Integración de WSN en entornos inteligentes

Grupo de Investigación ARCO

Escuela Superior de Informática
Universidad de Castilla-La Mancha
Paseo de la Universidad 4. Ciudad Real
Juancarlos.lopez@uclm.es

Abstract. Las redes de sensores inalámbricas están destinadas a jugar un rol crucial en numerosos ámbitos de aplicación. La evolución de plataformas, middlewares y protocolos ha sido importante en los últimos años tanto a nivel industrial como de investigación. En este trabajo vamos a analizar las distintas plataformas existentes para redes de sensores inalámbricas desde el punto de vista de la integración con el resto de la infraestructura en entornos inteligentes. Si abordar los problemas que inherentemente tienen este tipo de redes es un aspecto importante, no lo es menos el estudiar cómo integraremos dichas redes con aplicaciones de alto nivel que las usarán para llevar a cabo su cometido.

Introducción

Las redes de sensores inalámbricas o WSN (Wireless Sensor Network) son conjuntos de dispositivos de bajo coste con una interfaz inalámbrica y que nos permiten mediante elementos sensores recolectar todo tipo de información del mundo físico. Podemos incluir en este tipo de redes nodos que son a su vez actuadores y que permiten controlar todo tipo de dispositivos utilizando el mismo medio de comunicación inalámbrico. Este último tipo de redes ha acuñado el término WSAN's (Wireless Sensor Actuator Networks).

La facilidad y flexibilidad en el despliegue es uno de los principales puntos fuertes que hacen de este tipo de redes un candidato idóneo para un gran número de aplicaciones. Monitorización de entornos, control de todo tipo de dispositivos y recolección de datos en bruto son las principales aplicaciones de las WSN y WSAN.

Las WSN presentan nuevos retos dadas las características de los dispositivos que forman parte de la misma, entre los más importantes podemos destacar:

- La energía pasa a ser un punto crucial ya que, a menudo, los dispositivos sensores están alimentados por baterías. Todos los protocolos y algoritmos destinados a estos dispositivos alimentados por baterías deben tener en cuenta el consumo que supone la transmisión de datos inalámbricos (que constituye el mayor gasto en cuanto a energía dentro de este tipo de redes).

- El consumo de recursos es otro de los problemas asociados sobre todo cuando hablamos de dispositivos destinados a ser desplegados de forma masiva (monitorización de grandes espacios, control y sensorización de entornos inteligentes, etc.). Cuantos mas recursos se requieran (CPU, RAM, flash, etc.) mas caro resulta el coste por nodo y la competitividad y viabilidad de las soluciones basadas en WSN se ven mermados.

- El modelo de enrutamiento entre nodos es un aspecto importante al tener que establecer, en la mayoría de los casos, redes inalámbricas multisalto en la cual cada uno de los nodos debe realizar las funciones de enrutado de aquellos datos provenientes de otros nodos.

- La integración con el resto de la arquitectura. Generalmente y como ya hemos comentado la WSN no es sino un interfaz con el mundo físico que sirve para observar el mismo y controlarlo por parte de aplicaciones mas avanzadas. El cómo esas aplicaciones obtienen los datos o como proporcionan las ordenes en el caso de los actuadores es un problema que atañe a aspectos tan diversos como la seguridad, tolerancia a fallos, fiabilidad, etc.

Este tipo de problemas han sido abordados de forma amplia por la comunidad científica y ya existen soluciones que se han aplicado con éxito en diversas áreas de aplicación.

Un área donde la implantación de las WSN están destinadas a jugar un papel crucial es la de los entornos inteligentes. Efectivamente podemos considerar las WSN como un paso real en la dirección que Mark Weiser marcó con su artículo “The Computer for the 21st Century”. En dicho trabajo se acuñó el término computación ubicua que defínía entornos donde la capacidad de cómputo se trasladaba de unos pocos computadores ordinarios (PC de sobremesa, portátiles, etc.) a muchos dispositivos desplegados por el entorno físico con los cuales los usuarios interaccionaban de forma inconsciente y natural.

Podemos considerar los denominados entornos inteligentes como entornos ubicuos que son capaces de ser sensibles y responder de forma apropiada a la presencia de usuarios en ellos. Esas respuestas apropiadas se deben traducir en acciones concretas y por lo tanto, también exigen tener la capacidad de controlar el entorno donde están presentes los usuarios.

Es precisamente la enorme capacidad de sensorización y control sobre el mundo físico que se requiere para llevar a cabo esta definición de entornos inteligentes la que hace de las redes de sensores/actuadores inalámbricos una tecnología crucial para este tipo de entornos.

Dispositivos para WSN

En la actualidad existen numerosas plataformas para redes de sensores inalámbricas. La mayor parte de los dispositivos como veremos están compuestos de componentes de bajo consumo y escasa capacidad. No obstante algunos de los dispositivos que se han creado para este tipo de redes tienen unas prestaciones considerables que, a veces, exceden las características que generalmente se han considerado para este tipo de dispositivos.

Nombre	**uPart**	**Imote2**	**MICAz**	**SmartDust**	**Sun Spot**	**Multimedia mote [1]**
CPU	PIC12F675	Marvell PXA271	ATmega128L	Diseño específico	ARM920t	ARM7 32 bits
Memoria	1k words SRAM 64 bytes EEPROM 128 bytes	SRAM 256KB 32 MB Flash 32 MB SDRAM	FLASH 128K SRAM 4K		512 RAM 4M ROM 4M Flash	8-64 KB RAM 32 – 256 KB Flash
Interfaz inalámbrica	868-914 Mhz 433, 310/315 Mhz	802.15.4 2.4 GHz	802.15.4 2.4 GHz	Comunicación óptica	802.15.4	802.15.4 2.4 GHz
Rango	1-30 m indoor ~100m outdoor	~30m	~30m	~180m	~30m	~30m

Table 1. Ejemplos de plataformas para redes de sensores inalámbricas

En la tabla 1 podemos ver algunos ejemplos de dispositivos que se utilizan en la actualidad para formar WSN. Para evitar problemas en el despliegue generalmente se usan frecuencias libres que no requieren de licencias. Algunos estándares como Zigbee (802.15.4) o Wibree están teniendo éxito como interfaz inalámbrica aunque también existen soluciones particulares.

La capacidad tecnológica de la industria genera constantemente nuevas posibilidades en cuanto a miniaturización de las plataformas sensoras y aumento de las prestaciones de estos dispositivos. En este sentido, la plataforma *SmartDust* es el ejemplo no comercial mas avanzado en cuanto a miniaturización de este tipo de

dispositivos y en el futuro ya se apuntan nuevos paradigmas de computación como la computación pintable [22] , esto es, dispositivos menores de 2mm cuadrados con un área de conexión inalámbrica de entorno a 2 cm y que se extienden como la pintura.

Middlewares para WSN

Desde el punto de vista del modelo de programación, existen varios modelos que se están aplicando en este tipo de redes, los cuales los podemos resumir en:

- Base de datos distribuida: Con este modelo básicamente cada nodo sensor constituye una tupla dentro de una tabla. Desde el punto de vista de las aplicaciones, acceder a la información contenida en la red es como acceder a una base de datos. Ejemplos de este tipo de modelo son TinyLIME[13] , TinyDB [19] y Cougar [19] (los tres basados en el sistema operativo TinyOS, un sistema operativo que constituye un estándar de facto en las WSN). Generalmente cada uno de los nodos de la red debe tener la capacidad para recibir y procesar preguntas (similar a SQL) y los datos son almacenados en el nodo que realiza las funciones de pasarela.

- Maquinas virtuales: Las aplicaciones se distribuyen sobre máquinas virtuales que ejecutan el código. La principal ventaja de este tipo de modelo es la distribución de código sobre la máquina virtual que representa un modelo de abstracción sobre los diversos dispositivos existentes. En función de la máquina virtual (por ejemplo la JVM de los dispositivos *SunSpot*) ésta puede ser una sobrecarga importante para los nodos que forman parte de la WSN. El propio TinyOS tiene un mecanismo para distribuir aplicaciones que son guardadas de forma temporal en la EEPROM del dispositivo para, una vez terminado el proceso de transmisión, ejecutar dicha aplicación de forma dinámica.

- Basados en agentes: En este modelo pequeñas porciones de código que constituyen los agentes migran desde unos nodos a otros ejecutándose y realizando tareas de cooperación, en ocasiones se pueden ver como una evolución de la aproximación de máquinas virtuales. SensorWare[6] , e Impala [23] son ejemplos de este tipo de aproximación. En estos middlewares a menudo se pone especial atención en la transmisión y actualización eficiente de los agentes dejando a un lado los problemas de *networking* propiamente dichos.

- Basados en eventos: el mayor exponente de este tipo de modelo es DSWare [11] y TinyDiffussion[12] los cuales notifican de forma asíncrona cualquier cambio que de la variable monitorizada a todas las partes interesadas.

- Orientadas a objetos: En este modelo cada dispositivo de la red alberga uno o varios objetos distribuidos que se invocan de forma similar a como se invocan operaciones sobre objetos locales en el paradigma de programación orientada a objetos. En la arquitectura DUO desarrollada por el grupo Arco de la UCLM para entornos inteligentes la parte de la integración de redes WSN sigue este

paradigma. La forma en la cual se ha conseguido integrar las redes de sensores mediante estándares de objetos distribuidos (tipo CORBA) es presentada en profundidad en [14] .

Cada uno de estos modelos hereda las características de su modelo de programación y por lo tanto nos proporciona ventajas y desventajas, generalmente ligadas a la aplicación específica.

Mientras que el acceso a una base de datos es una forma cómoda y bien controlada por los desarrolladores para acceder a los datos de una red de sensores-actuadores, existe el problema de la localización física de esa base de datos. Generalmente reside en un dispositivo que realiza las labores de pasarela entre la tecnología empleada por la red de sensores (zigbee, wibree, etc.) y la tecnología empleada por el resto de la infraestructura de acceso a los datos (ethernet, wifi, etc.).

La base de datos representa un único punto de fallo y establecer varias pasarelas con replicas de la misma supone problemas de consistencia y fiabilidad en el acceso a los datos como veremos mas adelante.

Aquellos basados en agentes tienen el principal problema en los recursos necesarios para soportar el intérprete de los agentes así como, en función del tamaño de dichos agentes, el gasto de energía que supone enviarlos a través de los nodos mediante costosas (en términos de energía) transmisiones inalámbricas. Adicionalmente, los recursos necesarios para ejecutar este tipo de soluciones limitan el tipo de dispositivos en los cuales se pueden implantar este tipo de soluciones a aquellos mas potentes, y por lo tanto, mas caros.

Los middlewares basados en eventos aportan una buena solución en aquellas aplicaciones que son reactivas, es decir, que deben reaccionar ante una variación determinada de una magnitud. El problema de este tipo de soluciones es que, a menudo habilitan a los dispositivos que forman la WSN sólo para notificar información y a menudo no contemplan que esos mismos dispositivos los puedan recibir (por ejemplo si son actuadores).

Finalmente, los sistemas orientados a objetos distribuidos presentan una buena solución para aquellas aplicaciones que requieren de una interacción continua con la red. En el caso de DUO donde se utiliza un middleware orientado a objetos distribuidos estándar (parecido a CORBA) el principal reto es reducir los recursos necesarios para albergar este tipo de middleware dentro de un dispositivo con recursos limitados como son los dispositivos de bajo coste que se esperan que forme parte de una red de sensores inalámbricas.

Middlewares para entornos inteligentes

Si los middlewares destinados a WSN se han centrado generalmente en facilitar la programación de aplicaciones sobre las mismas, salvaguardando en la medida de lo posible los problemas de consumo, recursos, etc. de los nodos que forman parte de la WSN, los middlewares destinados a entornos inteligentes escasamente han tenido en cuenta estos requisitos a la hora de realizar sus planteamientos.

Los middlewares tradicionalmente asociados a entornos domóticos y sus respectivas evoluciones destinadas a entornos inteligentes se han orientado tradicionalmente a lidiar con la heterogeneidad de dispositivos, protocolos y servicios existentes en dichos entornos.

Precisamente para lidiar con esa heterogeneidad generalmente han adoptado modelos basados en la plataforma JAVA y/o la plataforma *Web Services*. Ambas aproximaciones plantean serios problemas para su implantación en redes inalámbricas de sensores debido a los recursos necesarios en el caso de la plataforma JAVA (sería necesario ejecutar la JVM en cada nodo) o los mensajes excesivamente complejos de las plataformas basadas en Web Services (básicamente archivos XML por cada invocación) que plantean problemas de consumo al ser el envío de información la mayor fuente de consumo de energía en este tipo de redes.

En el caso de la *Java Virtual Machine* existen dispositivos como los SPOT de la empresa SUN que tienen una JVM limitada en su memoria flash pero que difícilmente pueden soportar los middlewares para entornos inteligentes ya que, generalmente, estas restricciones no han sido tenidas en cuenta en su diseño y contemplan el uso de una JVM completa.

Por ejemplo en el proyecto OZONE [8] se establecen tres capas (denominadas plataforma de dispositivo, middleware y servicios) donde en la plataforma de dispositivo se proporcionan interfaces heterogéneas para dispositivos de toda índole, el middleware propiamente dicho se basa principalmente en WSAMI [7] , una arquitectura *open source* de servicios web basada en el lenguaje JAVA desarrollada para el proyecto OZONE. Por último los servicios se agrupan en la última capa.

Desde el punto de vista de la integración de WSN, la capa que mas nos interesa es la del middleware así como la plataforma de dispositivo. El uso de servicios web basados en Java para el middleware contrasta con el hecho de que intenten integrar dispositivos con bajas prestaciones como pueden ser sensores y actuadores, dispositivos de electrónica de consumo, móviles, etc. que generalmente pueden presentar serios problemas para albergar los requerimientos de una plataforma como WSAMI. El uso de WSAMI requiere por su concepción de servidor web, máquina virtual de java completa, librerías XML etc. alojados en todos los dispositivos del entorno.

Precisamente la falta de soporte para estos últimos dispositivos hizo descartar WSAMI para el proyecto IST Amigo [9] . El proyecto IST Amigo supedita su

arquitectura a la plataforma OSGi [10] principalmente y por eso sus componentes mínimos son los denominados *bundles*, estos *bundles* son la unidad mínima que se emplea en OSGi que consta básicamente de un archivo JAR con un denominado manifiesto que contiene información acerca de los recursos necesarios para su desempeño. Los principales servicios de la plataforma OSGI están relacionados con el registro, despliegue, gestión y actualización de los servicios como pueden ser el servicio de *log*, el servicio de registro, administración de permisos, etc.
La arquitectura OSGi se desarrolló con la arquitectura de pasarela residencial en mente, esto es, un único dispositivo que hace de nexo de todas las redes existentes en el entorno y que lo conecta con el exterior, y por lo tanto sigue un modelo centralizado. La arquitectura OSGI ha tenido gran aceptación como plataforma de soporte a servicios, paradójicamente encontrando en el entorno de desarrollo Eclipse, una aplicación que en nada tiene que ver con entornos inteligentes, uno de sus máximos exponentes de este éxito.

En la mayoría de los casos, para la integración de WSN se ha optado por separar lo que es la aplicación que va a utilizar los datos proporcionados por la WSN de la parte del software que se ejecuta en la misma WSN. Cuando la aplicación simplemente recoge datos de la WSN y esta no tiene actuadores, los middlewares que utilizan bases de datos para representar la WSN pueden ser una buena solución. El uso de bases de datos es bien conocido por la mayoría de los ingenieros software y por lo tanto su uso no plantea mayores problemas. No obstante para aquellas redes donde sí haya actuadores la interacción con la red WSN se vuelve algo más compleja y menos intuitiva.

Integración de redes de sensores en entornos inteligentes

Tal y como hemos visto en las secciones anteriores, esta claro que si queremos desarrollar una aplicación sobre una WSN o WSANs para un entorno inteligente, las soluciones actuales pasan por establecer un modelo como el representado en la figura 1.

Fig. 1. Integración de una Aplicación para un entorno AmI usando WSN

En el caso general, en el dispositivo de la WSN se usa uno de los middlewares específicos para WSN y se construye sobre él la parte dependiente de la aplicación relativa a la WSN. Esta aplicación en el sistema empotrado generalmente es

desarrollada por un experto en el área de desarrollo sobre dispositivos con escasos recursos. En la parte de la pasarela se debe realizar la parte de la aplicación que recopila los datos y que generalmente debe lidiar tanto con el middleware de la WSN (tinyDB, Cougar, Tinydiffusion, etc.) como con el middleware específico del entorno AmI (IST-Amigo, Ozone, etc.). En este caso, el desarrollo lo debe realizar tanto un experto en el middleware del WSN como en el middleware AmI.

Esta adaptación de middlewares generalmente es dependiente de la aplicación y por lo tanto es específica de la misma debiendo ser desarrollada para cada una de las aplicaciones. Existen plataformas para la integración de diferentes middlewares creados para diversas WSN como Sensation [21] , pero no existen ese tipo de plataformas para la integración de middlewares WSN y middlewares AmI.

Un ejemplo de las prácticas habituales a la hora de integrar las WSN con aplicaciones de mas alto nivel (en este caso *Web Services*) la podemos ver en [28] . En este trabajo se desarrolla una aplicación sobre el middleware TinyDB para, con posterioridad desarrollar un *front-end* en forma de *Web Services*. Un aspecto que si han integrado es el direccionamiento que en este trabajo se realiza mediante la utilización de IPv6.

El proyecto IST e-SENSE [27] si trabaja de forma integral las WSN mediante un middleware orientado a la publicación y suscripción de mensajes y diseña toda una pila de protocolos para las propias WSN. El objetivo final es proporcionar a los usuarios y sus dispositivos personales (teléfonos móviles, PDA's, consolas, etc.) información del contexto donde se mueven.

La función del gateway en e-Sense es la de conectar la WSN con el resto de la infraestructura. En este proyecto el resto de la infraestructura son las redes 2G y 3G de comunicaciones móviles. En esta pasarela, además de implementar la pila diseñada para WSN también incluye todo el middleware necesario para conectarse con las redes móviles tipo GSM y UMTS y futuras tecnologías con el mismo ámbito. No se aborda el tema de la existencia de varias pasarelas ni se dan pautas de diseño para las aplicaciones (salvo los ejemplos en la documentación del proyecto).

Un aspecto que consideramos crucial para el desarrollo de aplicaciones sobre WSN es la tolerancia a fallos y fiabilidad de la captura de datos provenientes de este tipo de redes. Desde el punto de vista del despliegue, estos requisitos pasan por establecer varias pasarelas con la WSN y adecuar los mecanismos para usar esta redundancia en un aumento de la fiabilidad que analizamos en el siguiente apartado.

Despliegue de redes de sensores inalámbricas.

Con independencia de la plataforma sensor y del midlleware que se empleen, el despliegue de redes de sensores inalámbricas ya sea en espacios abiertos o en entornos cerrados tienen como elemento común la necesidad de poner los datos que se recolectan de la red de sensores a disposición de las aplicaciones (de procesado de datos, de almacenamiento, de monitorización, control, etc.) que, generalmente,

acceden a esos datos mediante uno o varios nodos que realizan las funciones de pasarela.

Estas pasarelas tienen un interfaz con la tecnología inalámbrica que utiliza la red de sensores (e.j 802.15.4) y otra interfaz mas apropiada con el resto de la infraestructura (e.j Ethernet cableada, Wifi, GPRS, etc.).

Como ya hemos comentado, una vez desplegada la WSN en el entorno se debe establecer una topología que estructure la red a la hora de la transmisión de la información desde los nodos sensores a las pasarelas. Esta estructura generalmente adopta las siguientes formas:

- Totalmente ad-hoc: En esta topología todos los nodos tienen la misma función, reenvian la información mediante un algoritmo de enrutado a las pasarelas o directamente a otros sensores/actuadores.

- En estrella: Ya sea mediante configuración manual o mediante algoritmos específicos (de forma dinámica) se establecen una serie de nodos que dan cobertura a la red y que se encargan de recolectar la información de los sensores en su rango, preprocesar información si es necesario, coordinar las transmisiones, etc. Generalmente estos nodos pueden tener unas características superiores en cuanto a prestaciones, pueden tener baterías mas potentes e incluso utilizar una tecnología inalámbrica diferente para comunicarse entre ellos y la pasarela o pasarelas.

- En árbol: Esta estructura puede ser considerada un caso especial de la configuración en estrella y esta basada en crear un árbol cuya raíz es la pasarela y a partir del cual se establecen diversas ramas o niveles (que de nuevo debe dar cobertura a toda la WSN) por las cuales se enruta la información hacia la pasarela. Tanto TinyDB como Cougar utilizan esta aproximación en forma de árbol [3] .

En cualquiera de los casos, el objetivo último como ya hemos indicado es el de enviar la información a la pasarela o pasarelas con el objetivo de ponerla a disposición de las aplicaciones de mas alto nivel.

Si observamos los trabajos existentes en la literatura, generalmente el número de pasarelas se reduce a uno con las consiguientes desventajas que esto origina (figura 2):

- Un fallo en la pasarela deja incomunicada la red sin poder acceder a los datos.

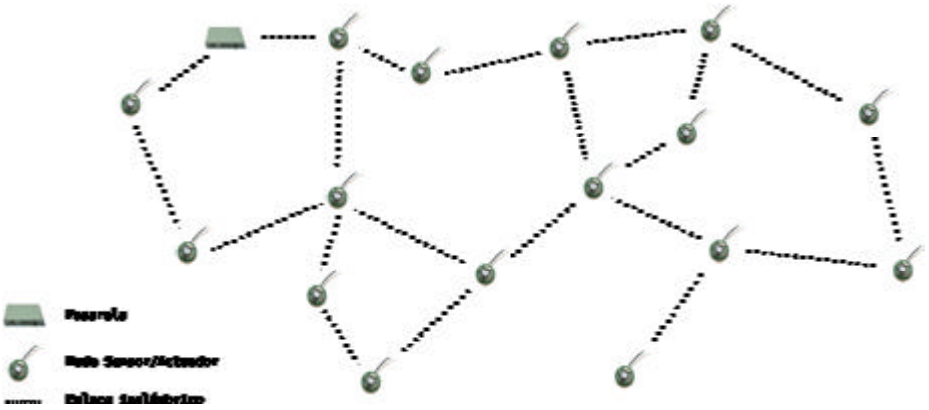

Fig. 2. Red de sensores/actuadores inalámbricos

- Los sensores o nodos cercanos geográficamente a la pasarela ven incrementada su carga de trabajo al tener que enrutar toda la información relativa a la red. En el caso de despliegues con un considerable número de nodos esta actividad puede llevar a dichos nodos a quedarse sin energía en un tiempo relativamente corto. De igual forma los problemas de comunicación en aquellos nodos cercanos a la pasarela (interferencias, problemas en el acceso al medio, etc.) se agravan considerablemente y soluciones como los algoritmos de enrutado multicamino no solucionan dicho problema al tener que converger en la pasarela.

- Obliga a tener toda la red cohesionada y que tenga una conectividad total al no ofrecer rutas alternativas.

La alternativa pasa por establecer una redundancia en las pasarelas de acceso a la red que permita aumentar la fiabilidad, balancear la carga, ser tolerantes a la creación de islas por fallo de sensores en la red, etc.

No obstante, a nivel de middleware esta redundancia en cuanto a pasarelas plantea nuevos requisitos que no siempre todos los middlewares diseñados para WSN pueden soportar. Vamos a ver estos requisitos en función del tipo de middleware en la sección siguiente.

Integracción de redes inalámbricas de sensores con multipasarela

El establecimiento de dos o mas pasarelas con una red WSN plantea diversos problemas en función del middleware que escogamos para el desarrollo de la aplicación (figura 3).

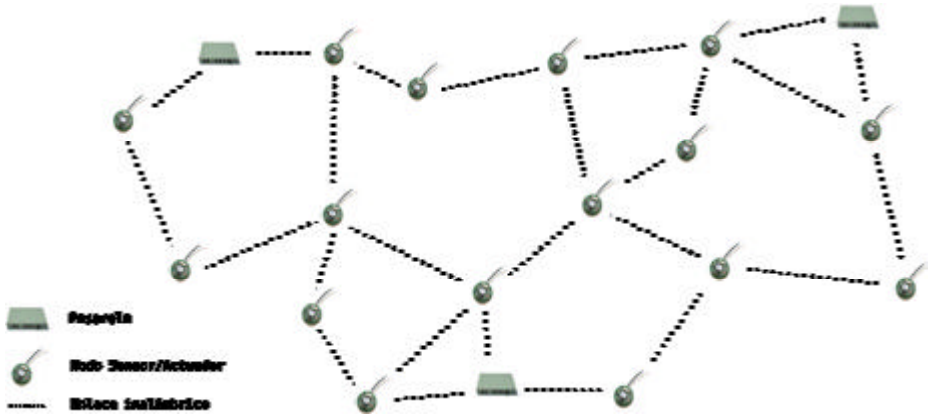

Fig. 3. Despliegue multipasarela

En aquellos middlewares basados en base de datos, el primer problema que se plantea es la consistencia de la tabla o tablas en la base de datos que representa la red. Es decir, debe existir una coherencia entre las tablas de las varias pasarelas. Generalmente las consultas a la red se realizan de forma periódica y continua desde el nodo pasarela. Los clientes acceden a la base de datos de la pasarela para recoger los datos.

Establecer varias pasarelas implica o bien establecerlas como nodos independientes con lo que se duplica el número de consultas con el consiguiente desgaste de recursos en los nodos de la red inalámbrica, o bien permitir una comunicación entre las pasarelas (a ser posible que no sea a través de la WSN) de forma que se actualicen las tablas sin que sea necesario duplicar el número de consultas a WSN. La inmensa mayoría de las optimizaciones que han sido tenidas en cuenta en este tipo de diseños (agregación de datos, optimización de consultas, etc.) asumen un modelo con una pasarela. Es necesario, por lo tanto, estudiar este tipo de optimizaciones para arquitecturas con varias pasarelas tal y como se apunta en [18] .

En el caso de redes inalámbricas que no sólo contienen sensores sino que poseen también actuadores existe el problema de la duplicidad en cuanto a las órdenes que podemos transmitir a la red. Generalmente, en este tipo de middlewares, si por ejemplo queremos incrementar un valor en un actuador (control de luz, temperatura, etc.) se realiza mediante el establecimiento de un valor en la tabla residente en las pasarelas que, con posterioridad se traducen en órdenes a través de la WSN. Si no se establece algún mecanismo de sincronización entre las pasarelas se pueden cursar varias órdenes redundantes, que, dependiendo de la semántica de estas, podrían desembocar en funcionamientos del sistema controlado no deseados.

Los middlewares basados en agentes raramente abordan el problema de la existencia de dos o mas pasarelas centrándose en aspectos mas “afines” a su campo de

investigación como es la movilidad del código, mecanismos de búsqueda de la información que requieren los agentes, etc.

De forma general la integración de redes de sensores/actuadores inalámbricos mediante una infraestructura que disponga de varias pasarelas pasa por:

- Establecer una estructura lógica o topología que considere la existencia de dos o mas pasarelas. La estructura en árbol descrita anteriormente es la que, a priori, representa mayores problemas con este requisito ya que generalmente se debe coger la pasarela como raíz del árbol, no obstante los trabajos que tienen en cuenta la posible existencia de varias pasarelas optan por crear *bosques* de árboles disjuntos que son agregados de forma externa a la WSN[19] . Tanto la topología en estrella como en ad-hoc permiten establecer varias pasarelas mas fácilmente delegando la responsabilidad al algoritmo de enrutado.

- Habilitar un algoritmo de enrutado que permita el balanceo de carga entre aquellas pasarelas que supongan alternativas entre si, es decir, que el envío de un dato hacia una de ellas permita comunicar ese mismo dato a las mismas aplicaciones que si se enviaran a ambas. En este punto es necesario resaltar que se podría optar por crear una división lógica de la WSN atendiendo a la pasarela mas cercana y que se hace cargo desde el punto de vista lógico de recoger datos de una parte de la red. Obviamente si se opta por este esquema se debe habilitar un mecanismo para proporcionar a las aplicaciones una visión homogénea de la WSN que no dependa de la pasarela a la cual accede. A este nivel empiezan a aparecer trabajos que asumen la existencia de varias pasarelas, también llamadas sumideros (*multi-sink*)[16] [17] [20] .

- Establecer los mecanismos apropiados para que una invocación a la WSN se traduzca, en la medida de lo posible, en una sola ejecución de la misma, con independencia del número de pasarelas que envíen dicha invocación a la WSN. Este requisito generalmente puede ser abordado mediante dos técnicas:

  - Incluir en la invocación un identificador que permita discernir al dispositivo sensor/actuador cuando se trata de dos invocaciones diferentes o cuando es la misma invocación enviada a través de caminos, es decir, pasarelas distintas.
  - Especificar un mecanismo de sincronización entre pasarelas de forma que ante una misma invocación sólo una de las pasarelas retransmita esa invocación a la WSN. Esta última aproximación suele ser mas difícil de implementar debido a que pueden existir pasarelas que no compartan la infraestructura, la sincronización necesaria puede ser compleja, etc.

- Habilitar un esquema de direccionamiento lógico que permita la interacción con los sensores o actuadores de forma univoca. De igual forma, si con una pasarela toda la información generada en la WSN converge en ella, en una estructura con varias pasarelas, se debe identificar la pasarela destino. En este sentido trabajos como uIP [15] permite establecer el direccionamiento IPv4 en WSN a un bajo coste. De no establecerse un direccionamiento lógico único (para la WSN y el resto

de la infraestructura) se debe realizar en las pasarelas las finciones de NAT (*Network Address Translation*) apropiadas para poder comunicarse de forma directa con los dispositivos de la WSN.

- Por último y con independencia del número de pasarelas que existan se deben habilitar los mecanismos necesarios para la interoperabilidad necesaria entre la WSN y el resto de la infraestructura si usan distintos middlewares.

Existen una enorme cantidad de algoritmos de enrutado propuestos en la arquitectura para redes multisalto, sobre todo para las denominadas MANET's. La clasificación mas común que se realiza para este tipo de algoritmos de enrutado son proactivos y reactivos. Mientras que los algoritmos preactivos mantienen una visión global de la red en cada nodo mediante el intercambio de información periodica, los algoritmos reactivos buscan la ruta en el momento que es necesaria. Generalmente los algoritmos proactivos de cara a las WSN tienen el problema del intercambio periodico de información (con el consumo de energía asociado), si bien siempre saben el camino a seguir por la información, mientras que los reactivos deben establecer la ruta cada vez que se requiera una conexión y generalmente los paquetes a enviar deben contener la ruta completa.

A nivel de middleware el problema de tener varias pasarelas ha sido poco estudiado o simplemente no ha sido abordado con lo que podemos considerar como un problema abierto.

DUO: Distributed Universal Object

En la arquitectura DUO desarrollada por el grupo Arco de la UCLM los entornos inteligentes se abordan desde el paradigma de la programación orientada a objetos distribuidos (figura 4). Los entornos inteligentes tal y como adelantó Mark Weiser están pasando a ser entornos distribuidos con dispositivos de toda índole en cuanto a recursos y que deben proporcionar una visión homogénea al programador de aplicaciones.

Las WSN (y mas específicamente las WSNAs) en DUO son tratadas como parte integral del entorno y por lo tanto se proporcionan las herramientas para automatizar su integración con el resto del software.

En DUO se aboga por insertar en cada uno de los nodos que conforman la red los elementos necesarios para que sean modelados como objetos software. El desarrollador de aplicaciones para entornos inteligentes debe ser un experto en su área de conocimiento y no debe preocuparse de aspectos de bajo nivel acerca de protocolos eficientes, programación para dispositivos con bajos recursos, etc. Ese tipo de problemas debe ser abordados por los ingenieros de sistemas empotrados ya que constituye su ámbito de dominio.

Con estos conceptos en mente, en DUO asumimos que todos los dispositivos con su funcionalidad asociada son objetos distribuidos que, desde el punto de vista de un desarrollador de aplicaciones son invocados de forma transparente a su localización, dispositivo que lo soporte o red en la que esté presente.

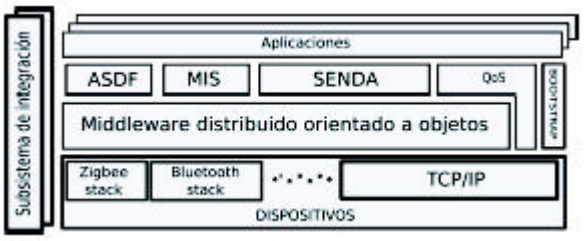

Fig. 1. Arquitectura general de DUO

En la arquitectura DUO (Figura 4) las aplicaciones usan los servicios de DUO y son modeladas conforme a sus principios. El modelado de aplicaciones generales y la descripción de cómo usar los servicios de DUO quedan fuera del ámbito de este documento.

No obstante el modelo de desarrollo para aplicaciones que incluyen WSN lo podemos visualizar en la figura 5. Una vez tenemos echo un adecuado análisis de requisitos de la aplicación a desarrollar y una vez identificadas las necesidades de sensorización/actuación que la aplicación necesita, se deben definir las interfaces que los objetos insertados en los dispositivos sensores o actuadores van a implementar.

Esta definición de interfaces constituye el contrato entre el servidor (residente en los dispositivos de la WSN) y el cliente (la aplicación interesada en invocar esas interfaces). Esta especificación de interfaces se realiza normalmente en un lenguaje IDL (*Interface definition language*) específico de la plataforma distribuida orientada a objetos a utilizar.

Una vez que establecemos estas interfaces se usan los compiladores apropiados para generar los denominados *stubs* y *skeletons* que nos proporcionarán la transparencia de acceso y que por lo tanto nos permitirán invocar los objetos de forma independiente a donde se instancien.

Obviamente los compiladores que las plataformas comerciales proporcionan (implementaciones de CORBA, ICE, JINI, etc.) generan implementaciones de los *stubs* y *skeletons* que no son apropiadas para los dispositivos habituales en las WSN. Dentro de DUO se desarrolla un compilador especial que genera los denominados *PicoObjetos*. Un *PicoObjeto* es el código necesario de las características fundamentales para presentar, de cara al exterior un comportamiento de un objeto estándar. Obviamente se sigue el protocolo estándar de la plataforma escogida para la

implementación de DUO y que en los desarrollos realizados en la actualidad están basados en el middleware ICE [26] (*Internet Communication Engine*).

El picoobjeto generado automáticamente parsea los mensajes relativos a las invocaciones de las interfaces que implementa y construye las respuestas de forma compatible al protocolo utilizado (ICEP en el caso de ICE).

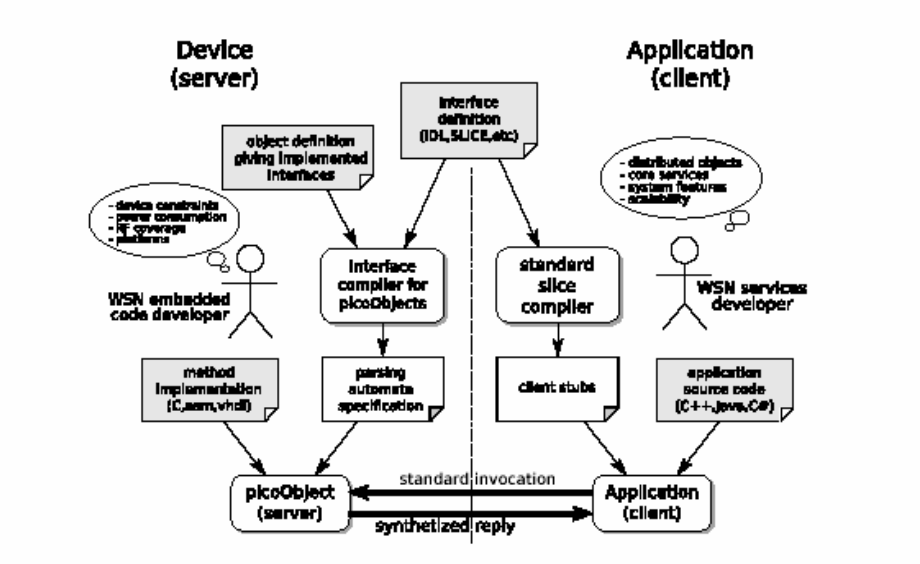

Fig. 2. Desarrollo de aplicaciones para WSN usando DUO.

Con este esquema, el desarrollador de la aplicación invoca los picoobjetos como si se tratara de objetos normales por lo que le es transparente todos los detalles de la plataforma utilizada para formar la WSN. En el caso del desarrollador de la parte dependiente de la aplicación residente en la WSN, se centra en implementar la funcionalidad de las interfaces sin preocuparse ni del software de comunicaciones (automáticamente generado) ni de las particularidades de DUO.

La utilización de WSN se realiza por lo tanto de forma integral en el proceso de desarrollo de las aplicaciones para entornos inteligentes y de forma mucho mas simple y transparente que las aplicaciones desarrolladas conforme a otros middlewares para este tipo de entornos podrían realizar.

Cada uno de los objetos distribuidos (residan o no en la WSN) tiene un ID que lo identifica unívocamente dentro de todo el entorno. Este ID a nivel de aplicación debe tener una correspondencia con el identificador del dispositivo donde reside, generalmente y en el caso de una red de área local con la pila de protocolos TCP/IP, este identificador de dispositivo se corresponde con su dirección IP. En el caso de los

objetos alojados en un dispositivo perteneciente a la WSN se opta por, en las pasarelas que conectan la WSN con el resto de la infraestructura, realizar NAT (*Network Address Translation*). La misión del NAT es detectar las invocaciones a los objetos residentes en la WSN y trasladar al algoritmo de enrutado utilizado en dicha WSN las invocaciones con el identificador de dispositivo correspondiente que alberga el objeto destino de la petición. En cualquier caso es necesario resaltar que la función de NAT puede ser realizada de forma dinámica conforme se añaden dispositivos y objetos a la WSN. Al contrario que ocurre con la mayoría de los middlewares para WSN, el software residente en la pasarela para realizar este NAT es genérico y totalmente independiente de la aplicación.

La correspondencia entre los identificadores de los objetos que residen en la WSN y los identificadores de los dispositivos donde se ejecutan puede ser realizada mediante configuración manual (en el caso de entornos estáticos) o de forma automática mediante un mecanismo de anuncios que se implementa en los dispositivos. Este mecanismo de anuncio constituye en si mismo un ejemplo de los objetos distribuidos que podemos integrar en los dispositivos que forman parte de la WSN. El interfaz que en este caso invoca cada uno de los dispositivos es la siguiente:

```
module ASD {

interface iListener {

idempotent void adv(Object* prx, iProperties* prop);

  };

};
```

La única operación es *adv* y tiene como argumentos el identificador del objeto residente en el dispositivo y el identificador del objeto que contiene que las propiedades del dispositivo. Este último objeto es nulo si el propio objeto (*prx*) tiene sus propiedades y tendrá el identificador del servidor de propiedades si esta funcionalidad está delegada. En cualquier caso, las implementaciones realizadas para objetos junto con sus anunciamientos en diversas plataformas representan, frente a propuestas similares, unos requisitos menores [24] .

En la tabla 2 podemos observar los requisitos para implementaciones mínimas de algunas propuestas para sistemas empotrados de plataformas orientadas a objetos distribuidos y middlewares para WSN junto con varias implementaciones de picoobjetos (en CORBA y ICE).

En principio, la infraestructura que se plantea en DUO para las WSN nos permite ser independientes de la topología y algoritmo de enrutado utilizados en el interior de la misma.

Las invocaciones que se generan en con el protocolo de ICE contienen un campo denominado *requestID* que determina de forma unívoca, junto con el cliente que ha generado la propia invocación, una petición. Por lo tanto, ante una invocación que llega varias veces a un objeto residente en un dispositivo que forma parte de la WSN, el picoobjeto puede distinguir cuando son dos invocaciones distintas o cuando es una misma invocación que ha seguido caminos distintos (en la propia WSN o reenviada por varias pasarelas).

Middleware	**Tamaño**	**Middleware**	**Tamaño**
TAO	1738KB+OS	picoCORBA (C)	5.2KB+OS
nORB	567KB+OS	picoCORBA (Java)	5 KB+OS
UIC/CORBA	35KB+OS	picoCORBA (PIC12C509)	415 words
JacORB (Java)	243KB	picoICE (tinyOS mica2 platform)	11KB
ZEN (Java)	53KB+OS	picoICE (C)	5.4KB+OS
MicroQoSCORBA (TINI)	21 KB	picoICE (PIC12C509)	503 words
TinyLime (mica2 platform)	16KB	picoICE (Atmel128)	1923 bytes
TinyDB	58KB	picoICE(Atmel128 + Zigbee communication library)	6184 bytes

Table 2. Implementaciones y recursos necesarios de middlewares[25]

En la figura 6 podemos ver la herramienta de monitorización *Inspector* desarrollada para la visualización y depuración de servicios DUO. Entre estos servicios podemos ver (servicio resaltado) sensores de presencia insertados en dispositivos Chipcom con interfaz Zigbee (Figura 7).

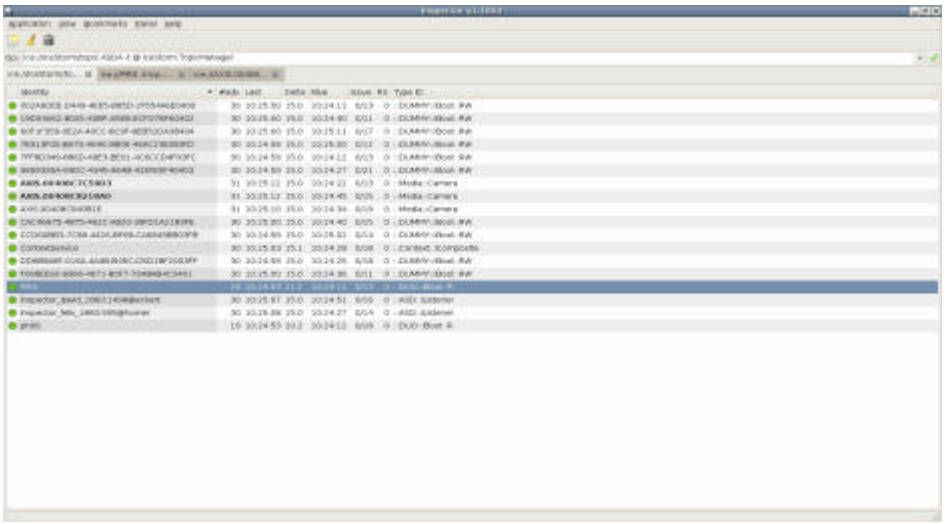

Fig. 3. Herramienta *Inspector*.

Es necesario resaltar como ejemplo de integración que, desde el punto de vista del *Inspector,* no se distingue cuando se accede a un servicio DUO residente en un PC, en un dispositivo perteneciente a una WSN o incluso una implementación hardware en una FPGA .

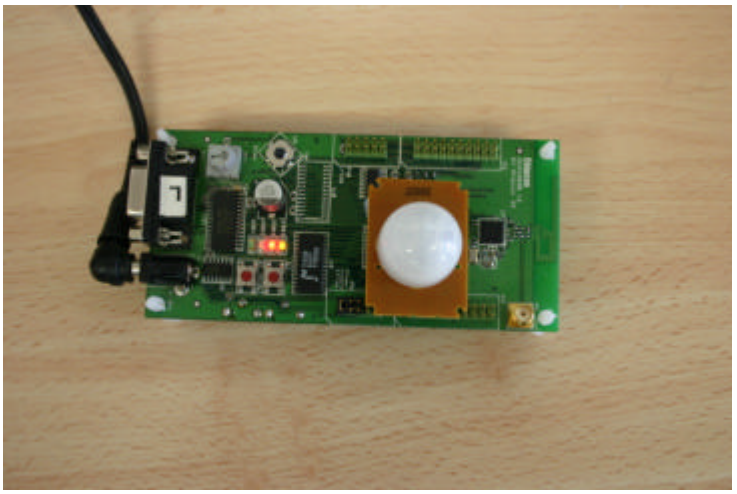

Fig. 4. Dispositivo Zigbee de Chipcom con sensor de presencia

Conclusiones

A lo largo de este documento hemos descrito varios de los problemas que existen en la actualidad en el desarrollo de aplicaciones para WSN en entornos inteligentes. A nivel de red se plantean problemas cuando el esquema de despliegue debe ser realizado mediante varias pasarelas mientras que a nivel de desarrollo la integración que se realiza pasa por establecer pasarelas entre diversos middlewares.

Para finalizar hemos descrito nuestra aproximación DUO que constituye, desde el punto de vista de la integración transparente de WSN en entornos inteligentes, un paso importante al homogeneizar y automatizar el desarrollo de aplicaciones sobre este tipo de redes.

Bibliografía

[1] Downes, L. Baghaei Rad, H. Aghajan. *Development of a Mote for Wireless Image Sensor Networks* in Proc if COGnitive systems with Interactive Sensors (COGIS), Paris, France. 2006.

[2] http://particle.teco.edu

[3] J. Gehrke, S. Madden *Query Processing in Sensor Networks* IEEE pervasive computing. January 2004.

[4] Y. Yao and J. Gehrke, *The Cougar approach to in-network query processing in sensor networks*, ACM SIGMOD Record, vol 31, no. 3, pp. 9-18, 2002.

[5] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W.Hong *The design of an acquisional query processor for sensor networks* in Proc. 22nd ACM SIGMOD International Conference on Management of Data, pp. 491-502, San diego, Calif, USA, June 2003.

[6] A. Boulis, C-C Han, and M. B. Srivastava, *Design and implementation of a framework for efficient and programmable sensor networks* in Proc. 1st International Conference on Mobile Systems, Applications, and Services (MobySys ' 03), San Francisco, Calif, USA, May 2003.

[7] F. Tartanoglu, F. Sailhan, R. Chibout, N. Levy, A Talamona Valerie Issarny, Daniele Sacchetti. *Developing ambient intelligence systems: A solution based on web services*. Journal of Automated Software Engineering, Vol 12, 2005.

[8] J. Gelissen. *Final report.* Technical report, IST OZONE PROJECT, 2005.

[9] IST Amigo Project. *State of the art analysis including assesment of system architectures for ambient intelligence.* Deliverable D2.2, Abril 2005.

[10] The OSGi Alliance. *OSGi Service Platform: Core Specification*, edición 4.0.1,Julio 2006. Release 4.

[11] S.Li, S.H. Son, and J.A. Stankovic. *Event detection services using data service middleware in distributed Sensor Networks*. In IPSN 2003, Palo Alto, USA, April 2003.

[12] D. Ganesan, *TinyDiffusion Application Programmer's Interface API 0.1*. http://www.isi.edu/scadds/papers/tinydiffusion-v0.1.pdf

[13] Carlo Curino , Matteo Giani , Marco Giorgetta , Alessandro Giusti , Amy L. Murphy , Gian Pietro Picco, *TinyLIME: Bridging Mobile and Sensor Networks through Middleware*, Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, p.61-72, March 08-12, 2005.

[14] F. Moya, D. Villa, F. J. Villanueva, J. Barba, F. Rincón, J. C. López, J. Dondo, *Embedding Standard Distributed Object-Oriented Middlewares in WSNs.* Special Issue on Distributed Systems of Sensors and Applications, WIRELESS COMMUNICATIONS AND MOBILE COMPUTING JOURNAL, WILEY. 2008.

[15] A. Dunkels. *Full TCP/IP for 8-bit architectures*. In Proceedings of The First International Conference on Mobile Systems, Applications, and Services, May 2003.

[16] P. Ciciriello, L. Mottola and G.P. Pietro Picco. *Efficient Routing from Multiple Sources to Multiple Sinks in Wireless Sensor Networks*. Lecture Notes in Computer Science. 2007.

[17] A. Lukosius. Opportunistic Routing in Multi-Sink Mobile Ad Hoc Wireless Sensor Networks. Master Thesis. 2007

[18] Q. Luo and H. Wu. *System Design Issues In Sensor Databases*, SIGMOD Tutorial. 2007.

[19] CRUISE project. Creating Ubiquitous Intelligent Sensing Environments, CRUISE deliverable, December 2006.

[20] S. Pack et al. "a dynamic load balancing scheme in Multi-Sink Wireless Sensor Networks". Technical-Report. December 2006.

[21] T. Hasiotis, G. Alyfantis, V. Tsetos, O. Sekkas and S. Hadjiefthymiades. *Sensation: A middleware Integration Platform for Pervasive Applications in Wireless Sensor Networks*. Proceedings of the second European workshop on wireless sensor networks. 2005.

[22] Jan M. Rabaey. Ubiquitous Sensor Networks and ambient intelligence- Technology Focusing on Society's Woes. 2005.

[23] Ting Liu and Margaret Martonosi. *Impala: a middleware system for managing autonomic, parallel sensor systems. In* PPoPP '03: Proceedings of the 9th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pages 107-118, New York, NY, USA, 2003. ACM Press.

[24] D. Villa, F. J. Villanueva, F. Moya, J. Barba, F. Rincón, J. C. López, *Minimalist Object Oriented Service Discovery Protocol for Wireless Sensor Networks* In Proc. of 2nd international conference on Grid and Pervasive Computing (GPC 2007), LNCS 4459, pp. 472-483, Paris (France), May 2-4, 2007.

[25] F. Moya, D. Villa, F. J. Villanueva, J. Barba, F. Rincón, J. C. López, J. Dondo. *Embedding Standard Distributed Object-Oriented Middlewares in WSNs*. Special Issue on Distributed Systems of Sensors and Applications, WIRELESS COMMUNICATIONS AND MOBILE COMPUTING JOURNAL, WILEY.

[26] M. Henning, M. Spruiell. Distributed Programming with Ice, 2006, available online at http://www.zeroc.com/

[27] W. Schott, A. Gluhak, M. Presser, U. Hunkeler and R. Tafazolli, e-SENSE Protocol Stack Architecture for Wireless Sensor Networks. Mobile and Wireless Communications Summit, 2007. 16th IST.

[28] Jeong-Hee k., Do-Hyeon K., Ho-Young K. Young-Cheol Byun. *Address Internetworking between WSNs and Internet supporting Web Services*. International Conference on Multimedia and Ubiquitous Engineering (MUE' 07).