seleccionar los

servicios que necesita desde las peticiones de registro recibidas en respuesta y dejar o rechazar el resto.

Implementación de Servicio.

Los objetos que implementan un servicio puede diseñarse para correr en un solo espacio de dirección con otros objetos sobre todo cuando hay cierta situación o requerimientos de seguridad. Tales objetos constituyen un grupo objeto. Un grupo objeto es garantía de que siempre los objetos en ejecución residen en una único espacio de dirección o maquina virtual. Los objetos que no están en el mismo grupo objeto están incomunicados entre sí normalmente, ya que se ejecutan en diferentes espacios de memoria o maquinas virtuales.

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

Desde el punto de vista del servicio cliente, no hay una distinción entre servicios que son implementados por máquinas diferentes, los servicios son descargados en un espacio de dirección local, y estos son implementados en el hardware. Todos estos servicios aparecen disponibles en la red como objetos escritos en Java. Un tipo de implementación podría ser reemplazado por otro tipo de implementación sin que el cliente conozca los cambio.

Anexo B

Implementación de Universal Plug and Play

Traducción oficial de "Understanding Universal Plug and Play" revisada por Jorge García Ruiz.

Sinopsis

Este documento ofrece una descripción general de Plug and Play universal (UPnP) y cómo funciona. Se incluyen escenarios en donde UPnP mejora la experiencia total de operación en red a través del descubrimiento automático e interoperabilidad de dispositivos. Se ofrecen detalles adicionales sobre los componentes, protocolos y procedimientos utilizados en UPnP, enfocándonos sobre cómo los protocolos estándares existentes se utilizan para crear dispositivos UPnP. Después, le indicaremos en dónde puede encontrar mayor información para incrementar su conocimiento sobre UPnP.

Introducción

¿Qué es UPnP?

Con la adición de las capacidades Plug and Play (PnP) para dispositivos al sistema operativo, ahora es mucho más fácil instalar, configurar y agregar periféricos a una PC. Plug and Play universal (UPnP) amplía esta facilidad de uso para incluir toda la red, habilitando el descubrimiento y control de dispositivos y servicios en red, tales como impresoras en red, conexiones de Internet y equipo electrónico del consumidor.

UPnP es más que sólo una extensión sencilla del modelo Plug and Play para periféricos. Está diseñado para soportar cero configuración, operación en red “invisible” y descubrimiento automático para una amplia gama de dispositivos de una extensa variedad de proveedores.

Con UPnP, un dispositivo puede unir dinámicamente una red, obtener una dirección IP, transportar sus capacidades y aprender sobre la presencia y capacidades de otros dispositivos todo automáticamente, habilitando verdaderamente redes de cero configuración. Los dispositivos pueden comunicarse más tarde entre ellos directamente, habilitando aún más una operación en red.

La variedad de dispositivos que se pueden beneficiar de una red habilitada por UPnP es amplia e incluye aparatos inteligentes, dispositivos inalámbricos y PCs de todo tipo.

El alcance de UPnP es suficientemente amplio para abarcar varios escenarios existentes, así como algunos nuevos y emocionantes, incluyendo automatización del hogar, impresión e imágenes, entretenimiento de audio / video, aparatos de cocina, redes de automóviles y redes disponibles en lugares públicos.

UPnP utiliza los protocolos estándares de TCP/IP e Internet, lo que le permite ajustarse de manera transparente a las redes existentes. Utilizar estos protocolos estandarizados permite a UPnP beneficiarse de una gran riqueza de experiencia y conocimiento y hace de la interoperabilidad una función inherente.

Debido a que UPnP es una arquitectura de red distribuida y abierta, definida por los protocolos utilizados, es independiente de cualquier sistema operativo en particular, lenguaje de programación o medio físico (como el caso de Internet). UPnP no especifica cuáles APIs de aplicaciones utilizará, permitiendo a los proveedores de sistemas operativos crear las APIs que cumplirán las necesidades de sus clientes.

¿Quién define UPnP?

El Foro universal de Plug and Play define las descripciones de dispositivos y servicio UPnP (originalmente llamados Protocolos de control de dispositivos o DCPs) de acuerdo a una arquitectura de dispositivos comunes aportada por Microsoft. El Foro universal Plug and Play es un grupo de compañías y personas de la industria que intentan desempeñar un papel líder en la autoría de especificaciones para los dispositivos y servicios UPnP. Constituida el 18 de octubre de 1999, es una asociación de más de 340 proveedores líderes en la industria de aparatos electrónicos para consumidores, computación, automatización y seguridad para el hogar, aparatos para el hogar, operación en red de computadoras y dispositivos portátiles.

Los objetivos del Foro son permitir el surgimiento de dispositivos fácilmente conectados y simplificar la implementación de las redes en el hogar y en los ambientes corporativos. El Foro logrará esto definiendo y publicando las descripciones del dispositivo y servicio UPnP desarrolladas en estándares de comunicación abiertos, basados en Internet. El sitio Web del Foro, http://upnp.org/, es el receptor central para el esquema que ha sido desarrollado y estandarizado por el Foro. Además, el sitio incluye el documento de la arquitectura del dispositivo, las plantillas para el dispositivo y las descripciones de servicio, así como los lineamientos para el diseño de la descripción del dispositivo y servicio. UPnP.org también distribuye información sobre las actividades y avances del Foro.

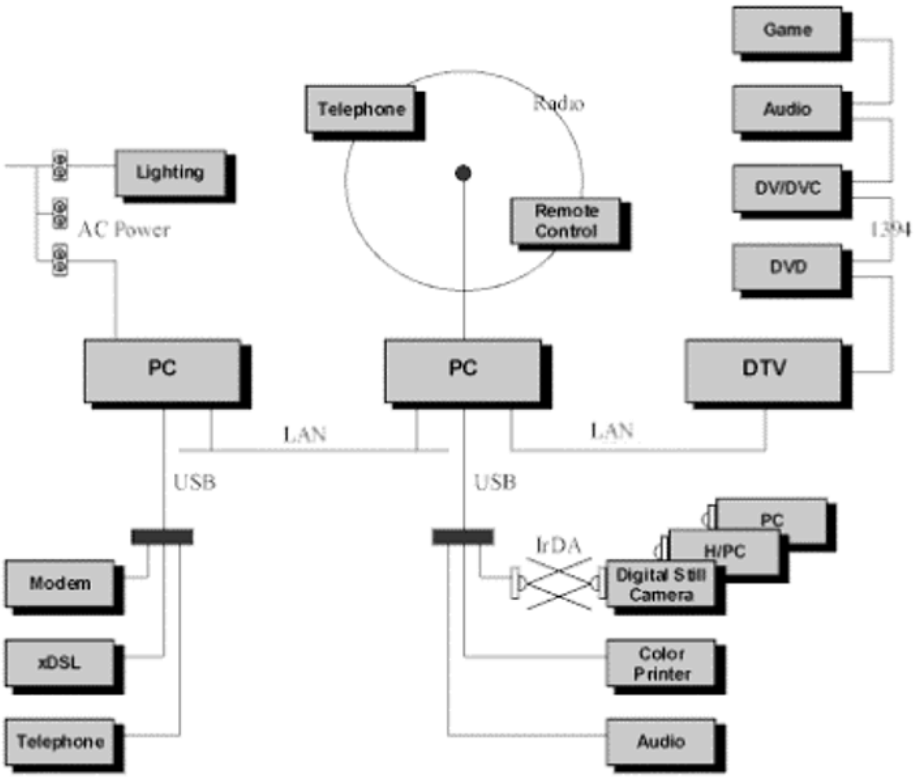


Ilustración 1: Una red de distintos proveedores con varios medios combinados

Escenarios habilitados para UPNP

UPnP abre un mundo de posibilidades

Existen varias maneras en que el descubrimiento automático y el control de dispositivos pueden lograr que la vida sea mejor y más fácil. La llegada de UPnP está abriendo nuevas posibilidades y usos todos los días. Nos atreveríamos a decir que la explosión de posibilidades no se ha igualado desde la llegada de las redes de cómputo. Los escenarios parecen ser ilimitados. Hemos incluido tan sólo algunas de muchas ideas en esta sección.

Algunos escenarios de UPnP:

Uso compartido de conexión a Internet

Juan navega la Web principalmente en la PC de su despacho en casa, que cuenta con una conexión DSL siempre activada y está protegida a través de un firewall. Le gustaría agregar un aparato WebTV a su sala de estar y una conexión MSN en su cocina para que sus papás y su esposa naveguen la Web desde sus lugares preferidos. Los dispositivos WebTV y MSN podrán descubrir y utilizar el servicio de uso compartido de la conexión a Internet UPnP desde su PC con Windows XP en una red HomePNA.

Monitoreo de bebés y cámara de vigilancia

Susana colocó dos pequeñas cámaras de monitoreo en las habitaciones de sus hijos para poder ver cómo están y asegurarse de que están haciendo su tarea, mientras ella trabaja en su PC. También colocó una a la entrada de la casa para ver quién llama a la puerta. A través de su red en el hogar, Susana también puede ver cualquiera de estas cámaras a través de su TV si se encuentra en la sala.

Sincronice sus relojes

Sucede todo el tiempo. Se va la luz por un par de horas y todos los relojes en su casa parpadean a las 12:00. Recorrer toda la casa y ponerlos de nuevo a la hora es un inconveniente y lleva tiempo.

Introduzca UPnP. Un script que se ejecuta en su PC con Windows XP que sincroniza periódicamente todos los relojes en la casa. El script busca todos los servicios de reloj en la red, sin importar el dispositivo. Después, simplemente interactúa con cada servicio y cambia la hora al valor que obtuvo del reloj por Internet.

Este script se puede ejecutar como una tarea que se repite todos los días a media noche. Se puede ejecutar de manera manual después de que vuelve la luz para reconfigurar cada reloj o después de que cambie el horario de verano.

La nueva impresora

Si hoy llegara a casa con una nueva impresora para utilizarla en la red de su casa, tendría que tomar varios pasos para asegurarse de que la impresora funcione con todas las PC en la red y hacerla disponible para cualquier otro dispositivo que tenga la necesidad de imprimir.

Por ejemplo, tendría que conectar la impresora a una PC existente, cargar el driver del dispositivo, compartir la impresora en la red, después ir a cada una de las otras PCs en la red y conectarlas a esa impresora compartida.

¿No sería maravilloso si sólo tuviera que conectar la impresora a cualquier puerto de la red disponible, ya sea una línea telefónica, enchufe o Ethernet y lograr que la impresora estuviera disponible inmediatamente para todos los dispositivos y usuarios de la red?

Sin espacio de almacenamiento nuevamente

Como de costumbre, y cada vez es más seguido, nos quedamos sin espacio de almacenamiento en nuestras PCs. Supongo que esto tiene que ver con la gran cantidad de imágenes digitales y películas que tomo o posiblemente porque he hecho un gran esfuerzo para catalogar toda mi colección de CDs como archivos de Windows Media™, pero por lo pronto no veo cómo puede mejorar esta situación.

Aunque los precios de almacenamientos secundarios han bajado dramáticamente, el dolor de agregar un disco duro a un sistema sólo ha mejorado marginalmente a través del Plug and Play de dispositivos. Peor aún es intentar aprender cómo pasar esas películas, fotografías y sonidos a través de aduanas cuando voy a visitar a mi familia durante las vacaciones.

Supongamos que tuviera un almacén de datos de alta velocidad, de alta capacidad y móvil habilitado por UPnP. Tal vez este dispositivo conectado directamente a una red Ethernet o línea telefónica. O, para permitir el flujo de mis películas familiares, el dispositivo podría estar conectado al bus IEEE 1394 de alta velocidad con el resto de mi sistema de entretenimiento. Siempre que conecto este dispositivo, todos los demás dispositivos en la red (incluyendo aquellos que producen o muestran medios) saben inmediatamente de su disponibilidad.

¡No estaría mi familia encantada cuando los visito con películas que parecen eternas!

Los nuevos dispositivos portátiles no están limitados a almacenamiento o impresoras, sino que pueden incluir otros dispositivos, incluyendo cámaras, reproductores MP3, scanners, dispositivos MIDI, controles remotos, TVs y dispositivos de video. La lista es interminable.

"¡Siempre está frío cuando me levanto!"

Usted está en la cama a punto de irse a dormir y mañana es sábado. La alarma de su reloj normalmente lo despierta a las 7 AM, pero mañana quiere dormir un poco más. Necesita que una alarma lo despierte y las 9 AM parece como una mejor hora. Pone la alarma a las 9 AM en lugar de las 7 AM.

Debido a que cuenta con un reloj despertador UPnP, todo es sencillo. Cuenta con un script que se ejecuta en su PC basada en Windows XP® que espera la notificación de la alarma del reloj despertador. Tan pronto como esto sucede, instruye al cronómetro en el dispositivo HVAC que ponga la hora para levantarse a la misma hora que la del reloj despertador.

Ahora, ¡su calefacción se encenderá lo suficientemente temprano para que no se congele cuando se pare de la cama! Con un sistema HVAC inteligente, se podrían agregar muchas otras funciones más allá de lo que proporcionan los termostatos programados de hoy. Esto podría incluir habilitar el sistema cuando los sensores detectan personas y el control remoto del sistema por Internet.

Si su reloj despertador tuviera acceso a su información programada, le podría advertir en caso de que hubiera elegido una hora después de una cita programada. En otras palabras, si tuviera una reunión a las 9 AM, e intentara poner la alarma a las 9 AM, le advertiría que en realidad tiene que levantarse a las 8 AM para poder llegar al trabajo a las 9.

El interruptor maestro

Regresa a casa después de un día arduo de trabajo y camina hacia la puerta principal. Gira un interruptor de pared que, para la mayoría de nosotros, sólo enciende la

luz del vestíbulo. Con UPnP, este interruptor es mucho más que eso. Este interruptor es sencillamente un servicio UPnP cuyo status está definido por una variable llamada "Posición". Cuando gira el interruptor, la "Posición" cambia a "Encendido" y un script que se ejecuta en su PC con Windows XP entra en acción.

El script recibe la notificación de que la posición del interruptor maestro ha cambiado a "encendido". Después, hace lo siguiente:

- La calefacción se enciende a su temperatura programada.
- La máquina empieza a reproducir mensajes nuevos.
- Su estéreo se enciende y sintoniza su estación de música clásica favorita y el volumen se programa a nivel ambiental.
- Las persianas eléctricas de la ventana se suben, pero sólo después de la puesta del sol (los datos de puesta y salida de sol se obtienen fácilmente de Internet).
- Opcionalmente, la TV se enciende en el canal de noticias, sin sonido y con la función de subtítulos encendida.
- Por supuesto, la luz en el vestíbulo también se enciende.

De manera similar, cuando cambia el interruptor a la posición de "apagado", ocurre lo contrario:

- La calefacción se apaga (o se programa a una temperatura menor).
- El estéreo se apaga.
- La TV se apaga.
- Las persianas eléctricas de la ventana se bajan para tener una mayor privacidad.

- Todas las luces en la casa se apagan.

Para terminar

El hijo se encontraba haciendo su tarea y alistándose para imprimir, por lo que pensó que su vida seria más fácil si movía la impresora a su habitación. No pensó que su papá se daría cuenta. Sin embargo, en la computadora de su papá apareció un cuadro de diálogo tan pronto como el hijo desconectó la impresora.

Mientras que esto lo pudo haber hecho a través de eventos, en este caso, fue parte del protocolo de descubrimiento. Cuando un dispositivo sale de la red, envía un anuncio de que ya no está conectado. Como resultado, todos los puntos de control tendrán un conocimiento perfecto del status de la red.

Bien, la cena se ha terminado, la película también y papá y mamá están sentados en el sofá. Mamá nota que la portátil de papá sigue encendida cuando él debería estarle poniendo atención a ella. Mamá alcanza la portátil y oprime el botón para el control de modalidad; las luces se desvanecen, las sombras desaparecen, surge música suave y la portátil se apaga.

Componentes de una red UpnP

Dispositivos, servicios y puntos de control

Los bloques de construcción básicos de una red UPnP son dispositivos, servicios y puntos de control. Estos se describen más detalladamente en esta sección

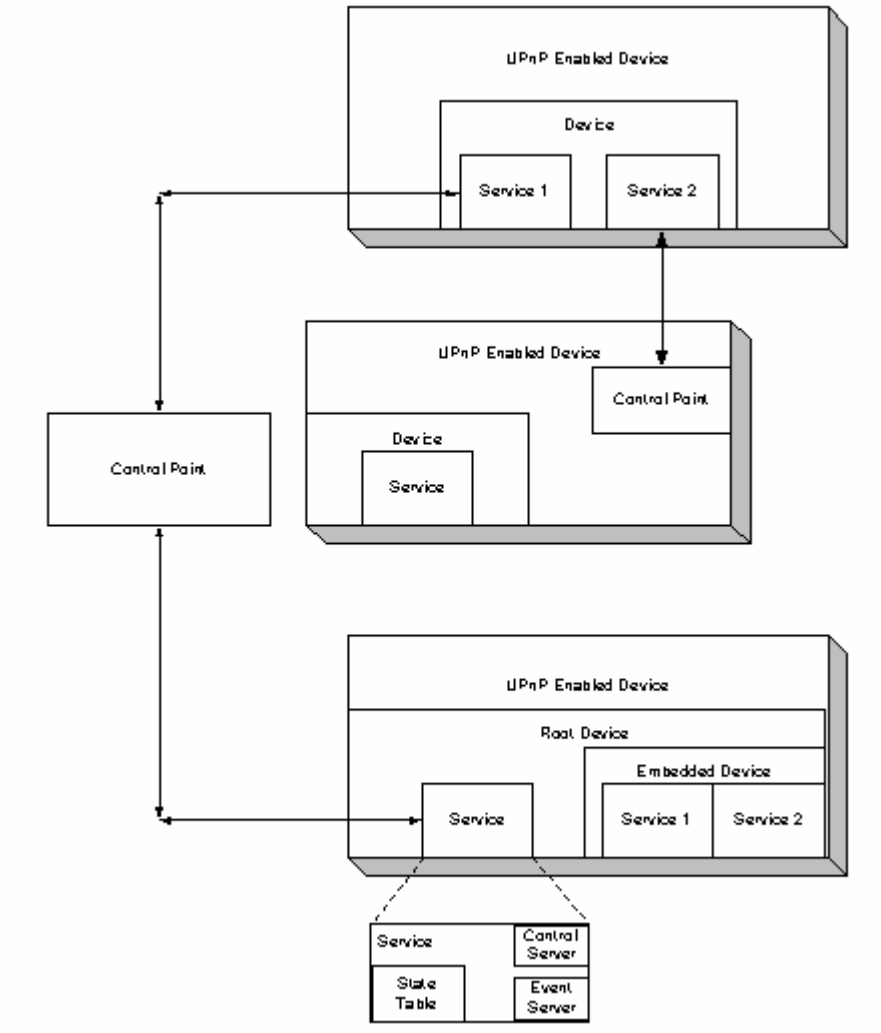


Ilustración 2: Puntos de control, dispositivos y servicios de UPnP

Dispositivos

Un dispositivo UPnP es un contenedor de servicios y dispositivos. Por ejemplo, una videograbadora puede consistir en un servicio de transporte de cinta, un servicio de afinación y un servicio de reloj. Un dispositivo integrado que incluye TV/videograbadora consistiría no sólo de servicios, sino también de un dispositivo integrado.

Diferentes categorías de dispositivos UPnP se asociarán con diferentes conjuntos de servicios y dispositivos integrados. Por ejemplo, los servicios de una videograbadora serán

diferentes a aquellos de una impresora. Como consecuencia, los diferentes grupos de trabajo se estandarizarán en el conjunto de servicios que proporcionará un tipo de servicio en particular. Toda esta información se captura en un documento de descripción de dispositivo XML que el dispositivo debe alojar. Además del conjunto de servicios, la descripción del dispositivo también enumera las propiedades (tales como nombre de dispositivo e iconos) asociadas con el dispositivo.

Servicios

La unidad de control más pequeña en una red UPnP es un servicio. Un servicio expone acciones y modela su status con variables de status. Por ejemplo, un servicio de reloj se puede modelar para tener una variable de status, hora actual, que define el status del reloj y dos acciones, definir hora y obtener hora, que le permiten controlar el servicio. Similar a la descripción del dispositivo, esta información es parte de una descripción de servicio XML estandarizada por el foro UPnP. Dentro del documento de descripción del dispositivo se incluye una dirección Web para estas descripciones de servicio. Los dispositivos pueden incluir varios servicios.

Un servicio en un dispositivo UPnP consiste en una tabla de status, un servidor de control y un servidor de eventos. La tabla de status modela el status del servicio a través de las variables de status y las actualiza cuando cambia el status. El servidor de control recibe solicitudes de acción (tales como establecer _ hora), las lleva a cabo, actualiza la tabla de status y devuelve respuestas. El servidor de eventos publica eventos para suscriptores interesados en cualquier momento en que cambie el status del servicio. Por ejemplo, el servicio de alarma contra incendios enviaría un evento a los suscriptores interesados cuando su status cambia a “sonar”.

Puntos de control

Un punto de control en una red UPnP es un controlador capaz de descubrir y controlar a otros dispositivos. Después del descubrimiento, un punto de control podría:

- Recuperar la descripción del dispositivo y obtener una lista de servicios asociados.
- Recuperar las descripciones de servicio para los servicios de interés.
- Invocar acciones para controlar el servicio.
- Subscribirse a la fuente de eventos del servicio. Siempre que cambie el status del servicio, el servidor de eventos enviará un evento al punto de control.

Se espera que los dispositivos incorporen la funcionalidad del punto de control (y viceversa) para permitir una verdadera operación en red de punto a punto ("peer-to-peer").

Descripción general del protocolo UpnP

Medios de operación en red para UpnP

UPnP aprovecha el conjunto estándar del protocolo IP para que los medios de la red sigan independientes. Los dispositivos en una red UPnP se pueden conectar utilizando cualquier medio de comunicación, incluyendo frecuencia de radio (RF, inalámbrica), línea telefónica, línea de energía, IrDA, Ethernet e IEEE 1394. En otras palabras, cualquier medio que se pueda utilizar para colocar dispositivos en red, UPnP puede habilitarlo. La única preocupación podría ser que los medios que se están utilizando no soporten el ancho de banda que se requiere para el uso que se pretende.

UPnP utiliza protocolos abiertos y estándares, tales como TCP/IP, HTTP y XML. Sin embargo, se podrían utilizar otras tecnologías para colocar en red los dispositivos por varias razones, incluyendo costos, requerimientos tecnológicos o soporte inherente. Estos

incluyen tecnologías de operación en red como HAVi, CeBus, LonWorks, EIB o X10. Estos también pueden participar en la red UPnP a través de un bridge o proxy UPnP. Una red UPnP que contenga dispositivos en red a través de un bridge, se podría ver de manera similar a la siguiente ilustración.

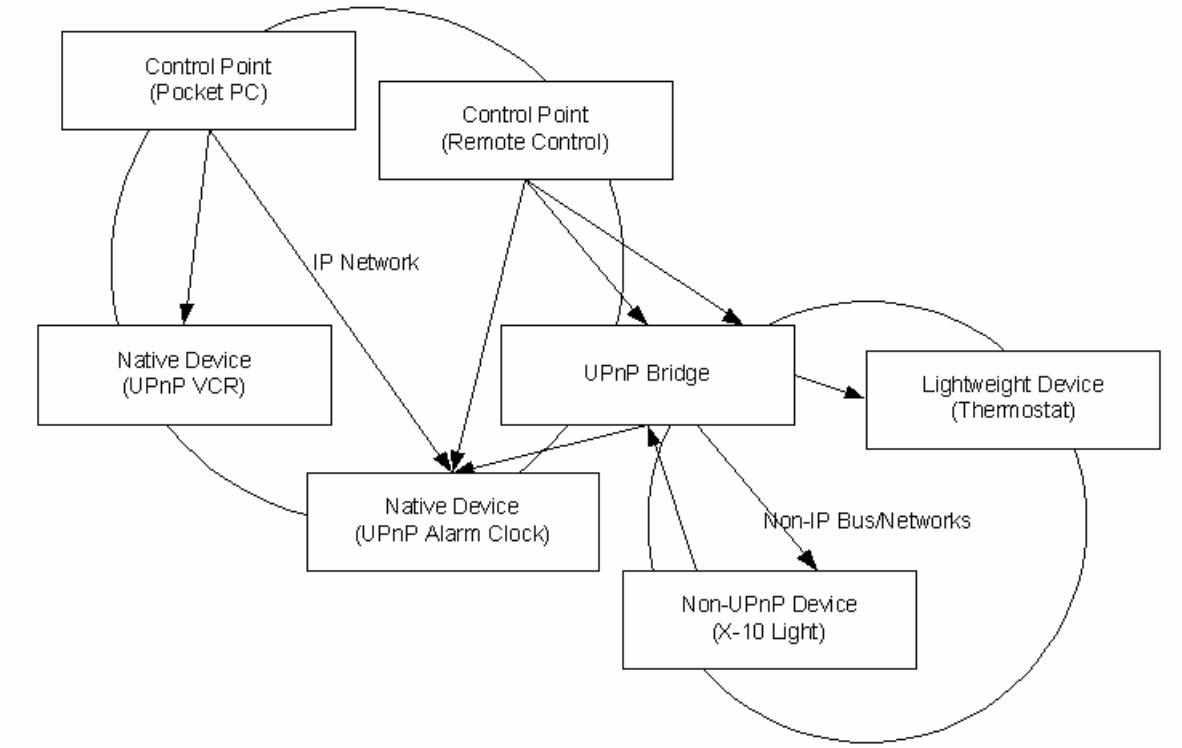


Ilustración 3: Una red UPnP utilizando un “bridge”

Protocolos utilizados por UpnP

UPnP aprovecha varios protocolos estándares existentes. Utilizar estos protocolos estandarizados ayuda a asegurar la interoperabilidad entre las implementaciones del proveedor. Los protocolos que se utilizan para implementar UPnP se encuentran en uso en Internet y en redes de área local en todas partes. Esta permanencia asegura que existe un

gran conjunto de personas con conocimientos para implementar e instalar soluciones basadas en estos protocolos. Debido a que los mismos protocolos ya están en uso, se necesitaría hacer muy poco para que los dispositivos UPnP funcionen en un ambiente en red existente.

Algunos de los protocolos que se utilizan para implementar UPnP se resumen en el resto de esta sección.

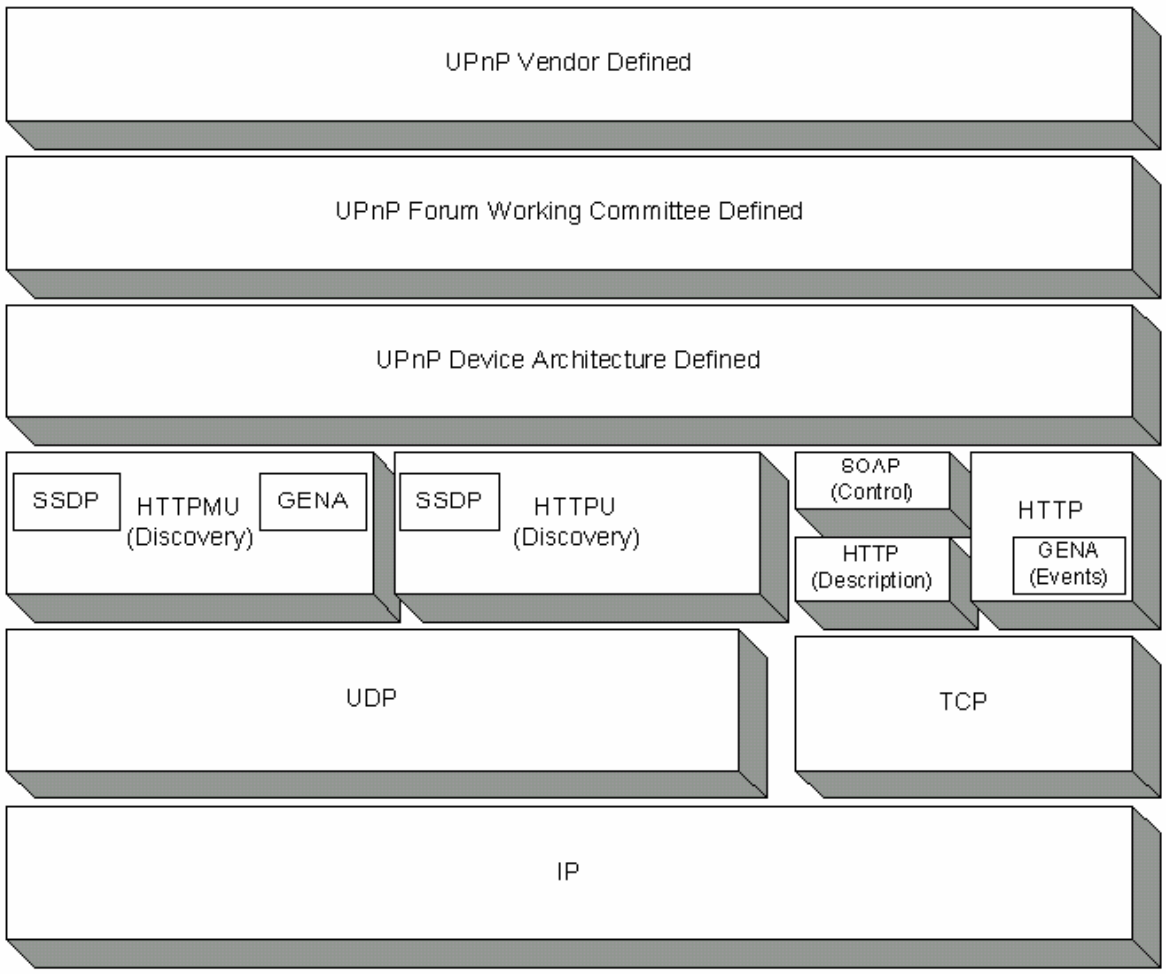


Ilustración 4: Pila del protocolo UPnP

Protocolos específicos de UPnP

Los proveedores UPnP, los Comités de trabajo del foro UPnP y el documento de Arquitectura del dispositivo UPnP definen los protocolos de más alto nivel que se utilizan para implementar UPnP. Con base en la arquitectura del dispositivo, los comités de trabajo definen las especificaciones para los tipos de dispositivos tales como videograbadoras, sistemas HVAC, lavadoras y otros aparatos. Subsecuentemente, los proveedores de dispositivos UPnP agregan datos específicos a sus dispositivos, como nombre de modelo, sitio Web, etc.

TCP/IP

La pila del protocolo de operación en red TCP/IP funciona como una base sobre la cual se desarrolla el resto de los protocolos UPnP. Al utilizar el conjunto de protocolo TCP/IP prevaleciente y estándar, UPnP aprovecha la habilidad del protocolo para aprovechar diferentes medios físicos y asegura la interoperabilidad de varios proveedores.

Los dispositivos UPnP pueden utilizar varios de los protocolos en la pila TCP/IP, incluyendo TCP, UDP, IGMP, ARP e IP, así como los servicios TCP/IP tales como DHCP y DNS. La forma en que se utilizan estos protocolos y servicios para proporcionar lo que se requiere para que UPnP funcione, será más claro conforme analicemos los otros protocolos en esta sección y analicemos cómo UPnP funciona en las secciones subsecuentes.

Debido a que TCP/IP es uno de los protocolos de operación en red más ubicuo, localizar o crear una implementación para un dispositivo UPnP que se ajusta para pie de impresión y/o rendimiento es relativamente fácil.

Este documento asume que el lector tiene un entendimiento básico del conjunto de protocolo y servicios TCP/IP. Se puede encontrar mayor información sobre TCP/IP en las referencias enumeradas al final de este documento.

HTTP, HTTPU, HTTPMU

TCP/IP proporciona la pila de protocolo base para proporcionar conectividad de red entre los dispositivos UPnP. HTTP, que es en gran medida responsable del éxito de Internet, también es una parte básica de UPnP. Todos los aspectos de UPnP se desarrollan por encima de HTTP o sus variantes.

HTTPU (y HTTPMU) son variantes de HTTP definido para entregar mensajes a través de UDP/IP en lugar de TCP/IP. Estos protocolos son utilizados por SSDP, que se describe a continuación. Los formatos de mensajes básicos utilizados por estos protocolos se adhieren a los de HTTP y se requieren ambos para una comunicación de multidifusión y cuando la entrega del mensaje no requiere los riesgos asociados con la confiabilidad.

Parte de las explicaciones de los protocolos de mayor nivel y de los trabajos de UPnP asumen un conocimiento básico del protocolo HTTP. Se puede encontrar mayor información sobre HTTP a través de las referencias enumeradas al final de este documento.

SSDP

El Protocolo de descubrimiento de servicio simple (SSDP), como lo implica su nombre, define cómo los servicios de red pueden ser descubiertos en la red. SSDP se desarrolla tomando como base HTTPU y HTTPMU y define métodos tanto para que un punto de control localice los recursos de interés en la red y para que los dispositivos anuncien su disponibilidad en la red. Al definir el uso tanto de las solicitudes de búsqueda como de los anuncios de presencia, SSDP elimina los gastos que serían necesarios si sólo

se utilizara uno de estos mecanismos. Como resultado, cada punto de control en la red cuenta con información completa sobre el status de la red, mientras reduce el tráfico de la misma.

Tanto los puntos de control como los dispositivos utilizan SSDP. Un punto de control UPnP, al iniciarse, puede enviar una solicitud de búsqueda SSDP (a través de HTTPMU) para descubrir dispositivos y servicios que están disponibles en la red. El punto de control puede refinar la búsqueda para encontrar sólo los dispositivos de un tipo en particular (como una videograbadora), servicios en particular (tales como dispositivos con servicios de reloj) o incluso un dispositivo en particular.

Los dispositivos UPnP escuchan al puerto de multidifusión. Al recibir una solicitud de búsqueda, el dispositivo examina el criterio de búsqueda para determinar si se correlacionan. Si se encuentra una relación, una respuesta SSDP de difusión única (a través de HTTPU) se envía al punto de control.

De manera similar, un dispositivo, al conectarse en la red, enviará varios anuncios de presencia SSDP que anunciarán los servicios que soporta.

Tanto los anuncios de presencia como los mensajes de respuesta del dispositivo de difusión única contienen una dirección Web para ubicar el documento de descripción de dispositivo que tiene información sobre el conjunto de propiedades y servicios soportados por el mismo.

Además de las capacidades de descubrimiento que se proporcionan, SSDP también proporciona un método para que un dispositivo y sus servicios asociados no se “alejen” de la red (envía una notificación) e incluye tiempos fuera de la memoria cache para purgar la información de auto reparación.

GENA

La Arquitectura de notificación de eventos genéricos (GENA) se definió para proporcionar la capacidad de enviar y recibir notificaciones utilizando HTTP sobre TCP/IP y UDP de multidifusión. GENA también define los conceptos de suscriptores y editores de notificaciones para habilitar eventos.

Los formatos GENA se utilizan en UPnP para crear los anuncios de presencia que se van a enviar utilizando el Protocolo de descubrimiento de servicios simple (SSDP) y para proporcionar la habilidad de señalizar cambios en el status del servicio para los eventos UPnP. Un punto de control interesado en recibir notificaciones de eventos se suscribirá a una fuente de eventos al enviar una solicitud que incluye el servicio de interés, una ubicación hacia dónde enviar los eventos y un tiempo de suscripción para la notificación del evento.

La suscripción se debe renovar periódicamente para seguir recibiendo notificaciones y también se puede cancelar utilizando GENA.

SOAP

El Protocolo simplificado para acceso de objetos (SOAP) define el uso del Lenguaje XML y HTTP para ejecutar llamadas de procedimiento remoto. Se está convirtiendo en el estándar para la comunicación basada en RPC por Internet. Al hacer uso de la infraestructura existente de Internet, puede funcionar de manera efectiva con firewalls y proxies. SOAP también puede hacer uso de SSL para seguridad y utilizar las facilidades de gestión de conexión de HTTP, logrando una comunicación distribuida a través de Internet tan fácil como se accede a las páginas Web.

Muy similar a una llamada de procedimiento remoto, UPnP utiliza SOAP para distribuir mensajes de control a los dispositivos y regresar los resultados o errores de nuevo a los puntos de control.

Cada solicitud de control UPnP es un mensaje SOAP que contiene la acción que se va a invocar junto con un conjunto de parámetros. La respuesta también es un mensaje SOAP y contiene el status, valor de retorno y cualquier parámetro de retorno.

XML

XML, para utilizar la definición de W3C, es el formato universal para datos estructurados en la Web. En otras palabras, XML es una manera de colocar casi cualquier tipo de datos estructurados en un archivo de texto.

XML se asemeja mucho a HTML en cuanto a que utiliza etiquetas y atributos. En realidad, es un poco diferente en cuanto a que estas etiquetas y atributos no se definen de manera global en cuanto a su significado, pero se interpretan dentro del contexto de su uso. Estas funciones de XML lo proyectan como la alternativa ideal para desarrollar esquemas para varios tipos de documentos. El uso de XML como un lenguaje esquema, está definido por el W3C.

XML es una parte básica de UPnP que se utiliza en las descripciones de dispositivos y servicio, mensajes de control y eventos.

Cómo funciona UPNP

Las Responsabilidades de UpnP

UPnP proporciona soporte para la comunicación entre los puntos de control y los dispositivos. Los medios de red, el conjunto de protocolo TCP/IP y HTTP proporcionan conectividad de red básica y direccionamiento necesario. Sobre estos protocolos abiertos estándares basados en Internet, UPnP define un conjunto de servidores HTTP para manejar el descubrimiento, la descripción, el control, los eventos y la presentación.

Esta sección describe cómo los protocolos definidos anteriormente en este documento se utilizan para resolver estas necesidades.

La pila de protocolo UpnP

Hemos descrito los protocolos que se utilizan para implementar UPnP, pero para comprender estos protocolos mejor, observemos este diagrama.

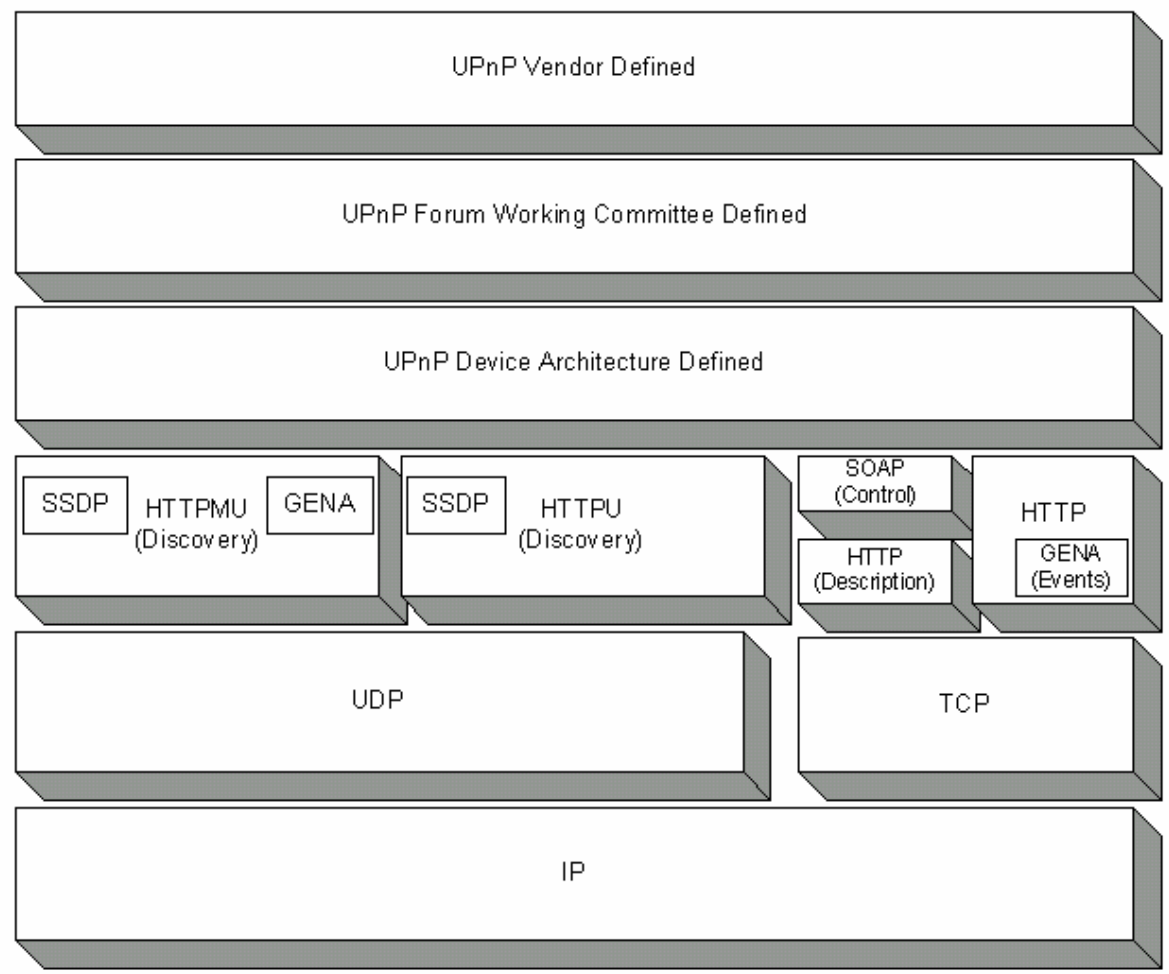


Ilustración 5: La pila del protocolo UPnP

La arquitectura de dispositivos UPnP define un esquema o plantilla para crear la descripción del dispositivo y de servicio para cualquier tipo de dispositivo o servicio.

Los comités de trabajo individual estandardizan subsecuentemente sobre varios tipos de dispositivos y servicio y crean una plantilla para cada tipo de dispositivo de servicio individual.

Por último, un proveedor llena esta plantilla con información específica del dispositivo o servicio, tal como el nombre del dispositivo, número de modelo, nombre del fabricante y sitio Web para la descripción del servicio.

Estos datos después se encapsulan en los protocolos específicos de UPnP definidos en el documento de la arquitectura de dispositivos UPnP (tal como la plantilla de descripción del dispositivo XML).

La información especifica de UPnP que se requiere se inserta en todos los mensajes antes de que se formatee utilizando SSDP, GENA y SOAP y se distribuyen a través de HTTP, HTTPU o HTTPMU.

Pasos involucrados en la operación en red de UpnP

Direccionamiento

El fundamento de la operación en red de UPnP es el conjunto del protocolo TCP/IP y la clave para este conjunto es el direccionamiento. Cada dispositivo debe tener un cliente de Protocolo con configuración dinámica de host (DHCP) y buscar un servidor DHCP cuando el dispositivo se conecta primero a la red. Si un servidor DHCP está disponible, el dispositivo debe utilizar la dirección IP que se le asigna. Si no está disponible ningún servidor DHCP, el dispositivo debe utilizar Auto IP para obtener una dirección.

En resumen, Auto IP define cómo un dispositivo elige de manera inteligente una dirección IP a partir de un conjunto de direcciones privadas reservadas y puede moverse fácilmente entre las redes administradas y no administradas.

Un dispositivo puede implementar protocolos con niveles más altos fuera de UPnP que utilicen nombres fáciles de identificar para los dispositivos. En estos casos, es necesario resolver nombres host (dispositivo) para la dirección IP. Normalmente, para realizar esta función se utiliza DNS. Un dispositivo que requiere utilizar esta funcionalidad puede ser un cliente DNS y puede soportar el registro DNS dinámico para que su propio nombre aborde la correlación.

Descubrimiento

Una vez que los dispositivos se conectan a la red y se dirigen adecuadamente, el descubrimiento toma lugar. El descubrimiento es manejado por SSDP como se analizó anteriormente. Cuando un dispositivo se agrega a la red, SSDP permite al dispositivo anunciar sus servicios a los puntos de control de la red. Cuando un punto de control se agrega a la red, SSDP permite a ese punto de control buscar los dispositivos de interés en la red.

El intercambio fundamental en ambos casos es un mensaje de descubrimiento que contiene algunos aspectos específicos e importantes acerca del dispositivo o uno de sus servicios, por ejemplo, su tipo, identificación y un sitio Web para su documento de descripción de dispositivo XML.

Descripción

El siguiente paso en la operación en red UPnP es la descripción. Después de que un punto de control ha descubierto un dispositivo, el punto de control sigue conociendo muy

poco acerca del dispositivo. Para que el punto de control aprenda más acerca del dispositivo y sus capacidades, o para interactuar con el dispositivo, el punto de control debe recuperar la descripción del dispositivo del sitio Web proporcionado por el dispositivo en el mensaje de descubrimiento.

Los dispositivos pueden contener otros dispositivos y servicios lógicos. La descripción UPnP para un dispositivo se expresa en XML e incluye información del fabricante especifica, que incluye el nombre y número de modelo, número de serie, nombre del fabricante, direcciones de sitios Web específicos del proveedor, etc. La descripción también incluye una lista de algunos dispositivos o servicios integrados, así como direcciones Web para control, eventos y presentación.

Control

Después de que un punto de control ha recuperado una descripción del dispositivo, el punto de control cuenta con los aspectos básicos para el control del dispositivo. Para aprender más sobre este servicio, un punto de control debe recuperar una descripción UPnP detallada para cada servicio. La descripción para un servicio también se expresa en XML e incluye una lista de los comandos, o acciones, a los que responde el servicio y los parámetros o argumentos para cada acción. La descripción de un servicio también incluye una lista de variables. Estas variables modelan el status del servicio durante el tiempo de ejecución y se describen en términos, ya sea de tipo de datos, rango y características del evento.

Para controlar un dispositivo, un punto de control envía una solicitud de acción a un servicio del dispositivo. Para hacer esto, un punto de control envía un mensaje de control adecuado a la dirección Web de control para el servicio (proporcionado en la descripción del dispositivo). Los mensajes de control también se expresan en XML utilizando SOAP.

En respuesta al mensaje de control, el servicio regresa valores específicos de la acción o códigos de falla.

Eventos

Una descripción UPnP para un servicio incluye una lista de acciones a los cuales responde el servicio y una lista de variables que modelan el status del servicio durante el tiempo de ejecución. El servicio publica actualizaciones cuando estas variables cambian y un punto de control puede suscribirse para recibir esta información.

El servicio publica actualizaciones al enviar mensajes de eventos. Los mensajes de eventos contienen los nombres de una o más variables de status y el valor actual de esas variables. Estos mensajes también se expresan en XML y se formatean utilizando GENA.

Un evento de mensaje inicial especial se envía cuando un punto de control se suscribe primero. Este mensaje de eventos contiene los nombres y valores de todas las variables con eventos y permite al suscriptor inicializar su modelo del status del servicio.

Para soportar varios puntos de control, a todos los suscriptores se le envían todos los mensajes de eventos, los suscriptores reciben mensajes de eventos para todas las variables con evento y los mensajes de evento se envían sin importar por qué cambió el status de la variable (en respuesta a una solicitud de acción o debido a un cambio de status).

Presentación

Si un dispositivo tiene una dirección Web para presentación, entonces el punto de control puede recuperar una página a partir de esta dirección, cargar la página en un navegador y, dependiendo de las capacidades de la página, permitir al usuario controlar el dispositivo y/o ver el status del mismo. El grado al cual cada uno de estos se pueda lograr, depende de las capacidades específicas de la página y dispositivo de presentación.

En resumen:

- UPnP se basa en protocolos conectados (como Internet), no en APIs, permitiéndoles ser verdaderamente independientes de los medios y de la plataforma.
- UPnP se basa en los estándares existentes, haciendo que la interoperabilidad sea pueda lograr con facilidad.
- UPnP ha venido ganando fuerza en el medio industrial, lo que asegura su éxito.
- Aunque se basa en estándares, UPnP es flexible y puede satisfacer las necesidades de los dispositivos que operan en las redes de hoy y del futuro.

Un ejemplo de red UpnP

Para comprender aún más cuándo y cómo toma lugar cada uno de los pasos en la operación en red UPnP, es bueno definir una red pequeña con sólo unos cuantos dispositivos UPnP. Después, podemos describir cómo estos dispositivos interactúan conforme se relacionan con UPnP.

Para este documento, hemos elegido una red que contiene los siguientes dispositivos habilitados con UPnP:

Una conexión a Internet. Este dispositivo podría ser un dispositivo individual de conexión o una PC que funcione como una salida. El dispositivo puede ser un punto de control, pero tal vez no. Los servicios proporcionados por el dispositivo pueden incluir acceso a Internet, un servidor DHCP, un proxy DNS y un servicio de almacenamiento. Esta salida también se conectará a varios medios LAN en el hogar y funcionará como un puente para estos medios. Los medios utilizados incluyen IEEE 802.11 inalámbrico, una red de línea de poder, una red de línea telefónica e IEEE 1394.

Varios aparatos inteligentes. Para los propósitos de este ejemplo, la red contendrá varios aparatos que están habilitados por UPnP. Esto incluirá un radio reloj, una cafetera y un horno microondas ,todos conectados en una red de línea eléctrica. La red también contendrá una impresora UPnP conectada a la red de la línea telefónica.

Un sistema de entretenimiento para el hogar. El sistema de entretenimiento para el hogar incluye varios dispositivos conectados conjuntamente a través de IEEE 1394 ó 'Firewire' y conectado al dispositivo de salida. Los componentes incluyen un estéreo con radio, receptor y reproductor de CDs; una TV y videograbadora; y conexiones para equipo A/V adicional, tal como video o cámara digital. También se agregará un nuevo reproductor de DVD a esta red.

Una portátil inalámbrica. El papá utiliza una computadora portátil con un adaptador de red inalámbrico en el trabajo y, ocasionalmente, trae esta computadora a la casa para terminar tareas no concluidas en el trabajo.

Aunque existen muchas otras posibilidades para que los componentes participen en esta red, la red se está manteniendo relativamente sencilla en este ejemplo con el propósito de explicar la operación de UPnP.

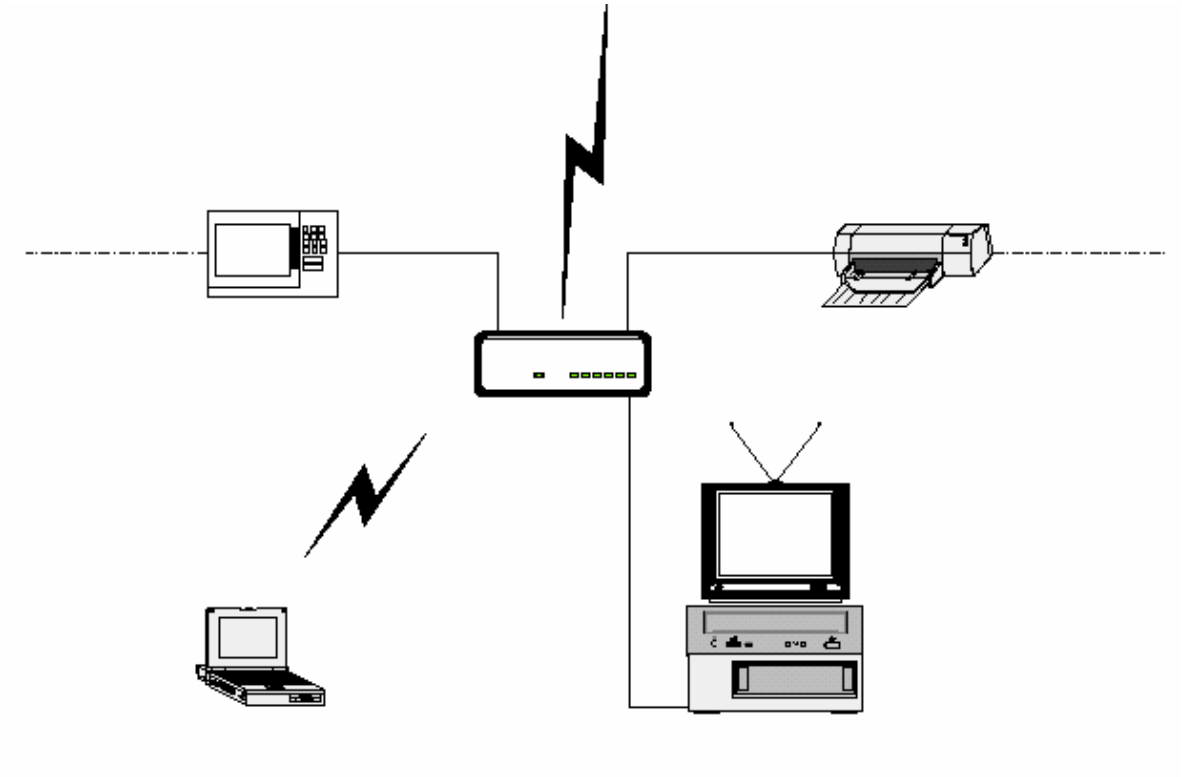


Ilustración 6: Ejemplo de una red

La historia empieza

Para iniciar este escenario, todos los componentes en la red que se muestra arriba están activos y funcionando y se han descubierto entre sí utilizando los protocolos UPnP, excepto por la portátil y el reproductor de DVD.

Durante la cena hace un par de semanas, la familia comentaba el éxito del DVD en el mercado de video y cómo era posible que ellos todavía no habían experimentado el video de alta calidad, ya que sólo cuentan con una videograbadora obsoleta.

Mamá recibió una invitación en el correo para unirse a un club de DVD. La familia se sentó y cada uno seleccionó varios DVDs para empezar su colección. Hoy, mamá llamó a papá al trabajo y le dijo que el pedido de los DVDs había llegado. Desafortunadamente,

no habían comprado la pieza clave del equipo necesario para verlos, el reproductor de DVD.

En su camino a casa desde el trabajo, papá (quien es el experto en dichas compras) se detuvo en una tienda y compró el juego de reproductor y parlantes más avanzado de DVD, asegurándose de obtener uno que estaba habilitado para UPnP.

Cuando papá llegó a casa, desenvolvieron el nuevo juguete y lo conectaron en el sistema de entretenimiento que utiliza IEEE 1394. Con UPnP, eso fue lo único que se necesitó para el resto de la red de la casa conociera su existencia.

Direccionamiento del dispositivo

Lo primero que el nuevo reproductor de DVD tenía que hacer era obtener una dirección para poder participar en la red. Cada dispositivo debe contar con un cliente DHCP y buscar un servidor DHCP cuando el dispositivo se conecta por primera vez a la red.

Si el cliente DHCP en el reproductor de DVD no obtiene una respuesta de un servidor después de esperar un corto periodo de tiempo, lo volverá a intentar para asegurarse que un servidor tiene oportunidad de responder. Si la red no cuenta con un servidor DHCP, el reproductor de DVD utilizará automáticamente la dirección IP (Auto IP) para elegir una dirección adecuada.

Con Auto IP, el dispositivo elige de manera inteligente una dirección IP en el rango 169.254/16. La primera y las últimas 256 direcciones en este rango están reservadas y no se deben utilizar. La dirección seleccionada deberá probarse para determinar si la dirección ya está en uso. Si la dirección está siendo utilizada por otro dispositivo, se debe elegir y probar otra dirección para que se pueda implementar.

Si la red cuenta con un servidor DHCP disponible, todo este proceso puede bien tomar un segundo para completarse. Si la red no cuenta con un servidor DHCP, que requiera que el dispositivo utilice Auto IP, el proceso tomará un poco más. Si la dirección es asignada utilizando Auto IP, el reproductor de DVD periódicamente verificará para ver si un servidor DHCP está disponible en la red para asegurar que la conectividad se mantiene entre los dispositivos.

En este punto, el reproductor de DVD tiene una dirección asignada por el servidor DHCP y todos los demás dispositivos en la red cuentan con una dirección en la misma subred o el reproductor de DVD cuenta con una dirección Auto IP. En cualquiera de los casos, el reproductor de DVD puede comunicarse con otros dispositivos en la red, utilizando TCP/IP.

Una vez que el reproductor de DVD cuenta con una dirección IP válida para la red, se puede localizar y hacer referencia en esa red a través de esa dirección. Pueden surgir situaciones en donde el usuario final necesite localizar e identificar un dispositivo. En estas situaciones, un nombre normal para el dispositivo es mucho más fácil en su uso para un humano que una dirección IP. Sin embargo, el uso de DNS para el nombre de correlación de dirección está fuera del alcance de UPnP.

Descubrimiento – anuncio

Ahora que nuestro nuevo dispositivo cuenta con una dirección y puede comunicarse en la red, necesita darse a conocer con aquellos puntos de control UPnP que ya estaban funcionando en la red. Esta es una forma de descubrimiento en UPnP. Cuando un dispositivo se agrega a la red, el protocolo de descubrimiento UPnP permite que este dispositivo anuncie sus servicios a los puntos de control en la red.

Cuando un dispositivo nuevo se agrega a la red, éste transmite varios mensajes de descubrimiento anunciando sus dispositivos y servicios integrados. Cualquier punto de

control interesado puede escuchar la dirección estándar para notificaciones en cuanto a que están disponibles nuevos servicios.

Los mensajes de descubrimiento que nuestro reproductor de DVD envía incluirán un sello de tiempo para indicar durante cuánto tiempo es válido el anuncio. Antes de que expire este tiempo, el reproductor de DVD debe volver a enviar su anuncio. De otra forma, los puntos de control pueden suponer que el dispositivo ya no está disponible. El reproductor de DVD también debe enviar un mensaje para comunicar explícitamente a la red que se va a retirar antes de salirse de línea.

La pila del protocolo que se muestra a continuación se utiliza para enviar y recibir anuncios.

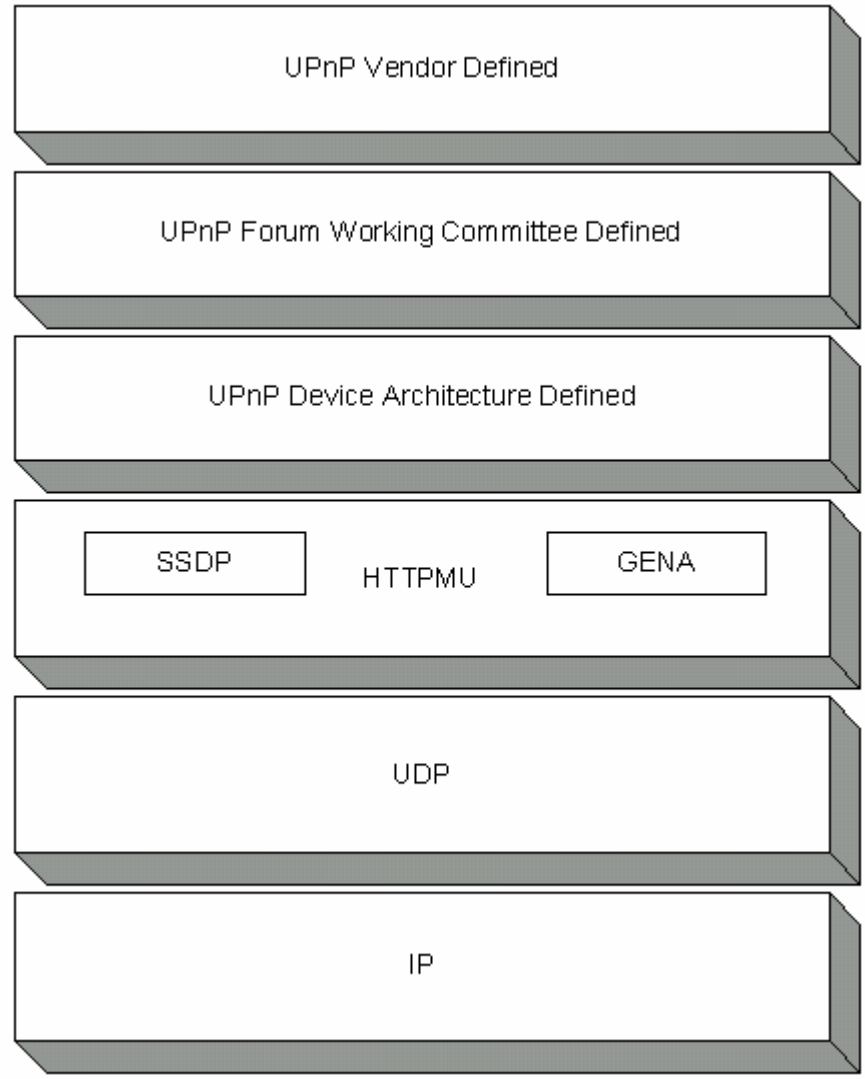


Ilustración 7: Pila del protocolo para anuncios de descubrimiento

Nuestro reproductor de DVD, una vez que se conecta a la red, enviará anuncios GENA para cada dispositivo y servicio, anunciando su presencia. Debido a que estos mensajes se distribuyen a través de UDP, un transporte no confiable, podrían enviarse varias veces para garantizar que son recibidos por todos los puntos de control interesados.

Búsqueda de descubrimiento

Una vez realizado el arduo trabajo de conectar el reproductor de DVD (papá también llenó el reproductor con los DVDs que recibieron por correo), papá se sienta en el sofá con su portátil para terminar la presentación que tiene que entregar el lunes.

La portátil de papá también está habilitada para UPnP, por lo que la dirección y los anuncios de descubrimiento toman lugar como lo harían con el nuevo reproductor de DVD. Papá empieza su trabajo como parte de la red en el hogar sin ninguna configuración adicional. Debido a que es viernes por la noche, todavía falta una hora para cenar, no tiene que terminar su presentación sino hasta el lunes y hay un nuevo juguete en casa. Además, papá quiere jugar con él.

Papá desea ver su película favorita de DVD y cómo funciona el nuevo juguete. Por supuesto que podría intentar descubrir el nuevo remoto que se incluye con el reproductor, ya que está sentado en el sofá con su portátil encendida y el DVD está disponible en la red

UPnP, ¿por qué levantarse?

Papá inicia una aplicación del control de video en su portátil. Al iniciar esta aplicación, se presenta un nuevo punto de control en la red. Todos los dispositivos de video en la red aparecen en pantalla y papá selecciona el reproductor de DVD. Papá luego selecciona el disco que desea reproducir y éste empieza. También puede utilizar la aplicación del control de video para encender la TV.

Varios otros pasos en la operación en red de UPnP acaban de realizarse. Por primera vez, un nuevo punto de control se ha introducido en la red. Cuando un nuevo punto de control se agrega a la red, éste hace una transmisión múltiple de un mensaje de descubrimiento SSDP, buscando así dispositivos y servicios interesados. Todos los dispositivos deben escuchar la dirección de multidifusión estándar de estos mensajes y deben responder si cualquiera de sus dispositivos o servicios integrados concuerda con el

criterio de búsqueda en el mensaje de descubrimiento. La aplicación del control de video que papá agregó se ve específicamente en los dispositivos de fuente de video.

La pila del protocolo que se utilizó para estos mensajes de búsqueda se muestra aquí:

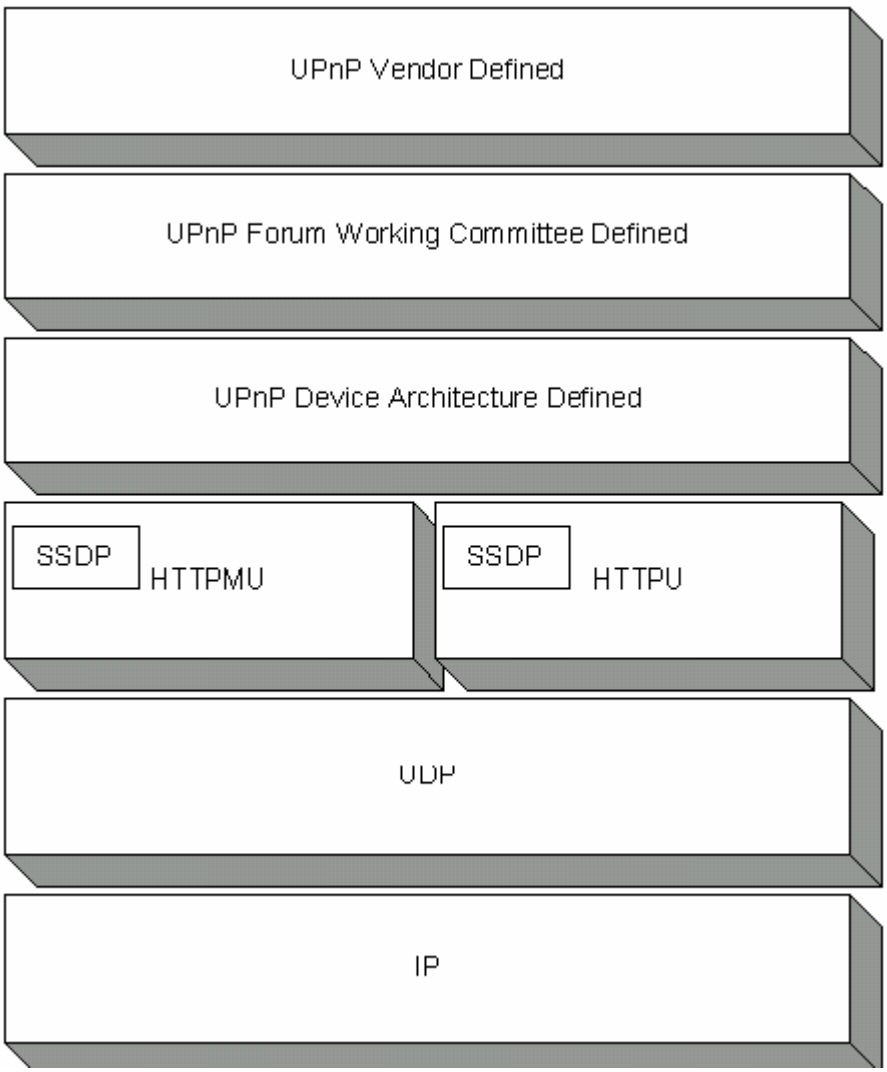


Ilustración 8: Pila del protocolo para la búsqueda de descubrimiento

Estos mensajes de búsqueda contienen información específica del proveedor, tal como los tipos de dispositivos o servicios y los identificadores. Se agregan los tipos de dispositivos o servicios definidos por un comité de trabajo UPnP para estos tipos de dispositivos, en este caso, dispositivos de fuente de video. Esta información se encapsula en una solicitud SSDP que se envía utilizando HTTPMU. Las respuestas a estas solicitudes de búsqueda se enviarán utilizando UDP de transmisión única con encabezados SSDP.

Las respuestas a estas solicitudes de búsqueda contienen la misma información que se incluye en los anuncios de descubrimiento. Estas respuestas se envían a la dirección IP del punto de control que inicia la búsqueda, en este caso la portátil de papá.

Descripción

El nuevo punto de control que se ejecuta en la portátil de papá ahora cuenta con información sobre todos los dispositivos de fuente de video en la red. Por primera vez en esta historia, tenemos una situación en donde el punto de control necesita más información sobre un dispositivo y, por lo tanto, necesitamos avanzar a la fase de descripción.

Las respuestas recibidas por la solicitud de descubrimiento de búsqueda contienen la dirección a través de la cual obtendremos las descripciones del dispositivo.

Para obtener una descripción del dispositivo UPnP, el punto de control emite una solicitud HTTP GET en la dirección a partir del mensaje de descubrimiento y el dispositivo regresa la descripción del dispositivo. Las direcciones para las descripciones de servicio son parte de la descripción del dispositivo, como consecuencia, las descripciones del servicio se pueden recuperar de la misma manera. La pila del protocolo que se utilizó para el paso de descripción es de la siguiente manera:

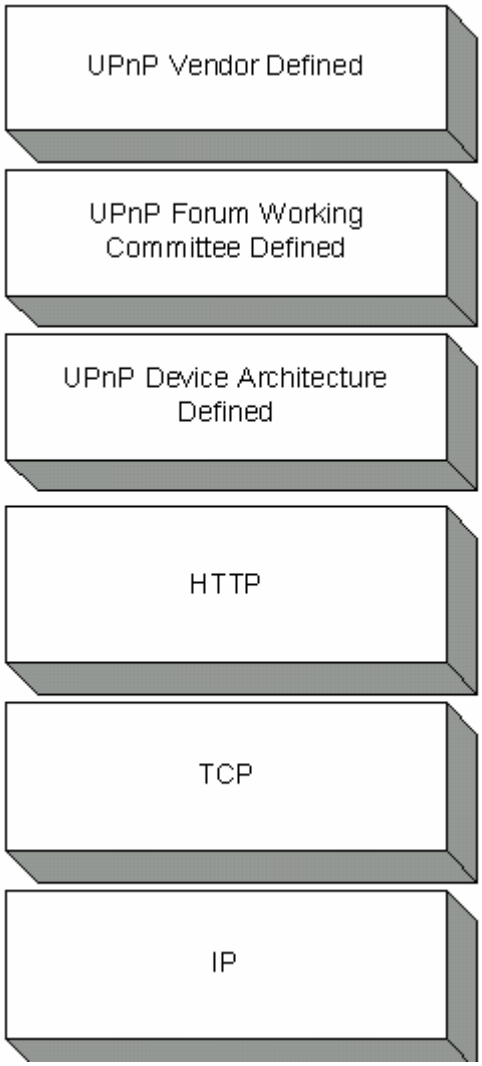


Ilustración 9: Pila del protocolo para la descripción

La descripción UPnP para un dispositivo es un documento XML que contiene varias piezas de información específica del proveedor, definiciones de todos los dispositivos integrados, dirección para la presentación del dispositivo y una enumeración de todos los servicios, incluyendo sus direcciones para control y eventos. Los proveedores de UPnP pueden ampliar la descripción estándar del dispositivo y servicio para que incluya variables adicionales del status, acciones e incluso servicios completos. De esta manera, UPnP permite flexibilidad mientras se adhiere a los estándares básicos. Las descripciones de

dispositivos y servicios de muestra se pueden encontrar en el documento de arquitectura del dispositivo UPnP.

Presentación

La aplicación que se está ejecutando en la portátil de papá puede determinar qué dispositivos y servicios presentar y cómo presentarlos. De manera alternativa, si el reproductor de DVD aloja una página de presentación (Web), esta página HTML se podría descargar y utilizar para controlar también el dispositivo.

La dirección de la página de presentación se contiene en la descripción del dispositivo. Para recuperar esta página se requiere el punto de control para emitir una solicitud HTTP GET URL de presentación. El dispositivo regresará entonces una página de presentación. La pila del protocolo se utiliza de la siguiente manera.

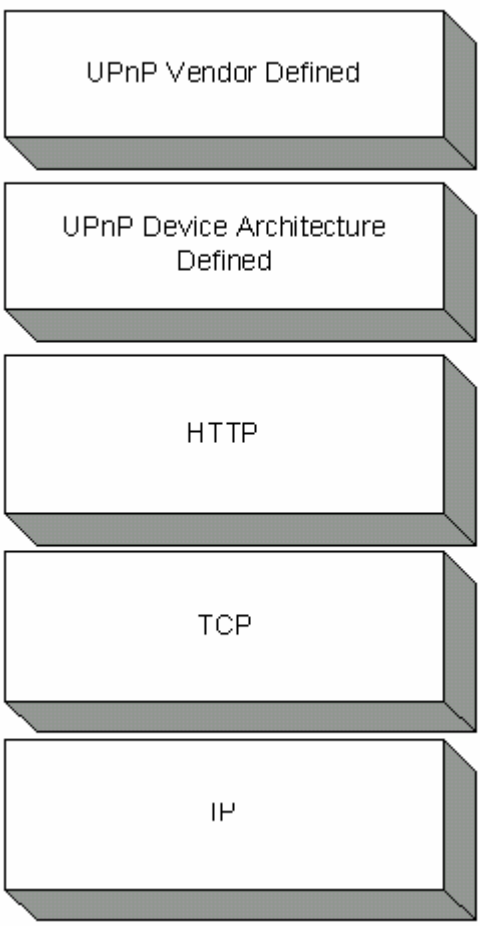


Ilustración 10: Pila del protocolo para presentación

El documento de la Arquitectura del dispositivo UPnP especifica que esta página se escriba en HTML. Esto es similar al navegador Web, excepto que aquí papá ve el dispositivo para controlarlo.

Las capacidades de la página de presentación son completamente especificadas por el proveedor UPnP. Para implementar una página de presentación, un proveedor UPnP puede desear utilizar los mecanismos UPnP para control y/o eventos, aprovechando las capacidades existentes del dispositivo. Observe que no existe un elemento del Foro UPnP definido en la presentación, ¡depende completamente del proveedor!

Control

Papá necesita controlar el reproductor de DVD para seleccionar una película en DVD e iniciar su reproducción. Para esto, podría utilizar ya sea la página de presentación o una aplicación genérica de control de video.

Una vez que el punto de control tiene conocimiento de un dispositivo y de sus servicios, puede invocar acciones sobre estos servicios y obtener valores de retorno. Al mismo tiempo, el punto de control puede agrupar esos servicios para los valores de sus variables de status.

Invocar acciones es un tipo de llamada de procedimiento remoto; un punto de control envía la acción al servicio del dispositivo y cuando la acción se ha terminado (o fallado) el servicio regresa los resultados o errores. El punto de control también puede agrupar el valor de las variables de status.

Para controlar el reproductor de DVD, la portátil de papá envía un mensaje de control a la dirección de control (contenido dentro de la descripción del dispositivo) para el servicio del DVD. El servicio de reproducción del DVD regresa los resultados o errores de la acción. Los efectos de la acción también se pueden monitorear a través de cambios en las variables de status del servicio. Estos cambios en las variables del status se publican en todos los puntos de control de interés, como se describió en los eventos, pero los valores de estas variables de status se pueden consultar, lo cual es una variante con respecto a una solicitud de control.

La siguiente pila de protocolos se utiliza para control.

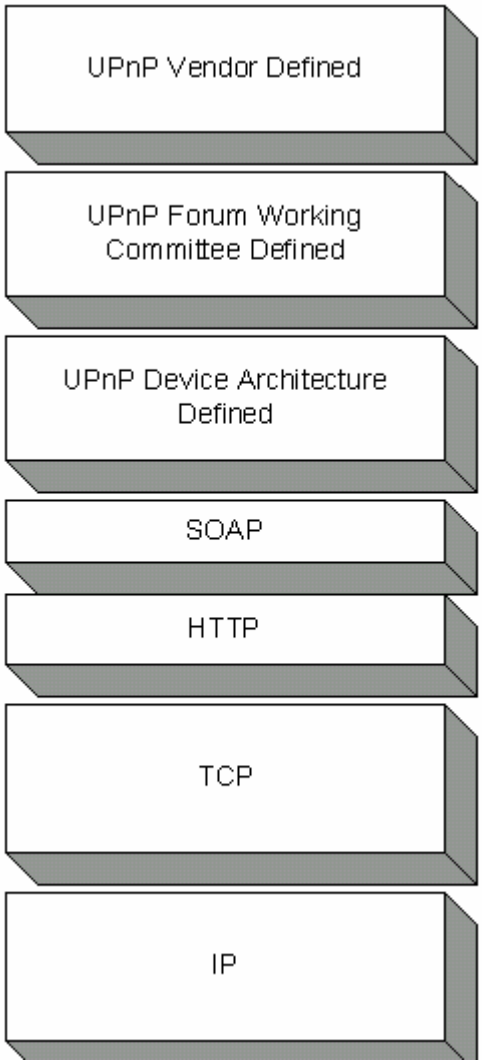


Ilustración 11: Pila del protocolo para control

La información específica del proveedor contenida en los mensajes de control incluye valores de argumento. El comité de trabajo del Foro UPnP define los nombres de acción, los nombres de argumento y las variables contenidas en estos mensajes. Esta información se encapsula en los formatos específicos UPnP y se formatean utilizando SOAP y luego se transmiten utilizando HTTP sobre TCP/IP.

El dispositivo debe responder a la solicitud de control en 30 segundos. La respuesta puede indicar que la acción sigue pendiente y que un evento significará su ejecución.

Nuestra aplicación de control tal vez también desee consultar el status de una variable del servicio en particular, por ejemplo, nuestro servicio de reproductor DVD puede contar con un servicio con una variable de status que contenga el tiempo de ejecución de un DVD en particular. Papá tal vez querrá saber esto para poder saber qué tan avanzada está la película cuando se sirva la cena. Los puntos de control también pueden consultar las variables de status de un servicio, pero sólo una variable de status única con cada consulta enviada.

Revisión de descubrimiento, descripción y control

Ahora el DVD se está reproduciendo y papá está listo para sentarse y empezar a trabajar en su presentación. Al ver su nueva adquisición, observa que el reloj está parpadeando. No sólo eso, sino que el reloj en la videograbadora también está parpadeando. Brevemente considera desconectarlo para que deje de seguir parpadeando al viejo estilo, pero mamá menciona que el reloj en el microondas, en la cafetera y en el despertador también están mal, ya que se fue la luz por unos minutos durante la tarde.

Papá recuerda que en la conexión a Internet venía una aplicación para configurar el reloj. Qué mejor oportunidad que ahora para intentarlo. No había cargado la aplicación en su portátil, pero dos factores han permitido que esta aplicación sea fácil de encontrar y ejecutar.

Primero, la conexión proporciona servicios de almacenamiento para la red del hogar. Cuenta con un servicio de almacenamiento que permite tener espacio disponible para la red. Esta aplicación para configurar la hora está disponible en el disco compartido por la salida. El segundo aspecto que permite que esta aplicación se ejecute fácilmente es que el sistema operativo en la portátil de papá está habilitada por UPnP, incluyendo su

explorador de archivos. Cuando abre el explorador, éste busca automáticamente dispositivos en la red que proporcionen servicios de almacenamiento de archivos y aparece el dispositivo de salida del almacenamiento.

Ahora papá puede hacer clic en la aplicación para configurar el reloj y hacer lo siguiente:

- Localizar la conexión de Internet y conectarla a una fuente de hora en Internet para obtener la hora correcta.
- Utilizar el descubrimiento UPnP para buscar en la red todos los dispositivos que proporcionan servicios de reloj.
- Recorrer cada uno de los dispositivos y enviar acciones “de configuración” para cada uno de los servicios de reloj.

Eso fue fácil. Al navegar más adelante, papá se da cuenta que la aplicación de reloj se puede configurar para ejecutarse periódicamente desde la salida que se ejecuta como un mismo punto de control. Establece la aplicación para ejecutarse a las 4:00 AM todas las mañanas y así nunca tendrá que preocuparse por configurar los relojes de nuevo.

Eventos

Ya se está acercando la hora de la cena y papá ha terminado su presentación. Desea quedarse con una copia original para poder revisarla durante la cena. Debido a que una impresora UPnP está conectada a la red de la línea telefónica en la cocina, ya está disponible en su portátil a través de su explorador de la impresora.

Papá elige imprimir en esta impresora y así lo hace. Coloca a un lado su portátil y se pone a ver de nuevo su película cuando de repente aparece un mensaje en su computadora notificándole que la impresora no tiene tinta. Mientras que esto es posible hoy con las

impresoras directamente conectadas a las PCs, con UPnP la impresora y el explorador de impresión utilizan eventos UPnP.

Papá estaba listo para llamar a su hijo y pedirle que cambie el cartucho de la tinta, cuando su hijo venía hacia él y le dijo que ya se había encargado de eso. Esto sucedió justamente cuando su hijo estaba haciendo su tarea en la habitación y su PC recibió la misma notificación. Todos los puntos de control en una red que registran eventos reciben las notificaciones.

Las variables de status que se describen en una descripción de servicio se pueden canalizar en eventos. El servicio publica actualizaciones cuando estas variables cambian. Un punto de control, tal como un explorador de impresión en este caso, puede suscribirse para recibir esta información al enviar un mensaje de suscripción. El editor del evento puede aceptar esta suscripción y responder con una duración para la suscripción. El suscriptor puede renovar esta suscripción o cancelarla cuando ya no esté interesado en ella.

La siguiente pila del protocolo se utiliza para los eventos:

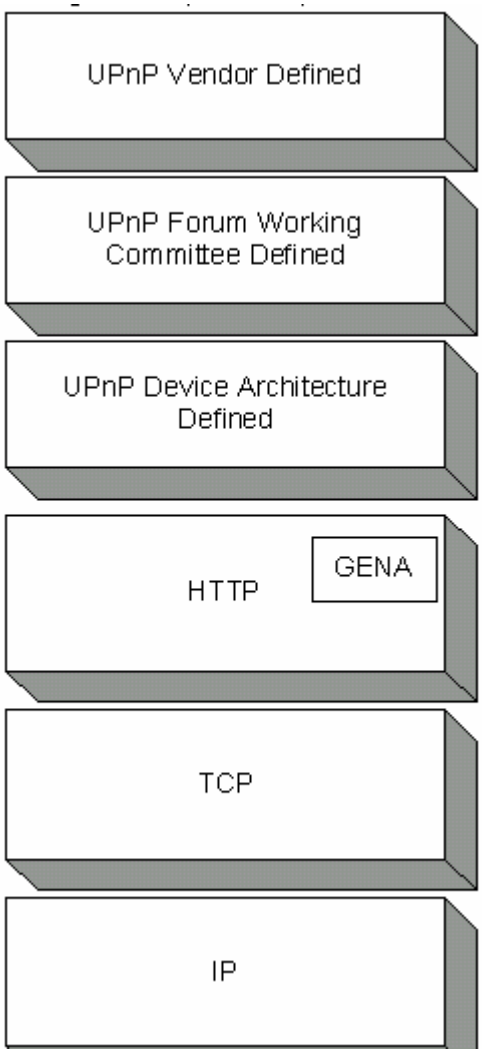


Ilustración 12: Pila del protocolo para eventos

Las direcciones para suscripción, duraciones de suscripción, valores de variables específicas y nombres de variables se formatean utilizando GENA y se distribuyen utilizando TCP/IP.

Acrónimos

API Interfaz de programación de aplicaciones

ARP Protocolo de resolución de dirección

DHCP Protocolo dinámico de configuración de host

DNS Sistema de nombre de dominio

FXPP Perfil de procesamiento XML flexible

GENA Arquitectura de notificación de eventos generales

HTML Lenguaje de marcación de hipertexto

HTTPMU Transmisión múltiple HTTP a través de UDP

HTTPU HTTP (transmisión única) a través de UDP

SOAP Protocolo de acceso sencillo a objetos

SSDP Protocolo de descubrimiento de servicio sencillo

UPC Código de producto universal

UPnP Plug and Play universal

URI Identificador de recursos uniformes

URL Localizador de recursos uniformes

URN Nombre de recursos uniformes

USN Nombre de servicio único

UUID Identificador único universal

XML Lenguaje de marcación extensible

Resumen:

Ahora más que nunca, el poder de la computación se está agregando a dispositivos más pequeños y comunes. Las tecnologías de medios en operación de red económicos y ubicuos ya están aquí o están cerca de lograrse. Las reducciones de precios en redes y cómputo de los últimos años son maravillosas.

Al combinar el poder de la computación y la conectividad en los dispositivos diarios con la operación en red, la cual es fácil de usar y configurar, nos está llevando a nuevos beneficios: las tareas diarias son más fáciles de realizar y las personas tienen más tiempo para disfrutar una calidad de vida superior. Las personas están más conectadas al mundo que nunca antes, un desarrollo que también corre el riesgo de abrumar a las personas. Así, las tareas pueden ser fáciles para que las personas las utilicen de manera efectiva.

Plug and Play universal es una iniciativa abierta para tomar los estándares existentes, las tecnologías existentes y el conocimiento existente, para darle un nuevo propósito y ofrecerlo sobre la promesa y la oportunidad de un mundo interconectado. Basado en estándares y lo suficientemente sencillo para que los aparatos más pequeños lo implementen, lo suficientemente poderoso para escalar la Internet global y con base en un enfoque probado de protocolos de Internet, Plug and Play universal es un esquema en desarrollo, pero un esquema que ha sido probado para que funcione.

Referencias

Arquitectura de dispositivos con Plug and Play universal
http://www.upnp.org/Device_Architecture_v0.92_.htm

RFC 2616
http 1.1. Solicitud IETF para comentarios.
http://search.ietf.org/rfc/rfc2616.txt?number=2616

RFC 2279
UTF-8, un formato de transformación de ISO 10646 (codificador de caracteres). Solicitud IETF para comentarios.
http://search.ietf.org/rfc/rfc2279.txt?number=2279

XML
Lenguaje de marcación extensible. Recomendación W3C.
http://www.w3.org/XML/

Auto IP
Seleccione automáticamente una dirección IP en una red IPv4 Ad-Hoc. Proyecto IETF.
http://search.ietf.org/internet-drafts/draft-ietf-dhc-ipv4-autoconfig-05.txt.

RFC1034
Nombres de dominio — Conceptos y recursos. Solicitud IETF para comentarios.
http://search.ietf.org/rfc/rfc1034.txt?number=1034

RFC1035
Nombres de dominio – Implementación y especificación. Solicitud IETF para comentarios.
http://search.ietf.org/rfc/rfc1035.txt?number=1035

RFC 2131
DHCP. Solicitud IETF para comentarios.
http://search.ietf.org/rfc/rfc2131.txt?number=2131

RFC 2136
Actualizaciones dinámicas en el sistema de Nombre de Dominio. Solicitud IETF para comentarios.
http://search.ietf.org/rfc/rfc2136.txt?number=2136

Actualizaciones DNS dinámicas por clientes y servidores DHCP
Interacción entre DHCP y DNS. Proyecto de IETF.
http://search.ietf.org/internet-drafts/draft-ietf-dhc-dhcp-dns-12.txt

GENA
Arquitectura de notificación de eventos generales. Proyecto de IETF.

HTTPMU, HTTPU
Transmisión múltiple de HTTP a través de UDP, transmisión única de HTTP a través de UDP. Proyecto de IETF.

SSDP
Protocolo de descubrimiento de servicio sencillo. Proyecto de IETF.

FXPP
Perfil del procesamiento XML flexible.
Especifica qué elementos XML desconocidos y sus subelementos se deben ignorar.
Proyecto de IETF.

RFC 1123
Incluye formato para fechas, por ejemplo, encabezado HTTP FECHA.
Solicitud IETF para comentarios.
http://search.ietf.org/rfc/rfc1123.txt?number=1123

RFC 1766
Formato para la etiqueta de idioma, por ejemplo, ENCABEZADO HTTP ACEPTAR - idioma.
Solicitud IETF para comentarios.
http://search.ietf.org/rfc/rfc1766.txt?number=1766

RFC 2387
Formato para representar el tipo de contenido; por ejemplo, un elemento mimetipo para un icono.
Solicitud IETF para comentarios.
http://www.ietf.org/rfc/rfc2387.txt?number=2387

UPC
Código de producto universal. 12 dígitos, todos códigos numéricos que identifican el paquete del consumidor. Administrado por el Consejo de Códigos Uniformes.
http://www.uc-council.org/main/ID_Numbers_and_Bar_Codes.html

XML
Lenguaje de marcación extensible. Recomendación W3C.
http://www.w3.org/XML/

Esquema de XML (Parte 1: Estructuras, Parte 2: Tipos de datos)
Gramática que define el lenguaje de la plantilla UPnP. Se define utilizando XML.
Proyecto en desarrollo W3C.
Parte 1: Estructuras http://www.w3.org/TR/xmlschema-1/
Parte 2: Tipos de datos http://www.w3.org/TR/xmlschema-2/

HTML
Lenguaje de marcación de hipertexto. Recomendación W3C.
http://www.w3.org/MarkUp/

Marco de extensión HTTP
Describe un mecanismo de extensión genérico para HTTP.
Solicitud de W3C para comentarios.
http://www.w3.org/Protocols/HTTP/ietf-http-ext/

SOAP
Protocolo de acceso de objetos sencillo.
Define un protocolo en XML, sobre HTTP, para llamadas de procedimiento remoto.
Proyecto de IETF y Reporte técnico de W3C.

Anexo C

Especificación de la Arquitectura Salutation

Extracto de "Salutation Architecture Specification V2.1(Part 1)" traducido por Jorge García Ruiz.

Introducción

Objetivos de la Arquitectura

La Arquitectura Salutation fue creada para resolver problemas de descubrimiento de servicio y utilización de un conjunto amplio de aparatos y equipos en un ambiente de conectividad extendida y movilidad.

La arquitectura provee un método estándar para aplicaciones, servicios y dispositivos para describir y anunciar sus capacidades a otras aplicaciones, servicios y dispositivos y para averiguar sus capacidades. La arquitectura permite también a aplicaciones, servicios y dispositivos buscar otras aplicaciones, servicios o dispositivos con una capacidad particular, y pedir establecer sesiones de interoperabilidad con ellos para utilizar sus capacidades.

Dado la diversa naturaleza de aparatos y equipos en un ambiente de conectividad extendida, arquitectura del procesador, sistema operativo, e independiente protocolo de comunicación, y permite aplicaciones escalables, incluso en los mismos dispositivos de bajo-precio.

Definición de términos

Unidad Funcional Todas Las Llamadas

El registro Unidad Funcional Todas Las Llamadas es un campo reservado usado para especificar una petición de capacidad buscada acerca de todas las unidades funcionales registradas en un administrador manager.

Equipo

Es un dispositivo físico o aparato informático, tal como pero no limitado a, una impresora, maquina de fax (FAX), teléfono, central telefónica privada (PBX), asistente personal digital (PDA) o computador (de cualquier tamaño).

Unidad Funcional

Es una subdivisión lógica de Equipamiento que realiza una tarea determinada tal como "IMPRESORA", "FAX" o "OCR".

Entidad de red

Es un dispositivo, aplicación, servicio o unidad funcional que tiene acceso a o puede acceder desde otras aplicaciones, servicios o dispositivos. Una entidad de red puede existir en el mismo equipamiento o en uno diferente conectado a una red.

Aplicación Salutation

Es un programa aplicación ejecutado en el equipamiento que provee servicio(s) y/o usa el servicio(s) de otro Equipo bajo el protocolo definido por la arquitectura Salutation.

Arquitectura Salutation

El asunto de esta especificación, la arquitectura Salutation es un marco de trabajo creado para resolver problemas de descubrimiento de servicio y utilización de un amplio conjunto de equipamiento.

Aplicación Salutation Cliente / Servicio

Una Aplicación Salutation Cliente usa Servicios de otros Equipos bajo el protocolo definido por la arquitectura Salutation.

Una Aplicación Salutation Servicio provee servicios a otro Equipo a través del protocolo definido por la arquitectura Salutation. Servicio es un término genérico que describe una funcionalidad o recurso proveído por una pieza de Equipo a otra. Algunos ejemplos de Servicios son:

- Imprimir un documento
- Enviar un documento por fax
- Hacer accesible (para mirar, agregar, actualizar, eliminar, etc) una libreta de direcciones (nombre, dirección, número telefónico,etc.) desde otra pieza de Equipo (o aplicación).
- Enviar un mensaje de voz
- Proveer una imagen para transformar vía reconocimiento de caracteres ópticos (OCR) en caracteres.
- Convertir un Disco de Video Digital (DVD) a una imagen de video análoga.

A veces una Aplicación Salutation puede funcionar como un servicio, otras veces como un Cliente, o como ambos al mismo tiempo. Colectivamente, esto es llamado una Aplicación Salutation Cliente / Servicio.

Equipo Salutation Cliente / Servicio

El equipo Salutation Cliente es un equipo que tiene más de una aplicación Salutation cliente ejecutándose.

El equipo Salutation Servicio es un equipo que tiene más de una aplicación Salutation servicio ejecutándose

A veces el mismo equipo puede funcionar como servicio, como cliente, o ambas al mismo tiempo. Generalmente, este es llamado Equipo Salutation Cliente / Servicio.

Apreciación de la arquitectura

Administrador Salutation

Un alto nivel de apreciación es incluido en esta sección a través de una descripción de corredores de tareas de servicios.

General

Como muestra la figura 2-1, la arquitectura Salutation define una entidad llamada Administrador Salutation (SLM) este funciona como un corredor de servicios para aplicaciones, servicios y dispositivos llamados Entidades de Red. El administrador Salutation permite a las Entidades en Red descubrir y utilizar las capacidades de otras Entidades en red.

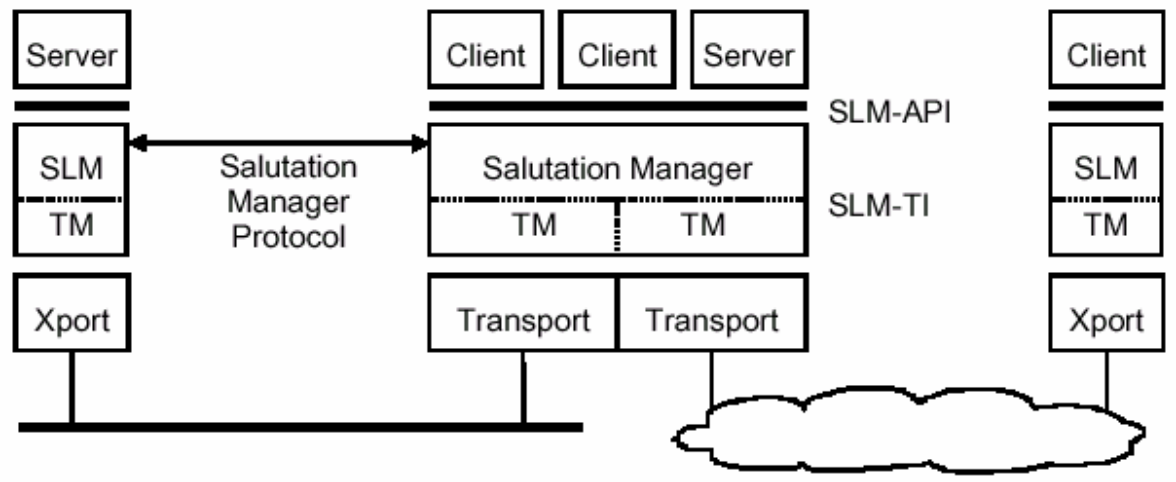


Figure 2-1: Model of the Salutation Manager

Una entidad de red puede ser un proveedor de servicio, llamado Servicio. El Servicio registra su capacidad con un Administrador Salutation. Una Entidad Conectada a una red de computadoras puede ser un servicio usuario, llamado un Cliente. El Cliente descubre los Servicios y pide usarlos a través de un Administrador Salutation. Una Entidad Conectada una red de computadoras puede servir como un Cliente o un Servicio, o ambos.

El Administrador Salutation provee una interfaz de transporte independiente, llamada Aplicación Interfaz de programa de Administrador Salutation (SLM-API), para servicios y clientes. La arquitectura define un abstracto procedural SLM-API.

El administrador Salutation se comunica con otros Administradores Salutation para cumplir el rol de corredor de servicios. El protocolo de comunicación entre Administrador Salutation y Administrador Salutation es definido por la Arquitectura Salutation y llamada Protocolo Administrador Salutation.

El Protocolo Administrador Salutation usa las Llamadas de Procesamiento Remoto en una Red de Computación Abierta de Sun Microsystems.

El protocolo Administrador Salutation requiere que el protocolo de transporte subyacente soporte múltiples comunicaciones bidireccionales.

Cada Administrador Salutation tiene un único identificador universal, SLM-ID. El SLM-ID es una cadena de octetos usada por Clientes, Servicios y Administradores

Salutation para identificar únicamente un Administrador Salutation de forma independiente del transporte.

El Administrador Salutation provee una interfaz independiente del transporte, llamada Interfaz de Transporte Administrador Salutation (SLM-TI), para dependientes del transporte son llamadas Administradores de Transporte. El administrador Transporte es introducido para hacer el Administrador Saluttion independiente del transporte. El Administrador Salutation y Administrador de Transporte juntos realizan el rol de corredor de servicios. En la especificación actual, no hay una descripción específica respecto al Administrador de Transporte. En futuras versiones, serán definidas separadamente la parte común de SLM y Administrador de Transporte y SLM-TI será definido conectado a esas partes.

Un administrador Salutation trabaja con uno o más Administradores de transporte. Cada Administrador de Transporte descubre otros Administradores Salutation conectados al transporte que lo soporta. Por cada uno descubierto, el Administrador Transporte busca el SLM-ID del Administrador Salutation, registrando el SLM-ID con el Administrador Salutation local, y manteniendo la asociación entre la dirección de transporte y el SLM-ID del Administrador Salutation.

El Administrador Transporte descubre otros Administradores Salutation de gran número de formas:

- El Administrador Transporte tiene una tabla estática de las direcciones de transporte de Administradores Salutation remotos. El formato y estructura de la tabla está fuera del alcance de esta arquitectura.

- El Administrador de Transporte realiza una consulta broadcast para buscar otros Administradores Salutation usando el protocolo definido por esta arquitectura. (Este método es posible sólo si el transporte suporta el mecanismo broadcast).

- Si hay un servidor central que contiene el directorio de equipos Salutation, Administrador Transporte puede consultar la dirección de transporte de otros Administradores Salutation. Aunque el protocolo entre el Administrador Transporte y tal directorio está fuera del alcance de esta Arquitectura.

- El Cliente especifica el tipo de transporte y la dirección de transporte del Administrador Salutation remoto a través del SLM-API.

Cada pieza del Equipo tiene a lo menos un Administrador Salutation. Cuando no hay Administrador Salutation, Clientes / Servicios pueden usar un Administrador Salutation remoto, a un Administrador Salutation en otra pieza de Equipo, a través de Llamadas a Procedimientos Remotos tal como muestra la figura 2-2. Depende de la implementación de cada Administrador Salutation si proporciona o no la interfaz de Llamada de Procedimiento Remota a los Clientes remotos.

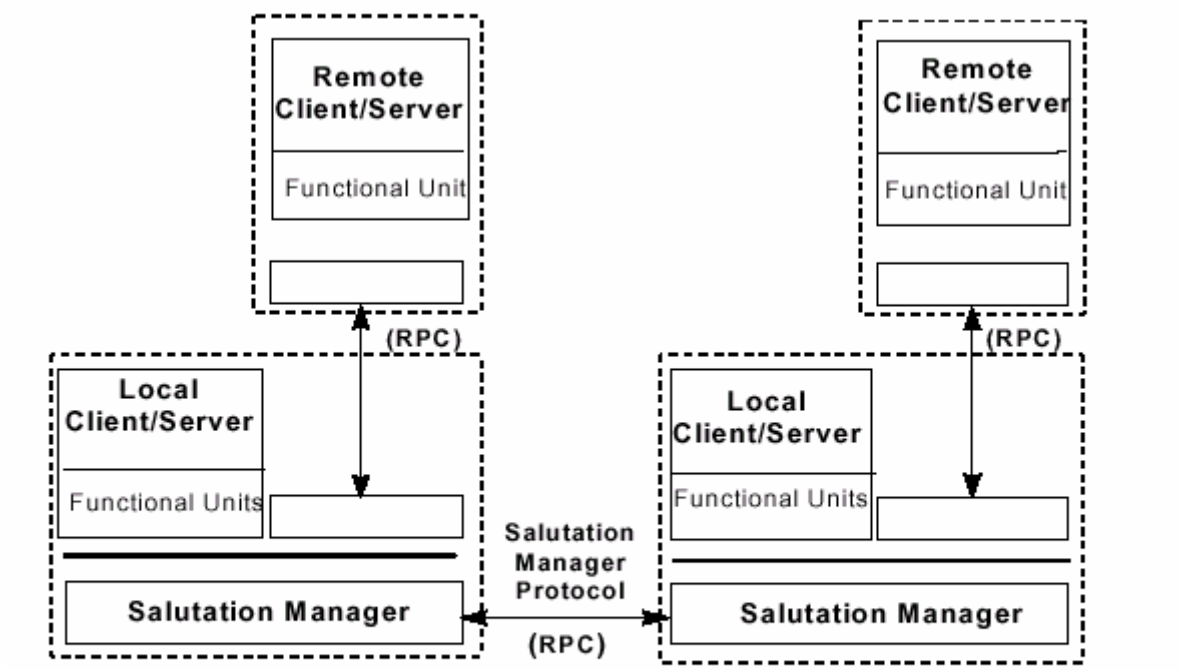


Figure 2.2: Use of Remote Procedure Call for Remote Clients/Services

Servicio Corredor de Tareas

Para realizar la función de servicio corredor, el Administrador Salutation provee cuatro tareas básicas:

- Servicio Registro.
- Servicio Descubrimiento.
- Servicio Disponibilidad.
- Servicio Administración de Sesión.

Servicio Registro

El Administrador Salutation contiene un registro que posee información de Servicios. El mínimo requerimiento para el registro es guardar información acerca de los servicios conectados al Administrador Salutation. Como describe la figura 2-2, estos servicios pueden residir en el Equipo Salutation local o puede conectarse al Administrador Salutation vía llamadas a procedimiento remoto.

Opcionalmente, el registro Administrador Salutation puede almacenar información de servicios que están registrados en otros Administradores Salutation. Esto expande la funcionalidad del registro permitiendo al Administrador Salutation mantener un "directorio" de Equipo Salutation que es el ambiente local. Por ejemplo, se el Administrador Salutation local mantiene un servicio de impresión, el Registro puede incluir información acerca de todos los servicios de impresión.

El Registro Administrador Salutation puede también servir en la función de directorio de red, proveyendo información acerca de todo el Equipo Salutation en el rango del Administrador Salutation, sin tomar en cuenta el tipo de Equipo. En este caso, uno de los Administradores Salutation en una red se designaría como el directorio central. Este tendrá la responsabilidad de buscar y registrar todo el Equipo Salutation. Todas las peticiones de otros equipos por recursos Salutation serán redireccionadas hacia este Administrador Salutation que responderá apropiadamente.

Servicio Descubrimiento

El Administrador Salutation puede descubrir otros Administradores Salutation remotos y determinar los Servicios ahí registrados. El Servicio Descubrimiento es utilizado para

comparar un tipo de servicio requerido, especificado por un Administrador Salutation, con los Tipos de Servicio disponibles en un Administrador Salutation remoto. Las Llamadas a Procedimiento remoto son usadas para transmitir el tipo de servicio requerido desde el Administrador Salutation local al Administrador Salutation remoto y transmitir la respuesta desde el Administrador Salutation al Administrador Salutation local. Utilizando la especificación de Servicios requeridos, el Administrador Salutation puede determinar:

- Las características de todos los servicios registrados en un Administrador Salutation remoto.
- Las características de un servicio específico registrado en un Administrador Salutation remoto.
- La presencia de un servicio en un Administrador Salutation remoto que satisfaga determinado conjunto de características.

Servicio Disponibilidad

El Administrador Salutation puede periódicamente chequear la disponibilidad de un Servicio. El Administrador Salutation local demanda al Administrador Salutation apropiado realizar un chequeo de disponibilidad. El chequeo de disponibilidad es realizado por intercambio de mensajes de llamada a procedimiento remoto entre los Administradores Salutation. El periodo del chequeo de disponibilidad es especificable.

Servicio Administrador de Sesión

Cuando un cliente desea usar un Servicio proveído por un equipo Salutation, el Administrador Salutation puede establecer un tubo virtual de datos entre un Cliente y un Servicio. Esto es llamado un Servicio Sesión. Comandos, respuestas y datos son intercambiados entre Clientes y Servicios en esta tubo de datos en bloques llamados

mensajes. Los mensajes tienen un formato definido y son intercambiados bajo un protocolo definido. Tales definiciones de formato de intercambio de mensaje y protocolo son llamado Protocolos Personalidad. Los Administradores Salutation pueden ser determinados para operar en uno de tres Protocolos de Personalidad distintos. Este tubo de datos en mostrado en la figura 2-3.

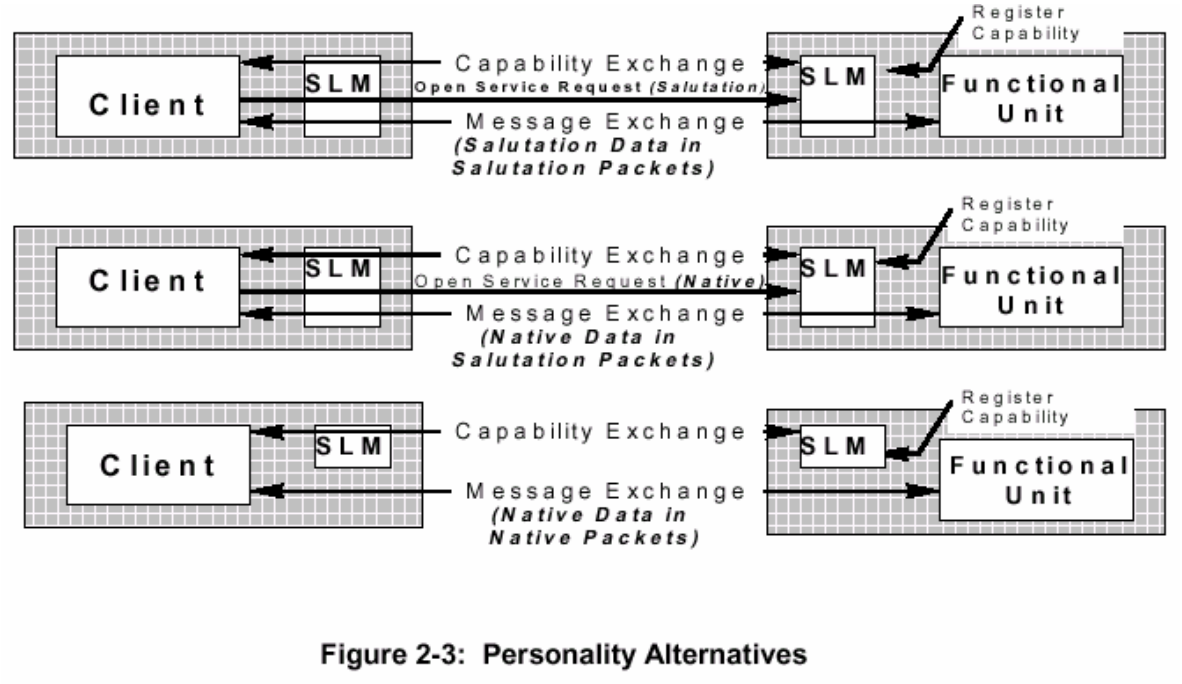


Figure 2-3: Personality Alternatives

- El Administrador Salutation puede preparar el tubo de datos luego andar en background, permitiendo al Cliente y Servicio administrar las cadenas de mensajes y formatos de datos. Esto es conocido como Personalidad Nativa. Esta personalidad es útil cuando Administrador Salutation es usado únicamente para descubrir las capacidades de otras Entidades de red, con las aplicaciones, servicios y dispositivos administrando las interacciones entre Clientes y Servicios descubiertos. Los mensajes son intercambiados entre Clientes y Servicios directamente, sin necesitar el Administrador Salutation. Los menajes no son transportados por el Protocolo Administración Salutation bajo un Protocolo Personalidad Nativo. El intercambio de mensaje es dato nativo en paquetes nativo. Note: Sin embargo el Protocolo Administrador Salutation no es usado con el Protocolo Personalidad Nativo.

Anexo D

Service Location Protocol, Version **2**

Extracto de "Service Location Protocol, Version 2" traducido por Jorge García Ruiz.

Abstracto

El Protocolo de Localización de Servicios (SLP, Service location Protocol) proporciona un escalable marco de trabajo para el descubrimiento y selección de servicios en red. Usando este protocolo, computadores que usan Internet necesitan poca o no estática configuración de servicios en red en su arquitectura actual. Este protocolo es especialmente importante para computadores más portables y usuarios menos tolerantes o responsables de cumplir las demandas del administrador de sistemas de red.

Introducción

El Protocolo de Localización de Servicios (SLP) provee un flexible y escalable marco de trabajo para proveer dispositivo con acceso a información acerca de la existencia, localización, y configuración de servicios en red.

Tradicionalmente, los usuarios tenian que buscar servicios conociendo el nombre del dispositivo de red (una frase de texto humanamente entendible) que es un alias para una dirección de red. SLP elimina la necesidad de que un usuario conozca el nombre de un dispositivo de red proveyendo un servicio. Más bien, el usuario proporciona el tipo de servicio deseado y un conjunto de atributos que describen el servicio. Basado en este descripción, el Protocolo de Localización de Servicios resuelve la dirección de red del servicio por el usuario.

SLP provee un mecanismo de configuración dinámico para aplicaciones en redes de área local. Las aplicaciones son modeladas como clientes que necesitan buscar servicios vinculados a cualquiera de las redes disponibles en una empresa. Para los casos donde hay muchos diferentes clientes y/o servicios disponibles, el protocolo es ajustado para hacer uso de Agentes de Directorio (DA, Directory Agents) que ofrecen un repositorio centralizado para el anuncio de servicios.

Declaración De Aplicabilidad

SLP está propuesto para funcionar en redes bajo un control administrativo cooperativo. Estas redes permiten una política de implementación considerando seguridad, ruteo multicast y organización de servicios y clientes en grupos que no son factibles, por ejemplo, en la escala de Internet.

SLP ha sido diseñada para servir en redes empresariales que comparten servicios, no siendo necesariamente escalable para descubrimiento de servicios en redes de área ancha a través de Internet, o en redes donde hay cientos o miles de clientes o decenas de miles de servicios.

Terminología

Agente Usuario (UA, User Agent): Un proceso trabajando a nombre de usuario para establecer contacto con algún servicio. Los UA obtienen información de servicio desde Agentes Servicio o Agentes Directorio

Agente Servicio (SA, Service Agent): Un proceso trabajando a nombre de uno o más servicios para anunciarlos.

Agente Directorio (DA, Directory Agent): Un proceso que recopila anuncios de servicios. Sólo puede haber un DA presente por cada dispositivo.

Tipo de Servicio: Cada tipo de servicio tiene una única secuencia de caracteres que lo identifica.

Autoridad de Nombramiento: La agencia o grupo que cataloga los tipos de servicio y sus atributos. La Autoridad de Nombramiento actual es IANA [1].

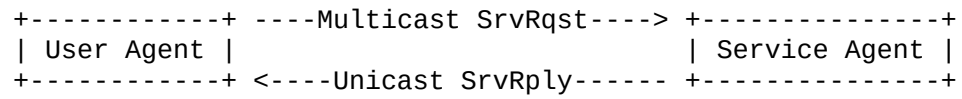
Ámbito: Un conjunto de servicios, típicamente creado para un grupo administrativo lógico.

Apreciación Del Protocolo

El Protocolo de Localización de Servicios apoya un marco de trabajo donde las aplicaciones clientes son modeladas como “Agentes Usuario” y servicios son anunciados por “Agentes Servicio”. Una tercera entidad, llamada “Agente Directorio” provee escalabilidad al protocolo.

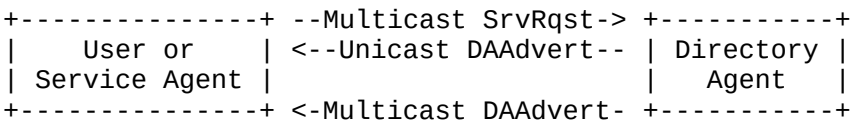
El Agente Usuario emite una “Petición de Servicio” (SrvRqst, Service Request) a nombre de la aplicación cliente, especificando las características de el servicio que el cliente requiere. El Agente Usuario recibirá una “Respuesta de Servicio” (SrvRply, Service Reply) especificando la localización de todos los servicios en la red que satisfagan la petición.

El marco de trabajo del Protocolo de Localización de Servicios permite al Agente Usuario emitir directamente peticiones a los Agentes Servicio. En este caso la petición es multicast. Los Agentes de Servicio que reciben la petición para un servicios que ellos anuncian emiten una respuesta unicast que contiene la localización del servicio.



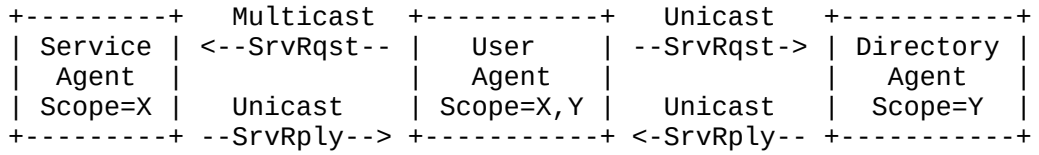
En grandes redes, uno o más agentes directorios son usados. El agente directorio funciona como un cache. Agentes servicio envían mensajes de registro (SrvReg) que contienen todos los servicios que él anuncia para los Agentes Directorio y recibe reconocimientos en respuesta (SrvAck). Esos anuncios deben ser actualizados con el Agente Directorio o expirarán. Agentes Usuario piden por unicast a los Agentes Directorio en lugar de a los Agentes Servicio si algún Agente Directorio es conocido.

Los Agentes Usuario y Servicio descubren Agentes Directorio de dos maneras. Primero, ellos emiten una Petición multicast de Servicio para el "Agente Directorio" cuando se inicializan. Segundo, el Agente Directorio envía un no solicitado anuncio infrecuentemente, que los Agentes Usuario y Servicio escuchan. En ambos casos los agentes reciben un anuncio DA (DAAdvert).



Los Servicios son agrupados usando "alcances". Estos son cadenas de caracteres que identifican servicios que están administrativamente identificados. Un alcance podría indicar una ubicación, agrupamiento administrativo, proximidad en una topología de red o algunas otras categorías. Los agentes servicio y agentes directorio estas siempre asignados a un alcance.

Un agente usuario es normalmente asignado a un alcance (en este caso el agente usuario estará solo habilitado para descubrir ese grupo particular de servicios). Esto permite a un administrador de red "abastecer" servicios a usuarios. Alternativamente, el agente usuario puede ser configurado con un alcance completo. En tal caso, descubrirá todos los alcances disponibles y permitirá a la aplicación cliente emitir peticiones para cualquier servicio disponible en la red.



En la ilustración anterior, los agentes usuario están configurados con alcance X y Y. Si un servicio es buscado en alcance X, la petición es multicast. Si este es buscado en alcance Y, la petición es unicast a el DA.

Finalmente, si la petición está hecha para ambos alcances, tendrá que ser ambos unicast y multicast.

Agentes servicio y agentes usuario pueden verificar las firmas digitales proveídas con los anuncios del DA. Agentes usuario y agentes directorio pueden verificar la información de servicio registrada por agentes servicio. El material codificado para usado para verificar firmas digitales es identificado usando un índice de parámetro de seguridad SLP, o SLP SPI.

Cada host configurado para generar una firma digital incluye el SLP SPI usado para verificar este en la autentificación del bloque a transmitido.

Cada host que puede verificar una firma digital debe ser configurado con material codificado y otros parámetros correspondientes con el SLP SPI de tal modo que pueda funcionar verificando los cálculos.

Agentes Servicio (AS) aceptan peticiones de servicio multicast y unicast. AS pueden aceptar otras peticiones (peticiones de un tipo o atributo de servicio). AS deben escuchar los anuncios multicast de los agentes Directorio (AD).

Las características descritas arriba deben ser implementadas . Una mínima implementación consiste de un Agente Usuario, Agente Servicio o ambos.

Hay varias características opcionales en el protocolo. Note que DA deben soportar todos estos mensajes tipo, pero DA su apoyo es opcional desplegado en redes usando SLP.

AU y AS pueden soportar esos mensajes tipos. Estas operaciones son principalmente para uso interactivo (hojeando o poniendo al día los registros de servicio selectivamente). AU y AS cualquiera los apoya o no dependiendo de los requisitos y restricciones del ambiente en el cual serán usados.

Petición de servicio tipo: Una petición para todos los tipos de servicio en la red. Esto permite construir un navegador de servicios genéricos.

Respuesta de servicio tipo: Una respuesta a la petición de servicio tipo.

Petición Atributo: Una petición por atributos de un tipo de servicio dado o atributos de un servicio dado.

Respuesta Atributo: Una respuesta a una petición atributo.

Desregistrar Servicio: Una petición para desregistrar un servicio o algunos atributos de un servicio.

Actualizar Servicio: Una subsiguiente Petición Servicio (SrvRqst) para un anuncio. Esto permite poner al día los atributos dinámicos individuales.

Anuncio Agente Servicio: En la ausencia de un agente directorio, un agente usuario puede pedir Agentes servicio para descubrir su canfiguración de alcance. El agente usuario puede usar ese alcance en peticiones.

En la ausencia de apoyo Multicast, Broadcast puede ser usado. La ubicación de DA puede ser estáticamente configurada, descubierta usando SLP como se describió anteriormente, o configurada usando DHCP.

Una implementación de SLPv2 debe soportar SLPv1.

Anexo E

Código Fuente Proyecto Vistazo

Ejemplo Unicast Registrar

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import java.rmi.RMISecurityManager;

public class UnicastRegister  {

   static public void main(String argv[]) {
      new UnicastRegister();
   }

   public UnicastRegister() {
          LookupLocator lookup = null;
          ServiceRegistrar registrar = null;

          System.setSecurityManager(new RMISecurityManager());

      try {
         lookup = new LookupLocator("jini://localhost");
      } catch(java.net.MalformedURLException e) {
         System.err.println("Lookup fallado: " + e.toString());
            System.exit(1);
      }

          try {
             registrar = lookup.getRegistrar();
          } catch (java.io.IOException e) {
          System.err.println("Busqueda Registrar fallada: " + e.toString());
             System.exit(1);
          } catch (java.lang.ClassNotFoundException e) {
          System.err.println("Busqueda Registrar fallada: " + e.toString());
             System.exit(1);
          }
          System.out.println("Registrar encontrado");

   }
}
```

Ejemplo Multicast Registrar

```
import net.jini.discovery.LookupDiscovery;
import net.jini.discovery.DiscoveryListener;
import net.jini.discovery.DiscoveryEvent;
import net.jini.core.lookup.ServiceRegistrar;

public class MulticastRegister implements DiscoveryListener {
    static public void main(String argv[]) {
       new MulticastRegister();
           try {
              Thread.currentThread().sleep(10000L);
           } catch(java.lang.InterruptedException e) {}
    }

    public MulticastRegister() {
           System.setSecurityManager(new java.rmi.RMISecurityManager());
       LookupDiscovery discover = null;
       try {
          discover = new LookupDiscovery(LookupDiscovery.ALL_GROUPS);
       } catch(Exception e) {
          System.err.println(e.toString());
              e.printStackTrace();
              System.exit(1);
       }
       discover.addDiscoveryListener(this);
    }

    public void discovered(DiscoveryEvent evt) {

       ServiceRegistrar[] registrars = evt.getRegistrars();

       for (int n = 0; n < registrars.length; n++) {
              ServiceRegistrar registrar = registrars[n];
              System.out.println("Lookup encontrado");
                     }
    }

           public void discarded(DiscoveryEvent evt) { }
}
```

Ejemplo Servicio Simple

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import java.io.Serializable;
import java.rmi.RMISecurityManager;

public class SimpleService implements Serializable {

   static public void main(String argv[]) {
      new SimpleService();
   }

   public SimpleService() {
          LookupLocator lookup = null;
          ServiceRegistrar registrar = null;
          System.setSecurityManager(new RMISecurityManager());
      try {
         lookup = new LookupLocator("jini://localhost");
      } catch(java.net.MalformedURLException e) {
         System.err.println("Lookup fallado: " + e.toString());
            System.exit(1);
      }

          try {
             registrar = lookup.getRegistrar();
          } catch (java.io.IOException e) {
          System.err.println("Busqueda Registrar fallada: " + e.toString());
             System.exit(1);
          } catch (java.lang.ClassNotFoundException e) {
          System.err.println("Busqueda Registrar fallada: " + e.toString());
             System.exit(1);
          }
          System.out.println("Encontrado un registrar");

          ServiceItem item = new ServiceItem(null, this, null);
          ServiceRegistration reg = null;
          try {
             reg = registrar.register(item, 10000000L);
          } catch(java.rmi.RemoteException e) {
             System.err.println("Register excepcion: " + e.toString());
          }
          System.out.println("Servicio registrado con id: " +
                                reg.getServiceID());
   }
}
```

MyHelloServer

```
import java.rmi.RMISecurityManager;
import net.jini.discovery.LookupDiscovery;
import net.jini.discovery.DiscoveryListener;
import net.jini.discovery.DiscoveryEvent;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import net.jini.core.lease.Lease;
import net.jini.core.lookup.ServiceID ;
import net.jini.lease.LeaseListener;
import net.jini.lease.LeaseRenewalEvent;
import net.jini.lease.LeaseRenewalManager;
import java.io.*;


public class MyServer implements DiscoveryListener,
                                 LeaseListener {

   protected LeaseRenewalManager leaseManager = new LeaseRenewalManager();
   protected ServiceID serviceID = null;
   protected      MyServerInterfazImpl impl;

   public static void main(String argv[]) {
          MyServer s = new MyServer();

          Object keepAlive = new Object();
          synchronized(keepAlive) {
             try {
                    keepAlive.wait();
             } catch(java.lang.InterruptedException e) { }
          }
   }

   public MyServer() {
          impl = new MyServerInterfazImpl();

          DataInputStream din = null;
          try {
             din = new DataInputStream(new FileInputStream("MyServerInterfaz.id"));
             serviceID = new ServiceID(din);
          } catch(Exception e) {}

       System.setSecurityManager(new RMISecurityManager());

          LookupDiscovery discover = null;
```

```
    try {
       discover = new LookupDiscovery(LookupDiscovery.ALL_GROUPS);
    } catch(Exception e) {
       System.err.println("Discovery fallado " + e.toString());
       System.exit(1);
    }

    discover.addDiscoveryListener(this);
}

public void discovered(DiscoveryEvent evt) {

    ServiceRegistrar[] registrars = evt.getRegistrars();

    for (int n = 0; n < registrars.length; n++) {
       ServiceRegistrar registrar = registrars[n];

          ServiceItem item = new ServiceItem(serviceID,
                                             impl,
                                             null);
          ServiceRegistration reg = null;
          try {
                reg = registrar.register(item, Lease.FOREVER);
          } catch(java.rmi.RemoteException e) {
                System.err.println("Register excepcion: " + e.toString());
                continue;
          }
          System.out.println("Servicio registrado con id " + reg.getServiceID());

          leaseManager.renewUntil(reg.getLease(), Lease.FOREVER, this);

          if (serviceID == null) {
                serviceID = reg.getServiceID();

                DataOutputStream dout = null;
                try {
                   dout = new DataOutputStream(new FileOutputStream
                         ("MyServerInterfaz.id"));
                   serviceID.writeBytes(dout);
                   dout.flush();
                } catch(Exception e) {}
          }
       }
}

public void discarded(DiscoveryEvent evt) {  }

public void notify(LeaseRenewalEvent evt) {
       System.out.println("Lease expired " + evt.toString());  } }
```

ClienteHelloUnicast

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import java.rmi.RMISecurityManager;
import net.jini.core.lookup.ServiceTemplate;

public class MyClientUnicast {
   public static void main(String argv[]) {
          new MyClientUnicast();
   }

   public MyClientUnicast() {
          LookupLocator lookup = null;
          ServiceRegistrar registrar = null;
          MyServerInterfaz myinterfaz = null;
      try {
         lookup = new LookupLocator("jini://delfin");
      } catch(java.net.MalformedURLException e) {
         System.err.println("Lookup fallado: " + e.toString());
            System.exit(1);
      }

          System.setSecurityManager(new RMISecurityManager());

          try {
             registrar = lookup.getRegistrar();
          } catch (java.io.IOException e) {
          System.err.println("Busqueda Registrar fallada: " + e.toString());
             System.exit(1);
          } catch (java.lang.ClassNotFoundException e) {
          System.err.println("Busqueda Registrar fallada: " + e.toString());
             System.exit(1);
          }

          Class[] classes = new Class[] {MyServerInterfaz.class};
          ServiceTemplate template = new ServiceTemplate(null, classes, null);
          try {
             myinterfaz = (MyServerInterfaz) registrar.lookup(template);
          } catch(java.rmi.RemoteException e) {
             e.printStackTrace();
             System.exit(1);
          }

          if (myinterfaz == null) {
             System.out.println("myinterfaz null");
```

```
            System.exit(2);
        }

        try {
            System.out.println
        ("Cliente: Llamando a sayHello() --> " + myinterfaz.sayHello() +"<--");
        } catch(java.rmi.RemoteException e) {
            System.err.println(e.toString());
        }
        System.exit(0);
    }
}
```

ClienteHelloMulticast

```
import java.rmi.RMISecurityManager;
import net.jini.discovery.LookupDiscovery;
import net.jini.discovery.DiscoveryListener;
import net.jini.discovery.DiscoveryEvent;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceTemplate;

public class MyClientMulticast implements DiscoveryListener {

   public static void main(String argv[]) {
          new MyClientMulticast();
     try {
         Thread.currentThread().sleep(100000L);
      } catch(java.lang.InterruptedException e) {}
   }

   public MyClientMulticast() {
          System.setSecurityManager(new RMISecurityManager());

          LookupDiscovery discover = null;
      try {
         discover = new LookupDiscovery(LookupDiscovery.ALL_GROUPS);
      } catch(Exception e) {
         System.err.println(e.toString());
         System.exit(1);
      }
      discover.addDiscoveryListener(this);
   }

   public void discovered(DiscoveryEvent evt) {

   ServiceRegistrar[] registrars = evt.getRegistrars();
          Class [] classes = new Class[] {MyServerInterfaz.class};
          MyServerInterfaz myinterfaz = null;
          ServiceTemplate template = new ServiceTemplate(null, classes,
                                                         null);

      for (int n = 0; n < registrars.length; n++) {
             System.out.println("Servicio encontrado");
         ServiceRegistrar registrar = registrars[n];
             try {
                    myinterfaz = (MyServerInterfaz) registrar.lookup(template);
             } catch(java.rmi.RemoteException e) {
                    e.printStackTrace();
                    continue;
             }
```

```
            if (myinterfaz == null) {
                  System.out.println("myinterfaz null");
                  continue;
            }
            try {
            System.out.println
                  ("Cliente: Llamando a sayHello() --> " + myinterfaz.sayHello() +"<--");
         } catch(java.rmi.RemoteException e) {
            System.err.println(e.toString());
         }
         System.exit(0);
         }
   }

   public void discarded(DiscoveryEvent evt) {  }
}
```

MyServerRMI

```
import net.jini.discovery.LookupDiscovery;
import net.jini.discovery.DiscoveryListener;
import net.jini.discovery.DiscoveryEvent;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import net.jini.core.lease.Lease;
import net.jini.lease.LeaseRenewalManager;
import net.jini.lease.LeaseListener;
import net.jini.lease.LeaseRenewalEvent;
import java.rmi.RMISecurityManager;
import net.jini.config.*;
import net.jini.export.*;
import java.io.*;


public class MyServerRMI implements DiscoveryListener, LeaseListener {

        protected LeaseRenewalManager leaseManager = new LeaseRenewalManager();
        protected MyServerInterfazImpl impl;
   protected RemoteMyServerInterfaz proxy;

   private static String CONFIG_FILE = "My_server_rmi.config";

   public static void main(String argv[]) {
        new MyServerRMI();

        Object keepAlive = new Object();
        synchronized(keepAlive) {
           try {
                keepAlive.wait();
           } catch(java.lang.InterruptedException e) {
                }
        }
   }

   public MyServerRMI() {
        try {
           impl = new MyServerInterfazImpl();
        } catch(Exception e) {
        System.err.println("Nuevo impl: " + e.toString());
        System.exit(1);
        }

        String[] configArgs = new String[] {CONFIG_FILE};
```

```
        try {
          Configuration config = ConfigurationProvider.getInstance(configArgs);
           Exporter exporter = (Exporter) config.getEntry( "MyServerRMI",
                                                           "exporter",
                                                           Exporter.class);
           proxy = (RemoteMyServerInterfaz) exporter.export(impl);
        } catch(Exception e) {
           System.err.println(e.toString());
           e.printStackTrace();
           System.exit(1);
        }
        System.setSecurityManager(new RMISecurityManager());
        LookupDiscovery discover = null;
    try {
       discover = new LookupDiscovery(LookupDiscovery.ALL_GROUPS);
    } catch(Exception e) {
       System.err.println(e.toString());
       System.exit(1);
    }
    discover.addDiscoveryListener(this);
}

public void discovered(DiscoveryEvent evt) {

ServiceRegistrar[] registrars = evt.getRegistrars();
        RemoteMyServerInterfaz service;

    for (int n = 0; n < registrars.length; n++) {
       ServiceRegistrar registrar = registrars[n];

          ServiceItem item = new ServiceItem(null,
                                             proxy,
                                             null);
           ServiceRegistration reg = null;
           try {
                 reg = registrar.register(item, Lease.FOREVER);
           } catch(java.rmi.RemoteException e) {
                 System.err.print("Register excepcion: ");
                 e.printStackTrace();
                 continue;
           }
           try {
                 System.out.println("Servicio registrado en" +
                                    registrar.getLocator().getHost());
           } catch(Exception e) {
           }
           leaseManager.renewUntil(reg.getLease(), Lease.FOREVER, this);
        }
}
```

```
    public void discarded(DiscoveryEvent evt) {

    }

    public void notify(LeaseRenewalEvent evt) {
            System.out.println("Arrendamiento expirado " + evt.toString());
    }

}
```

MyClientUnicast – byte[ ]

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import java.rmi.RMISecurityManager;
import net.jini.core.lookup.ServiceTemplate;
import java.io.*;
import java.awt.image.*;
import javax.imageio.*;

public class MyClientUnicast {

   public static void main(String argv[]) {
         new MyClientUnicast();
   }

         public MyClientUnicast() {
         LookupLocator lookup = null;
         ServiceRegistrar registrar = null;
         MyServerInterfaz myinterfaz = null;
         BufferedImage input=null;

      try {
         lookup = new LookupLocator("jini://delfin");
      } catch(java.net.MalformedURLException e) {
         System.err.println("Lookup fallado: " + e.toString());
            System.exit(1);
      }

         System.setSecurityManager(new RMISecurityManager());

         try {
            registrar = lookup.getRegistrar();
         } catch (java.io.IOException e) {
         System.err.println("Busqueda Registrar fallada: " + e.toString());
            System.exit(1);
         } catch (java.lang.ClassNotFoundException e) {
         System.err.println("Busqueda Registrar fallada: " + e.toString());
            System.exit(1);
         }

         Class[] classes = new Class[] {MyServerInterfaz.class};
         ServiceTemplate template = new ServiceTemplate(null, classes, null);
         try {
            myinterfaz = (MyServerInterfaz) registrar.lookup(template);
         } catch(java.rmi.RemoteException e) {
```

```
            e.printStackTrace();
            System.exit(1);
         }

         if (myinterfaz == null) {
            System.out.println("myinterfaz null");
            System.exit(2);
         }

         File finput = new File("screen.png");

         byte[] bimage = new byte[(int)finput.length()];
         try
         {
                  FileInputStream in = new FileInputStream("screen.png");
                  in.read(bimage);
         }catch(Exception e){System.out.println(e);}


         try {
            System.out.println("Cliente: Llamando a sayHello() --> " + myinterfaz.sayHello(bimage)
+"<--");
         } catch(java.rmi.RemoteException e) {
            System.err.println(e.toString());
         }
         System.exit(0);
   }
}
```

ClientUnicast – Screenshot

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import java.rmi.RMISecurityManager;
import net.jini.core.lookup.ServiceTemplate;
import java.io.*;

public class ClientUnicast {

   public static void main(String argv[]) {
          new ClientUnicast(argv);
   }

   public ClientUnicast(String argv[]) {
          LookupLocator lookup = null;
          ServiceRegistrar registrar = null;
          ProyectarInterfaz proyectarinterfaz = null;
          byte[] bimage = null;

      try {
         lookup = new LookupLocator("jini://delfin");
      } catch(java.net.MalformedURLException e) {
         System.err.println("Error Lookup: " + e.toString());
            System.exit(1);
      }

          System.setSecurityManager(new RMISecurityManager());

          try {
             registrar = lookup.getRegistrar();
          } catch (java.io.IOException e) {
          System.err.println("Error Registrar: " + e.toString());
             System.exit(1);
          } catch (java.lang.ClassNotFoundException e) {
          System.err.println("Error Registrar: " + e.toString());
             System.exit(1);
          }

          Class[] classes = new Class[] {ProyectarInterfaz.class};
          ServiceTemplate template = new ServiceTemplate(null, classes, null);
          try {
             proyectarinterfaz = (ProyectarInterfaz) registrar.lookup(template);
          } catch(java.rmi.RemoteException e) {
             e.printStackTrace();
             System.exit(1);
```

```
        }

        if (proyectarinterfaz == null) {
           System.out.println("proyectarinterfaz null");
           System.exit(2);
        }
        System.out.println("Voy a screenshot");

        Screenshot screenshot= new Screenshot();

        try{
        bimage = screenshot.Screenshot();
        }catch(java.lang.Exception e) {System.err.println(e.toString());}


        try {
           System.out.println
("Cliente: Llamando a Proyectar() --> " + proyectarinterfaz.Proyectar(bimage) +"<--");
        } catch(java.rmi.RemoteException e) {System.err.println(e.toString());}
        System.exit(0);
    }
}
```

ClientUnicast – ImageIcon

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import java.rmi.RMISecurityManager;
import net.jini.core.lookup.ServiceTemplate;

import java.awt.*;
import java.awt.image.*;
import javax.swing.*;


public class ClientUnicast {

   public static void main(String argv[]) {
          new ClientUnicast(argv);
   }

   public ClientUnicast(String argv[]) {
          LookupLocator lookup = null;
          ServiceRegistrar registrar = null;
          ProyectarInterfaz proyectarinterfaz = null;
          byte[] bimage = null;

      try {
         lookup = new LookupLocator("jini://delfin");
      } catch(java.net.MalformedURLException e) {
         System.err.println("Error Lookup: " + e.toString());
            System.exit(1);
      }

          System.setSecurityManager(new RMISecurityManager());

          try {
             registrar = lookup.getRegistrar();
          } catch (java.io.IOException e) {
          System.err.println("Error Registrar: " + e.toString());
             System.exit(1);
          } catch (java.lang.ClassNotFoundException e) {
          System.err.println("Error Registrar: " + e.toString());
             System.exit(1);
          }

          Class[] classes = new Class[] {ProyectarInterfaz.class};
          ServiceTemplate template = new ServiceTemplate(null, classes, null);
          try {
```

```
            proyectarinterfaz = (ProyectarInterfaz) registrar.lookup(template);
         } catch(java.rmi.RemoteException e) {
            e.printStackTrace();
            System.exit(1);
         }

         if (proyectarinterfaz == null) {
            System.out.println("proyectarinterfaz null");
            System.exit(2);
         }
         System.out.println("Voy a screenshot");
                  Toolkit toolkit = Toolkit.getDefaultToolkit();
                  Dimension screenSize = toolkit.getScreenSize();
                  Rectangle screenRect = new Rectangle(screenSize);
                  BufferedImage imagebuff=null;

                  try{
                  Robot robot = new Robot();
                  imagebuff = robot.createScreenCapture(screenRect);
                  }catch (Exception e) {e.printStackTrace();}
                  System.out.println("Screenshot realizado");

                   ImageIcon image = new ImageIcon(imagebuff);

         try {
            System.out.println
("Cliente: Llamando a Proyectar() --> " + proyectarinterfaz.Proyectar(image) +"<--");
         } catch(java.rmi.RemoteException e) {System.err.println(e.toString());}
         System.exit(0);
   }
}
```

ClientUnicast-TimerTask

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import java.rmi.RMISecurityManager;
import net.jini.core.lookup.ServiceTemplate;
import java.awt.*;
import java.awt.image.*;
import javax.swing.*;
import java.util.Timer;
import java.util.TimerTask;

public class ClientUnicast {
        Timer timer;
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Dimension screenSize = toolkit.getScreenSize();
        Rectangle screenRect = new Rectangle(screenSize);
        BufferedImage imagebuff=null;
        Robot robot = new Robot();
        ProyectarInterfaz proyectarinterfaz = null;

        public static void main(String argv[]) throws Exception{    }

   public ClientUnicast(String argv[]) throws Exception{
                LookupLocator lookup = null;
                ServiceRegistrar registrar = null;

                byte[] bimage = null;
                String urlLookup = argv[0];
                String delay = argv[1];

      try {
         lookup = new LookupLocator(urlLookup);
      } catch(java.net.MalformedURLException e) {
                        System.err.println("Error Lookup: " + e.toString());
                System.exit(1);
         }

                System.setSecurityManager(new RMISecurityManager());

                try {
                registrar = lookup.getRegistrar();
                } catch (java.io.IOException e) {
         System.err.println("Error Registrar: " + e.toString());
                System.exit(1);
                        } catch (java.lang.ClassNotFoundException e) {
```

```
        System.err.println("Error Registrar: " + e.toString());
                System.exit(1);
                                }
                Class[] classes = new Class[] {ProyectarInterfaz.class};
                ServiceTemplate template = new ServiceTemplate(null, classes, null);
                try {
                proyectarinterfaz = (ProyectarInterfaz) registrar.lookup(template);
                } catch(java.rmi.RemoteException e) {e.printStackTrace();
        System.exit(1);}

                if (proyectarinterfaz == null) {
                System.out.println("proyectarinterfaz null");
                System.exit(2);
                }
                System.out.println("Voy a screenshot");

                timer = new Timer();
                timer.schedule(new RemindTask(),0,1*1000);
        }

        class RemindTask  extends TimerTask {
           public void run() {
                while(true){
                try{
                        imagebuff = robot.createScreenCapture(screenRect);
                }catch (Exception e) {e.printStackTrace();}
                System.out.println("Screenshot realizado");
                ImageIcon image = new ImageIcon(imagebuff);
                try {
                        System.out.println("Cliente: Llamando a Proyectar() --> " +
                proyectarinterfaz.Proyectar(image) +"<--");
                } catch(java.rmi.RemoteException e) {System.err.println(e.toString());}
                }
           }
   }
}
```

ClientUnicast – Final

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import java.rmi.RMISecurityManager;
import net.jini.core.lookup.ServiceTemplate;

import net.jini.core.lookup.ServiceMatches;
import net.jini.core.entry.Entry;
import net.jini.core.lookup.ServiceID;
import net.jini.lookup.entry.Name;


public class ClientUnicast {
        ProyectarInterfaz proyectarinterfaz= null;
        ServiceMatches matches;
        protected int maxProyectores = 10;

        public static void main() throws Exception{

  }

  public ClientUnicast(String urlLookup) throws Exception{
                LookupLocator lookup = null;
                ServiceRegistrar registrar = null;
                byte[] bimage = null;


    try {
       lookup = new LookupLocator(urlLookup);
    } catch(java.net.MalformedURLException e) {
                        System.err.println("Error Lookup: " + e.toString());
                System.exit(1);
        }

                System.setSecurityManager(new RMISecurityManager());

                try {
                registrar = lookup.getRegistrar();
                } catch (java.io.IOException e) {
       System.err.println("Error Registrar: " + e.toString());
                System.exit(1);
                        } catch (java.lang.ClassNotFoundException e) {
        System.err.println("Error Registrar: " + e.toString());
                        System.exit(1);
                                }
```

```
            Class[] classes = new Class[] {ProyectarInterfaz.class};
            ServiceTemplate template = new ServiceTemplate(null, classes, null);

            //Buscar los proyectores publicados
            try{
                  matches = registrar.lookup(template, maxProyectores);
            } catch(java.rmi.RemoteException e) {e.toString();}
            //



      }

      public int getMatches(){
            return matches.items.length;
      }

      public String getNameProyector(int numMatch){
            ServiceItem[] item=matches.items;
            Entry[] atributosEntry = item[numMatch].attributeSets;
            Name name= (Name) atributosEntry[0];
            String nombre = name.name;
            return nombre;
      }

       public ProyectarInterfaz getInterfaz(int numMatch) {
            proyectarinterfaz = (ProyectarInterfaz) matches.items[numMatch].service;
            if (proyectarinterfaz == null) {
            System.out.println("proyectarinterfaz null");
            System.exit(2);
            }
            return proyectarinterfaz;
       }

}
```

ClientMulticast – Final

```
import java.rmi.RMISecurityManager;
import net.jini.discovery.LookupDiscovery;
import net.jini.discovery.DiscoveryListener;
import net.jini.discovery.DiscoveryEvent;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceTemplate;
import java.io.*;

import net.jini.core.lookup.ServiceMatches;
import net.jini.core.entry.Entry;
import net.jini.core.lookup.ServiceID;
import net.jini.lookup.entry.Name;
import net.jini.core.lookup.ServiceItem;

public class ClientMulticast implements DiscoveryListener {
        ProyectarInterfaz proyectarinterfaz= null;
        ServiceMatches matches, matchesChoose=null;
        protected int maxProyectores = 10;
        ServiceRegistrar registrar;
        ServiceTemplate template;


  public static void main(String argv[]) {
                //new ClientMulticast();

  }

  public ClientMulticast() {


                System.setSecurityManager(new RMISecurityManager());
                LookupDiscovery discover = null;
    try {
      discover = new LookupDiscovery(LookupDiscovery.ALL_GROUPS);
    } catch(Exception e) {
      System.err.println(e.toString());
      System.exit(1);}
    discover.addDiscoveryListener(this);

                try {
      Thread.currentThread().sleep(2000L);
    } catch(java.lang.InterruptedException e) { }
  }

  public void discovered(DiscoveryEvent evt) {
                byte[] bimage = null;
```

```
        ServiceRegistrar[] registrars = evt.getRegistrars();
                Class [] classes = new Class[] {ProyectarInterfaz.class};
                proyectarinterfaz = null;
                template = new ServiceTemplate(null, classes,
                                                        null);
                for (int n = 0; n < registrars.length; n++) {
                        registrar = registrars[n];

                        try{
                                matches = registrar.lookup(template, maxProyectores);
                        } catch(java.rmi.RemoteException e) {e.toString();}

                        if (matches.items.length > 0) {
                                matchesChoose = matches;
                                continue;
                        }else{
                                System.out.println("Lookup sin Proyectores");
                                continue;
                        }
                }
    }

        public int getMatches(){
                if(matchesChoose==null){
                        return 0;
                }
                else{
                        return matchesChoose.items.length;
                }
        }
        public String getNameProyector(int numMatch){
                ServiceItem[] item=matchesChoose.items;
                Entry[] atributosEntry = item[numMatch].attributeSets;
                Name name= (Name) atributosEntry[0];
                String nombre = name.name;
                return nombre;
        }

        public ProyectarInterfaz getInterfaz(int numMatch) {
                proyectarinterfaz = (ProyectarInterfaz) matchesChoose.items[numMatch].service;
                if (proyectarinterfaz == null) {
                        System.out.println("proyectarinterfaz null");
                        System.exit(2);
                }
                return proyectarinterfaz;
        }

    public void discarded(DiscoveryEvent evt) { }
}
```

Gui

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Gui implements ActionListener {
	JPanel mainPanel,superiorPanel,serviciosPanel,medioPanel,inferiorPanel;
	JButton proyectarButton, detenerButton, buscarButton;
	JRadioButton multicastRadio, unicastRadio;
	ButtonGroup groupGroup;
	JTextField urlText;
	JTextArea mensajesEditor;
	JSpinner velocidadSpinner;
	JLabel segundosLabel;
	JScrollPane editorScrollPane;
	JOptionPane optionPane;
	JFrame dialogFrame;
	JComboBox serviciosBox;
	ClientUnicast cliUnicast;
	ClientMulticast cliMulticast;
	String lastBuscar;
	ProyectarInterfaz proyectarInterfaz;
	ProyectarTimer proyectarTimer;


	//************CONSTRUCTOR******************
	public Gui(){
		mainPanel = new JPanel();
		superiorPanel = new JPanel();
		inferiorPanel = new JPanel();
		medioPanel = new JPanel();
		serviciosPanel = new JPanel();

		//Layout Paneles
		mainPanel.setLayout(new BoxLayout(mainPanel,BoxLayout.Y_AXIS));
		superiorPanel.setLayout(new GridLayout(3,1));
		serviciosPanel.setLayout(new
		BoxLayout(serviciosPanel,BoxLayout.LINE_AXIS));
		medioPanel.setLayout(new FlowLayout(FlowLayout.CENTER,15,0));
		inferiorPanel.setLayout(new BorderLayout());

		//Borde Paneles
		superiorPanel.setBorder(BorderFactory.createCompoundBorder(
		BorderFactory.createTitledBorder("Tipo de Busqueda"),
		BorderFactory.createEmptyBorder(5,5,5,5)));
```

```
            serviciosPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Proyectores disponibles"),
            BorderFactory.createEmptyBorder(5,5,5,5)));


            medioPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Velocidad de captura"),
            BorderFactory.createEmptyBorder(5,5,5,5)));

            inferiorPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Acciones"),
            BorderFactory.createEmptyBorder(5,5,5,5)));

            mainPanel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));

            //Adjuntando todo al panel principal
            mainPanel.add(superiorPanel);
            mainPanel.add(serviciosPanel);
            mainPanel.add(medioPanel);
            mainPanel.add(inferiorPanel);


            //Pegar Widgets a Paneles
            addWidgets();

            //Agregar Listeners
            addListeners();

    }

    //************WIDGETS***********************
    private void addWidgets() {
            proyectarButton= new JButton("Proyectar");
            detenerButton= new JButton("Detener");
            detenerButton.setEnabled(false);
            unicastRadio = new JRadioButton ("Unicast");
            unicastRadio.setActionCommand("Unicast");
            unicastRadio.setSelected(true);
            multicastRadio = new JRadioButton ("Multicast");
            multicastRadio.setActionCommand("Multicast");
            groupGroup = new ButtonGroup();
            urlText = new JTextField(1);
            mensajesEditor = new JTextArea("Cliente Vistazo v.2.9\n");
            mensajesEditor.setEditable(false);
            editorScrollPane = new JScrollPane(mensajesEditor);
            editorScrollPane.setVerticalScrollBarPolicy(
           JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
editorScrollPane.setPreferredSize(new Dimension(200, 100));
editorScrollPane.setMinimumSize(new Dimension(10, 10));
```

```
        velocidadSpinner = new JSpinner(new SpinnerNumberModel(5,0, 300,1));
        segundosLabel = new JLabel("Segundos");
        optionPane = new JOptionPane();
        dialogFrame = new JFrame();
        serviciosBox = new JComboBox();
        buscarButton = new JButton("Buscar");

        //Crea grupo de RadioButton
        groupGroup.add(unicastRadio);
        groupGroup.add(multicastRadio);

        //agregar a Panel Superior
        superiorPanel.add(multicastRadio);
        superiorPanel.add(unicastRadio);
        superiorPanel.add(urlText);

        //agregar a Panel Servicios
        serviciosPanel.add(buscarButton);
        serviciosPanel.add(serviciosBox);

        //agregar a Panel Medio
        medioPanel.add(velocidadSpinner);
        medioPanel.add(segundosLabel);

        //agregar a Panel Inferior
        inferiorPanel.add(proyectarButton,BorderLayout.WEST);
        inferiorPanel.add(detenerButton,BorderLayout.EAST);
        inferiorPanel.add(editorScrollPane,BorderLayout.SOUTH);
}

//************Listeners******************
private void addListeners() {
        unicastRadio.addActionListener(this);
        multicastRadio.addActionListener(this);
        proyectarButton.addActionListener(this);
        detenerButton.addActionListener(this);
        buscarButton.addActionListener(this);

}
//************EVENTOS************************
public void actionPerformed(ActionEvent event) {
        String accionString;
        accionString= event.getActionCommand();

        //>>>ComboBox Unicast
        if(accionString == "Unicast"){
                //habilitar
                urlText.setEnabled(true);
        }
```

```
//>>>ComboBox Multicast
if(accionString == "Multicast"){
        //desabilitar
        urlText.setEnabled(false);
}

//>>>Button Proyectar
if(accionString == "Proyectar"){
        if((String) serviciosBox.getSelectedItem()==null){
        //NO Seleccionado Proyector
        optionPane.showMessageDialog(dialogFrame,"Debe buscar y seleccionar
        un Proyector");
        }
        else{
        //Proyector Seleccionado
                cambiarEstadoWidgets(false);
                //insertarTexto((String) serviciosBox.getSelectedItem() +" ;
                "+velocidadSpinner.getValue().toString());
                if(lastBuscar=="Unicast"){
                //con Resultado Unicast
                        //obtener Interfaz Remota
                        proyectarInterfaz = cliUnicast.getInterfaz
                        (serviciosBox.getSelectedIndex());
                        insertarTexto("Interfaz remota obtenida");
                        //Enviar la imagen con una determinada velocidad
                        proyectarTimer = new ProyectarTimer
                        (proyectarInterfaz,velocidadSpinner.getValue().toString(),t
                        his);
                }
                else{
                //con Resultado Multicast
                        //obtener Interfaz Remota
                        proyectarInterfaz = cliMulticast.getInterfaz
                        (serviciosBox.getSelectedIndex());
                        insertarTexto("Interfaz remota obtenida");
                        //Enviar la imagen con una determinada velocidad
                        proyectarTimer = new ProyectarTimer
                        (proyectarInterfaz,velocidadSpinner.getValue().toString(),t
                        his);

                }
        }
}

//>>>Button Detener++++++++++++++++++++++++++++++++++++++++++++
if(accionString == "Detener"){
        cambiarEstadoWidgets(true);
        proyectarTimer.Stop();
```

```
                    insertarTexto("Proyeccion detenida");
                }

        //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                //>>Button Buscar
                if(accionString == "Buscar"){
                        //seleccionado Unicast
                        if (groupGroup.getSelection().getActionCommand()=="Unicast"){
                                //Url vacia
                                if(urlText.getText().equalsIgnoreCase("")){
                                        optionPane.showMessageDialog(dialogFrame,"Debe
ingresar la URL o IP del servidor Lookup");
                                }
                                //Url ingresada++++++++++++++++++++++++++++++++++
                                else{
                                        //Crea ClienteUnicast
                                        try{
                                                cliUnicast = new ClientUnicast(urlText.getText());
                                        }catch  (java.lang.Exception  e)  {System.err.println("Error
                                        GUI: " + e.toString());System.exit(1);}
                                        insertarTexto("->Cliente unicast creado");

                                        //Crea ComboBox con Proyectores encontrados
                                        serviciosBox.removeAllItems();
                                        for (int n=0; n < cliUnicast.getMatches() ; n++){
                                                serviciosBox.insertItemAt
                                                (cliUnicast.getNameProyector(n),n);
                                        }
                                        lastBuscar="Unicast";
                                        insertarTexto("Proyectores enlistados");
                                }

        //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        }
                        //seleccionado Multicast
                        if (groupGroup.getSelection().getActionCommand()=="Multicast"){

                                //Crea ClienteMulticast
                                try{
                                        cliMulticast = new ClientMulticast();
                                }catch (java.lang.Exception e)
                                {System.err.println("Error GUI: " + e.toString());System.exit(1);}
                                insertarTexto("->Cliente Multicast creado");

                                try {
                Thread.currentThread().sleep(2500L);
                } catch(java.lang.InterruptedException e) { }
```

```
                        //Crea ComboBox con Proyectores encontrados
                        serviciosBox.removeAllItems();
                        for (int z=0; z < cliMulticast.getMatches() ; z++){
                                serviciosBox.insertItemAt
                                (cliMulticast.getNameProyector(z),z);
                        }

                        lastBuscar="Multicast";
                        insertarTexto("Proyectores enlistados");

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                }
  }

}

//***************INSERTAR TEXTO EN TEXT AREA**************
public void insertarTexto(String text){
        mensajesEditor.append(text+"\n");
        mensajesEditor.setCaretPosition(mensajesEditor.getDocument().getLength());
}

//***************Cambiar estado a Widgets***************
private void cambiarEstadoWidgets(boolean estado){
        proyectarButton.setEnabled(estado);
        velocidadSpinner.setEnabled(estado);
        multicastRadio.setEnabled(estado);
        unicastRadio.setEnabled(estado);
        serviciosBox.setEnabled(estado);
        buscarButton.setEnabled(estado);
        urlText.setEnabled(estado);
        detenerButton.setEnabled(!estado);

        if (groupGroup.getSelection().getActionCommand() =="Multicast" && estado ==
        true){
        urlText.setEnabled(false);}
}

//***************PRINCIPAL********************
public static void main(String[] args) {
        Gui guiVistazo = new Gui();
        JFrame ClientFrame = new JFrame("Cliente Vistazo");

        try {
        UIManager.setLookAndFeel(
                UIManager.getCrossPlatformLookAndFeelClassName());
        } catch(Exception e) {}

        ClientFrame.setContentPane(guiVistazo.mainPanel);
```

```
    ClientFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            ClientFrame.pack();

            //Centrar la ventana
            int WIDTH = 250;
            int HEIGHT = 425;
            ClientFrame.setSize(WIDTH,HEIGHT);
            Dimension screenDim = Toolkit.getDefaultToolkit().getScreenSize();
            ClientFrame.setLocation( (screenDim.width-WIDTH)/2, (screenDim.height-
            HEIGHT)/2);

            ClientFrame.setVisible(true);
}}
```

ProyectarTimer

```
import java.util.Timer;

public class ProyectarTimer{
	Timer timer;
	CapturarTask capturarTask;

	public  void main(String argv[]) throws Exception{
	}

	public ProyectarTimer(ProyectarInterfaz proyectarInterfaz, String delay, Gui guiCliente) {
		timer = new Timer();
		capturarTask = new CapturarTask(proyectarInterfaz, delay,guiCliente);
		if(delay=="0"){
		timer.schedule(capturarTask,0,1);
		}
		else{
		timer.schedule(capturarTask,0,Long.parseLong(delay)*1000);
		guiCliente.insertarTexto("Schedule iniciado");
		}

	}

	public void Stop(){
		timer.cancel();
	}

}
```

CapturarTask

```
import java.awt.*;
import java.awt.image.*;
import javax.swing.*;
import java.util.TimerTask;
import java.util.Timer;


class CapturarTask extends TimerTask  {
		Toolkit toolkit;
		Dimension screenSize;
		Rectangle screenRect;
		BufferedImage imagebuff;
		ProyectarInterfaz prointerfaz;
		Robot robot;
		String delay;
		Gui guiCliente;


		public  void main(String argv[]) throws Exception{

		}

		public CapturarTask(ProyectarInterfaz prointerfaz, String delay,Gui guiCliente) {
			toolkit = Toolkit.getDefaultToolkit();
			screenSize = toolkit.getScreenSize();
			screenRect = new Rectangle(screenSize);
			imagebuff=null;

			try{
					robot = new Robot();
				}catch (Exception e) {e.printStackTrace();}

			this.prointerfaz=prointerfaz;
			this.delay = delay;
			this.guiCliente= guiCliente;

		}


	public void run() {
		try{
					imagebuff = robot.createScreenCapture(screenRect);
				}catch (Exception e) {e.printStackTrace();}

				guiCliente.insertarTexto("Screenshot realizado");
			ImageIcon image = new ImageIcon(imagebuff);
```

```
                        guiCliente.insertarTexto("Llamando a Proyectar( )");
                        try {
                        guiCliente.insertarTexto("**" + prointerfaz.Proyectar(image) );
                        } catch(java.rmi.RemoteException e) {
                                guiCliente.insertarTexto("Error Proyector no responde");
                                guiCliente.detenerButton.doClick();
                                System.err.println(e.toString());
                                }

                        guiCliente.insertarTexto("Esperando "+delay+" segundos...");
                }
}
```

Anexo F

Acrónimos

2.5G	Generation Between Second And Thirty
2G	Second Generation
3G	Thirty Generation
3GPP	3rd Generation Partnership Project
ANSA	Advanced Network Systems Architecture
API	Application Programming Interfaz
CAL	California Institute Of Technology
CEPT	Conferencia Europea De Administraciones De Correos Y Telecomunicaciones
CEPT	Centre For Environmental Planning & Technology
CISC	Complex Instruction Set Computer
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DCOM	Distributed Component Object Model
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
EDGE	Enhanced Data Rates For Gsm Evolution
EEPROM	Electrically Erasable Programmable Read-Only Memory
EHS	European Home System
EIA	Electronics Industry Association
ETSI	European Telecommunications Standards Institute
ETSI	European Telecommunications Standards Institute
GPRS	General Packet Radio Service
GSM	Global System For Mobile Communications
H2GF	Hiperlan2 Global Forum
HAN	Home Area Network
HOMECNA	Home Cable Network Alliance
HOMEPNA	Home Personal Area Network
HOMERF	Home Networking Radio Frecuency
HTTP	Hypertext Transfer Protocol
IEEE	Institute Of Electrical And Electronics Engineers
IETF	Engineering Task Force
IMT	International Mobile Telecommunications
IP	Internet Protocol
IRDA	Infrared Data Association
ISM	Industrial, Scientific And Medical
ISM	Industrial, Scientific And Medical
ISO	International Organization For Standardization
JVM	Java Virtual Machine
L2CAP	Logical Link Layer
LAN	Local Area Network

M2M	Machine To Machine
MAC	Capa De Acceso Al Medio
MI	Mensajería Instantánea
MMS	Multimedia Messaging Service
MMU	Memory Management Unit
OFDM	Orthogonal Frequency Digital Multiplexing
OMG	Object Management Group
P2P	Peer To Peer
PAN	Personal Area Network
PC	Personal Computer
PDA	Personal Digital Assistants
QoS	Quality Of Service
RISC	Reduced Instruction Set Computer
RMI	Remote Method Invocation
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SCSI	Small Computer Systems Interfaz
SDP	Service Discovery Protocol
SDRAM	Synchronous Dynamic Random Access Memory
SLP	Service Location Protocol
SOAP	Simple Object Access Protocol
SRAM	Static Random Access Memory
SSDP	Simple Service Discovery Protocol
SWAP	Shared Wireless Access Protocol
TCP	Transmission Control Protocol
TINY	Internet Interfaz
UDDI	Universal, Description, Discovery And Integration
UDP	User Datagram Protocol
UIT	Unión Internacional De Telecomunicaciones
UIT	Unión Internacional De Telecomunicaciones
UMTS	Universal Mobile Telecomunications System
UPNP	Universal Plug And Play
URL	Uniform Resource Locator
WAN	Wide Area Network
WAP	Wireless Application Protocol
WEP	Wired Equivalent Protocol
WEP	Wired Equivalent Protocol
WI-FI	Wireless Fidelity
WLAN	Wireless Local Area Network
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language