(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 551 301**

(51) Int. CI.:

H04L 29/06 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **23.03.2005 E 05729180 (9)**

(97) Fecha y número de publicación de la concesión europea: **16.09.2015 EP 1730929**

(54) Título: **Método y aparato para comunicar datos entre dispositivos de ordenador**

(30) Prioridad:

31.03.2004 GB 0407388

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
17.11.2015

(73) Titular/es:

BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY (100.0%)
81 NEWGATE STREET
LONDON EC1A 7AJ, GB

(72) Inventor/es:

ROXBURGH, DAVID;
BEDDUS, SIMON ALEXANDER;
FARLEY, PATRICK BRIAN y
HOSKING, MICHAEL ROBERT

(74) Agente/Representante:

CURELL AGUILÁ, Mireia

DESCRIPCIÓN

Método y aparato para comunicar datos entre dispositivos de ordenador.

Campo de la invención

La presente invención se refiere a un método y a un aparato de transmisión de datos entre dispositivos de ordenador y, en particular, a un método y un aparato para permitir que servicios *backend*, los cuales son proporcionados por un único sistema *backend*, tal como una red inalámbrica móvil, y los cuales posibilitan un acceso controlado a una o más características del sistema de fondo por desarrolladores de aplicaciones (internos o externos), inicien, de una manera eficiente, comunicaciones con aplicaciones que están usando los servicios.

Antecedentes de la invención

Los presentes inventores han desarrollado un sistema (al que en lo sucesivo en la presente se hace referencia como pasarela o plataforma) por el cual una red móvil (u otro sistema de "fondo" similar) puede proporcionar una serie de servicios (mayoristas básicos) a desarrolladores de aplicaciones terceros los cuales pueden entonces desarrollar y desplegar fácilmente nuevos servicios (minoristas) para clientes finales. Esto proporciona oportunidades de negocio tanto para operadores de redes móviles como para desarrolladores de aplicaciones terceros.

Por ejemplo, la mayoría de redes de telefonía móvil GSM disponen de la capacidad de localizar un conjunto telefónico móvil de mano (es decir, una unidad de abonado) conectado a la red (es decir, cuando se enciende y está dentro del alcance de una estación base adecuada). No obstante, la mayoría de estas redes no ofrecen actualmente este servicio para usuarios finales. Esto es debido a que la red no puede ofrecerlo simplemente de una manera incontrolada a terceros por motivo, por ejemplo, de la posibilidad de un ataque de tipo denegación de servicio (en el cual terceros podrían solicitar constantemente la posición de unidades de abonado hasta que el nivel de servicio ofrecido por la red se deteriorase a causa de una carga excesiva hasta el punto de que pudiera reducirse la capacidad de otras unidades de abonado de realizar y recibir llamadas telefónicas). Por otro lado, el desarrollo de servicios rentables y totalmente operativos, controlados, que incorporan (en mayor o menor medida) dicha capacidad es una actividad que con frecuencia es llevada a cabo eficientemente por desarrolladores de aplicaciones terceros que son especialistas. No obstante, esto sigue requiriendo que el operador de la red controle el acceso a dichas capacidades, y llevar a cabo esto de manera individual para cada nueva aplicación propuesta es una tarea relativamente lenta e ineficiente.

Por ello, los presentes inventores han desarrollado una pasarela de servicio inalámbrico de Interfaz de Programación de Aplicaciones (API) para proporcionar la infraestructura que cubra las necesidades de desarrolladores de aplicaciones y operadores de redes. La pasarela posibilita que los operadores ofrezcan servicios inalámbricos de API comerciales, tales como localización, mensajería y tarificación, al mercado general de los desarrolladores, lo cual a su vez permite el desarrollo y lanzamiento rentables de aplicaciones nuevas. La pasarela ofrece a los desarrolladores una amplia gama de servicios que son sencillos de utilizar, están protegidos y son seguros.

Una característica importante de la pasarela es que incluye un Motor de Exposición de Servicio (SEE) con el cual se puede contactar a través de una red convencional, tal como Internet, por parte de dispositivos remotos en los cuales se están ejecutando aplicaciones individuales. Esto proporciona acceso limitado a los servicios ofrecidos por el operador de la red a cada aplicación permitida. El SEE incluye una serie de características comunes que garantizan que la conexión entre una aplicación y la pasarela está segura y autenticada, y la pasarela incluye también un mecanismo para garantizar que la carga máxima que puede provocar en la red una aplicación es gestionable en todo momento.

Para acceder a un servicio de red por medio de la pasarela, una aplicación inicia una conexión segura con la pasarela. Obsérvese que la conexión es iniciada siempre por la aplicación y no por la pasarela. La razón de esto es que hay una cierta cantidad de procesado que es necesario que sea llevado a cabo por la parte no iniciadora para garantizar, entre otras cosas, que la parte iniciadora es quién reivindica ser (es decir, para garantizar una autenticación correcta). Esto se realiza proporcionando a la pasarela la funcionalidad denominada "servidor web". Si todas las aplicaciones también tuvieran que disponer de la capacidad de hacer frente al establecimiento de conexiones iniciadas por la pasarela utilizando una funcionalidad de servidor web, la complejidad de todas estas aplicaciones tendría que incrementarse considerablemente. Si la complejidad adicional fuera proporcionada por los desarrolladores de pasarelas, esto provocaría problemas adicionales de mantenimiento para los administradores de pasarelas (es siempre preferible evitar situaciones que requieren un soporte remoto de software instalado en máquinas terceras remotas). Alternativamente, esto impondría una carga adicional considerable sobre los desarrolladores/administradores de aplicaciones, e idealmente ello debería reducirse al mínimo siempre que sea posible.

Por lo tanto, en la actualidad, para que una aplicación reciba datos de un servicio en algún momento desconocido, la misma debe contactar continuamente con el servicio a intervalos regulares y preguntar si el servicio en cuestión dispone de algún dato nuevo para suministrar a la aplicación. Este método de comunicación intermitente se usa

ampliamente en sistemas de comunicación de datos y en general funciona bien siempre que se encuentre el equilibrio correcto entre un contacto con la pasarela que sea tan frecuente que provoque una carga excesiva sobre la misma y un contacto con la pasarela que sea tan poco frecuente que se pueda producir un retardo indeseablemente grande entre algunos datos que llegan a un servicio de pasarela y la entrega de los mismos a la aplicación respectiva. Desafortunadamente, en el presente caso los operadores de pasarelas y los operadores de aplicaciones, cuyos deseos están completamente en oposición mutua, ya que cuanto menos frecuente sea el sondeo menos carga habrá en la pasarela pero más probable será que la aplicación tenga que esperar por algunos datos, son entidades comerciales independientes, y por lo tanto no existe en absoluto ninguna motivación por parte de ninguno de los participantes en intentar renunciar a sus empeños.

La presente invención busca proporcionar un método el cual en cierta medida al menos resuelva el conflicto antes mencionado sin quitar valor sustancial a las características de diseño de la pasarela antes descrita.

Obsérvese que el uso de una pasarela de este tipo no se limita al caso de redes de telefonía móvil y podría aplicarse el mismo tipo de pasarela a cualquier sistema grande que tenga algunas funciones básicas que podrían usarse para generar muchas aplicaciones diversas de usuario final con solo que estas funciones se pudieran ofrecen a desarrolladores terceros de aplicaciones de una manera directa, simple, segura y protegida.

Los documentos US 6.510.464, WO 02/15523 y US 6.094.485 se refieren todos ellos a métodos para permitir que un usuario de un dispositivo conectado a una red no segura, tal como Internet, establezca una conexión segura con un ordenador servidor situado detrás de un cortafuegos de una manera eficiente.

Sumario de la invención

Según un primer aspecto de la presente invención, se proporciona un método tal como se expone en la reivindicación 1.

De acuerdo con un segundo aspecto de la presente invención, se proporciona un sistema de servidor de cliente tal como se expone en la reivindicación 7.

Así, una forma de realización de la presente invención proporciona un sistema para ofrecer servicios proporcionados por un primer subsistema a uno o más subsistemas de alojamiento de aplicaciones por medio de un dispositivo de pasarela, estando dispuestos la pasarela y el subsistema de alojamiento de aplicaciones o cada uno de los subsistemas de alojamiento de aplicaciones para permitir que el o cada subsistema de alojamiento de aplicaciones inicie una conexión segura y autenticada con la pasarela por medio de una conexión de red de datos no segura, y estando conectada lógicamente la pasarela al primer subsistema para posibilitar que servicios proporcionados por el primer subsistema sean proporcionados al o a cada subsistema de alojamiento de aplicaciones, incluyendo la pasarela medios de notificación para notificar a uno o más de entre los subsistemas de alojamiento de aplicaciones que debería o deberían iniciar una conexión autenticada segura con la pasarela.

Preferentemente, las notificaciones proporcionadas por el servidor de notificaciones no incluyen ningún dato sensible o valioso (en el sentido de que la información no pueda ser usada indebidamente para provocar datos o molestias a ninguno de los participantes que tenga algún interés en cualquier parte del sistema, si fuera interceptada por un tercero malicioso o si fuera dirigida de manera errónea involuntariamente a este último a través de la red de datos no segura). Preferentemente, las notificaciones son completamente pasivas en el sentido de que no incluyen ningún código ejecutable. Preferentemente, las notificaciones son archivos de texto simple, preferentemente en la forma específica de archivos del Lenguaje de Marcado Extensible (XML).

Un sistema de este formato tiene la ventaja de que los subsistemas de alojamiento de aplicaciones requieren una complejidad adicional mínima en comparación con el sistema antes descrito en el cual los subsistemas de alojamiento de aplicaciones deben sondear continuamente la pasarela para ver si hay algún mensaje nuevo, etcétera, esperando por ellos, y sin embargo se evita una carga excesiva sobre la pasarela sin aumentar el retardo máximo entre un servicio en el primer subsistema que recibe alguna información para una aplicación y la aplicación que recibe esos datos.

Preferentemente, las notificaciones especifican exactamente qué servicio es el que desea contactar (o de manera más precisa, el que desea ser contactado por) el subsistema de alojamiento de aplicaciones. Preferentemente, hay una interfaz común entre el servidor de notificaciones y todos los servicios proporcionados por el primer subsistema y una interfaz común entre el servidor de notificaciones y cada uno de los subsistemas de alojamiento de aplicaciones de tal manera que el servidor de notificaciones puede llevar a cabo una gestión, un encaminamiento y un despacho de notificaciones, los cuales son independientes del origen e independientes del destinatario.

Preferentemente, la interfaz común entre el servidor de notificaciones y cada uno de los servicios ofrecidos por el primer subsistema permite que un servicio especifique con respecto a una notificación individual un identificador que identifique la aplicación de destino, y un identificador que identifique el servicio solicitante y uno o más parámetros de comportamiento de gestión. Preferentemente, los parámetros de comportamiento de gestión pueden incluir una o

ambas opciones de entre el número de intentos que se debería realizar para enviar la notificación en caso de uno o más intentos insatisfactorios de envío de la notificación y el retardo entre intentos nuevos de enviar la notificación en caso de uno o más intentos insatisfactorios de envío de la notificación.

Preferentemente, el servidor de notificaciones se puede hacer funcionar para generar hilos de ejecución diferentes con el fin de controlar la transmisión de notificaciones diferentes respectivas a través de la red de datos no segura. Esto tiene la ventaja de posibilitar que el servidor de notificaciones proporcione un caudal adecuado de notificaciones para múltiples servicios generadores de notificaciones a pesar de la posibilidad de que cada notificación pueda tardar una cantidad de tiempo significativa en atravesar la red, puesto que se pueden enviar muchas notificaciones diferentes “en paralelo” si las mismas se envían utilizando hilos diferentes.

Preferentemente, el servidor de notificaciones conserva notificaciones para las cuales ha fallado la entrega y vuelve a intentar su entrega después de un retardo especificado por el servicio o por defecto correspondiente hasta un número de intentos especificado por el servicio o por defecto.

La presente invención proporciona también un servidor de notificaciones para su uso en el sistema del segundo aspecto de la presente invención antes descrito, y programas de ordenador y productos de programa de ordenador que se corresponden con el primer y el segundo aspectos de la presente invención.

Breve descripción de las figuras

Para que la presente invención pueda ser entendida mejor, a continuación se describirán, únicamente a título de ejemplo, formas de realización de la misma, en referencia a los dibujos adjuntos en los cuales:

la Figura 1 es una ilustración esquemática de un sistema según una forma de realización de la presente invención;

la Figura 2 es una ilustración esquemática de una aplicación de ejemplo que utiliza el sistema de la Figura 1 para proporcionar un servicio de usuario final de localización;

la Figura 3 es un diagrama de señales que muestra las señales intercambiadas entre varios elementos del sistema de las Figuras 1 y 2 en la provisión del servicio de usuario final de localización al que se ha hecho referencia anteriormente;

la Figura 4 es un diagrama de flujo que ilustra las etapas llevadas a cabo por una aplicación de cliente en la interacción con usuarios finales y una pasarela entre la aplicación de cliente y un sistema de fondo;

la Figura 5 es un diagrama de flujo que ilustra las etapas llevadas a cabo por un *plug-in* de servicios en una pasarela entre una aplicación de cliente y un sistema de fondo;

la Figura 6 es una ilustración esquemática de las estructuras funcionales principales del servidor de notificaciones de la Figura 1; y

las Figuras 7a a 7e son diagramas de flujo de las etapas llevadas a cabo por cinco módulos/objetos significativos del servidor de notificaciones de las Figuras 1 y 6.

Descripción detallada de una forma de realización preferida

En referencia a la Figura 1, el sistema de la forma de realización preferida tiene tres dominios: un dominio de aplicación de cliente 100, un dominio de pasarela 200, y un dominio de sistema de fondo 300. El dominio de aplicación de cliente 100 y el dominio de pasarela 200 están conectados por una red de datos no segura. En la presente forma de realización, el dominio de pasarela 200 y el dominio de sistema de fondo 300 están conectados también por una red no segura, aunque, en formas de realización alternativas, esta puede ser típicamente una conexión de red segura.

El dominio de aplicación de cliente 100 comprende una serie de aplicaciones de cliente 110, 120, 130, 140, 150 alojadas en varias máquinas de servidor de terceros (no mostradas) (Nota Bene. Típicamente las aplicaciones de cliente se comportarán como aplicaciones tanto de cliente como de servidor ya que se comportarán como aplicaciones de servidor para clientes de usuarios finales que contacten con ellas, pero se comportarán como aplicaciones de cliente cuando entren en contacto con el dominio de pasarela 200). En estos ejemplos específicos que se muestran en la Figura 1, las aplicaciones 110, 120, 130 son operadas todas ellas por una única organización tal como se indica por medio del recuadro circundante 101 con una sola cuenta de cliente con el operador del dominio de pasarela 200, cuya importancia se explicará de forma más detallada posteriormente. Además, tal como se muestra en la Figura 1, algunas de las aplicaciones, tales como la aplicación 150, pueden acceder a diferentes recursos, tales como la base de datos 152, considerándose que todos ellos se sitúan dentro del dominio de aplicación actual 100.

El dominio de pasarela 200 comprende los siguientes componentes principales: un portal de inicio 205 y un portal de operador de plataforma 210; un componente de software de Colas de Espera de Servidor de Mensajería Java (JMS) 215; un componente de software de servidor de notificaciones 220 (que se describirá de forma más detallada posteriormente) y un Motor de Exposición de Servicio (SEE) 250. El SEE 250 incluye un conjunto de capas de servicio jerárquicas 252 y un conjunto de módulos de *plug-in* de servicio 254 a 257. El conjunto de capas de servicio jerárquicas 252 proporciona una serie de servicios básicos a todos los *plug-ins* de servicio 254 a 257 incluyendo la gestión del cifrado y descifrado de datos para su transferencia a través de la red no segura, la autenticación de los participantes que se comunican entre sí a través de la red no segura, etcétera. Uno de los *plug-ins* de servidor 254 es un *plug-in* de Lado Servidor de Administrador de Integridad 254 que lleva a cabo la función especial de regular cómo se permite que aplicaciones de cliente individuales 110 a 150 se comuniquen con el SEE 250.

Detalles completos del IMSS (y los componentes correspondientes de Lado Cliente de Administrador de Integridad (IMCS) contenidos dentro de cada aplicación de cliente 110 a 215) se pueden encontrar en la solicitud de patente internacional publicada n.º WO 03/029975. Por comodidad, las enseñanzas relevantes del documento WO 03/029975 se resumen brevemente exponiendo que el IMSS 254 se comunica con un IMCS correspondiente en cada una de las aplicaciones de cliente con el fin de regular la frecuencia con la cual cada aplicación de cliente puede contactar con el SEE 250; el operador de plataforma de pasarela puede utilizar este mecanismo para "contener" la frecuencia con la cual cualquier aplicación particular puede contactar con el SEE con el fin de reducir la carga en el SEE en los intervalos de tiempo de mucha utilización. El mecanismo implica un denominado "latido" en el cual el IMCS de la aplicación de cliente respectiva entra en contacto con el IMSS y actualiza, a partir del IMSS, un conjunto de parámetros actuales que especifican la frecuencia con la cual la aplicación actual puede contactar con *plug-ins* de servicio especificados, dentro del SEE (o, en formas de realización alternativas, simplemente podrían especificar la frecuencia con la cual la aplicación puede contactar con el SEE – es decir, con independencia de con qué *plug-in* de servicio concreto se desee contactar).

En la presente forma de realización, los otros *plug-ins* de servicio mostrados son un *plug-in* de servicio de localización GSM 255, un *plug-in* de servicio de localización GPS 255, un *plug-in* de servicio de localización GPS 256 y un servicio de mensajes cortos (*plug-in* de servicio SMS 257). La funcionalidad de los *plug-ins* de servicios se describe de forma más detallada posteriormente.

El portal de inicio 205 y el portal de operador de plataforma 210, en la presente forma de realización, son proporcionados por un servicio de sitio web que puede ser revisado tanto por el operador de plataforma de pasarela como por los clientes de la pasarela y, cuando resulte apropiado, los mismos pueden modificar detalles de la relación comercial entre el operador de pasarela y el aparato de aplicación de cliente. Por ejemplo, una aplicación de cliente particular podría negociar con el operador de plataforma de pasarela para tener acceso a servicios especificados y realizar un número máximo de solicitudes del servicio dentro de un periodo de tiempo particular por un precio acordado. Si a continuación el operador de aplicación actual requiere aumentar el número máximo de solicitudes que desea poder realizar de un *plug-in* de servicio particular dentro del SEE en un periodo de tiempo particular, entonces esto se puede modificar usando el portal de inicio siempre que haya disposiciones adecuadas a punto para permitir aumentar de manera correspondiente el precio con el que se tarifica la aplicación actual. De modo similar, el portal de operador de plataforma se puede usar para identificar y corregir todos los detalles de cada cuenta de cliente en caso de que el cliente contacte con el operador de pasarela mediante algún mecanismo alternativo (por ejemplo, por correo electrónico) para solicitar dicho cambio en la relación contractual entre las partes.

En la presente forma de realización, el dominio de sistema de fondo 300 es una red de telefonía móvil GSM 310 que incluye un centro de localización de móviles 312 y un centro de SMS 314 cuya funcionalidad se describirá de forma más detallada posteriormente en referencia a las Figuras 2 y 3, así como las diversas características asociadas comúnmente a una infraestructura de red móvil, tales como estaciones base y torres de radiodifusión asociadas 305 e interconectadas por una red de datos y comunicadas por interfaz con otras diversas redes de telecomunicaciones y de datos.

Como ejemplo del funcionamiento del sistema mostrado en la Figura 1, a continuación se describirá en referencia a la Figura 2 el funcionamiento de la aplicación de cliente 110 de la Figura 1. A la aplicación actual 110 se le hace referencia en la presente en lo sucesivo como iLocate. La aplicación iLocate 110 permite que un usuario (tal, por ejemplo, "Papá") 9 establezca la posición de otro usuario, (por ejemplo, la hija de "Papá" "Kate") 19, por el teléfono móvil GSM 20 de ella utilizando el ordenador personal 10 de él, conectado a la aplicación iLocate 110 a través de Internet. Para realizar esto, la aplicación de cliente iLocate 110 utilizará el *plug-in* de servicio de localización GSM 255 y/o el servicio de localización GPS 256 con el fin de hallar la posición del aparato telefónico de mano 20 de Kate y utilizará también el *plug-in* de servicio SMS 257 con el fin de enviar y recibir SMSs hacia y desde el aparato telefónico de mano 20 de Kate. Los *plug-ins* de servicios 255, 256 y 257 se comunican a su vez con el sistema de fondo 300 el cual transmite y obtiene datos hacia y desde el aparato telefónico de mano 20 de Kate por medio de una torre de estación base 305.

Para determinar la posición de un dispositivo de aparato telefónico móvil de mano, los *plug-ins* de servicios de localización GSM y GPS 255, 256 de manera esencial simplemente contactan con el centro de localización de

móviles 312 y le solicitan a este la obtención de la información. La forma con la cual el Centro de localización de móviles obtiene la información no es relevante para la presente invención y no se describirá de forma detallada. En pocas palabras, tal como indican las denominaciones de estos servicios, para el servicio GSM el centro de localización de móviles determina en qué célula se encuentra el usuario y asociando esta a la región geográfica que cubre la célula se obtiene una posición aproximada del móvil; para el servicio GPS (que está únicamente disponible cuando el aparato telefónico de mano a localizar incluye un receptor GPS), el centro de localización de móviles envía un mensaje SMS el cual provoca que el aparato telefónico de mano busque automáticamente su posición utilizando su receptor GPS incorporado y a continuación envíe un SMS de vuelta al centro de localización de móviles con detalles de la posición determinada. Lo significativo es que el sistema de fondo que explota el sistema de localización de móviles confía en la pasarela para garantizar que no sobrecargará al centro de localización de móviles ni usará sus servicios de manera inapropiada.

En referencia a continuación a la Figura 3, se describen las señales intercambiadas entre los diversos componentes ilustrados en la Figura 2 con el fin de permitir que el usuario 9 (Papá) transmita y reciba un SMS hacia y desde el usuario 19 (su hija Kate).

Para permitir dicho intercambio de SMS's, la aplicación iLocate 110 inicia una conexión con la pasarela 200. Lleva a cabo esto enviando en primer lugar una señal de inicio de solicitud 405 desde la parte controlada por el desarrollador, de la aplicación iLocate 111, al módulo de Lado Cliente de Administrador de Integridad (IMCS) 114 que es proporcionado por el operador de plataforma. El IMCS 114, tras recibir la señal de solicitud de inicio 405, genera otra señal de inicio 410 la cual se transmite al módulo de *plug-in* de servicio de Lado Servidor de Administrador de Integridad (IMSS) 254 situado dentro del dominio de pasarela 200. La señal 410 es una comunicación totalmente autenticada y cifrada entre el IMCS 114 y el IMSS 254 la cual aprovecha el conjunto de capas de servicio jerárquicas 252 proporcionadas como parte del motor de exposición de servicio 250. Durante esta comunicación, el IMCS 114 proporciona varios detalles sobre la aplicación iLocate 110 al IMSS incluyendo, en la presente forma de realización, detalles sobre una dirección de oyente (es decir, la dirección del Protocolo de Internet (IP) del servidor y el número de puerto) en la cual la aplicación iLocate 110 está a la escucha de mensajes de notificación proporcionados por el servidor de notificaciones 220. Adicionalmente, como parte de la comunicación 410, el IMSS 254 comunica al IMCS 114 detalles sobre la frecuencia con la cual debería iniciar una comunicación de latidos.

Tras completarse la comunicación 410, el IMCS 114 inicia una comunicación de latidos 415a. Esta es nuevamente una comunicación autenticada segura que aprovecha el conjunto de capas de servicio jerárquicas 252 las cuales garantizan que la parte llamante es de hecho la aplicación iLocate 110 y no un impostor, y que el IMCS está autorizado verdaderamente para contactar con el IMSS 254 en ese momento. La función principal del latido regular para comunicación 415a (y comunicaciones posteriores de latidos 415n) es especificar la frecuencia con la cual la aplicación iLocate 110 puede solicitar un servicio de cualquiera de los *plug-ins* de servicios 255 a 257 en los cuales está registrada debidamente la aplicación 110. (Registro que es llevado a cabo por el portal de inicio 205 ó el portal de operador de plataforma 210). Las comunicaciones subsiguientes de latidos 415 se producen a intervalos regulares y se usan como mecanismo para contener la frecuencia con la cual la aplicación 110 puede contactar con el motor de exposición de servicio 250 así como para proporcionar un mecanismo con el fin de garantizar que tanto el motor de exposición de servicio 250 como la aplicación 110 (así como la red de datos de interconexión) están funcionando todavía correctamente. En el documento WO 03/029975 pueden encontrarse detalles completos de la comunicación de latidos.

Habiéndose establecido de esta manera una línea de comunicación entre la aplicación 110 y la pasarela 200, a continuación la aplicación 110 se sitúa a la espera de solicitudes de usuario. Así, tras la recepción de una comunicación de solicitud de inicio de sesión 420 desde el PC 10, la aplicación 110 proporcionará al PC 10 acceso a su sitio web desde el cual se ofrecen varios servicios al usuario ("Papá") del PC 10. Tras iniciar la sesión, el usuario del PC 10 selecciona del sitio web proporcionado por la aplicación 110 la opción de enviar una señal de SMS con el texto insertado "¿Cómo estás?" que el usuario especifica que quiere que se envíe a un teléfono móvil identificado por su número de teléfono móvil lo cual provoca la transmisión de una señal de envío de SMS 425 desde el PC 10 a la aplicación iLocate 110.

Tras completarse la comunicación de envío de SMS 425, la aplicación 110 intenta iniciar una comunicación de envío de SMS 430 con el *plug-in* de SMS 257. Para lograr esto, se llevan a cabo varias actividades de nivel inferior, para cuyos detalles se remite al lector al documento WO 03/029975, aunque los mismos incluyen comprobar con el IMCS 114 que la aplicación 110 está autorizada en ese momento a iniciar dicha conexión así como negociar el conjunto de capas de servicio jerárquicas 252 dentro del motor de exposición de servicio 250. Una vez que se ha completado satisfactoriamente la comunicación 430, el *plug-in* de servicio de SMS 257 inicia otra comunicación de envío de SMS 435 al aparato telefónico de mano 20 (por medio de varios intermediarios incluyendo el centro de SMS 314 y la infraestructura de red móvil 305).

En el presente ejemplo, tras producirse la recepción del SMS, el usuario ("Kate") del aparato telefónico de mano 20 desea enviar un SMS de respuesta. Obsérvese que, en la presente forma de realización, el *plug-in* de servicio de SMS 257 tiene solamente un único número de teléfono asociado al mismo al cual deben dirigirse todos los mensajes

de texto SMS entrantes. En la presente forma de realización, el *plug-in* de servicio de SMS 257 mantiene un registro de números de teléfono de destino a los cuales ha enviado previamente mensajes de texto y usa la ID de llamante incluida en el mensaje de texto para determinar a qué aplicación se supone que se va a entregar el SMS recibido. En formas de realización alternativas, el *plug-in* de servicio SMS 257 podría tener un número arbitrariamente grande de números de teléfono móvil a los cuales se pueden dirigir SMSs entrantes o puede incluir un mecanismo de identificación alternativo para especificar de manera más explícita a quién debería entregarse el SMS entrante.

De este modo, en el presente ejemplo, el aparato telefónico de mano 20 genera un SMS de respuesta ("bien") que se transmite en forma de comunicación 440 desde el aparato telefónico de mano 20 al *plug-in* de servicio SMS 257 por medio de los mismos intermediarios que se usaron en la comunicación de envío de SMS 435.

Tras producirse la recepción del SMS de respuesta 440, el *plug-in* de servicio 257 "deja caer" un "evento" en la Cola de espera de JMS 215, de manera que dicho evento especifica que se va a enviar una notificación a la aplicación 110 solicitándole contactar con el *plug-in* de servicio SMS 257. Posteriormente, el servidor de notificaciones 220 saca el evento de la Cola de espera de JMS y lo procesa (de una manera tal como se describirá de forma más detallada posteriormente). La combinación de dejar caer el evento en la Cola de espera de JMS y su posterior recuperación de dicha Cola de espera de JMS por el servidor de notificaciones 220 constituye la comunicación de eventos 445 que se ilustra en la Figura 3.

Como resultado del procesado llevado a cabo por el servidor de notificaciones 220, el servidor de notificaciones 220 inicia una conexión de TCP/IP simple (no autenticada y no cifrada) 450 con el oyente 112 y transmite a través de esta conexión una notificación (cuya naturaleza se describirá de forma más detallada posteriormente) al oyente 112.

Tras producirse la recepción de la notificación, el oyente 112 reenvía la notificación por medio de una comunicación de reenvío de notificaciones 455 a un módulo de procesado de notificaciones (no mostrado) dentro de la parte principal (específica de la aplicación de cliente) 111 de la aplicación 110 la cual procesa la notificación, y establece de este modo que debería intentar contactar con el *plug-in* de servicio SMS 257.

Por ello la aplicación 110 intenta establecer una comunicación de recogida de SMS 460 (de una manera correspondiente a aquella en la que se estableció la comunicación de envío de SMS 430); obsérvese que, en la presente forma de realización, la notificación no especifica específicamente la finalidad para la cual la aplicación 110 debería contactar con el *plug-in* de servicio SMS 257, la notificación es en cambio simplemente de un tipo retrollamada (*callBack*) que provoca que la aplicación 110 por esta razón simplemente contacte con el *plug-in* de servicio 257 e informe al *plug-in* de servicio que está llevando a cabo esto como resultado de haber recibido una notificación. Al establecer la comunicación de recogida de SMS 460, el *plug-in* de servicio 257 traslada a la aplicación 110 cualesquiera datos que tenga para la aplicación 110 incluyendo el SMS que ha recibido del aparato telefónico de mano 20.

Como resultado de recibir el SMS de respuesta, la aplicación 110 actualiza la información que presenta al PC 10 en su sitio web, y cuando el PC 10 a continuación refresca, es decir recarga, la página web de la aplicación 110, se le notificará de que hay un SMS esperándole y que este se puede recoger seleccionando un enlace apropiado de la página web según la manera normal.

Método realizado por la aplicación iLocate 110

Tras haber descrito un conjunto ejemplificativo de interacciones entre los diversos elementos del sistema mostrado en las Figuras 1 y 2 en referencia a la Figura 3, a continuación se describirán de forma más general las etapas llevadas a cabo por la aplicación 110 cuando se produce un funcionamiento de manera normal dentro del sistema de la Figura 1 en referencia a la Figura 4. De este modo, tras comenzar el método, el flujo pasa a la etapa 405 en la cual la aplicación lleva a cabo cualquier inicialización incluyendo, y lo que es más importante, estableciendo el oyente 112. Tal como se ha mencionado anteriormente, el oyente 112 es una simple construcción de programación que permite que terceros establezcan una conexión TCP con el oyente siempre que se proporcionen la dirección IP y el número de puerto correctos. Tal como entenderán bien aquellos versados en la materia, especificando que al oyente solamente se pueden transmitir documentos de texto simple, es posible de esta manera establecer un oyente del tipo mencionado incluso por detrás de un cortafuegos bastante fuerte tal como el puesto a punto normalmente por organizaciones que desean proteger su red interna contra un acceso no autorizado por medio de Internet.

Al completarse la etapa 405, el método prosigue la etapa 410 en la cual el IMCS 114 establece una comunicación con el IMSS 254, con el fin de registrar la aplicación en la pasarela 200 según la manera antes descrita y que se describe también de forma más detallada en el documento WO 03/029975.

Al completarse la etapa 410, el método prosigue a la etapa 415 en la cual se determina si la aplicación ha recibido o no una solicitud de usuario para un servicio de la aplicación 110. Si se recibe una solicitud, el proceso se mueve a la etapa 420 en la cual la solicitud es procesada debidamente; si fuera necesario con el fin de cumplir esta solicitud, esto incluye contactar con uno o más servicios en el motor de exposición de servicio 250. Al completarse la etapa 420, el método vuelve a la etapa 415.

Si, en la etapa 415, se determina que no hay ninguna respuesta de un receptor esperando a ser procesada, el método prosigue a la etapa 425 en la cual se determina si la aplicación ha recibido o no, por medio de su oyente 112, una notificación del servidor de notificaciones 220. Si no se ha recibido dicha notificación, el método realiza un bucle de vuelta a la etapa 415 y continúa a la espera, después de esto, o bien de una nueva solicitud de usuario o bien de una nueva notificación.

Si en la etapa 425 se determina que se ha recibido una notificación nueva, el método prosigue a la etapa 430 en la cual se procesa la notificación para identificar qué *plug-in* de servicio 255 a 257 inició el envío de la notificación.

Tras completarse la etapa 430, el método prosigue a la etapa 435 en la cual se contacta con el *plug-in* de servicio identificado en la etapa 430. Al producirse el contacto con el *plug-in* de servicio relevante, a continuación la aplicación lleva a cabo cualesquiera etapas necesarias como consecuencia de la información que reciba del *plug-in* de servicio. Por ejemplo, en el caso del *plug-in* de SMS 257, es probable que se haya recibido un nuevo mensaje de texto SMS para su transmisión hacia adelante por parte de la aplicación 110 a uno de sus usuarios. De manera similar, si el *plug-in* de servicio relevante es el *plug-in* de localización GPS 256, entonces es probable que el *plug-in* de localización GPS 256 haya recibido recientemente información con respecto al paralelo de un dispositivo móvil según haya solicitado previamente un usuario de la aplicación 110.

Tras completarse la etapa 435, el método realiza un bucle de vuelta a la etapa 415 donde espera adicionalmente solicitudes de usuario o notificaciones para un procesado apropiado.

Método realizado por *plug-ins* de servicios 255 a 257

En referencia a continuación a la Figura 5, se describe seguidamente el método básico de funcionamiento de un *plug in* de servicio (tal como uno cualquiera de los *plug-ins* de servicio 255 a 257). Tras la iniciación de un *plug-in* de servicio, el método prosigue a la etapa 505 en la cual se determina si se ha recibido una solicitud de servicio nueva desde una aplicación, tal como la aplicación 110 en la cual está registrado el *plug-in* de servicio. En caso afirmativo, el método prosigue a la etapa 510 en la cual se procesa la solicitud. El proceso incluye comprobar que la solicitud es válida y, si fuera necesario para cumplir la solicitud, el *plug-in* de servicio contacta con un servicio remoto, tal como uno de los servicios 312, 314 proporcionados por el sistema de fondo 310. Tras completarse la etapa 510, el método realiza un bucle de vuelta a la etapa 505.

Si en la etapa 505 se determina que no se ha recibido ninguna solicitud nueva desde una aplicación, entonces el método prosigue a la etapa 515 en la cual se determina si el *plug-in* de servicio ha recibido cualquier dato para una de las aplicaciones en las cuales está registrado (tal como en la aplicación 110) (Nota Bene “datos” pretende referirse en este caso a cualquier clase de evento del cual tenga conocimiento el *plug-in* de servicio y sobre el cual considere que debería notificarse a una de las aplicaciones en las que está registrado). Si se determina que no se ha recibido ningún dato del tipo mencionado, entonces el método toma un bucle de vuelta a la etapa 505. En caso contrario, el método prosigue a la etapa 520 en la cual se determina si la aplicación a la que se va a notificar sobre los datos recién recibidos tiene o no un oyente registrado en la exposición de servicio 250, si la aplicación en cuestión no tiene un oyente adecuado el método toma un bucle de vuelta a la etapa 505. En este caso, el *plug-in* de servicio simplemente esperará hasta que la aplicación en cuestión intente de nuevo contactar con el *plug-in* de servicio y notificará a la aplicación los datos u otro evento automáticamente en ese momento.

No obstante, si se determina en la etapa 520 que la aplicación en cuestión sí tiene un oyente adecuado, entonces el método prosigue la etapa 525 en la cual la aplicación genera una solicitud de notificación. En la presente forma de realización, la solicitud de notificación adopta la forma de un objeto de mensaje tal como se usa en el ampliamente conocido Servicio de Mensajería Java (JMS).

El método a continuación prosigue a la etapa 525 en la cual el mensaje de solicitud de notificación generado se envía al servidor de notificaciones utilizando el Servicio de Mensajería Java. En particular, en la presente forma de realización, esto se realiza mediante la colocación del mensaje de solicitud de notificación, por parte del *plug-in* de servicio, en la Cola de espera de JMS 215, de la cual posteriormente el servidor de notificaciones 220 toma el mensaje de solicitud de notificación. Tras tomar el mensaje de notificación el servidor de notificaciones, dicho servidor de notificaciones procesa la solicitud y genera y transmite al oyente de aplicación apropiado una notificación en forma de un documento XML. Las operaciones llevadas a cabo por el servidor de notificaciones se ilustran en la Figura 5 por medio de la sub-rutina 700. La conexión indirecta (des-acoplada) entre el *plug-in* de servicio y el servidor de notificaciones se ilustra por medio de la línea de puntos en la Figura 5, la cual indica que la conexión es asíncrona a través de la Cola de espera de JMS 215.

Tras completarse la etapa 525, el método del *plug-in* de servicio toma un bucle de vuelta a la etapa 505. Obsérvese que las etapas realizadas en la sub-rutina 700 se describen de forma más detallada posteriormente en referencia a las Figuras 7a a 7d.

Servidor de notificaciones 220

En referencia a continuación a la Figura 6, se describen seguidamente las características o módulos de arquitectura de software del servidor de notificaciones 220 en la presente forma de realización.

Un módulo de oyente 610 lleva a cabo la función de sacar mensajes de la Cola de espera de JMS 215 que van dirigidos al Servidor de notificaciones 220 y los almacena como eventos (que se pueden modificar en cuanto a formato con respecto a los mensajes que se reciben de la Cola de espera de JMS 215 en aspectos menores) en unos medios de almacenamiento de eventos del tipo Primero en Entrar Primero en Salir (FIFO) 620.

Un módulo de *Spooler* 625 toma eventos almacenados en los medios de almacenamiento de eventos FIFO 620 y los ofrece a "Despachadores" los cuales recurren periódicamente al *spooler* para ver si hay algún evento pendiente con el *Spooler* 625. El *Spooler* se comunica con un módulo de Memoria caché 630 que a su vez se comunica con unos medios de almacenamiento de Mapa de Direcciones 635, con el fin de permitir que el *Spooler* averigüe la dirección IP y el número de puerto del oyente de una aplicación para la cual está destinado un evento que se va a despachar (la Memoria caché 630 y el Mapa de Direcciones 635 funcionan de manera normal de tal modo que si los detalles de la dirección deseada no se encuentran en la Memoria caché, la Memoria caché solicita los mismos del Mapa de Direcciones, traslada los detalles al *Spooler* 625 y almacena una copia de ellos durante un periodo de tiempo predeterminado para posibilitar que una consulta posterior de los mismos detalles sea respondida directamente por la Memoria caché sin tener que contactar con el Mapa de Direcciones 635). El *Spooler* 625 se comunica también con un módulo de Controlador de Reintentos 640 el cual recibe eventos que un despachador ha intentado despachar de manera no satisfactoria y determina si se debería reintentar su despacho en algún instante de tiempo posterior; en caso afirmativo, los mismos se almacenan hasta dicho momento y a continuación se vuelven a presentar a los medios de almacenamiento de eventos FIFO 620.

El Servidor de Notificaciones 220 incluye también un módulo de Fábrica de Despachadores 670 que monitoriza los medios de almacenamiento de eventos FIFO 620 y, si detecta que hay eventos nuevos en los medios de almacenamiento de eventos FIFO que requieren ser despachados, entonces genera un nuevo objeto de Despachador 671, 672. Cada objeto de Despachador 671, 672 se ejecuta en su propio hilo de ejecución y es responsable de solamente una única notificación en un instante de tiempo cualquiera, de manera que los retardos de red no se suman acumulativamente para retardar el envío de notificaciones posteriores hasta que se hayan enviado satisfactoriamente notificaciones precedentes (o se haya intentado el envío de las mismas y este haya fallado). Una vez que un Despachador 671, 672 ha finalizado el procesado de una notificación (o bien enviándola satisfactoriamente o bien con un informe de vuelta al *Spooler* 625), el mismo realiza una comprobación con el *Spooler* 625 para ver si hay una notificación pendiente que debe ser despachada, y en caso afirmativo la toma e intenta despacharla, si no se destruye a sí mismo; este comportamiento de dimensionamiento automático de las reservas equilibra la capacidad de despacho con respecto a la demanda del momento.

En la Figura 6 se muestra también el mecanismo mediante el cual Aplicaciones tales como la aplicación iLocate Live 111 pueden registrar la ID de *socket* (es decir, la dirección IP y el número de puerto) de su Oyente 112 en el Servidor de notificaciones. Tal como se muestra en la Figura 6, en la presente forma de realización, esto se puede realizar o bien mediante la notificación, por parte de la aplicación, a un *plug-in* de servicio (por ejemplo, el *plug-in* de Servicio SMS 257) directamente, el cual a continuación traslada los detalles de la dirección al propio Mapa de Direcciones 635, o bien de manera alternativa, la aplicación contacta con un punto de acceso común Módulo de control 680, con los detalles de la dirección de su Oyente que a continuación son trasladados por el Control 680 al Mapa de Direcciones 635.

Visión general del funcionamiento del servidor de notificaciones

Por lo tanto, el funcionamiento general del Servidor de Notificaciones se puede describir de la manera siguiente. La Fábrica de despachadores 670 monitoriza los medios de almacenamiento de eventos FIFO 620, y cuando determina que hay un exceso de eventos a la espera de su despacho, da inicio a un Despachador 671, 672 nuevo. Tal como se ha mencionado anteriormente, los Despachadores 671, 672 se ejecutan en su propio hilo de ejecución; esto es necesario debido a que los retardos de la red pueden retardar operaciones de despacho individuales durante periodos de tiempo extremadamente grandes (varios segundos). Los Despachadores 671, 672 recurren al *Spooler* 625 para que les proporcione un evento para su despacho; si el *Spooler* 625 informa de que no hay eventos a la espera, el Despachador 671, 672 se autodestruye.

El desarrollador de aplicaciones de cliente consigue un oyente compatible con XML de entre uno de los muchos proveedores que suministran dichos componentes y lo integra con el resto de la aplicación. El desarrollador entiende que todos los eventos de la plataforma de servicio se ajustarán a un formato normalizado (por ejemplo, ID de servicio, tipo de evento, parámetro de evento).

Durante el funcionamiento, se llevan a cabo las siguientes etapas:

1) El cliente, autenticado para la plataforma de servicio por una ID única, comunica la dirección (dirección IP y puerto de su oyente) o bien por medio de un punto común de acceso 680 ó bien por medio de cualquier servicio 256, 257.

2) La [ID+dirección] del oyente de cliente se presenta al Mapa de Direcciones 635.

3) El cliente hace uso de múltiples servicios 256, 257 por medio de sus APIs.

4) Se produce un evento y el servicio relevante 256, 257 produce un evento el cual se envía a la cola de eventos 215 del Servidor de Notificaciones. El evento incorpora la ID del cliente 110 al cual está destinado.

5) El Oyente 610 recupera el evento y lo coloca en los medios de almacenamiento de eventos FIFO (primero en entrar, primero en salir) 620.

6) Un Despachador 671, 672 recurre al *Spooler* 625 para servirle un evento.

7) El *Spooler* 626 coge el siguiente evento de los medios de almacenamiento de eventos FIFO 620.

8) El *Spooler* 625 extrae la ID de cliente y consulta una dirección en la Memoria caché 630. Si es necesario, la Memoria caché 630 consulta en los medios de almacenamiento de Mapa de Direcciones 635. El *Spooler* 625 suministra el evento y la dirección de entrega al Despachador 671, 672. El Despachador traduce el evento en un documento XML e intenta entregarlo al oyente de cliente 112.

9) Suponiendo que el primer intento de entrega falle, el Despachador 671, 672 informa de que la entrega ha fallado.

10) El *Spooler* 625 presenta el evento al Controlador de Reintentos 640 el cual descarta el evento (en caso de que el mismo haya alcanzado su límite de reintentos) o lo conserva durante un periodo de tiempo (el periodo de reintento).

11) Una vez que ha transcurrido el periodo de reintento del evento, el Controlador de Reintentos 640 lo presenta a los medios de almacenamiento de eventos FIFO 620.

12) Un Despachador 671, 672 recurre al *Spooler* para servirle un evento.

13) El *Spooler* 625 coge el siguiente evento de los medios de almacenamiento de eventos FIFO 620.

14) El *Spooler* 625 extrae la ID de cliente y consulta una dirección en la Memoria caché 630. La Memoria caché 630 consulta los medios de almacenamiento de Mapa de Dirección 635 si fuera necesario. El *Spooler* 625 suministra el evento y la dirección de entrega al Despachador 671, 672.

15) El Despachador 671, 672 traduce el evento en un documento XML y lo entrega al oyente de cliente 112.

16) El Despachador 671, 672 recurre al *Spooler* 625 para servirle un evento. Si el *Spooler* responde indicando que no hay más eventos en cola de espera, el Despachador 671, 672 se autodestruye.

Variaciones de este proceso permiten que el Servidor de Notificaciones 220 proporcione a eventos, diferentes límites de tentativas de reintento, periodos de reintento, persistencia, etcétera. El formato del registro de eventos usado por los servicios para presentar eventos (anterior etapa 4) en la presente forma de realización incorpora marcadores para indicar el comportamiento requerido de gestión de los eventos.

El Servidor de Notificaciones 220 puede detectar un fallo de la entrega y alertar al propietario del cliente por medios alternativos (por ejemplo, correo electrónico).

Al producirse la recepción de la notificación, el oyente 112 interacciona con la aplicación de cliente 111 la cual lleva a cabo una acción apropiada (invocando típicamente una llamada en el servidor en la API de servicio apropiada, por ejemplo para recuperar mensajes SMS entrantes).

En la presente forma de realización se ofrece también un método de prueba a partir del punto de acceso común 680 ó a través de cualquier servicio 255, 256, 257. Bajo solicitud, se produce un evento de prueba. El evento de prueba se gestiona como cualquier otro evento y se entrega al oyente registrado apropiado. El cliente 111 puede detectar un fallo del mecanismo de entrega del evento por la carencia de la entrega de la notificación del evento de prueba.

Visión detallada de operaciones de módulos específicos del Servidor de Notificaciones 220

En referencia a continuación a las Figuras 7a a 7e, se describen seguidamente las etapas específicas realizadas por cinco módulos/objetos significativos dentro del Servidor de Notificaciones 220.

Método del Módulo de Oyente 610

Por lo tanto, haciendo referencia en primer lugar a la Figura 7a, el método llevado a cabo por el Módulo de Oyente 610 comienza, después del inicio del método, en la etapa 705 en la cual se determina si hay o no un mensaje nuevo en la Cola de Espera de JMS 215, que vaya dirigido al Servidor de Notificaciones 220. Si no existe tal mensaje que está a la espera, entonces el método realiza un bucle de vuelta continuamente a la etapa 705 hasta que se detecte dicho mensaje.

Si, en la etapa 705, se detecta un mensaje dirigido al Servidor de Notificaciones 220, entonces el método prosigue a la etapa 710 en la cual se recupera el mensaje de la Cola de Espera de JMS. Tras completarse la etapa 710, el método prosigue a la etapa 715 en la cual el evento recuperado se almacena en los medios de almacenamiento de eventos FIFO 620. En la presente forma de realización, esto se realiza simplemente cogiendo el objeto de mensaje tal como se recupera de la Cola de Espera de JMS 215 y almacenándolo sin modificaciones. No obstante, en formas de realización alternativas, se podría leer el contenido significativo del mensaje y el mismo se podría poner en un formato alternativo (por ejemplo, un objeto Java diferente), y a continuación se podría almacenar en los medios de almacenamiento de eventos FIFO en este formato modificado. Tras completarse la etapa 715, el método realiza un bucle de vuelta a la etapa 705 y se sitúa a la espera de otro mensaje para su procesado.

Método del módulo de Fábrica de Despachadores 770

En referencia a continuación a la Figura 7b, el método llevado a cabo por la Fábrica de Despachadores 670 prosigue, después de la iniciación del método, a la etapa 720 en la cual se determina si hay algún evento en los medios de almacenamiento FIFO 620 a la espera de ser despachado. Si no existen tales eventos que se encuentren a la espera de ser despachados, entonces el método realiza un bucle continuamente de vuelta a la etapa 720 hasta que en los medios de almacenamiento FIFO 620 se detecte un evento a procesar.

Si en los medios de almacenamiento FIFO 620 se detecta un evento, entonces el método prosigue a la etapa 725 en la cual se genera un nuevo objeto de Despachador. En la presente forma de realización, tras completarse la etapa 725, el método realiza un bucle de vuelta a la etapa 720 y consulta nuevamente para ver si hay algún evento a la espera de ser despachado. Esto significa, en la presente forma de realización, que si el evento todavía no se ha recuperado de los medios de almacenamiento FIFO por parte del *spooler* antes del momento en el que el método vuelva a la etapa 720, podrían generarse otro u otros objetos de Despachador aún cuando solamente se requiera en realidad uno. Esto no generará ningún problema ya que dichos objetos de despachador en exceso simplemente se autodestruirán en su debido momento. No obstante, podrían usarse métodos más complicados para evitar que ocurriese esto, tales como la introducción de un pequeño retardo después de generar un despachador antes de volver a etapa 720 ó una comprobación para ver si ya se ha generado un despachador con respecto a un evento particular, etcétera. La línea de puntos entre la etapa 725 de la Figura 7b y la etapa inicial de la Figura 7d representa la instanciación de un nuevo objeto de Despachador cuyo método de funcionamiento se ilustra en la Figura 7d.

Método del módulo de *Spooler* 625

En referencia a continuación a la Figura 7c, el método llevado a cabo por el módulo de *Spooler* 625, después de iniciarse el método, prosigue a la etapa 730 en la cual se determina si un despachador ha solicitado un evento nuevo para su despacho a una aplicación de cliente (tal como la aplicación 110). Si no se detecta ninguna de estas solicitudes por parte de un objeto de despachador, entonces el método prosigue a la etapa 735 en la cual se determina si un despachador está intentando devolver un evento fallido (es decir, un evento que el despachador ha intentado despachar al oyente apropiado y no lo ha conseguido satisfactoriamente). En caso afirmativo, entonces en la etapa 740 el *Spooler* 625 toma el evento fallido y lo traslada al Controlador de reintentos 640. En caso contrario, el método realiza un bucle de vuelta a la etapa 730 y espera que un despachador contacte con él o bien para recibir un evento nuevo a despachar o bien para devolver un evento fallido.

Si, en la etapa 745, se determina que hay un despachador libre a la espera de un evento nuevo a despachar, el método prosigue a la etapa 745 en la cual el siguiente evento que se va a despachar se coge de los medios de almacenamiento FIFO 620 (obsérvese que aún cuando no se muestra explícitamente en la Figura 7c, si no hay eventos a la espera de su despacho en los medios de almacenamiento FIFO, entonces la etapa 750 se salta y, en la etapa 735, se informa al Despachador de que no hay eventos a la espera tras lo cual el despachador simplemente termina).

Tras completarse la etapa 745, el método prosigue a la etapa 750 en la cual, en la memoria caché (o el mapa de direcciones a través de la memoria caché, si así fuera necesario, se busca la dirección del oyente al cual está destinada la transmisión del evento, y a continuación el método prosigue a la etapa 755 en la cual el evento se

traslada al Despachador que a continuación intenta despachar el evento apropiadamente. Tras completarse la etapa 755, el método realiza un bucle de vuelta a la etapa 730 y espera nuevamente a que un despachador contacte con él o bien para recibir un evento nuevo a despachar o bien para devolver un evento fallido.

Método del objeto de Despachador 671, 672

En referencia a continuación a la Figura 7d, el método que lleva a cabo cada uno de los objetos de Despachador 671, 672, después de que haya sido instanciado (por el módulo de Fábrica de Despachadores 670 en la etapa 725), prosigue hacia la etapa 760 en la cual el Despachador 671, 672 contacta con el *Spooler* 625 para solicitar un nuevo evento a despachar. En la etapa 765, el Despachador determina si ha recibido un evento nuevo para su despacho desde el *Spooler* 625 ó si el *Spooler* ha informado de que en ese momento no tiene ningún evento a la espera de ser despachado. Si se cumple esto último (es decir, que no hay ningún evento a la espera de ser despachado), entonces el objeto de Despachador auto-finaliza (esta capacidad de autofinalización es una característica convencional en Java y otros lenguajes orientados a objetos, y evita que objetos no utilizados consuman recursos informáticos).

No obstante, si en la etapa 765 se determina que al Despachador se le ha proporcionado un evento nuevo para ser despachado, entonces el método prosigue a la etapa 770 en la cual el Despachador genera, a partir de la información contenida en el evento trasladado al mismo por el *Spooler*, un documento XML que constituye la sustancia de la notificación a entregar a la aplicación de cliente; después de generar el documento XML, a continuación el Despachador intenta transmitir la notificación al Oyente utilizando los detalles de dirección trasladados al Despachador por el *Spooler* en el momento de trasladarle el evento.

Tras completarse la etapa 770, el método prosigue a la etapa 775 en la cual se determina si la entrega resultó satisfactoria. Si la entrega resultó satisfactoria, el método toma un bucle de vuelta a la etapa 760 para solicitar otro evento de cara a despacharlo. No obstante, si la entrega resultó insatisfactoria por algún motivo, entonces el método prosigue a la etapa 780 en la cual el evento entregado insatisfactoriamente (es decir, fallido) se traslada de vuelta al *Spooler* que a su vez pasará el evento fallido al Controlador de Reintentos 640. Tras completarse la etapa 780, el método realiza nuevamente un bucle de vuelta a la etapa 760.

Método del módulo de Controlador de Reintentos 640

En referencia a continuación a la Figura 7e, el método llevado a cabo por el módulo de Controlador de Reintentos 640, después de la iniciación del método, prosigue a la etapa 782 en la cual se determina si se ha trasladado un evento fallido al mismo desde el *Spooler* 625. En caso afirmativo, el método prosigue a la etapa 784 en la cual se determina si el evento es tal que debe realizarse un reintento del mismo. En la presente forma de realización, esto se determina leyendo un parámetro de reintento asociado al evento para ver si el mismo tiene valor cero. El parámetro de reintento especifica el número de veces que se va a reintentar el evento, en caso de fallo en su entrega, y puede adoptar cualquier valor entero entre cero, que indica que no se va a producir ningún reintento en absoluto, y algún valor máximo predeterminado; el parámetro de reintento es fijado primero por el *plug-in* de servicio que genera originariamente el evento el cual está situado en la Cola de Espera de JMS 215.

Si se determina en la etapa 784 que no se va a producir ningún reintento del evento, el método prosigue a la etapa 786 en la cual el evento simplemente es eliminado por el Controlador de Reintentos y el método realiza un bucle de vuelta a la etapa 782. Alternativamente, si en la etapa 784 se determina que se va a producir un reintento del evento, entonces el método prosigue a la etapa 788 en la cual el evento es almacenado por el Controlador de Reintentos, y se calcula y monitoriza el momento para reintentar el evento. En la presente forma de realización, el momento para realizar un reintento del evento se calcula leyendo un parámetro de intervalo de reintento y sumando este al momento actual dado por el reloj del sistema (no mostrado). Después de calcular este tiempo de reintento, el método vuelve a la etapa 782.

Si en la etapa 782 se determina que no se ha recibido ningún evento fallido, entonces, en lugar de proseguir hacia la etapa 784, el método prosigue a la etapa 790 en la cual se determina si se ha alcanzado o superado el tiempo de reintento (el primero) correspondiente al(a los) evento(s) que está(n) a la espera de reintento. En caso negativo, entonces el método realiza un bucle de vuelta a la etapa 782. En caso contrario, el método prosigue a la etapa 792 en la cual el parámetro de reintento del evento cuyo tiempo de reintento ha sido alcanzado o superado se decrementa en uno. Tras completarse la etapa 792, el método prosigue a la etapa 794 en la cual el evento se coloca a continuación en los medios de almacenamiento FIFO 620 desde los cuales el mismo será cogido posteriormente de nuevo por el *Spooler* 625 y este último realizará un reintento del mismo según la manera descrita previamente. Tras completarse la etapa 794, el método del Controlador de Reintentos 640 realiza un bucle de vuelta a la eta 782.

Formato de la notificación

En la presente forma de realización, cada notificación, según es enviada a través de la red no protegida entre el dominio de pasarela 200 y el dominio de aplicación 100, adopta la forma de un documento de Lenguaje de Marcado Extensible (XML) validable por el siguiente archivo de Definición de Tipo de Documento (DTD):

```
<!ELEMENT event (parameter*)>
<!ATTLIST event
      appAccID   CDATA       #REQUIRED
      serviceID  CDATA       #REQUIRED
      type CDATA       #REQUIRED
>
<!ELEMENT parameter (#PCDATA)>
<!ATTLIST parameter
    name  CDATA       #REQUIRED
>
```

Esto establece en esencia que una notificación que se ha proporcionado en forma de un documento XML validable por este DTD será un “evento” el cual puede contener cero o más “parámetros” (que constituyen los “elementos” hijos del “evento”). El evento debe tener tres atributos denominados “appAccID” (que es la identidad de la aplicación de cliente para la cual va destinada la notificación), “serviceID” (que es la identidad del *plug-in* de servicio que está enviando la notificación) y “tipo” (que especifica el tipo de la notificación, por ejemplo notificaciones de prueba, o notificaciones de retrollamada). Especifica también que cada parámetro debe tener un atributo denominado “nombre” (que aporta el nombre del parámetro) y contiene datos de caracteres analizables sintácticamente (es decir, prácticamente cualquier cosa).

Por ejemplo, el siguiente archivo XML sería validable por este DTD:

```
<?xml version="1.0"?>
<!DOCTYPE event SYSTEM "event.dtd">
<event appAccID="iLocate" serviceID="SMS" type="callBack">
<parameter name="Event_Number">1</parameter>
<parameter name="Param2">Val2</parameter>
<parameter name="Param2">Val3</parameter>
</event>
```

el cual, esencialmente, especifica que la notificación va destinada a la aplicación iLocate 110, que se ha enviado desde el *plug-in* de servicio SMS 257 y que su tipo es un tipo de retrollamada. Adicionalmente, contiene 3 parámetros, el primero de los cuales se denomina Event_Number y tiene el valor 1 (es decir, este es un número de serie para permitir que la aplicación receptora distinga esta notificación con respecto a posteriores notificaciones similares y también para evitar que la aplicación actúe sobre la misma notificación más de una vez en caso de que sea enviada erróneamente más de una vez por la red). Contiene también otros dos parámetros denominados Param2 y Param3 que tienen respectivamente los valores Val2 y Val3 los cuales son redundantes en este ejemplo, pero se podrían usar para especificar, por ejemplo, las identidades de las partes 9, 19 entre las cuales se está enviando un mensaje de texto que será cogido por la aplicación de cliente, etcétera.

Oyente 112

En la presente forma de realización, el Oyente 112 se forma utilizando el conjunto de herramientas API Simple para XML (SAX) el cual es un juego de herramientas para analizar sintácticamente XML y enviar sin solicitud previa (*pushing*) eventos genéricos de la XML a aplicaciones. Está disponible para Java así como otros entornos tales como Microsoft. En la presente forma de realización, la aplicación iLocate 110 está escrita en Java.

En la presente forma de realización, el Oyente 112 se forma utilizando el juego de herramientas SAX que, tal como entenderán claramente aquellos versados en la materia, requiere registrar un administrador de contenido, el cual es un método (objeto) (Java) dentro de la aplicación de cliente 110, que sabe qué elementos esperar dentro del XML y cómo gestionar dichos elementos (y sus valores) cuando los mismos son entregados. Además, tal como apreciarán nuevamente aquellos versados en la materia, el oyente 112 se establece también con el DTD, utilizando el juego de herramientas SAX, de tal manera que valide el XML entrante con respecto al DTD y recurra a un método de administrador de errores (nuevamente dentro del cliente 110) si hay algún error en el XML. A continuación, la aplicación de cliente 110 abre un *socket* en la red, y cuando obtiene una conexión del servidor abre esa conexión como un flujo continuo de entrada y lo traslada al administrador de contenido (como parámetro en el método *parse*() proporcionado por el juego de herramientas SAX). A medida que analiza sintácticamente el XML, los métodos heredados proporcionados por el juego de herramientas SAX permiten que el método de administrador de contenido tenga conocimiento de cada punto significativo en el análisis sintáctico del XML (documento de inicio, elemento de inicio, etcétera). De esta manera (la cual constituye el uso totalmente normalizado del SAX), el administrador de contenido puede construir un registro de todos los valores de dentro del XML (appAccID, serviceID, tipo, parámetros). Cuando el administrador de contenido llega a tener conocimiento de que se ha alcanzado el final del elemento de evento, el administrador de contenido toma el objeto de evento que ha poblado con los valores trasladados al mismo durante el análisis sintáctico del documento XML y lo traslada a un administrador de eventos el

cual reacciona al evento según la manera adecuada (por ejemplo, contactando con el *plug-in* de Servicio identificado – y por lo tanto, por ejemplo, sacando mensajes SMS del servidor).

El objeto de evento del lado cliente (es decir, en el dominio de aplicación 100) es distinto al objeto que representa el evento en el lado servidor (es decir, en el dominio de pasarela 200) por diversos motivos – hay muchos parámetros dentro del objeto de evento en el lado servidor que se utilizan en la entrega del evento (por ejemplo, el parámetro de reintento) y por tanto no es apropiado que el cliente los vea. Además, la finalidad de utilizar XML es desacoplar los lados de servidor y de cliente. Resultará también evidente que el objeto de evento en el lado cliente se define fuera del SAX. El SAX únicamente le dice al administrador de contenido lo que está encontrando en el XML y el administrador de contenido utiliza esa información según requiera la aplicación.

Obsérvese que las aplicaciones de cliente, tales como la aplicación iLocate 110, se pueden desarrollar en cualquier paradigma de desarrollo deseado, tal como Java, Perl, Microsoft, etcétera. Los desarrolladores de Java pueden decidir usar otros juegos de herramientas XML para desarrollar el módulo de Oyente (equivalente al Oyente 112), tales como DOM o JDOM. Los desarrolladores de Microsoft podrían utilizar el SAX o el Microsoft XML Parser (MSXML). MSXML. Los desarrolladores de Perl también pueden utilizar el SAX o algún otro juego de herramientas proporcionado específicamente para Perl. Otros entornos de desarrollo de cliente podrían usar sus propios juegos de herramienta XML.

REIVINDICACIONES

1. Sistema, que comprende un primer subsistema (250, 255, 256, 257) y una pasarela (250, 220, 252, 215, 254) para ofrecer unos servicios (255, 256, 257) proporcionados por el primer subsistema (250) a uno o más subsistemas de alojamiento de aplicaciones (110) por medio de la pasarela (250, 220, 252, 215, 254), estando la pasarela (250, 220, 252, 215, 254) y el o cada subsistema de alojamiento de aplicaciones (110) dispuestos para permitir que el o cada subsistema de alojamiento de aplicaciones (110) inicie una conexión segura y autenticada desde el o cada subsistema de alojamiento de aplicaciones (110) a la pasarela (250, 220, 252, 215, 254) por medio de una conexión de red de datos no segura, y estando la pasarela (250, 220, 252, 215, 254) lógicamente conectada al primer subsistema (250, 255, 256, 257) para posibilitar que los servicios (255, 256, 257) proporcionados por el primer subsistema (250) sean proporcionados al o a cada subsistema de alojamiento de aplicaciones (110) por medio de una conexión segura y autenticada, incluyendo la pasarela (250, 220, 252, 215, 254) un servidor de notificaciones (220) para enviar notificaciones a uno o más de entre los subsistemas de alojamiento de aplicaciones (110) por medio de una conexión no autenticada y no cifrada para notificarle o notificarles que debería o deberían iniciar una conexión autenticada segura con la pasarela (250, 220, 252, 215, 254) cuando al servidor de notificaciones (220) se le solicite hacerlo por cualquiera de los servicios (255, 256, 257) ofrecidos por el primer subsistema (250); estando el sistema dispuesto para pasar los datos, a los cuales se denominará en lo sucesivo datos que se van a transmitir, desde uno de los servicios (255, 256, 257) a uno de entre el subsistema o subsistemas de alojamiento de aplicaciones (110) al:

tener el respectivo servicio unos medios para detectar la aparición de un evento que da como resultado unos datos que se van a transmitir en el respectivo servicio;

tener además el respectivo servicio unos medios para solicitar al servidor de notificaciones que envíe una notificación al respectivo subsistema de alojamiento de aplicaciones (110) por medio de una conexión no autenticada y no cifrada; y

tener el respectivo subsistema de alojamiento de aplicaciones (110) unos medios para recibir la notificación, para establecer una conexión segura y autenticada con la pasarela y para invocar una llamada al respectivo servicio por medio de la conexión segura y autenticada con el fin de recuperar los datos que se van a transmitir a través de la conexión segura y autenticada.

2. Sistema según la reivindicación 1, en el que la notificación adopta la forma de un archivo de datos no ejecutable.

3. Sistema según la reivindicación 2, en el que la notificación adopta la forma de un archivo de texto simple que contiene un documento de Lenguaje de Marcado Extensible, XML.

4. Sistema según la reivindicación 1, en el que los medios de notificación (220) se pueden hacer funcionar para ejecutar unos hilos de procesado independientes, con el fin de controlar el reenvío de notificaciones independientes a la aplicación de cliente.

5. Sistema según la reivindicación 1, en el que los medios de notificación (220) incluyen unos medios para permitir que cada servicio (255, 256, 257) proporcionado por el primer subsistema (250) especifique el número de veces que se va a reintentar una notificación en caso de un fallo de entrega de la notificación y unos medios para reintentar la entrega de la notificación hasta el número especificado de veces, en caso de fallo de entrega de la notificación a través de la red no segura.

6. Sistema según cualquiera de las reivindicaciones anteriores, en el que los medios de notificación (220) reciben unas notificaciones de diversos servicios (255, 256, 257) ofrecidos por el primer subsistema (250) y las reenvía a diversos subsistemas de alojamiento de aplicaciones (110).

7. Método para pasar datos, a los cuales se denominará en lo sucesivo datos que se van a transmitir, desde un respectivo servicio de entre uno o más servicios (255, 256, 257) proporcionados por un primer subsistema (250, 255, 256, 257) a uno o más subsistemas de alojamiento de aplicaciones por medio de una pasarela (250, 220, 252, 215, 254), estando la pasarela (250, 220, 252, 215, 254) y el o cada subsistema de alojamiento de aplicaciones (110) dispuestos para permitir que el o cada subsistema de alojamiento de aplicaciones (110) inicie una conexión segura y autenticada desde el o cada subsistema de alojamiento de aplicaciones (110) a la pasarela (250, 220, 252, 215, 254) por medio de una conexión de red de datos no segura, estando la pasarela (250, 220, 252, 215, 254) lógicamente conectada al primer subsistema (250, 255, 256, 257) para posibilitar que los servicios (255, 256, 257) proporcionados por el primer subsistema (250, 255, 256, 257) sean proporcionados al o a cada subsistema de alojamiento de aplicaciones (110) por medio de una conexión segura y autenticada, e incluyendo la pasarela un servidor de notificaciones (220) para enviar notificaciones a uno o más de entre los subsistemas de alojamiento de aplicaciones (110) por medio de una conexión no autenticada y no cifrada para notificarle o notificarles que debería o deberían iniciar una conexión autenticada segura con la pasarela cuando al servidor de notificaciones se le solicite hacerlo por cualquiera de los servicios (255, 256, 257) ofrecidos por el primer subsistema (250), comprendiendo el método:

detectar, por parte de un respectivo servicio (255, 256, 257), la aparición de un evento que da como resultado los datos que se van a transmitir en el servicio respectivo;

solicitar, por parte del respectivo servicio, que los medios de notificación envíen una notificación al subsistema de alojamiento de aplicaciones respectivo (110) por medio de una conexión no autenticada y no cifrada, con el fin de notificarle o notificarles que debería o deberían iniciar una conexión autenticada segura con la pasarela (250, 220, 252, 215, 254); y a continuación

establecer, por parte del subsistema de alojamiento de aplicaciones (110), una conexión segura y autenticada con la pasarela (250, 220, 252, 215, 254) como respuesta a la recepción de la notificación y a continuación, invocar una llamada al servicio iniciador (255, 256, 257) por medio de esta conexión para recuperar los datos que se van a transmitir a través de la conexión segura y autenticada.

8. Programa de ordenador o conjunto de programas de ordenador para controlar uno o más procesadores de ordenador con el fin de llevar a cabo las etapas de la reivindicación 7 durante la ejecución del programa de ordenador o conjunto de programas.

9. Soporte legible por ordenador, que es portador del programa de ordenador o conjunto de programas de la reivindicación 8.

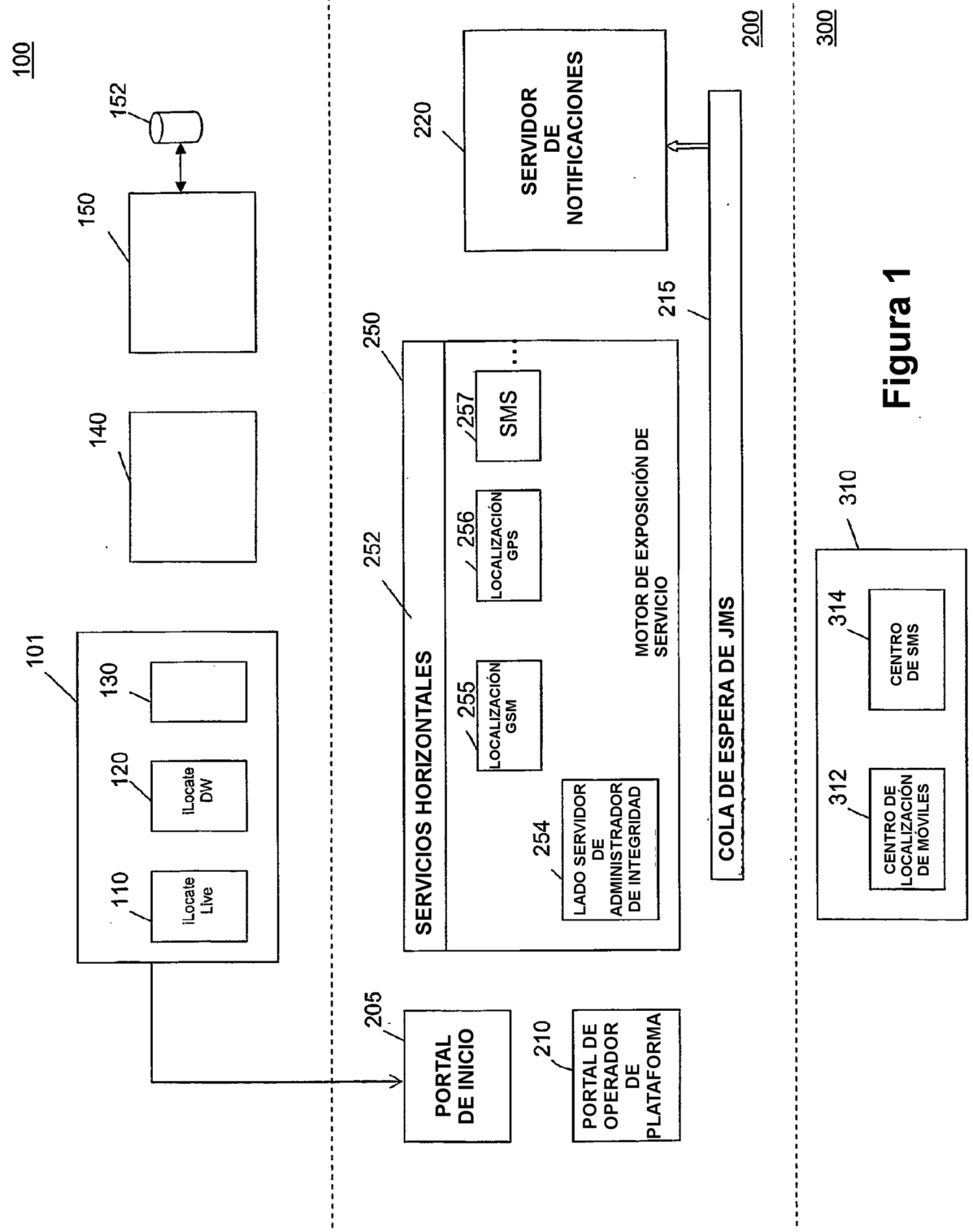
100
152
150
140
101
130
120
iLocate DW
110
iLocate Live
205
PORTAL DE INICIO
210
PORTAL DE OPERADOR DE PLATAFORMA
250
252
SERVICIOS HORIZONTALES
255
LOCALIZACIÓN GSM
256
LOCALIZACIÓN GPS
257
SMS
254
LADO SERVIDOR DE ADMINISTRADOR DE INTEGRIDAD
MOTOR DE EXPOSICIÓN DE SERVICIO
215
COLA DE ESPERA DE JMS
220
SERVIDOR DE NOTIFICACIONES
200
300
310
314
CENTRO DE SMS
312
CENTRO DE LOCALIZACIÓN DE MÓVILES

Figura 1

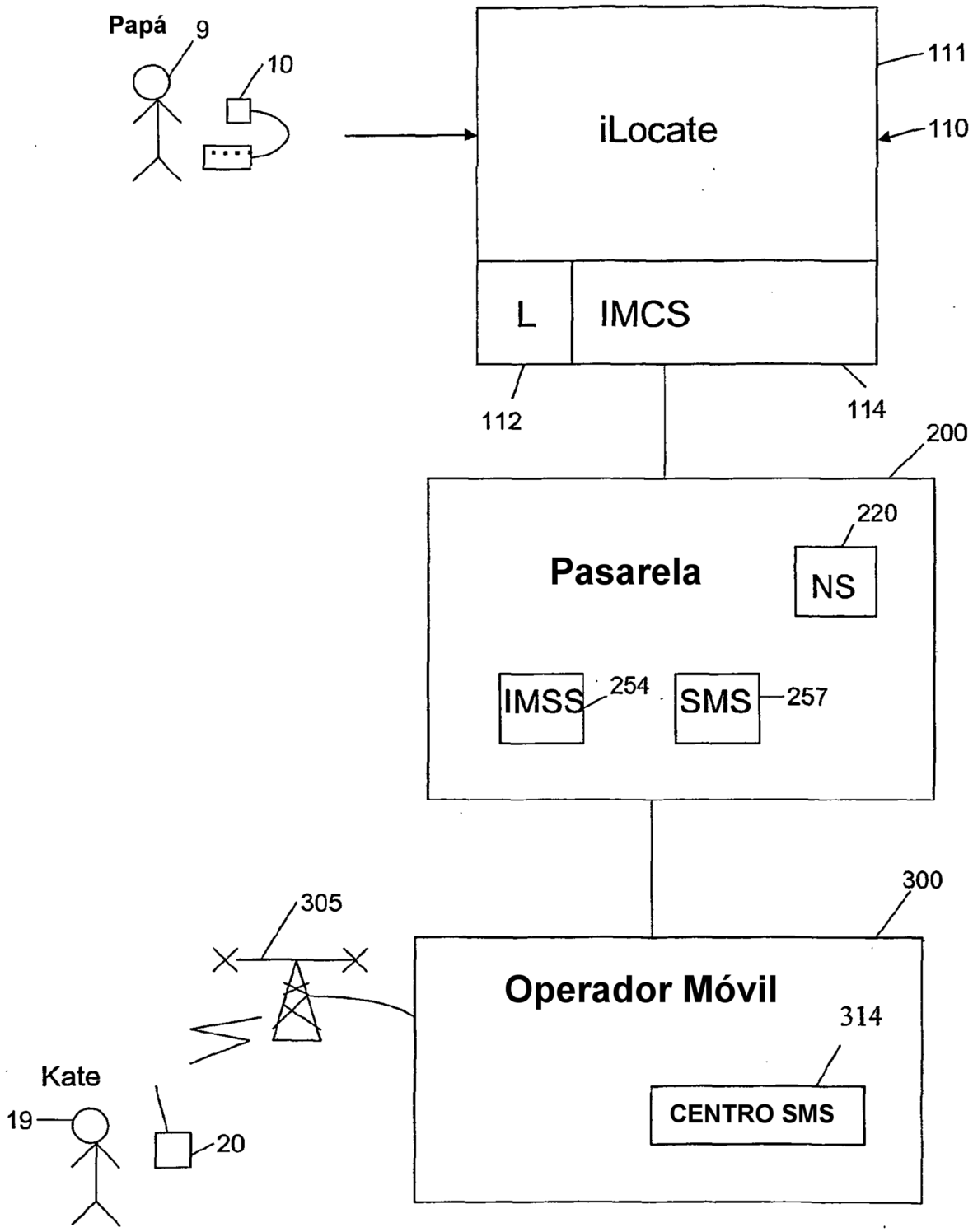
Papá
9
10
iLocate
111
110
L
IMCS
112
114
200
Pasarela
220
NS
IMSS
254
SMS
257
300
305
Operador Móvil
314
CENTRO SMS
Kate
19
20

FIGURA 2

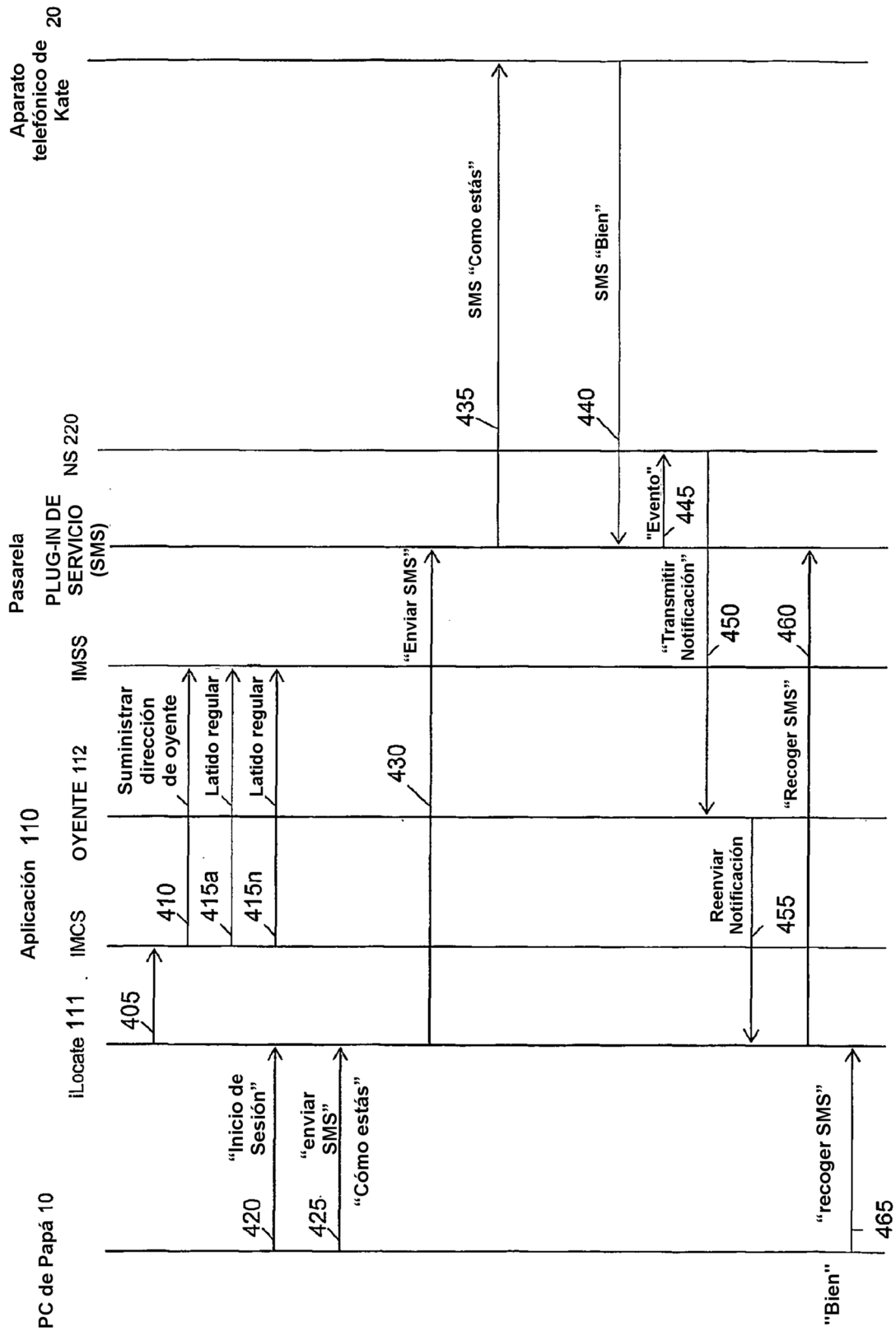
PC de Papá 10
iLocate 111
Aplicación 110
IMCS
OYENTE 112
IMSS
Pasarela
PLUG-IN DE SERVICIO (SMS)
NS 220
Aparato telefónico de Kate 20
405
410
415a
415n
420
425
430
435
440
445
450
455
460
465
"Inicio de Sesión"
"enviar SMS"
"Cómo estás"
Suministrar dirección de oyente
Latido regular
Latido regular
"Enviar SMS"
SMS "Como estás"
SMS "Bien"
"Evento"
"Transmitir Notificación"
Reenviar Notificación
"Recoger SMS"
"recoger SMS"
"Bien"

Figura 3

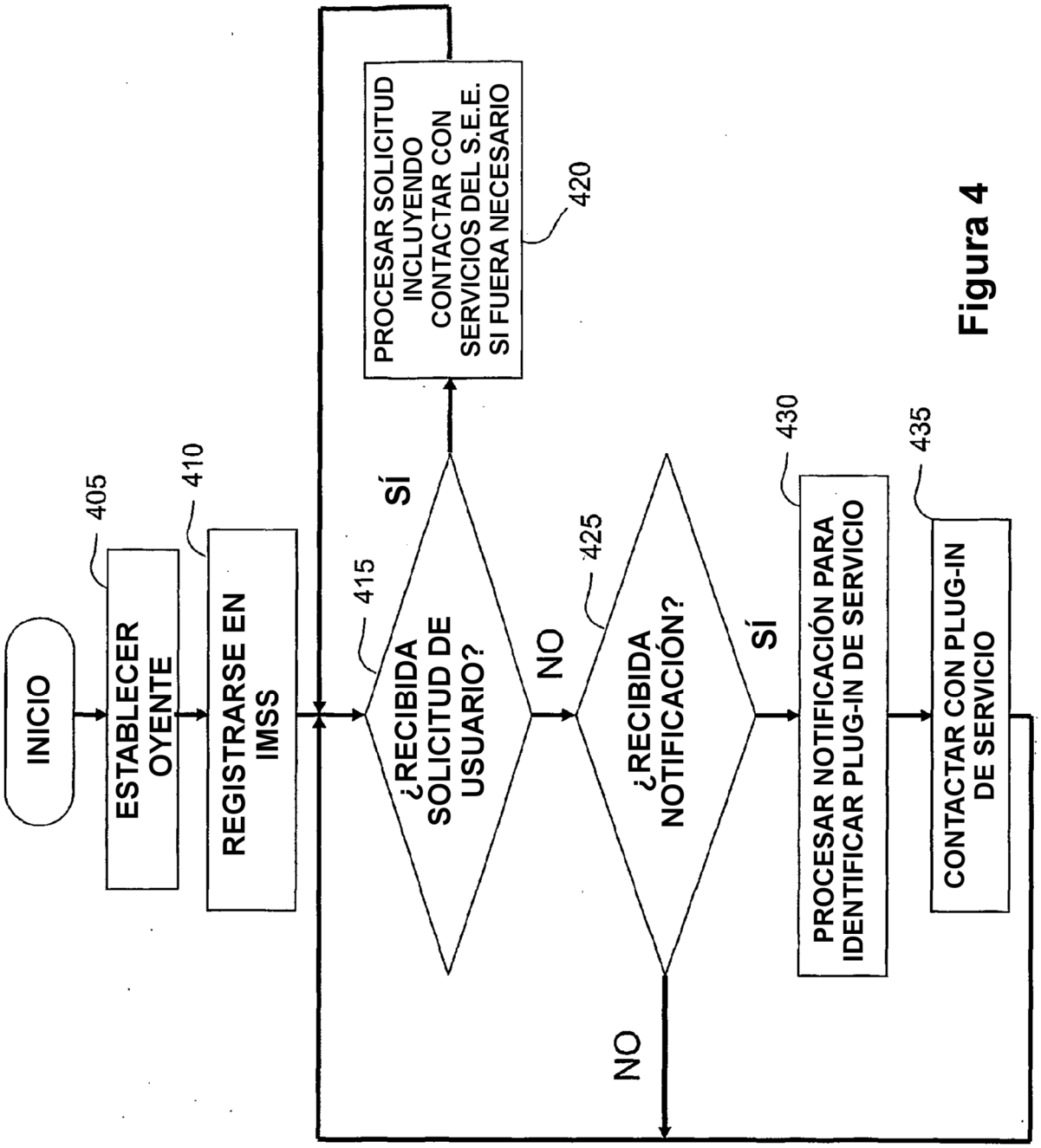
INICIO
ESTABLECER OYENTE
405
REGISTRARSE EN IMSS
410
¿RECIBIDA SOLICITUD DE USUARIO?
415
SÍ
PROCESAR SOLICITUD INCLUYENDO CONTACTAR CON SERVICIOS DEL S.E.E. SI FUERA NECESARIO
420
NO
¿RECIBIDA NOTIFICACIÓN?
425
SÍ
PROCESAR NOTIFICACIÓN PARA IDENTIFICAR PLUG-IN DE SERVICIO
430
CONTACTAR CON PLUG-IN DE SERVICIO
435
NO

Figura 4

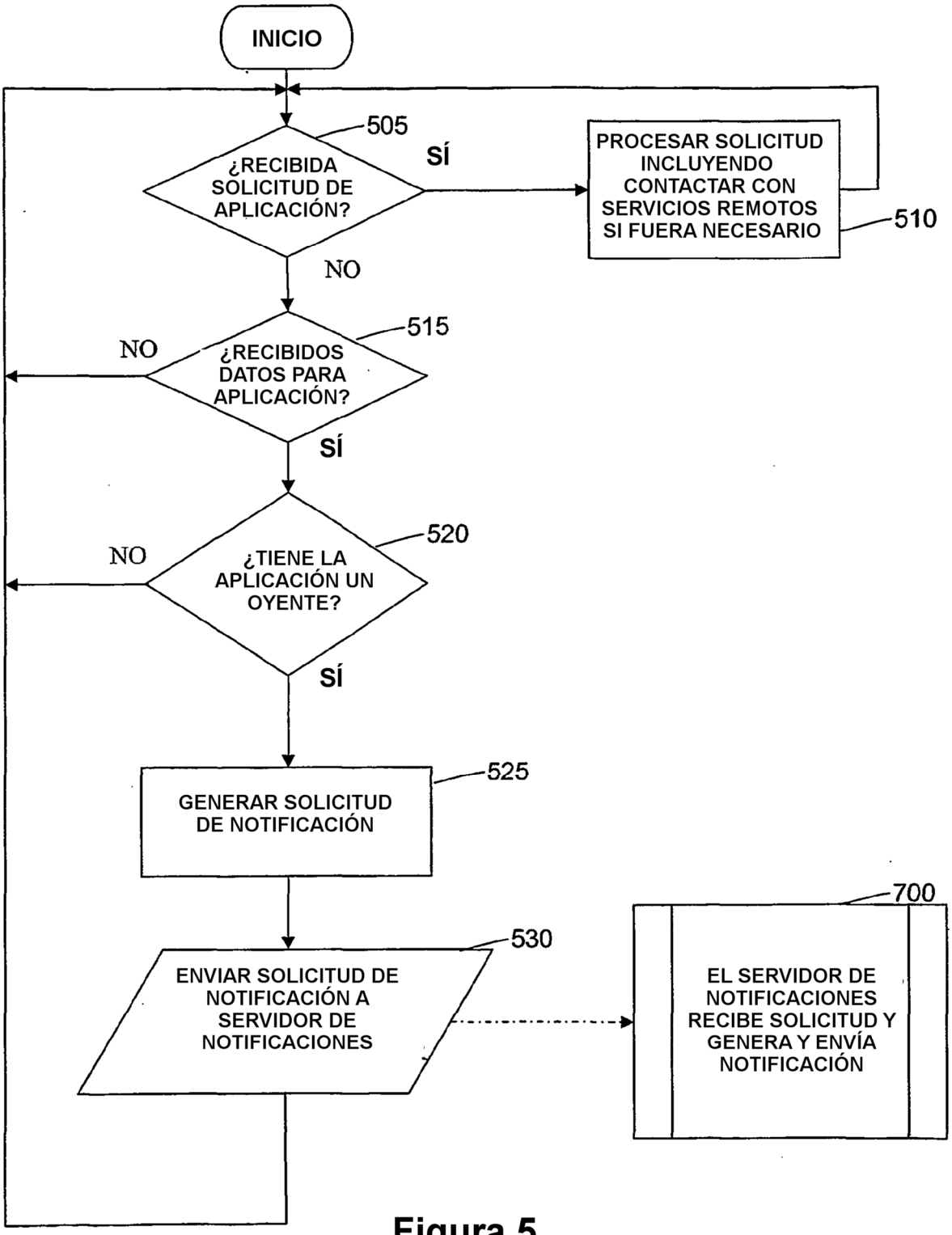
INICIO
505
¿RECIBIDA SOLICITUD DE APLICACIÓN?
SÍ
PROCESAR SOLICITUD INCLUYENDO CONTACTAR CON SERVICIOS REMOTOS SI FUERA NECESARIO
510
NO
515
NO
¿RECIBIDOS DATOS PARA APLICACIÓN?
SÍ
520
NO
¿TIENE LA APLICACIÓN UN OYENTE?
SÍ
525
GENERAR SOLICITUD DE NOTIFICACIÓN
700
530
ENVIAR SOLICITUD DE NOTIFICACIÓN A SERVIDOR DE NOTIFICACIONES
EL SERVIDOR DE NOTIFICACIONES RECIBE SOLICITUD Y GENERA Y ENVÍA NOTIFICACIÓN

Figura 5

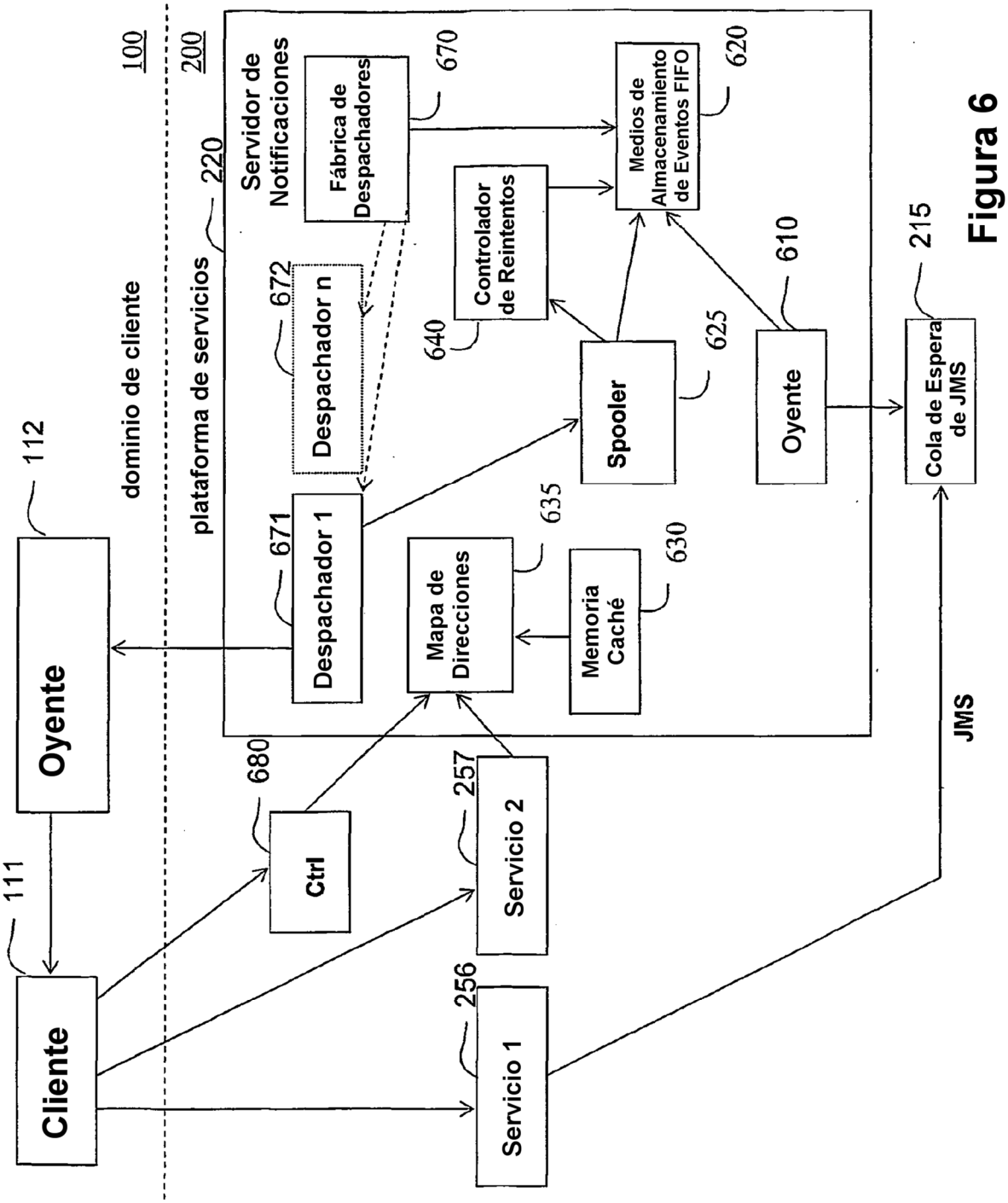
100
dominio de cliente
200
plataforma de servicios
Servidor de Notificaciones
220
Fábrica de Despachadores
670
Medios de Almacenamiento de Eventos FIFO
620
Controlador de Reintentos
640
Despachador n
672
Despachador 1
671
Spooler
625
Oyente
610
Cola de Espera de JMS
215
Mapa de Direcciones
635
Memoria Caché
630
Oyente
112
Ctrl
680
Servicio 2
257
Servicio 1
256
Cliente
111
JMS

Figura 6

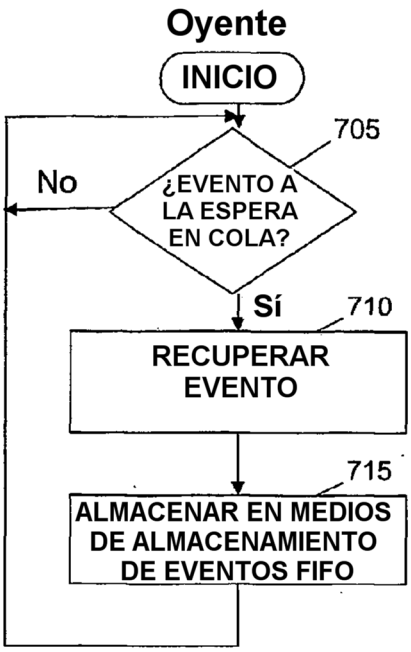

Figura 7a

Figura 7b

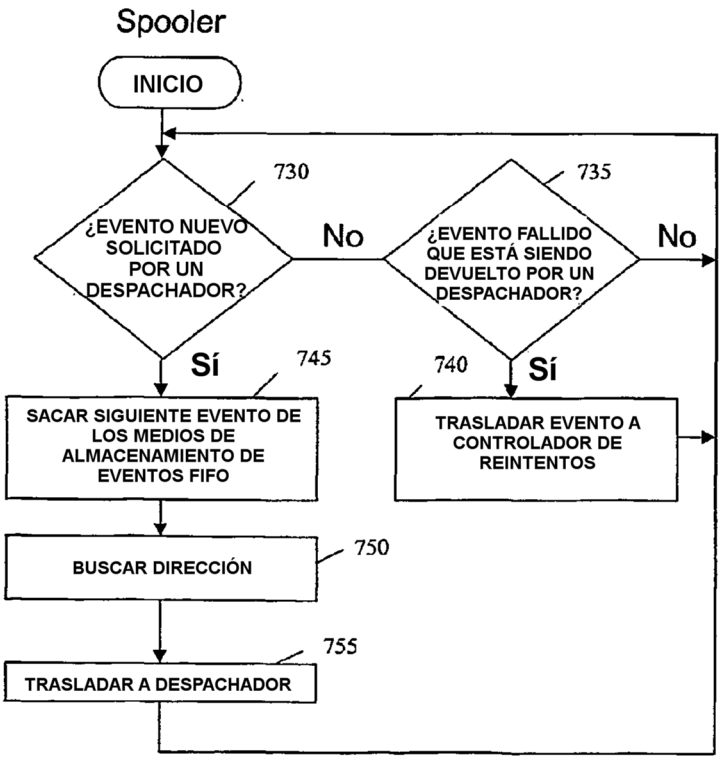

Figura 7c

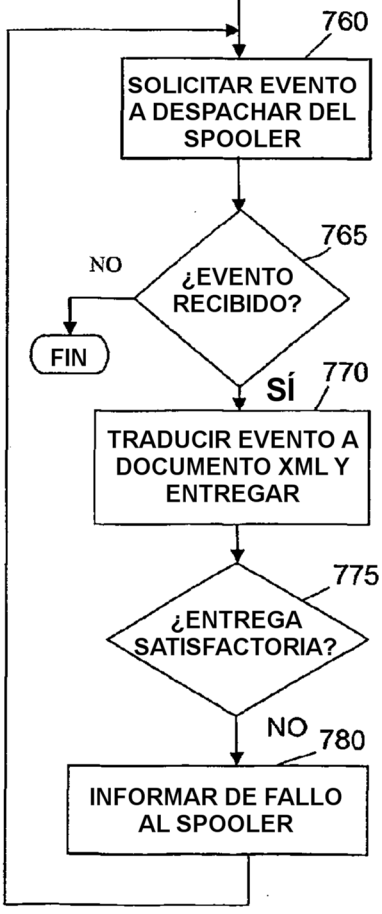

Figura 7d

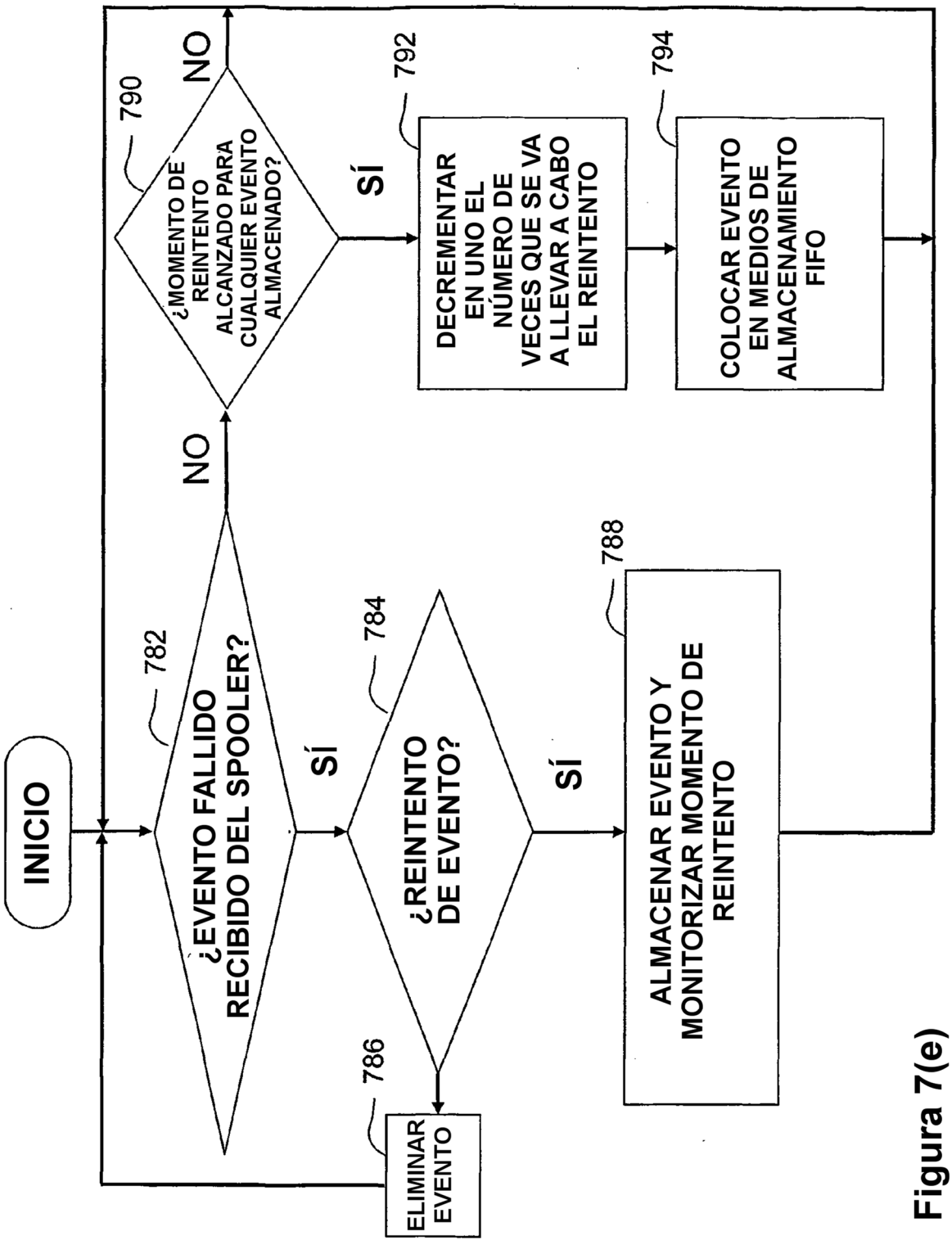
INICIO
782
¿EVENTO FALLIDO RECIBIDO DEL SPOOLER?
NO
SÍ
784
¿REINTENTO DE EVENTO?
NO
SÍ
786
ELIMINAR EVENTO
788
ALMACENAR EVENTO Y MONITORIZAR MOMENTO DE REINTENTO
790
¿MOMENTO DE REINTENTO ALCANZADO PARA CUALQUIER EVENTO ALMACENADO?
NO
SÍ
792
DECREMENTAR EN UNO EL NÚMERO DE VECES QUE SE VA A LLEVAR A CABO EL REINTENTO
794
COLOCAR EVENTO EN MEDIOS DE ALMACENAMIENTO FIFO

Figura 7(e)