Sistema de coordinación de servicios en redes ad-hoc para situaciones de catástrofes

Laura Díaz-Casillas, Marifeli Sedano, Mercedes Garijo, Gregorio Fernández
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
ldcasillas@gsi.dit.upm.es, {marifeli, mga, gfer}@dit.upm.es

***Resumen*—En este artículo se describe SCSANES (Service Coordination System adapted to Ad-hoc Networks in Emergency Situations), un sistema de coordinación de servicios capaz de adaptarse a un entorno dinámico, compuesto por un conjunto de dispositivos con diferentes capacidades, característico de las redes ad-hoc, y apto para situaciones de emergencia.**

El objetivo de SCSANES es facilitar la comunicación y el uso de servicios a los usuarios involucrados en una catástrofe. SCSANES gestiona el intercambio y la representación de información asociada a la situación de emergencia, como pueden ser alarmas, víctimas o recursos disponibles, de forma que el usuario pueda informar y conocer el estado de la catástrofe fácilmente. Además, permite la comunicación entre los usuarios mediante el intercambio de mensajes y la provisión y el acceso a servicios.

El sistema desarrollado se basa en un *middleware* implementado mediante una arquitectura de memoria compartida basada en espacios de tuplas. El sistema se adapta al entorno distribuido en el que se encuentra al permitir trabajar a cada nodo de manera independiente, pero a la vez coordinarse con el resto de nodos activos para ampliar sus funcionalidades y por tanto, las del sistema en conjunto.

I. Introducción

Los sistemas de comunicación actuales empleados en situaciones de emergencia permiten cierta interacción entre los equipos de trabajo implicados, aunque con un intercambio de información normalmente limitado a comunicaciones vocales, requiriéndose el desarrollo de una nueva tecnología que facilite dicha tarea.

La solución tecnológica actual se basa en el uso de redes ad-hoc, éstas se encuentran formadas por una serie de nodos autónomos y heterogéneos, que se encuentran comunicados mediante enlaces inalámbricos y sin una infraestructura de red fija. Los dispositivos involucrados en la red pueden presentar características muy distintas en cuanto a capacidad de computación, memoria disponible y batería se refiere, aunque, al ser en su mayoría dispositivos móviles, sus capacidades serán reducidas. La limitada funcionalidad de los dispositivos junto con la naturaleza *peer to peer* de la red conducen a una dependencia mutua entre sus integrantes, los cuales deberán cooperar para alcanzar objetivos comunes.

SCSANES en un sistema de coordinación de servicios desarrollado en base a un *middleware*, el cual permite abstraerse de las características de bajo nivel del sistema, implementado a través de una arquitectura de memoria compartida basada en el uso de espacios de tuplas. Cada agente tiene su propio espacio sobre el que puede insertar y extraer datos mediante patrones, que comparte con el resto de agentes activos en la red. De esta manera, se construye un sistema distribuido en el cual un nodo introduce información y el resto de integrantes de la red tienen acceso a ella.

El objetivo de SCSANES es mejorar las comunicaciones entre los usuarios involucrados en una situación de emergencia, conectados mediante una red de tipo ad-hoc. Para ello, SCSANES representa el estado de la catástrofe y facilita a los usuarios la interacción con el sistema y el intercambio de mensajes con otros usuarios involucrados en el desastre. Además, SCSANES permite la provisión y el acceso a servicios, incrementando así la funcionalidad del sistema.

En la sección II se muestran los resultados obtenidos del estudio previo realizado sobre las distintas tecnologías aplicables a dicho entorno. A continuación, en la sección III, se expone la arquitectura del sistema desarrollado y los distintos subsistemas que intervienen en él. Posteriormente, en la sección IV se describe el funcionamiento general del sistema, indicándose los requisitos considerados a la hora de desarrollar SCSANES y las distintas funcionalidades implementadas. Por último se exponen trabajos relacionados en la sección V y las conclusiones y propuestas futuras de trabajo en la sección VI.

II. Estudio previo

II-A. Modelos de coordinación

Una de las primeras arquitecturas de sistemas cooperantes es la arquitectura de pizarra [1], en la cual diversas fuentes de conocimiento (FC) cooperan entre sí para alcanzar un objetivo común a través de la pizarra. Las FC examinan la pizarra y cuando existen datos con los que poder trabajar, los recogen y procesan, dejando los resultados obtenidos en la pizarra, de manera que sean accesibles a otras FC.

Si se sustituye la pizarra por un espacio de tuplas, la información se almacena de forma estructurada: una tupla contiene una serie de campos diferenciados por un nombre en los que se almacenan datos. El acceso a la información se realiza mediante patrones. Un patrón se corresponde con un conjunto de restricciones que determinan un predicado asociado a un tipo de campo, así un patrón concuerda con una tupla si cada una de las restricciones definidas en el patrón encajan con uno de los campos de la tupla.

El espacio de tuplas puede verse como un canal de comunicaciones a través del cual los procesos se comunican mediante un conjunto simple y reducido de operaciones de

lectura y escritura. Para controlar dichas operaciones se emplea un modelo de coordinación. Linda [2] es históricamente el primer modelo de coordinación, con su mismo enfoque han surgido otros, algunos de ellos orientados a su uso en redes ad-hoc, al permitir una comunicación desacoplada en tiempo y espacio.

LIME (Linda in a Mobile Enviroment) [3] [4] adapta Linda a un entorno móvil y provee una capa de coordinación que permite el desarrollo de aplicaciones que requieren movilidad física, lógica o ambas, al eliminar el espacio de tuplas único y persistente empleado en Linda. Las entidades fundamentales en LIME son agentes, *hosts* (contenedores de agentes, a los que proporcionan conectividad y soporte de ejecución; los cuales también pueden ser móviles) y el espacio de tuplas (medio de coordinación entre agentes). En [5] se analizan los problemas que aparecen a la hora de implementar un sistema basado en LIME como son dificultades a la hora de escalar, problemas derivados de la atomicidad de las operaciones y un nivel de seguridad mínimo, impidiendo desarrollar una aplicación eficiente y segura. Esto ha dado lugar al desarrollo de nuevos modelos que intentan mejorarlo.

CoreLIME [5] mantiene la sintaxis y la semántica de la mayoría de las operaciones de LIME, aunque con algunas diferencias que intentan solventar los problemas mencionados anteriormente. En primer lugar, reduce el ámbito de las operaciones a nivel de *host*, lo que provoca que desaparezca el concepto de espacio de tuplas compartido entre agentes pertenecientes a diversos *hosts*. De esta manera, los agentes únicamente pueden acceder a los espacios de tuplas de otros agentes situados en el mismo *host* y si desean comunicarse con un agente ubicado en otro, deberán migrar a él. Por otro lado, se introduce seguridad a través del uso de *capabilities* o capacidades, las cuales regulan las operaciones sobre los espacios de tuplas. MARS [6] es otro modelo de coordinación en el cual cada nodo mantiene un espacio de tuplas local. MARS se adapta a entornos móviles permitiendo a los agentes capturar las conexiones como eventos, que indican la presencia de un nodo remoto al cual pueden migrar. El diseño de MARS, al igual que el CoreLIME, es ineficiente debido a que requiere una migración de un agente por cada operación, lo cual es bastante más costoso que un simple intercambio de mensajes.

Limone [7] presenta una perspectiva para la coordinación centrada en el agente, permitiendo que cada uno defina su propia política de conocimiento y limitando las interacciones con otros agentes en función de dicha política. Los agentes que satisfacen las condiciones impuestas son almacenados en una lista de conocidos, la cual es mantenida de forma automática por el sistema. Limone adapta las primitivas de Linda a entornos móviles eliminado el bloqueo remoto y la complejidad de las operaciones en grupo. Además, provee tiempos para todas las operaciones distribuidas y reacciones, que habilitan la comunicación asíncrona entre los agentes.

II-B. Mecanismos de descubrimiento y provisión de servicios

El modelo más extendido para la provisión de servicios es el de cliente servidor, en el cual el cliente conoce la ubicación del servidor y el protocolo necesario para poder comunicarse con él. Pero este modelo no es válido para entornos dinámicos, como el característico de las redes ad-hoc, siendo necesario emplear otras estrategias que permitan a los servidores anunciar sus servicios, y a los clientes buscarlos y acceder a ellos, como son Service Location Protocol (SLP) [8], DNS-Service Discovery (DNS-SD) [9], Salutation [10] o Jini [11]. Este último describe una arquitectura distribuida orientada a servicios, en la que se definen tres componentes: clientes o usuarios de un servicio; servicios o entidades software capaces de realizar una determinada tarea; y servicios de directorio, utilizados para el registro y la búsqueda de servicios.

SPAWN [12] es un *middleware* que adapta y extiende el modelo de Jini al proporcionar un servicio de anuncios descentralizado y un sistema de peticiones adaptado al dinamismo e impredecibilidad de las redes ad-hoc, apoyado sobre un sistema de gestión de código automático que puede obtener, usar y eliminar el código binario bajo demanda y mecanismos de mejora que extienden el ciclo de vida de los servicios. Además, presenta un ligero modelo de seguridad que asegura las interacciones.

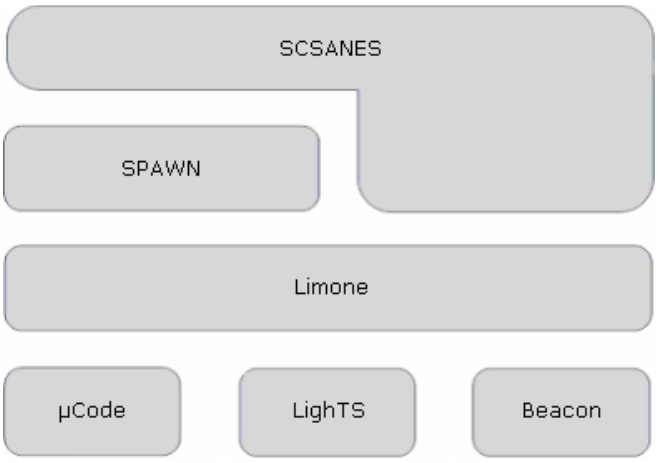

Figura 1. Arquitectura del sistema

III. SCSANES: ARQUITECTURA DEL SISTEMA

SCSANES utiliza y adapta los recursos de varios sistemas sobre los que se ubica con el objetivo de obtener un sistema de coordinación de servicios para su uso en redes ad-hoc en situaciones de catástrofes.

En la figura 1 se observa la relación entre los distintos subsistemas empleados para su implementación. μCode [13] [14] provee un conjunto reducido de primitivas que proveen movilidad de código y estado. LighTS [15] [16] es una implementación Java del concepto de espacio de tuplas propuesto en Linda. Actualmente existen varias implementaciones basadas en Java del espacio de tuplas, entre las cuales destacan TSpaces de IBM y JavaSpace de Sun, pero éstas presentan un gran número de características como persistencia, seguridad, acceso remoto o transacciones, entre otras, que complican el sistema, desperdiciando recursos y limitando con ello la capacidad de

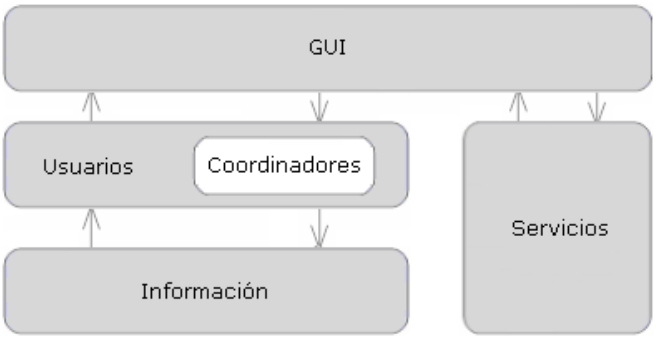

Figura 2. Arquitectura de SCSANES

extensión. Debido al entorno dinámico característico de las redes ad-hoc, en el que los agentes se conectan y desconectan sin previo aviso de manera habitual, es necesario emplear algún mecanismo para mantener actualizada la información relativa al estado del sistema. Beacon [7] es un protocolo de descubrimiento basado en *beacons* o balizas encargado de llevar a cabo dicha misión. Su funcionamiento se basa en que cada *host* emite de forma periódica un *beacon* con los perfiles de todos los agentes activos contenidos en dicho *host*. Un perfil es una colección de tripletas compuestas por el nombre de la propiedad, el tipo y el valor asociados. Limone [17] permite establecer una comunicación entre los agentes y SPAWN [18] implementa un sistema de provisión de servicios.

La figura 2 muestra la arquitectura interna de SCSANES, la cual se encuentra dividida en varios módulos. El módulo de información es el encargado de gestionar la información asociada a la catástrofe e intercambiada entre los usuarios, para ello se apoya sobre los recursos proporcionados por el subsistema Limone. El módulo de servicios administra el descubrimiento y la provisión de los servicios, éste se encuentra situado sobre SPAWN. En el módulo usuarios se ubica el agente del sistema Limone asociado a cada usuario, responsable de administrar la información. Los coordinadores son un tipo especial de agentes de usuario, con funcionalidad extra. En el nivel superior se encuentra el módulo encargado de la interfaz de usuario, responsable de la representación y la recopilación de información de los usuarios. Una explicación más en profundidad de la funcionalidad desarrollada en cada módulo se realiza en la sección IV.

IV. SCSANES: Descripción del sistema

El objetivo de SCSANES es obtener un sistema de coordinación de servicios capaz de adaptarse a un entorno dinámico, compuesto por un conjunto de dispositivos con diferentes capacidades, que favorezca la comunicación entre los integrantes de la red y el uso de servicios.

El escenario de aplicación del sistema son situaciones de catástrofes en las que se requiere la colaboración entre las distintas personas involucradas en el suceso y no existe a priori una infraestructura fija de comunicaciones, lo que lleva a la utilización de redes ad-hoc. Éstas se caracterizan por estar formadas por un conjunto de nodos autónomos y en su mayoría móviles, que se encuentran comunicados entre sí mediante enlaces inalámbricos, la topología la red es dinámica y la administración se lleva a cabo de manera descentralizada.

En el diseño de SCSANES se han considerado en primer lugar las restricciones impuestas por las características inherentes a las redes de tipo ad-hoc, como son: su estructura dinámica; el uso de enlaces inalámbricos, en los cuales el ancho de banda se encuentra limitado; y las restricciones de los dispositivos involucrados, en los cuales existen limitaciones de energía y de capacidad de procesamiento. Pero además, se ha tenido en cuenta que su uso se encuentra destinado a situaciones de emergencia, en las cuales los usuarios serán bomberos, policías o médicos, entre otros, siendo la información intercambiada entre ellos de suma importancia.

Teniendo en cuenta las consideraciones realizadas y las recomendaciones expuestas en [19], los requisitos que debe cumplir SCSANES son:

- Usabilidad, presentando una interfaz sencilla, que pueda ser utilizada casi intuitivamente, es decir, sin ningún tipo de conocimiento previo en cuanto a tecnología se refiere.
- Comunicación efectiva, informando de forma clara del estado del desastre y de las acciones apropiadas a realizar en cada caso.
- Flexibilidad, pudiendo adaptarse a distintos tipos de desastres, en los que varíen el número de personas involucradas, el número de recursos disponibles o el tipo de servicios requeridos, entre otros.
- Interoperabilidad o habilidad para trabajar con diversas tecnologías, admitiendo la compatibilidad entre diferentes dispositivos o aplicaciones, que en determinados casos podrán demandar requisitos especiales.
- Redundancia, como seguridad para poder manejar las situaciones de emergencia de manera rápida y eficiente. La redundancia implica coste, por tanto es necesario determinar un equilibrio, el cual dependerá del escenario considerado, teniendo en cuenta la posibilidad de riesgo y fallos de cada caso.
- Interdependencia. Se deben minimizar las dependencias entre dispositivos, cualquiera puede fallar y el resto deben seguir funcionando.
- Calidad de servicio. La carga de la red puede variar rápidamente, por lo que es necesario incluir servicios de apoyo para asegurar la transmisión de datos prioritarios, se debe asegurar que los datos se transmiten a tiempo, intentando evitar la sobrecarga de la red.
- Privacidad, se debe compartir la información necesaria con las personas adecuadas en el momento propicio.

IV-A. Descripción general de la interfaz de usuario

La interfaz de usuario es la herramienta que permite al usuario interaccionar con el sistema. En este caso se busca establecer una comunicación sencilla, cómoda y efectiva, adecuada al usuario y al entorno en el que se encuentra.

En la pantalla principal, mostrada en la figura 3, se observan todos los elementos que intervienen en la catástrofe, cada uno

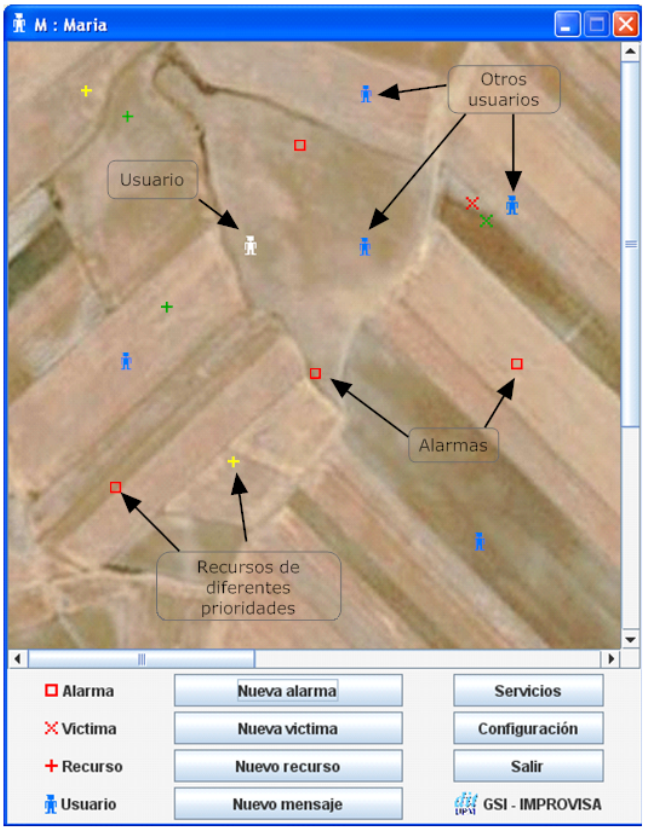

Figura 3. Pantalla principal (PC)

de ellos se representa con una imagen diferente dependiendo de:

- Su tipo: alarma, víctima o recurso; lo que determinará su forma.
- Su estado, lo que definirá su color.

De esta manera, el usuario puede de un simple vistazo conocer el estado de la situación en la que se encuentra involucrado. Además, es posible interactuar con dichos elementos, ya que puede conocer toda la información asociada al seleccionarlos, y modificarla o eliminarla en caso necesario.

En dicha pantalla también aparecen los usuarios, con un icono característico. En este caso el color distingue al propio usuario del resto de usuarios activos. Al seleccionar cualquiera de ellos, aparecerá una nueva ventana con su información y la opción de enviar un mensaje directo. Cuando se recibe un mensaje, se modifica el icono del usuario origen para avisar del nuevo evento.

Toda esta información se representa sobre una imagen, que se corresponde con la escena del lugar de la catástrofe. Esta imagen, proporcionada por un servicio de mapas, va asociada con unas coordenadas geométricas que permiten posicionar a cada uno de los integrantes del sistema. Sin embargo, la calibración del mapa resulta más sencilla si se utilizan otro tipo de coordenadas, conocidas como UTM (Universal Transversor Mercator), las cuales entienden el mundo como

Figura 4. Pantalla principal (PDA)

un plano en dos dimensiones. La conversión entre coordenadas geométricas y UTM se lleva a cabo aplicando el modelo WGS84 [20].

En la parte inferior de la pantalla principal, existe un panel con una serie de botones que permiten:

- Crear nuevos elementos, ya sean alarmas, víctimas, recursos o mensajes globales. Al activarlo, aparece una nueva ventana en la que se insertan los datos asociados a dichos elementos.
- Modificar la configuración del usuario.
- Visualizar los servicios disponibles.
- Salir del sistema.

En el diseño de todas las ventanas se ha mantenido la misma estructura para facilitar el uso de la aplicación, no existe la opción de tener más de una ventana activa cada vez, pero siempre existe la posibilidad de volver hacia atrás.

Además del diseño de esta interfaz, orientada para su uso en PCs, se ha realizado otra adaptada a las pantallas de las PDAs, en la que se han eliminado los gráficos, al consumir muchos recursos del sistema. El resultado puede observarse en la figura 4. En este caso, la pantalla principal muestra únicamente un panel con una serie de botones que permiten interactuar con el sistema. Estos botones se corresponden con los que aparecían en el panel inferior de la aplicación para PC, con la diferencia de que muestran el estado del sistema mediante una lista de elementos. Por ejemplo, al pulsar sobre el botón de alarmas, aparece la opción de crear una nueva alarma y una lista con todas las alarmas activas. Si se pulsa sobre alguna de ellas se visualizará su información asociada, pudiendo modificar o eliminar dicha alarma.

El objetivo ha sido mantener la funcionalidad del sistema desarrollado en una interfaz más sencilla, pero muy similar a

la realizada para PC, con el fin de evitar que el usuario tenga que adaptarse al nuevo entorno.

IV-B. Gestión de usuarios

Al inicio, el usuario debe indicar su configuración mediante una serie de parámetros: nombre, descripción, ubicación, y dos datos clave: el tipo de información que desea visualizar y su papel en la coordinación del sistema, los cuales determinarán su interacción con el sistema.

Cada nuevo usuario implica la aparición de un nuevo agente Limone, encargado de gestionar la comunicación con el resto de usuarios mediante el espacio de tuplas compartido. El agente almacena en su perfil los datos del usuario, dicha información es enviada periódicamente a todos los agentes activos, de manera que cuando un agente detecta un nuevo perfil, que se ha modificado uno ya existente o que no ha recibido alguno pasado un cierto tiempo establecido, deberá realizar las acciones pertinentes para mantener actualizada la información asociada a su entorno. Este mecanismo de actuación se encuentra basado en el subsistema Beacon.

Asociado a cada agente existen:

- Un espacio de tuplas, que actúa como un contenedor de datos y un canal de comunicaciones.
- Una lista de conocidos, en la que se almacenan los perfiles del resto de agentes con los que se desea comunicar. Esta lista permite obtener objetos *proxy* de agentes remotos, usados para la interacción entre agentes.
- Una lista de reacciones, en la que se guardan los patrones reactivos a aplicar al espacio de tuplas local.
- Un registro de reacciones, en el cual se almacenan las reacciones creadas y registradas por el agente.

Una reacción está compuesta por el patrón reactivo, el cual indica dónde debe propagarse la reacción y qué tuplas son sensibles a dicha reacción; y una función de llamada, la cual determina el código que debe ejecutarse cuando se encuentra una tupla sensible a dicha reacción.

El usuario podrá modificar su configuración cuando lo desee y visualizar los datos del resto de usuarios activos.

IV-C. Coordinación

Para un mejor funcionamiento del sistema se han incluido unos agentes especiales, denominados coordinadores, los cuales, además de realizar las acciones propias de cualquier otro agente, son los encargados de almacenar la información y avisar al resto de agentes interesados (éstos reaccionarán a tal evento realizando las acciones oportunas).

Durante su configuración, un usuario puede decidir ser coordinador, no serlo o que su estado sea variable, es decir, que dependa del resto de dispositivos que se encuentren activos en un determinado instante. Para implementar este último caso es necesario monitorizar el estado del resto de usuarios. Esta operación es llevada a cabo por los coordinadores cada vez que detectan un nuevo agente o que uno de los anteriores (incluido él mismo) ha cambiado, momento en el cual entra en marcha un algoritmo que determina si el estado de coordinación debe

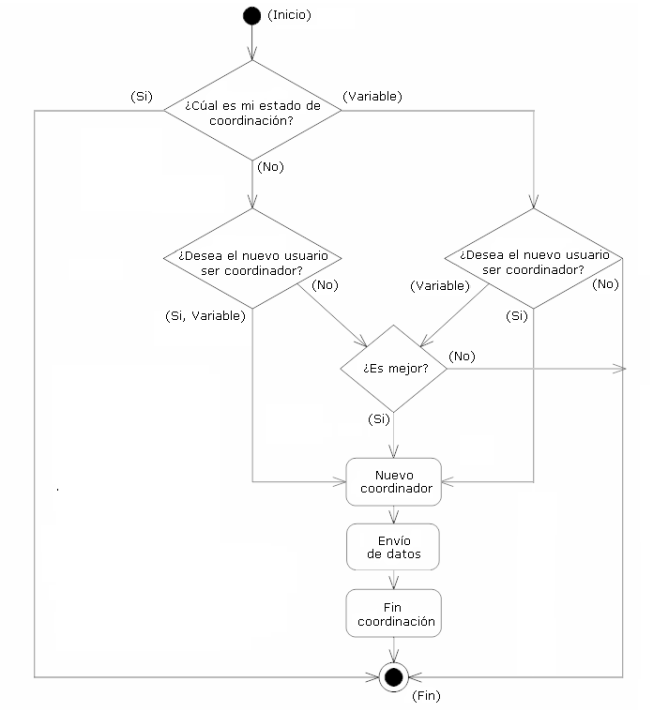

Figura 5. Algoritmo de coordinación

cambiar. Su funcionamiento, mostrado en la figura 5, se detalla a continuación:

- Si el usuario asociado al agente coordinador actual no desea ser coordinador, pero ha tenido que asumir el papel de coordinación, cuando se conecte un nuevo usuario que desee ser coordinador, no le importe serlo o tampoco quiera serlo, pero sus características sean mejores, el coordinador actual lo proclamará nuevo coordinador, le enviará la información que había almacenado hasta el momento y él dejará de ejercer dicha función.
- Si al usuario asociado al agente coordinador actual no le importa ser coordinador y aparece un nuevo usuario al que tampoco le importe serlo, pero presente mejores prestaciones, éste se convertirá en el nuevo coordinador, recibiendo la información almacenada hasta el momento.
- Si el usuario desea ser coordinador, esta condición se mantendrá hasta su salida del sistema.

Además, es necesario considerar dos momentos especiales: el inicio y la salida.

- Al principio, puede darse el caso de que no exista ningún usuario que desee ser coordinador. El primero que intente introducir información en el sistema se dará cuenta de este hecho y pondrá en funcionamiento un algoritmo de búsqueda del mejor agente disponible, al que nombrará coordinador. Esta situación también puede darse si el agente que había actuado como coordinador hasta el momento se ha desconectado de la red sin previo aviso.
- Antes de desconectarse, si un agente actúa como coordinador debe comprobar si existen más coordinadores

activos, en caso afirmativo abandonará el sistema sin más, de lo contrario, deberá buscar un nuevo coordinador (aquel que presente mejores prestaciones) y enviarle la información almacenada. En cualquier caso, antes de abandonar, registrará en un archivo de texto toda la información almacenada en su espacio de tuplas.

El objetivo es asegurar que siempre que se necesite haya al menos un coordinador activo en la red, aunque puede haber más de uno si así se desea.

La forma en la que se determina si un agente es mejor que otro es mediante el tipo de información que gestiona, definida por el usuario durante la configuración. Se supone que cuanta más información maneje, mejor será el dispositivo sobre el que trabaja, pero si esta suposición es errónea, el usuario puede negarse u ofrecerse a ser coordinador.

IV-D. Gestión de información

La información es intercambiada a través del espacio de tuplas. Existen tres posibles tipos de información a almacenar en dicho espacio:

- Los elementos involucrados en la catástrofe: alarmas, víctimas y recursos, representados a través del identificador del agente origen de la información, el elemento en sí mismo y la prioridad asociada a dicha información.
- Los mensajes globales, en los que se almacena el contenido del mensaje y la prioridad asociada a este.
- Los mensajes personales, que contienen únicamente el identificador del agente origen y el propio texto del mensaje.

Estos tres patrones de información dan lugar a dos patrones reactivos diferentes: en los dos primeros casos, la información se almacena en los coordinadores, los cuales serán los encargados de avisar al resto de agentes interesados de la nueva información, es decir, aquellos asociados a los usuarios que hayan decidido visualizar la información cuya prioridad es inferior o igual a la de la información contenida; mientras que en el último caso la información sólo se almacena en el agente destino, avisándose únicamente a sí mismo de este hecho.

Existen tres opciones a la hora de seleccionar el tipo de información a visualizar: mínima, media o máxima. La elección de una u otra estará influenciada por varios factores. Principalmente, el rol del usuario en el sistema. Es probable que el jefe de los bomberos desee conocer el estado completo del sistema, es decir, desee visualizar toda la información presente en este. Mientras que a lo mejor un médico sólo desea conocer la información más prioritaria. Aunque también son importantes las capacidades del dispositivo empleado, si un usuario posee un dispositivo muy potente puede que desee visualizar toda la información, aunque no lo requiera estrictamente, mientras que si el usuario dispone de un dispositivo de capacidades limitadas, se centrará únicamente en la información más prioritaria.

Cuando un usuario desea actualizar un elemento que existía con anterioridad, el sistema únicamente deshabilita la información con una prioridad igual o inferior a la enviada, de esta forma se consigue mostrar la información más actualizada, adecuada a las características de cada usuario. Al eliminar un elemento, se deshabilitan todos los datos asociados.

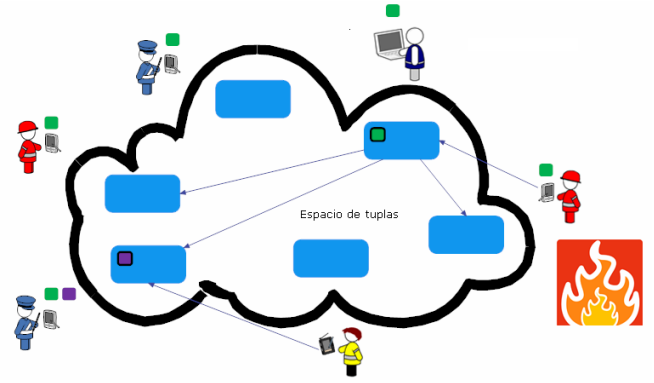

Figura 6. Funcionamiento del sistema

En el diagrama representado en la figura 6 puede observarse el estado del sistema en un determinado instante. Observamos como uno de los bomberos ha descubierto un nuevo fuego y desea informar al resto de agentes de ello mediante una alarma, para lo cual introduce la nueva información e indica su prioridad asociada, como en este caso el fuego es pequeño, considera que está controlado y le asocia una prioridad baja, esta información se representa en el diagrama mediante un cuadrado verde. Éste es almacenado en el coordinador, el cual es el encargado de avisar a los usuarios interesados, en este caso los bomberos y un policía, los cuales reciben la nueva información y la representan en la pantalla. El único caso en el que la información no se almacena en el coordinador es cuando se envía un mensaje personal, ya que éste se guarda directamente en el destino. Observamos esta posibilidad en la parte inferior del diagrama, representado mediante un cuadrado morado, ya que el médico decide enviar un mensaje al policía.

IV-E. Descubrimiento y provisión de servicios

La implementación de la provisión y el uso de servicios se realiza sobre SPAWN, a través de un servicio de directorio distribuido basado en un espacio de tuplas, en el cual el agente que desea ofertar un servicio indica su propuesta mediante una tupla en la que indica el nombre del servicio y la interfaz y el objeto *proxy* asociados. Los clientes buscarán en dicho directorio servicios que presenten una determinada interfaz y opcionalmente un nombre. En función de dicha elección se creará un patrón empleado en una operación de lectura que devolverá la interfaz del objeto buscado, el cual se comunicará con la tarea asociada en el servidor.

El proceso descrito requiere que el código del *proxy* se encuentre disponible en el *host* del cliente; para ello se incluye un sistema de gestión automática de código, de forma que se asegure que el código requerido es encontrado en un proveedor de servicios y puesto a disposición del cliente cuando lo necesite.

Se ha implementado un servicio de mapas que permite al usuario visualizar las imágenes proporcionadas por el servidor y almacenarlos, para poder usarlas posteriormente y representar el lugar de la catástrofe.

V. Trabajos relacionados

Los sistemas basados en *middlewares* de espacios de tuplas se utilizan en diferentes áreas, como computación distribuida [21], web semántica [22] o computación móvil [23].

Con este enfoque, se han desarrollado varios sistemas para permitir la comunicación en situaciones de emergencia. Siren [24] es un sistema destinado a la comunicación entre bomberos en situaciones de catástrofes. Se caracteriza por presentar múltiples niveles de redundancia para asegurar la transmisión de información. Otro ejemplo es MIDAS Data Space (MDS) [25], el cual simula un espacio de datos relacional para compartir información en casos de emergencia. Una de sus principales características es que implementa replicación optimista para garantizar la disponibilidad de la información. Esto provoca un aumento de la complejidad y del coste de la gestión de la red, que se intenta resolver evitando las operaciones de actualización. Se observa como tanto Siren como MDS se basan en la replicación para asegurar la transmisión de datos. SCSANES sólo implementa redundancia en los coordinadores, favoreciendo su uso en dispositivos con recursos limitados. Por otra parte, gracias a la provisión y el acceso a los servicios, SCSANES permite ampliar la funcionalidad del sistema, adecuándolo a las necesidades requeridas en cada situación.

Otro enfoque aplicado en situaciones de emergencia es el uso de redes de sensores como CodeBlue [26] o Agilla [27]. SCSANES no utiliza sensores, si no que se basa en la información de los usuarios para representar el estado del desastre, presentando la información considerada más relevante por los usuarios en cada momento.

VI. Conclusiones

SCSANES es un sistema que permite la provisión y el uso de servicios entre diversos usuarios involucrados en una situación de emergencia a través de un espacio de tuplas compartido y distribuido, el cual permite trabajar sobre un entorno descentralizado, en el que cada nodo funciona de manera independiente, pero a la vez es capaz de coordinarse con el resto de nodos activos para ampliar sus funcionalidades y por tanto las del sistema en conjunto.

Algunas de las principales características de SCSANES son:

- Usabilidad, al presentar una interfaz de uso sencillo e intuitivo, adaptada a las necesidades de los usuarios, mediante la cual se muestra el estado de los distintos elementos involucrados en la catástrofe (usuarios, alarmas, víctimas y recursos) sobre el mapa en el que tiene lugar la catástrofe, permitiendo al usuario conocer y modificar sus propiedades, en caso necesario.
- Comunicación efectiva entre los usuarios involucrados en la catástrofe a través del intercambio de mensajes globales e individuales y de la información asociada a las alarmas, víctimas y recursos presentes.
- Flexibilidad, adaptándose a las características de cada situación al permitir definir los elementos involucrados y proporcionar los servicios adecuados a cada caso. Además, tiene en cuenta el entorno dinámico de las redes ad-hoc, determinando los dispositivos más adecuados para las labores de coordinación en cada caso.
- Fiabilidad, gracias a la actuación de los coordinadores, encargados de almacenar la información e informar a los agentes interesados en ella.
- Adaptabilidad. El usuario puede adaptar el sistema a sus necesidades o a las del dispositivo con el que trabaje (necesario si presenta capacidades limitadas), al poder definir el tipo de información que desea visualizar (en función de su prioridad) y su papel en las tareas de coordinación.
- Interdependencia entre los dispositivos. Si en un instante determinado el único coordinador activo sale del sistema, el sistema proclamará uno nuevo y continuará funcionando.
- Calidad de servicio. El uso de prioridades permite controlar la cantidad de información intercambiada entre los usuarios, facilitando su transmisión.
- Privacidad. Además de la información global es posible el intercambio de mensajes individuales entre usuarios.
- Extensibilidad, al admitir el anuncio y provisión de servicios, lo que permite aumentar su funcionalidad en cualquier momento.

Para el desarrollo de la versión de SCSANES para PC se ha utilizado J2SE. Mientras que para la segunda versión, adaptada a dispositivos portátiles, se ha empleado J2ME con la configuración Connected Device Configuration (CDC) y el Personal Profile (PP). La aplicación ha sido probada únicamente en un entorno de desarrollo. Concretamente, sobre un PC, con sistema operativo Windows XP, y una PDA HP iPAQ, con Windows Mobile 2003 Second Edition.

Como líneas de trabajo futuras se proponen los siguientes temas:

- Simular escenarios reales para probar el rendimiento del sistema.
- Mejorar la usabilidad: facilitando la diferenciación entre distintos tipos de usuarios, alarmas, víctimas y recursos (bomberos, policías, fuegos, inundaciones, heridos, ambulancias,...); mejorando la interfaz gráfica para las PDAs; e integrando GPS en el sistema.
- Introducir opciones de seguridad: cifrando la información y/o utilizando contraseñas para controlar la lectura y el borrado de los datos.
- Desarrollar nuevos servicios para aumentar la funcionalidad del sistema, lo que se acompaña de la necesidad de mejorar el algoritmo de búsqueda de servicios.

Agradecimientos

Este trabajo se ha desarrollado dentro del proyecto IMPROVISA (TSI2005-07384-C03), perteneciente al programa

de Tecnologías de Servicios de la sociedad de la Información (TSI) del MEC.

REFERENCIAS

[1] L. Erman, F. Hayes-Roth, V. Lesser, and R. Reddy, "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," *ACM Computing Surveys (CSUR)*, vol. 12, no. 2, pp. 213–253, 1980.

[2] D. Gelernter, "Generative communication in Linda," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 80–112, 2005.

[3] G.P. Picco, A. Murphy, and G.-C. Roman, "Lime: A Coordination Model and Middleware Supporting Mobility of Hosts and Agents," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 15, no. 3, pp. 279–328, 2006.

[4] A. L. Murphy, "LIME's web page," http://lime.sourceforge.net, 2007.

[5] B. Carbunar, M. T. Valente, and J. Viteky, "Coordination and Mobility in CoreLime," in *ConCoord: Workshop on Concurency and Coordination*, 2001.

[6] G. Cabri, L. Leonardi, and F. Zambornelli, "Mars: A programmable coordination architecture for mobile agents," in *IEEE Internet Computing*, 2000.

[7] C.-L. Fok, G.-C. Roman, and G. Hackmann, "A Lightweight Coordination Middleware for Mobile Computing," in *6th International Conference on Coordination Models and Languages (Coordination 2004)*, 2004.

[8] E. Guttman, C. Perkins, J. Veizades, and M. Day, "RFC 2608: Service Location Protocol, Version 2," 1999.

[9] S. Cheshire and M. Krochmal, "DNS-Based Service Discovery, Internet-Draft," 2006.

[10] B. Miller and R. Pascoe, "Salutation service discovery in pervasive computing environments," IBM, Tech. Rep., 2000.

[11] Sun Microsystems, "Jini Community Resources: Jini Technology Architectural Overview," Sun Microsystems, Tech. Rep., 1999.

[12] R. Handorean, G. Roman, G. Hackmann, and C. Gill, "SPAWN: Service Provision in Ad-hoc Wireless Networks," Washington University, Department of Computer Science and Engineering, Tech. Rep., 2005.

[13] G.P. Picco, "μCode: A Lightweight and Flexible Mobile Code Toolkit," in *2nd International Workshop on Mobile Agents 98*, no. 1447, 1998, pp. 160–171.

[14] ——, "μCode's web page," http://mucode.sourceforge.net, 2000.

[15] D. Balzarotti, P. Costa, and G.P. Picco, "The LighTS Tuple Space Framework and Its Customization for Context-Aware Applications," *Journal of Web Intelligence and Agent Systems*, vol. 5, no. 2, 2007.

[16] G.P. Picco, "LighTS's web page," http://lights.sourceforge.net, 2001.

[17] C.-L. Fok, G.-C. Roman, and G. Hackmann, "Limone's web page," http://www.cs.wustl.edu/mobilab/projects/limone, 2004.

[18] R. Handorean, G. Roman, G. Hackmann, and C. Gill, "SPAWN's web page," http://www.cs.wustl.edu/mobilab/Projects/SPAWN.

[19] R. Dilmaghani and R. Rao, "On Designing Communication Networks for Emergency Situations," in *International Symposium on Technology and Society (ISTAS)*, 2006.

[20] B. Hoffmann-Wellenhof, H. Lichtenegger, and J. Collins, *GPS: Theory and Practice, 3rd ed*. New York: Springer-Verlag Wien, 1994.

[21] K. Hawick, H. James, and L. Pritchard, "Tuple-Space Based Middleware for Distributed Computing," Computer Science Division, School of Informatics University of Wales, Tech. Rep., 2002.

[22] R. Tolksdorf, E. P. Bontas, and L. J. B. Nixon, "Towards a tuplespace-based middleware for the Semantic Web," in *International Conference on Web Intelligence (WI'05)*, 2005.

[23] C. Mascolo, L. Capra, and W. Emmerich, *Mobile Computing Middleware*. Lecture Notes in Computer Science, Springer Berlin, Heidelberg, 2002.

[24] X. Jiang, N. Chen, J. Hong, K. Wang, L. Takayama, and J. Landay, "Siren: Context-aware Computing for Firefighting," in *2nd International Conference on Pervasive Computing (Pervasive 2004)*, 2004.

[25] J. Gorman, "The MIDAS Project: Interworking and Data Sharing," in *8th International Symposium on Interworking Santiago (Chile)*, 2007.

[26] K. Lorincz, D. J. Malan, T. R. Fulford-Jones, A.Ñawoj, A. Clavel, V. Shnayder, G. Mainland, and M. Welsh, "Sensor Networks for Emergency Response: Challenges and Opportunities," *IEEE Pervasive Computing, Special Issue on Pervasive Computing for First Response*, 2004.

[27] C.-L. Fok, G.-C. Roman, , and C. Lu, "Mobile Agent Middleware for Sensor Networks: An Application Case Study," Departament of Computer Science and Engineering, Washington University in Saint Louis, Tech. Rep., 2004.