Bizkaiko Foru Aldundia
Diputación Foral de Bizkaia
Berrikuntza eta Ekonomi Sustapen Saila
Departamento de Innovación y Promoción Económica

Programa Ekinberri 2007

SmartMotes

Nodos inalámbricos de redes de sensores con inteligencia semántica

D2.2 Análisis de modelos de inteligencia empotrada para nodos de sensores

Tecnológico Fundación Deusto
Teknologikoa Deustu Fundazioa
robotiker tecnalia

RESUMEN

El presente documento aborda el tema de las redes de sensores desde el punto de vista de la inteligencia que se les puede aplicar a los diferentes nodos.

En el capítulo 1 se realiza una presentación del ámbito y la situación actual en el área a nivel general.

El capítulo 2 define las características más relevantes y diferenciadoras que tienen las redes de sensores y que hay que tener en cuenta a la hora de desarrollar aplicaciones para ellas.

El capítulo 3 analiza el estado del arte en los modelos de middleware que se pueden utilizar para programar en las redes de sensores y el capítulo 4 va un paso más allá y analiza los modelos de inteligencia que se están aplicando en redes de sensores.

Por último, en el capítulo 5 se presentan unas conclusiones a modo de resumen, haciendo hincapié en el futuro de los modelos de inteligencia y las características que necesitan ser aplicadas en futuras investigaciones.

HISTORIAL DE CAMBIOS

Versión	Descripción	Autor	Fecha	Comentarios
V0.1	Estructura inicial del documento.	Iker Larizgoitia	07/01/2008	
V1.0	Terminada la primera versión completa del documento.	Iker Larizgoitia	01/02/2008	
V2.0	Versión Final: Revisada y añadidas las referencias bibliográficas.	Iker Larizgoitia	14/02/2008	Revisado por Iñaki Vázquez

TABLA DE CONTENIDOS

1 INTRODUCCIÓN

A lo largo de los últimos años las redes de sensores (*Wireless Sensor Networks* ó abreviando *WSN*) han sufrido un auge considerable, debido principalmente a la importante evolución en los elementos hardware que se utilizan. Aunque los sensores y dispositivos son cada vez más pequeños, más potentes y más baratos, todavía existe un hueco considerable entre los sistemas propiamente dichos y las aplicaciones que van sobre ellos.

La investigación en el campo de las redes de sensores es relativamente reciente, por lo que todavía se arrastran los problemas típicos de cualquier comienzo, siendo uno de ellos la falta de consenso en ciertos aspectos fundamentales. Hasta ahora los esfuerzos se han concentrado principalmente en dos áreas: determinar qué paradigma de comunicación es más adecuado para este tipo de redes y definir las primitivas básicas necesarias sobre dicha comunicación para este tipo de sistemas.

La evolución en el desarrollo de dos puntos, como veremos a lo largo del presente documento, a nuevos enfoques que intentan dar un paso más y adaptar aspectos del middleware tradicional a las redes de sensores.

Por otro lado, el diseño de aplicaciones para redes de sensores condicionaba en cierta manera su estructura [1]. Debido a las limitaciones que tiene el hardware utilizado, tradicionalmente el diseño seguía un enfoque *bottom-up.* Con este enfoque las aplicaciones se construyen sobre un único sensor que utiliza la red como mero mecanismo de comunicación. A medida que se consiga homogeneizar ciertos aspectos de diseño de bajo nivel, se podrá pasar al enfoque contrario, el *top-down.* Este enfoque ha resultado muy útil en los sistemas sensibles al contexto, los cuales veremos que tienen posibilidad de convertirse en el complemento ideal para las redes de sensores.

2 CARACTERÍSTICAS DE LAS REDES DE SENSORES

Las redes de sensores tienen características que las hacen cada vez más atractivas desde el punto de vista industrial: bajo coste, bajo consumo, fácil despliegue, escalabilidad,… Sin embargo, estas particularidades hacen que el desarrollo de aplicaciones sobre ellas no sea trivial, llegando en la mayoría de los casos a soluciones *ad-hoc* para cada aplicación concreta.

Basándonos en las características especiales de las *WSN*, los retos que se plantean a la hora de diseñar y desarrollar aplicaciones de manera eficaz son los siguientes [2]:

- ***Gestión de recursos y energía***: Los sensores son dispositivos miniaturizados con limitaciones, tanto de procesamiento como de energía. El entorno de despliegue puede a su vez llevarlos a sitios inaccesibles, donde en la mayoría de los casos dependen únicamente de su capacidad de comunicación. En base a estas restricciones, las tres operaciones básicas de este tipo de dispositivos: la sensorización, el procesamiento de los datos y la comunicación, deben realizarse y planificarse con la intención de no malgastar recursos innecesariamente, ya que ello puede repercutir negativamente en la vida y operación de la red.

- ***Escalabilidad, movilidad y topología de red dinámica***: El entorno altamente dinámico donde operan las redes de sensores necesita de mecanismos robustos de funcionamiento tolerantes a fallos. A su vez, los propios sensores necesitarán auto-configurarse y auto-mantenerse para adaptarse a los cambios en la red.

- ***Heterogeneidad***: La multitud de dispositivos diferentes obliga a unificar de alguna manera las operaciones más utilizadas, como son la configuración, la ejecución y sobre todo la comunicación. Las aplicaciones o plataformas tienen que ser capaces de abstraer las particularidades del hardware en funciones abstractas de alto nivel.

- ***Organización dinámica de la red***: Muchos de los recursos con los que se cuenta en las redes de sensores son dinámicos (la energía, el ancho de banda, capacidad de procesamiento, número de nodos,…). La organización de dichos elementos es una parte esencial de la propia red. Con este objetivo tiene que existir un mecanismo de descubrimiento que permita saber en todo momento cómo está la red y quiénes están accesibles.

- ***Integración en el mundo físico***: Gran parte de los sistemas que se despliegan con redes de sensores están basados en escenarios donde el tiempo y el espacio (la localización) son fundamentales para el sistema. Es por ello que los servicios que proporcionen las plataformas tienen que soportar las características de tiempo real que demandan las aplicaciones.

- ***Conocimiento de las aplicaciones***: Otro de los aspectos a considerar es el grado en que las aplicaciones van a conocer las características de la red de sensores. En el mejor de los casos, la abstracción total, serán necesarios mecanismos de mapeo entre, por ejemplo, las necesidades de comunicación de las aplicaciones y los parámetros de red necesarios para conseguirlas.

- ***Agregación de datos***: El despliegue de multitud de sensores con características similares puede provocar la existencia de datos redundantes en ciertas localizaciones. La gestión y agregación de estos datos puede ahorrar energía y recursos, por lo que pueden ser necesarios nuevos enfoques de comunicación centrados en los datos.

- ***Calidad de servicio***: La calidad de servicio es un término muy amplio que puede tener diferentes interpretaciones. En lo referente a las redes de sensores, son importantes dos ámbitos, el de aplicación y el de red. El de aplicación tendrá en cuenta las necesidades de la aplicación en cuanto a medidas de los nodos, el despliegue o los nodos activos, entre otros. El ámbito de red se centrará en cómo cumplir las necesidades de las aplicaciones, gestionando el ancho de banda y la energía. A la vista está que hacen falta también nuevos modelos de gestión y medida de la calidad de servicio para el ámbito de las redes de sensores.

- ***Seguridad***: En muchas ocasiones la información con la que tratan los sensores puede ser sensible para el usuario, como por ejemplo, los datos de su estado físico o salud. Asimismo, las condiciones de despliegue y el uso de tecnologías inalámbricas para las comunicaciones los hacen más vulnerables a los ataques malintencionados. Los métodos tradicionales de seguridad no se adaptan a estos entornos, ya que los recursos son limitados. A las necesidades básicas de seguridad (confidencialidad, autenticación o integridad de los datos) hay que sumarles también otras más específicas como la disponibilidad o la frescura de los mismos.

3 MIDDLEWARE PARA REDES DE SENSORES

Como hemos comentado anteriormente, la investigación sobre las redes de sensores se centraba básicamente en el hardware y los mecanismos de comunicación básicos. A medida que la tecnología ha ido madurando, se ha visto que las necesidades de las redes de sensores van más allá. Tener un conjunto de sensores desplegados de manera más o menos aleatoria en un área hace que aspectos como los recursos limitados de dichos nodos, los fallos típicos de este tipo de redes o la pérdida de energía sean factores que hay que tener muy en cuenta a la hora de desarrollar aplicaciones en este entorno (ver Capítulo 2).

Los primeros enfoques trataban cada nodo de forma independiente y únicamente utilizaba al resto para comunicarse. Sin embargo, la idea de enfocar la red hacia la consecución de un objetivo común por parte de los nodos que la forman, pone de manifiesto la necesidad de mecanismos que permitan la coordinación, compartición y gestión de los datos y los objetivos correspondientes. Por este motivo la solución natural es la creación de un *middleware* que sea capaz de “enganchar” las aplicaciones a la red, facilitando su diseño y potenciando todas las características especiales que tienen las *WSN*.

Desde un punto de vista objetivo, el uso de un *middleware* puede en un primer momento parecer inadecuado, ya que todo *middleware* supone por naturaleza una sobrecarga al sistema. Sin embargo, la abstracción que se consigue permite encapsular funcionalidades con lo que se potencia la reusabilidad y la simplicidad de los sistemas. Los problemas que surgen en la creación de plataformas para redes de sensores desde el punto de vista de la programación se centran principalmente en dos ámbitos:

- Cómo dar soporte a la programación en sus tareas más fundamentales: cómo distribuir el código, cómo ejecutarlo o qué servicios de programación y ejecución proporcionar.

- Mecanismos de abstracción de la programación, orientados a facilitar la visión que hay de la red y facilitar las referencias a los nodos y los datos que contienen.

En base a las diferentes aproximaciones que existen [3] [2], podemos realizar una taxonomía de los modelos de programación de *WSN,* como se muestra el la figura siguiente.

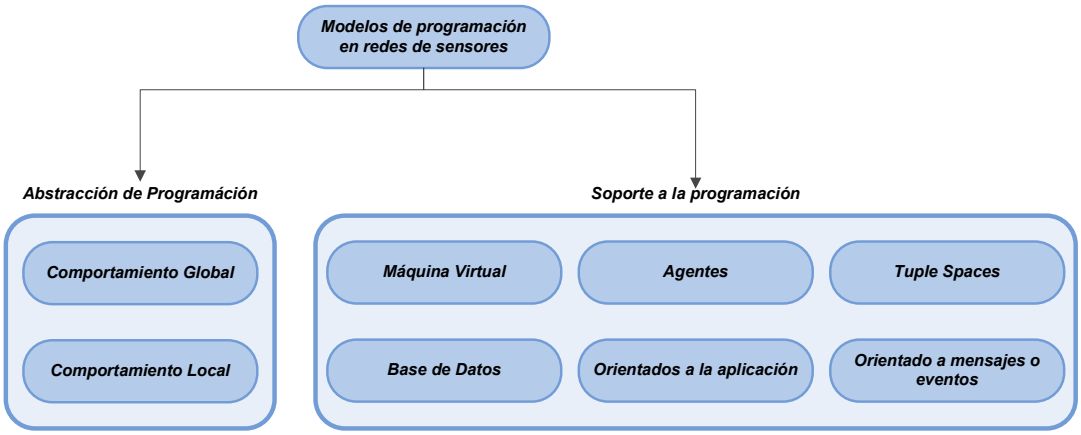

Figura 1 .- Taxonomía de los modelos de programación de sensores.

A continuación analizamos las características de cada enfoque y los proyectos más destacados dentro de cada uno.

3.1 Máquina Virtual

Los enfoques basados en máquina virtual se componen principalmente del entorno de la máquina virtual, el intérprete correspondiente y los mecanismos necesarios para que el código pueda cargarse de manera remota y flexible. Las aplicaciones se dividen en pequeños módulos que se distribuyen en la red a través de algoritmos que tratan de minimizar el uso de la energía. El principal inconveniente de este tipo de modelos es la sobrecarga que introducen los intérpretes de la máquina virtual. Siguiendo este modelo podemos citar los siguientes proyectos:

Maté [4]: Los componentes del sistema *Maté* son una máquina virtual, la red, un sistema de *log* y un planificador de arranque. Se base en un intérprete de *bytecode* sobre *TinyOS.* Los programas se dividen en cápsulas de 24 bytes (una especie de componentes) y el modelo de funcionamiento se basa en eventos síncronos, que se lanzan bien por la recepción de un paquete concreto o bien por *timeouts*. Este modelo no cuenta con opciones de *buffering* ni almacenamiento de grandes cantidades de datos.

Mate provee de una buena interacción y adaptación a la naturaleza cambiante de las redes de sensores. Permite también la actualización de los componentes y los protocolos que se utilizan de manera dinámica. Sin embargo, el consumo de energía derivado de la interpretación que lleva a cabo la máquina virtual sólo lo hacen adecuado para aplicaciones que tengan poca actividad. Además resulta complicado de utilizar, ya que carece de un entorno de programación adecuado.

MagnetOS [5]: Este proyecto se presenta como un sistema operativo *power-aware*, donde la red se contempla como una única máquina virtual. Siguiendo este patrón, existen componentes estáticos y dinámicos, que se cargan y migran a los diferentes nodos. Este sistema operativo está pensado para que los componentes se muevan a los nodos más próximos, ahorrando así energía en la recolección y ejecución de los datos.

El enfoque de máquina virtual utilizado por *MagnetOS* es adecuado para sistemas heterogéneos, ya que la programación de los detalles de bajo nivel está abstraída. El sistema de ejecución está optimizado para reducir el consumo de energía y la programación es sencilla, ya que se basa en una implementación Java, sin embargo este mismo hecho hace que la sobrecarga de memoria sea grande y que no sea apropiado para redes basadas en dispositivos más limitados.

3.2 Agentes móviles

Este modelo persigue diseñar las aplicaciones de manera modular, para que se puedan mover y transmitir por la red de forma más eficiente. Se basa en la idea de que transmitir pequeñas piezas de software es mucho más sencillo que transmitir aplicaciones enteras. Además hereda los conceptos de la tecnología de agentes adaptándolos a las redes de sensores.

Los proyectos más significativos dentro de este enfoque son:

Impala [6]: Este sistema soporta actualizaciones de software, es tolerante a fallos y permite auto-organización de la red. El inconveniente es que no soporta hardware diferente, ya que está pensado para una plataforma específica. El modelo de comunicación es asíncrono basado en eventos. Los diferentes agentes móviles se compilan en instrucciones binarias y se inyectan en los nodos a través de un protocolo, que puede variarse. El comportamiento de los eventos se puede también modificar en base al escenario de aplicación. El agente encargado de ello lleva para tal fin implementada una máquina de estados que gestiona por ejemplo la eficiencia energética u otros atributos que la aplicación especifique. El uso de agentes permite potenciar características como la movilidad o la adaptación en tiempo de ejecución. *Impala* permite asimismo cambiar el protocolo de enrutamiento en tiempo de ejecución. El ahorro de energía en el despliegue se consigue diseñando las aplicaciones de forma modular, ya que de esta manera se podrán actualizar los módulos de forma independiente.

3.3 Enfoques inspirados en base de datos

En este enfoque, la red de sensores a programar se considera una base de datos virtual a la cual se le pueden realizar consultas a través de un interfaz definido, en la mayoría de los casos siguiendo una sintaxis similar a SQL. Utilizar esquemas parecidos a SQL hace que estos sistemas sean relativamente fáciles de utilizar, pero lo que se gana en sencillez se pierde en otros aspectos fundamentales. Si bien se hace hincapié en el enrutamiento de las consultas y el procesamiento de las mismas, existen una serie de desventajas en el uso de estos sistemas [3] :

- En la mayoría de los casos suponen que los sensores son homogéneos, proporcionando sus datos o medidas en el mismo formato o escala, lo cual no es adaptable a ciertas aplicaciones.

- El modelo de base de datos virtual necesita que los nodos tengan un consenso sobre los tipos de datos que utilizan para poder proporcionar la vista de base de datos.

- Este modelo no es adecuado para sensores que tengan datos complejos, como por ejemplo, el video.

- El lenguaje de consulta utilizado, normalmente variantes de SQL no es del todo apropiado, ya que los sensores capturan observaciones en el tiempo, no hechos.

- Por la naturaleza de los sensores siempre habrá cierto grado de incertidumbre o imprecisión en sus medidas. El tratamiento de estos aspectos no suele contemplarse en el enfoque de base de datos.

- La reactividad que presenta este tipo de sistemas es limitada, ya que los nodos se contemplan como una mera fuente de información y las decisiones de actuación las toma generalmente el servidor central que recoge los datos.

Siguiendo este modelo podemos destacar los siguientes proyectos:

Cougar [7]: *Cougar* adopta el modelo de base de datos distribuida para representar la red de sensores. Proporciona una interfaz similar a SQL con la cual se pueden lanzar consultas a la red. Como aspecto significativo está el uso de abstracciones de datos para representar los sensores físicos y contempla las consultas incrementales y temporales en base a series de datos recogidos.

Este modelo es adecuado para redes de sensores densas y el sistema de consulta que ofrece es sencillo de usar. Sin embargo, la comunicación de los sensores con el servidor de base de datos genera demasiada sobrecarga y la gestión de la energía por lo tanto no es muy eficiente. Además tampoco considera otros aspectos fundamentales como la movilidad o la heterogeneidad, limitaciones ambas condicionadas por su enfoque centralizado.

TinyDB [8]: Este sistema también se basa en la idea de una base de datos virtual a la cual se pueden realizar consultas usando un lenguaje similar a SQL. Las consultas se envían a la red a través de inundación y en los paquetes va información de la ruta, lo cual facilita la agregación de los datos de los sensores. Cada nodo cuenta con su propio procesador de consultas que valora si agregar sus datos o no, manteniendo la ruta de la consulta.

Al igual que el resto de modelos basados en base de datos, el sistema es sencillo de utilizar. Derivado del propio *TinyOS*, el sistema ya cuenta con utilidades para gestionar la heterogeneidad. *TinyDB* incluye también mecanismos de agregación de los datos de los sensores, lo cual reduce considerablemente la cantidad de datos transmitidos. Existen problemas de escalabilidad, ya que el comportamiento es local, por lo que la modificación del comportamiento es costosa.

Sina [9]: Este enfoque se basa en el concepto de objetos distribuidos basado en clusters, a los cuales se les puede consultar y monitorizar. Los nodos se dividen en celdas y cada celda se gestiona en base a cuatro situaciones: petición de información bajo demanda, preparación de contenido, actualización periódica de contenido o actualización de contenido en base a eventos.

Sina incorpora como novedad mecanismos de clustering de sensores para ahorrar energía. Sin embargo el middleware necesita que la capa de aplicación sea más compleja y no soporta bien la compatibilidad con hardware diferente.

3.4 Tuple spaces

Este enfoque es parecido al de base de datos, pero el mecanismo de acceso es diferente, ya que utiliza un modelo de memoria distribuida compartida donde los diferentes actores pueden meter y sacar información.

Este enfoque es apropiado cuando las aplicaciones únicamente necesitan consultar datos, ya que no suele haber propagación de información, sino que todo funciona mediante consultas explícitas. Este aspecto limita la aplicabilidad del modelo a distintos escenarios.

Un proyecto destacado en este enfoque es ***TinyLIME*** [10]. En él se crean suscripciones en el *Tuple Space* para los datos de los sensores que están en rango de proximidad 1 a 1. Permite aplicar ciertos filtrados a los datos, pero muy limitado (por ejemplo, pedir los datos de temperatura siempre que sea menor que 30).

El reto de este tipo de sistemas es crear un espacio que considere la distribución, federación y movilidad de los nodos de la red de sensores.

3.5 Enfoques orientados a la aplicación

En este tipo de modelos, se va un paso más allá, incluyendo la pila de red en la propia gestión. Esto permite optimizar la red en función de la calidad de servicio y operaciones que realizan las aplicaciones. El inconveniente de este tipo de enfoques es que el acoplamiento que se crea con las aplicaciones hace que el middleware pierda generalidad.

Siguiendo este enfoque encontramos los siguientes proyectos:

Milan [11]: Este proyecto se centra en las necesidades de alto nivel de las aplicaciones para gestionar la calidad de servicio, ajustando la configuración de la red. Mediante el uso de grafos de estados determina la combinación de sensores que satisfacen la calidad de servicio que necesita la aplicación. *Milan* está especialmente diseñado para adaptar la gestión de la red a las aplicaciones, tratando de manera conveniente la escalabilidad del sistema. Provee de mecanismos para adaptar las aplicaciones, pero dejando de lado la eficiencia energética.

El descubrimiento se basa en protocolos existentes (no siempre adecuados a estos entornos), pero tiene un mecanismo de plugins que permite cambiarlo. Las aplicaciones definen la calidad del servicio que precisan en base a grafos de dependencias sobre variables. Las variables definen los tipos de sensores que hay y la relación entre las variables y los sensores se lleva a cabo en el descubrimiento. También permite la definición de sensores virtuales como mecanismo de agregación. Asimismo en este middleware no se contemplan mecanismos de movilidad.

3.6 Enfoques basados en eventos y mensajes

Los sistemas basados en eventos han resultado ser uno de los modelos que más relación tienen con el funcionamiento natural de las redes de sensores. A pesar de ello, las soluciones propuestas pecan de inmadurez.

Existen intentos por unificar los operadores necesarios básicos (por ejemplo, la unión, conjunción, negación, etc. de eventos) pero hay dos características fundamentales que son problemáticas: las relaciones espaciales y las temporales. Las relaciones espaciales requieren de una información de posición precisa, que no siempre puede obtenerse y las temporales necesitan de un sistema de sincronización que suele resultar complejo de implementar.

El enfoque orientado a eventos a su vez persigue facilitar el intercambio de mensajes mediante el uso de un sistema de publicación-suscripción. La principal ventaja es el soporte para el asincronismo de forma natural, lo cual facilita la programación orientada a eventos.

Los proyectos más significativos dentro de este enfoque son:

Mires [12]: este modelo resulta algo más pragmático que el resto, ya que es una adaptación del modelo *MOM* sobre *TinyOS* utilizando *NesC*. Se basa también en un modelo de componentes e implementa servicios adicionales como el encaminamiento o la agregación de datos de los sensores. El proceso es bien simple, los productores publican los datos de los sensores en la red y se van enrutando hacia los consumidores que están suscritos a cada evento. Los eventos se configuran a través de un interfaz de usuario donde se definen los eventos que se quieren recibir.

El mecanismo de publicación-suscripción utilizado reduce de manera considerable el uso de la energía. Al basarse en *TinyOS* es adecuado para hardware heterogéneo. La escalabilidad y la movilidad son parciales, ya que el protocolo de comunicaciones que se utiliza no soporta descubrimiento.

En [13] la principal desventaja de este proyecto es que consideran que los sensores son homogéneos en tipos de datos y utilizan un mecanismo demasiado simple para la definición de los mismos (por ejemplo, los datos de temperatura se marcan con una T).

DsWare [14]: este enfoque combina la abstracción de una base de datos virtual con la detección de eventos. Mejora la ejecución en tiempo real mediante el uso de técnicas de agregación y almacenamiento de datos. También provee de ciertos servicios a las aplicaciones como son almacenamiento, cacheo, detección de eventos, suscripción a eventos de datos o planificación, todo ello mediante un interfaz similar a SQL.

DsWare tiene como ventajas su facilidad de uso y la incorporación de ciertos servicios de serie que suelen ser útiles para el programador, como el mecanismo de gestión de nodos entrantes y salientes o la gestión de fallos en la red. Además incorpora también mecanismos de agregación de datos, pero se queda un poco corto en cuanto a soporte para la heterogeneidad y la flexibilidad, ya que se apoya en un modelo fijo global de red.

En lo relativo al modelo de eventos, incorpora cierta gestión de la incertidumbre en las correlaciones de los eventos determinando la confianza y el tiempo de las ocurrencias.

3.7 Comportamiento global

Este modelo proporciona una visión diferente de cómo programar las redes de sensores. Se basa en el concepto de macroprogramación, que significa que lo que se define con los programas es el comportamiento global que tiene que tener la red en su conjunto.

Este comportamiento se define a través de especificaciones de alto nivel, que son traducidas después de manera automática al comportamiento que va a tener cada nodo, liberando al programador de los detalles de bajo nivel de la red.

Proyectos destacados que utilizan este modelo son los siguientes:

Kairos [15]: En Kairos, los programadores definen un único programa general (el comportamiento global) que se descompondrá después en comportamientos específicos para cada nodo. Proporciona servicios tanto de compilación como de ejecución, pero haciendo los detalles de bajo nivel transparentes, como por ejemplo la generación del código distribuido, el acceso a los datos de los sensores de forma remota o la coordinación que tiene que seguir el flujo del comportamiento que se defina. Asimismo define abstracciones de los nodos y sus relaciones, así como de los datos que están disponibles en la red.

En el modelo de macroprogramación de Kairos, los programadores no se abstraen por completo de las implicaciones sobre el rendimiento que tiene la organización de los programas. La escalabilidad y la gestión de energía se consiguen parcialmente, ya que las aplicaciones no tienen control sobre ciertas funciones del middleware. La movilidad está soportada totalmente, con mecanismos robustos de localización y enrutado de los nodos.

RuleCaster [1]: la propuesta de RuleCaster consiste en proporcionar un lenguaje de alto nivel para definir las aplicaciones, un modelo dinámico que describa la red y los sensores que contiene, un compilador que es capaz de descomponer las aplicaciones en tareas y asignarlas a los diferentes nodos y un runtime que permita a los nodos recibir nuevas tareas y ejecutarlas en colaboración con otros nodos de la red.

Podemos destacar también otros proyectos que encajan en este modelo como ***Regimient*** [16] que define un lenguaje funcional orientado a las demandas de las aplicaciones, ***Abstract Task Graph*** [17] o ***Semantic Streams*** [18] que define el acceso a los servicios de la red a través de búsquedas semánticas.

3.8 Comportamiento local

La mayoría de las aplicaciones que utilizan redes de sensores dan más importancia a la naturaleza de los datos sensorizados, que normalmente se obtienen a partir de un grupo específico de nodos. A una aplicación concreta le interesa más ciertos datos localizados que todas las medidas que puede proporcionar la red. Cuando el enfoque se centra en los datos, los nodos tienden a direccionarse en base a los datos que son capaces de producir.

Siguiendo este enfoque encontramos los siguientes proyectos:

Abstract Regions [19]: Este proyecto potencia la idea de que casi todas las aplicaciones basadas en redes de sensores se pueden reducir en mecanismos de coordinación de ciertos nodos que agregan los datos pertenecientes a una zona concreta. Proporciona primitivas de comunicación de propósito general para redes de sensores, como son el direccionamiento, la agregación y la compartición de los datos, todo ello de forma localizada, lo que permite un ahorro de energía y reducción del ancho de banda.

Este enfoque como tal no es un middleware completo, sino una serie de primitivas de comunicación para el desarrollo de middleware.

EnviroTrack [20]: Este modelo está pensado para potenciar las aplicaciones de seguimiento, mediante el uso de etiquetas de contexto. Estas etiquetas se asocian a elementos físicos, cuyos datos sensorizados serán agregados para poder monitorizarlo de manera más eficiente. *EnviroTrack* soporta movilidad de los elementos físicos que se monitorizan.

Otros proyectos dentro de este modelo son ***Hood*** [21] que utiliza la abstracción de barrios de nodos o ***Generic Role Assignment*** [22], donde en base a la aplicación se especifica el rol que tiene que tener cada nodo.

3.9 Otros modelos

Existen también otros modelos que extienden los anteriores o los combinan. Entre ellos podemos destacar ***AutoSec*** [23] que sigue un enfoque orientado a la aplicación basada en *brokering* para optimizar el uso de los recursos y la gestión de la red, ***Agilla*** [24] basado en agentes móviles que son capaces de migrarse y clonarse en base a los cambios que sufra la red o ***Garnet*** [25] que explora el uso de flujos de información como abstracción para los datos de los sensores.

4 MODELOS DE INTELIGENCIA PARA REDES DE SENSORES

Analizando la trayectoria que han tenido las redes de sensores [26], podemos destacar varias fases. En un primer momento los esfuerzos estaban concentrados en el desarrollo de hardware, diseñando dispositivos adecuados tanto en tamaño como en precio para que resultara viable su despliegue. Con el surgimiento de los nuevos estándares de comunicación inalámbricos (Zigbee, 802.15.4) se potenció el desarrollo de protocolos de encaminamiento adecuados a las redes de sensores, cuya característica principal es su dinamismo y heterogeneidad.

La creciente madurez en el área, así como la rápida incursión en el sector comercial, abre nuevas cuestiones relacionadas con el desarrollo y despliegue de aplicaciones basadas en redes de sensores. Debido a ello, como hemos analizado en el apartado anterior, empiezan a surgir de forma natural diferentes abstracciones para facilitar la programación de los sensores a nivel de middleware.

El siguiente reto en este tipo de redes es conseguir dotarlas de cierta inteligencia, bien a la red en conjunto o a los propio nodos, de manera que éstos sean capaces de tomar ciertas decisiones que mejoren no sólo el funcionamiento de la red, sino también las aplicaciones que van sobre ellas, así como todos los aspectos de importancia (gestión de la energía, descubrimiento, gestión de los servicios,…).

Cómo dotar a los nodos de inteligencia es un área que últimamente se está estudiando activamente. Qué modelo de inferencia puede resultar más adecuado está todavía por definir. Los esfuerzos actuales están orientados a la inclusión de motores de inferencia adaptados a las redes de sensores.

Por otro lado tenemos la Web Semántica, que ha alcanzado también cierto grado de desarrollo, pero en el sentido contrario. Ésta se concibe como una red de información y actuación a gran escala e interoperable. Parece por lo tanto que hay cierta posibilidad de integración de ambos mundos, donde las ontologías pueden convertirse en el nexo de unión entre ellos.

En la figura siguiente se muestran los modelos de inteligencia para sensores que serán analizados en los apartados siguientes.

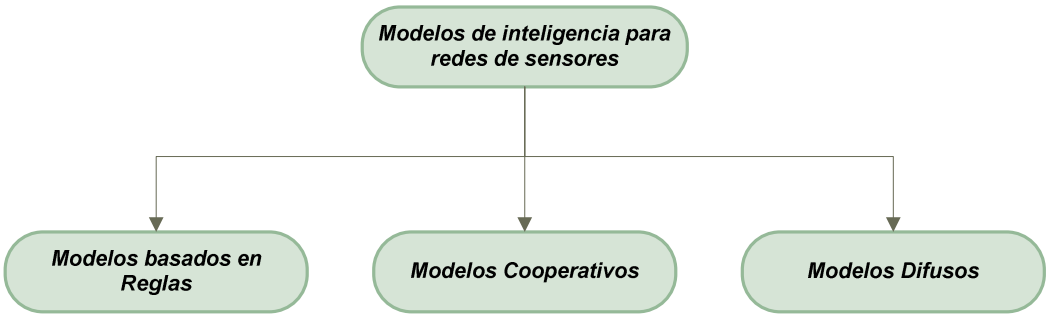

Figura 2.- Taxonomía de los modelos de inteligencia para redes de sensores

4.1 Modelos basados en reglas

Los sistemas de reglas se presentan como una alternativa sencilla para la introducción de inteligencia en las redes de sensores. Si bien ciertas técnicas de razonamiento basado en reglas pueden resultar excesivas para este tipo de entornos, correctamente adaptadas se convierten en un enfoque mucho más natural para definir el comportamiento esperado de los nodos.

Entre los proyectos que utilizan y adaptan los mecanismos basados en reglas podemos citar los siguientes:

Ubiquitous Chip [27] utiliza un lenguaje de descripción de comportamiento basado en reglas *ECA* (Event, Condition, Action) con funciones de entrada y salida simple. Su motivación parte de la idea de que es necesario un modelo que permita cambiar el comportamiento de los dispositivos de manera dinámica para que éstos reaccionen al usuario de forma adecuada. Esto lo consiguen separando las entradas y salidas de los dispositivos utilizando un enfoque basado en reglas.

El modelado con reglas permite el diseño de un sistema más natural y permite modelar mejor el comportamiento que espera el usuario. Las reglas que utiliza están optimizadas para que ocupen lo mínimo (4 bytes), a cambio de reducir la expresividad, ya que únicamente se permiten dos condiciones máximo por regla. La implementación se basa en un PIC 16F873.

RULE 1	RULE 2	RULE 3
E:	E: TIMER	E:
C: I1=0, S1=0	C:	C: I1=1, S1=1
A: S1=1, TIMER(1min)	A: O1=1	A: S1=0, TIMER(0), O1=0
RULE 4	**RULE 5**	**RULE 6**
E:	E: RECEIVE(M2)	E: RECEIVE(M3)
C: I2=1	C:	C:
A: SEND(M1)	A: SEND(M3)	A: O2=1, SEND(M4), S2=1
RULE 7	**RULE 8**	**RULE 9**
E:	E:	E: RECEIVE(M4)
C: I3=1, S2=1	C: I3=0	C:
A: O3=1	A: O3=0	A: S3=1
RULE 10	**RULE 11**	
E:	E:	
C: I4=1, S3=1	C: I4=1, S3=0	
A: O4=1	A: O4=0	

I1: door sensor	O1: buzzer	S1: door flag	M1: unlock request
I2: key sensor	O2: room light	S2: enter flag (for chip B)	M2: unlock complete
I3: heat gauge	O3: air conditioner	S3: enter flag (for chip C)	M3: enter (to chip B)
I4: phone call	O4: phone		M4: enter (to chip C)

Figura 3.- Ejemplo de reglas utilizadas en Ubiquitous Chip.

Cooperative Artefacts [28]: Este proyecto se basa en dos conceptos principales, la existencia de conocimiento del dominio en los dispositivos empotrados y la inclusión de inteligencia perceptual a través de un sistema de inferencia basado en reglas.

Utiliza un modelo de agentes, con un sistema de base de conocimiento compartida que se compone de hechos y reglas. Los hechos pueden ser de dominio, de observaciones o inferidos. Las reglas pueden a su vez ser de inferencia o de actuación.

Uno de los objetivos principales de este proyecto era que los dispositivos cooperaran entre ellos. Dicha colaboración se puede dar en dos supuestos, a nivel de aplicación, todos los nodos que pertenezcan a una misma aplicación o a nivel de cercanía, todos los nodos que se encuentran en un rango determinado.

El motor de reglas es un subconjunto del intérprete de Prolog basado en lógica de predicados con encaminamiento hacia atrás. Tiene implementado un algoritmo de inferencia con *depth first search*, cuyo funcionamiento básico es intentar probar los objetivos que se definen con las reglas. El razonamiento se lleva a cabo en tiempo real y es dependiente del contexto, el cual se comparten mediante un protocolo de consulta-respuesta entre los nodos.

Inference rules

```
(R1) hazard_unapproved:-      content(me, CH),
                              critical_time(CH, T1),
                              location(me, out, T2),
                              T1 < T2.

(R2) hazard_incompatible:-    content(me, CH1),
                              proximity(me, C),
                              content(C, CH2),
                              reactive(CH1, CH2).

(R3) hazard_critical_mass:-   content(me, CH),
                              cond_sum(
                                 M1,
                                 (proximity(me,C),
                                  content(C,CH),
                                  mass(C,M1)),
                                 S),
                              mass(me, M2),
                              sum(S, M2, SUM)
                              critical_mass(CH, MASS),
                              MASS < SUM.
```

Actuator rules

```
(R4) alert_hazard:- hazard_unapproved
(R5) alert_hazard:- hazard_incompatible
(R6) alert_hazard:- hazard_critical_mass
```

Figura 4.- Ejemplo de regla utilizada en Cooperative Artefacts.

El principal inconveniente de este sistema es que la independencia de los dispositivos puede llevar a inconsistencias, las cuales no están tratadas. Además, no contempla el tiempo en los razonamientos, por lo que no se pueden correlacionar hechos o eventos.

FACTS [29]: *Facts* es un middleware para redes de sensores que combina un sistema basado en eventos con un motor de reglas. Combinando varios de los enfoques orientados a redes de sensores, han definido una abstracción basada en tres componentes: reglas, hechos y funciones. La información de contexto se presenta en forma de hechos que se guardan en un repositorio y se procesan con reglas. Las reglas representan la información relevante del sistema en un lenguaje de alto nivel que tiene condiciones de *triggering* y acciones a realizar.

El motor de inferencia está en fase de prototipado, programado en Haskell y utiliza encaminamiento hacia delante para que la semántica sea parecida a la de los eventos. Las reglas también pueden llamar a funciones integradas en el *firmware* o el sistema operativo, lo cual resulta más eficiente. Es importante destacar que las reglas son locales a cada nodo y cada nodo tiene su propio motor de reglas. Los hechos también se usan como abstracción para el intercambio de información entre nodos, no distinguiendo entre hechos locales y remotos.

```
ruleset PingPong
rule button 150
<- exists {button}
-> retract {button}
-> define ping
-> send 0 15 {ping}
-> retract {ping}

rule ping 100
<- exists {ping}
-> retract {ping}
-> define pong
-> send 0 15 {pong}
-> retract {pong}
```

Figura 5.- Ejemplo de reglas utilizadas en FACTS.

4.2 Modelos cooperativos

El modelo cooperativo se basa en la idea de distribuir funciones a diferentes nodos y que a nivel general la red se comporte como esperamos, siendo capaz de llevar a cabo adaptaciones en base a pequeñas reglas o condiciones. Se puede considerar como un reparto inteligente de tareas sobre los nodos para llevar a cabo un objetivo común.

Entre los proyectos que utilizan y adaptan los mecanismos basados en reglas podemos citar ***GRA*** [22] el objetivo es definir de forma sencilla la configuración de los nodos y su comportamiento como grupo. Dichos comportamientos se definen con roles. En base a dichos roles cada nodo puede variar su comportamiento, cooperando con otros nodos, actualizando su código o adaptando su comportamiento.

Los roles principales que han definido se dividen en tres bloques:

- ***Cobertura***: definen si el nodo tiene que estar activo (ON) o no (OFF) en base a, por ejemplo, la batería que tiene o el número de nodos de alrededor.

- ***Clustering***: definen el comportamiento del nodo como elemento de la red, siendo CLUSTERHEAD, GATEWAY o SLAVE.

- ***Agregación de datos***: especifica si el nodo realizará agregación de datos y en qué condiciones, pudiendo ser SOURCE (fuente de datos), AGGREGATOR (agregador de datos) o SINK (consumidor).

Los roles que se asignan tienen asociadas una serie de condiciones bajo las cuales se activan en cada nodo, lo cual resulta útil para automatizar las acciones de evaluación de su estado, comparación con los vecinos y actuación en consecuencia.

La asignación de los roles se lleva a cabo mediante un algoritmo que se basa en analizar las condiciones localmente, propagando también las propiedades al resto de nodos vecinos. La reevaluación de las condiciones puede provocar un cambio de rol en los nodos. Debido a la naturaleza dinámica de la red, es necesario reevaluar cuando se produce un cambio. Los cambios que se contemplan son de tres tipos:

- Cambios en las propiedades del nodo (por ejemplo, la batería).
- Nuevos nodos que entran en la red.
- Ha ocurrido un fallo en algún nodo.

En resumen, mediante esta organización grupal y definición de roles se pueden automatizar las tareas genéricas siguientes:

1. Hacer accesible el estado de los nodos (sensores disponibles, resolución, batería y localización física). Para este fin cada nodo tiene un directorio de propiedades sobre los que implementa un interfaz de consulta.
2. En base a las propiedades de los nodos se definen los roles para configurar las tareas del entorno. Cada rol define un comportamiento en base a la red y a las propiedades que utiliza. Cada rol tiene un pequeño conjunto de reglas a evaluar. Un cambio en alguna condición puede requerir reasignar roles.
3. Es necesario un conjunto básico de servicios, como el enrutado de los mensajes, la gestión del tiempo, gestión de energía,…etc.

4.3 Modelos basados en razonamiento difuso

Los modelos basados en reglas tienen el principal inconveniente de que su lógica de actuación suele limitarse a reglas de negocio precisas, que pueden resultar problemáticas si los sensores son imprecisos o erróneos.

Una alternativa para intentar paliar estos efectos es utilizar técnicas de fusión de datos, pero suelen requerir de más potencia de computación y a veces utilizan información que en realidad no es práctica.

Los modelos de razonamiento difuso se presentan como una alternativa para el tratamiento de este tipo de problemas, sobre todo cuando la incertidumbre está presente en el sistema, lo cual, por otra parte, es una característica inherente de las redes de sensores.

Los FIS (Fuzzy Inference Systems) han probado su validez en otras áreas como la automoción o la medicina. Por ello y por sus características pueden ser aplicados con éxito en las redes de sensores. Las ventajas de los FIS se pueden resumir en los siguientes puntos:

- Son simples, por lo que se adaptan bien a las limitaciones del hardware.
- Toleran la imprecisión de los datos y resulta más fácil añadirles modelos de confianza sobre los mismos.
- Reducen el tiempo de desarrollo, ya que son más intuitivos.
- Son flexibles, se construyen a partir de conocimiento experto, pero se pueden cambiar y adaptar de forma automática.
- Son rápidos computacionalmente hablando y eficientes en cuanto a la sobrecarga de memoria.
- Combinados con redes neuronales se pueden diseñar sistemas adaptativos y con capacidad de aprender de los resultados de su funcionamiento.

A pesar de las aparentes ventajas que puede proporcionar un modelo difuso y de la antigüedad de la teoría correspondiente, el uso de modelos difusos para redes de sensores es todavía muy reciente y pocos son los proyectos que lo utilizan.

D-FLER [30]: D-FLER implementa un motor de inferencia difuso y distribuido que utiliza reglas simples IF-THEN para detectar incendios.

El sistema está diseñado para recibir como entradas las observaciones individuales de los nodos, pero también las observaciones (previamente fuzzificadas) de los nodos que están a su alrededor. Con esta información cada nodo toma sus decisiones en base a las reglas que tenga programadas. Cada nodo puede a su vez valorar el nivel de consenso que hay en el vecindario mediante una serie de operadores. Sin embargo, lo que se difunde son las medidas fuzzificadas, no las decisiones que toma cada nodo, para no condicionar la confianza que se asigna a las medidas.

También contemplan cierto aprendizaje, ya que las informaciones que vienen de los nodos vecinos, así como las reglas, pueden tener asignado un peso que puede variar dinámicamente. Por ejemplo, si un sensor reporta muchas veces un valor diferente al de la mayoría, puede marcarse con pesos menores para que influya menos en las decisiones.

```
IF (Smoke is High) AND

   (Temp is Low) AND

   (most(SmokeNeighbour)) is High AND

   (most(TempNeighbour)) is High

THEN

FireDecision is High
```

Figura 6.- Ejemplo de reglas utilizadas en D-FLER.

5 CONCLUSIÓN

Las redes de sensores cuentan con la versatilidad y potencial suficiente para aplicarse con éxito en multitud de campos diferentes (medicina, agricultura, defensa). Sin embargo, debido a su actual inmadurez, todavía faltan aspectos por desarrollar que están siendo objeto de las investigaciones actuales en este campo. Los aspectos más desafiantes que se plantean a partir de ahora en este tipo de sistemas son los siguientes [3]:

- Hacer ***converger el middleware de redes de sensores con los sistemas sensibles al contexto***, permitiendo así que cada campo se beneficie mutuamente de sus respectivos avances. El área de *sensibilidad al contexto* cuenta con características que aplicadas a las redes de sensores pueden ser muy interesantes:
    - Las consultas y fusión de datos son más sofisticadas.
    - El razonamiento se apoya en ontologías, potenciando el uso de reglas y aprendizaje en las aplicaciones.
    - Dichas ontologías se basan en estándares (OWL, RDF,…), algo de lo que las redes de sensores carecen. Ya se están desarrollando algunos borradores de estándares para la información que proporcionan los sensores, como el SensorML [31] o el O&M (Observations and Measurements) [32].
- Actualmente existe una excesiva preocupación por las restricciones de los sistemas embebidos (energía, capacidad de procesamiento y memoria) y se dejan de lado otros aspectos incluso más importantes como es la heterogeneidad de los sistemas. Vemos como los sistemas embebidos evolucionan en estos aspectos (por ejemplo la evolución que ha sufrido la plataforma iMote de Intel, con su reciente versión 2), por lo que de cara al futuro esto no será un problema tan determinante. Hay que centrarse por lo tanto ***en conseguir plataformas más genéricas y que soporte multitud de sensores y dispositivos diferentes***.
- Los modelos de razonamiento que se utilicen tienen que incluir ***mecanismos que gestionen la incertidumbre y la imprecisión de los datos recogidos***. El concepto de calidad y frescura de los datos tiene que incluirse en los sistemas de redes de sensores.

- ***La cooperación y coordinación de los diferentes nodos***, no sólo a nivel de red, sino también a nivel de aplicación se convierte en un aspecto fundamental para que los objetivos de las aplicaciones se cumplan de manera eficaz. Cada nodo tiene que ser consciente no sólo de sus propios datos o medidas, sino también de los de los demás, ayudando así a tomar decisiones más consistentes.

- Otro aspecto importante es la necesidad de definir un ***framework de evaluación*** estándar para este tipo de sistemas que ayude en la validación de los mismos.

En resumen y a la espera de que nuevos avances en este campo permitan validar las ventajas o desventajas de las diferentes técnicas, podemos concluir que todos los modelos de inteligencia aplicados a redes de sensores deben incluir al menos de los siguientes puntos:

- Un ***modelo de representación común de los datos sensorizados*** por los diferentes sensores, que permita después trabajar con ellos de forma independiente al hardware.

- Un ***modelo de coordinación o cooperación entre los nodos a nivel de aplicación***. Este modelo bien puede contemplar un mecanismo de descubrimiento o intercambio de la información disponible en cada nodo.

- Un ***mecanismo de razonamiento adaptado en los nodos*** que permita detectar situaciones concretas, agregar datos o llevar a cabo pequeñas acciones con el objetivo de optimizar el comportamiento de la red y su aplicabilidad. El modelo más utilizado, por su adecuación a las características de funcionamiento de las redes es el basado en reglas.

- Un ***mecanismo de tratamiento de la incertidumbre***. Los datos que proporcionan los sensores en un entorno heterogéneo pueden ser imprecisos, erróneos o contradictorios unos con otros, por lo que hace falta aplicar alguna técnica que permita paliar los efectos de dichas características.

- En modelos más avanzados, sobre todo en el ámbito de la sensibilidad al contexto, se pueden empezar a considerar ***mecanismos de aprendizaje*** que haga que los sensores se adapten a las condiciones del entorno aprendiendo del feedback que obtengan, tanto del resto de nodos como de las variables que monitoricen.

6 REFERENCIAS

[1] U. Bischoff, U. Bischoff, and G. Kortuem, "A State-Based Programming Model and System for Wireless Sensor Networks" in *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, 2007, pp. 261-266.

[2] S. Hadim and N. Mohamed, "Middleware: middleware challenges and approaches for wireless sensor networks," *Distributed Systems Online, IEEE,* vol. 7, 2006.

[3] H. Karen and R. Ricky, "A survey of middleware for sensor networks: state-of-the-art and future directions," in *Proceedings of the international workshop on Middleware for sensor networks* Melbourne, Australia: ACM, 2006.

[4] L. Philip and C. David, "Maté: a tiny virtual machine for sensor networks," in *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems* San Jose, California: ACM, 2002.

[5] B. Rimon, C. B. John, S. D. Daniel, D. Bowei, T. W. D. Kim, Z. Bing, G. Emin, and S. n, "On the need for system-level support for ad hoc and sensor networks," *SIGOPS Oper. Syst. Rev.,* vol. 36, pp. 1-5, 2002.

[6] L. Ting, M. S. Christopher, Z. Pei, and M. Margaret, "Implementing software on resource-constrained mobile sensors: experiences with Impala and ZebraNet," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services* Boston, MA, USA: ACM, 2004.

[7] B. Philippe, G. Johannes, and S. Praveen, "Towards Sensor Database Systems," in *Proceedings of the Second International Conference on Mobile Data Management*: Springer-Verlag, 2001.

[8] R. M. Samuel, J. F. Michael, M. H. Joseph, and H. Wei, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.,* vol. 30, pp. 122-173, 2005.

[9] S. Chavalit, J. Chaiporn, and S. Chien-Chung, "Sensor Information Networking Architecture," in *Proceedings of the 2000 International Workshop on Parallel Processing*: IEEE Computer Society, 2000.

[10] C. Carlo, G. Matteo, G. Marco, G. Alessandro, L. M. Amy, and P. Gian Pietro, "TinyLIME: Bridging Mobile and Sensor Networks through Middleware," in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*: IEEE Computer Society, 2005.

[11] W. B. Heinzelman, W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. A. P. M. A. Perillo, "Middleware to support sensor network applicationsMiddleware to support sensor network applications," *Network, IEEE,* vol. 18, pp. 6-14, 2004.

[12] S. Eduardo, G. Germano, es, V. Glauco, V. Mardoqueu, R. Nelson, and F. Carlos, "A message-oriented middleware for sensor networks," in *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing* Toronto, Ontario, Canada: ACM, 2004.

[13] E. Yoneki and J. Bacon, "Unified Semantics for Event Correlation over Time and Space in Hybrid Network Environments," in *IFIP International Conference on Cooperative Information Systems (CoopIS'05)* Cyprus, 2005.

[14] S. Li, Y. Lin, S. H. Son, J. A. Stankovic, and Y. Wei, "Event Detection Services Using Data Service Middleware in Distributed Sensor Networks: Wireless Sensor Networks (Guest Editors: Yuh-Shyan Chen, Yu-Chee Tseng, Ying Zhang, Feng Zhao),"

Telecommunication Systems, vol. 26, pp. 351-368, 2004.

[15] R. Gummadi, O. Gnawali, and R. Govindan, "Macro-programming Wireless Sensor Networks Using Kairos," in *Distributed Computing in Sensor Systems*, 2005, pp. 126-140.

[16] N. Ryan and W. Matt, "Region streams: functional macroprogramming for sensor networks," in *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004* Toronto, Canada: ACM, 2004.

[17] B. Amol, K. P. Viktor, R. Jim, and L. Daniel, "The Abstract Task Graph: a methodology for architecture-independent programming of networked sensor systems," in *Proceedings of the 2005 workshop on End-to-end, sense-and-respond systems, applications and services* Seattle, Washington: USENIX Association, 2005.

[18] W. Kamin, Z. Feng, and L. Jie, "Automatic programming with semantic streams," in *Proceedings of the 3rd international conference on Embedded networked sensor systems* San Diego, California, USA: ACM, 2005.

[19] W. Matt and M. Geoff, "Programming sensor networks using abstract regions," in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1* San Francisco, California: USENIX Association, 2004.

[20] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru, and A. Wood, "EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks," in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*: IEEE Computer Society, 2004.

[21] M. Samuel, J. F. Michael, M. H. Joseph, and H. Wei, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.,* vol. 36, pp. 131-146, 2002.

[22] R. Kay, mer, F. Christian, J. Pedro, Marr, and B. Christian, "Generic role assignment for wireless sensor networks," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop* Leuven, Belgium: ACM, 2004.

[23] Q. Han and Venkatasubramanian, "Autosec: An integrated middleware framework for dynamic service brokering," in *IEEE Distributed Systems Online*, 2001.

[24] C. L. Fok, C. L. Fok, G. C. Roman, and C. Lu, "Mobile agent middleware for sensor networks: an application case study" in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 382-387.

[25] L. St Ville, L. St Ville, and P. Dickman, "Garnet: a middleware architecture for distributing data streams originating in wireless sensor networks" in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, 2003, pp. 235-240.

[26] K. Taylor and A. Ayyagari, "Research Topics in Semantic Sensor Networks," *Proceedings of the Semantic Sensor Network Workshop,* p. Preface, 2006.

[27] T. Terada, M. Tsukamoto, K. Hayakawa, T. Yoshihisa, Y. Kishino, A. Kashitani, and S. Nishio, "Ubiquitous Chip: A Rule-Based I/O Control Device for Ubiquitous Computing," in *Pervasive Computing*, 2004, pp. 238-253.

[28] M. Strohbach, H. Gellersen, G. Kortuem, and C. Kray, "Cooperative artefacts: Assessing real world situations with embedded technology," 2004.

[29] K. Terfloth, K. Terfloth, G. Wittenburg, and J. Schiller, "FACTS; A Rule-based Middleware Architecture for Wireless Sensor Networks" in *Communication System*

Software and Middleware, 2006. Comsware 2006. First International Conference on, 2006, pp. 1-8.

[30] M. Marin-Perianu and P. J. M. Havinga, "D-FLER - A Distributed Fuzzy Logic Engine for Rule-Based Wireless Sensor Networks," Germany: Springer Verlag, 2007, pp. 86-101.

[31] O. G. Consortium, "SensorML V1.0," 2007.

[32] O. G. Consortium, "Observations and Measurements," 2006.