International Journal on Artificial Intelligence Tools


# Sensor Web Interaction

GREGORY M. P. O'HARE
*CLARITY: The Centre for Sensor Web Technologies,*
*School of Computer Science and Informatics,*
*University College Dublin, Belfield, Dublin 4, Ireland*
*gregory.ohare@ucd.ie*

CONOR MULDOON
*School of Computer Science and Informatics,*
*University College Dublin, Belfield, Dublin 4, Ireland*
*conor.muldoon@ucd.ie*

MICHAEL J. O'GRADY
*CLARITY: The Centre for Sensor Web Technologies,*
*School of Computer Science and Informatics,*
*University College Dublin, Belfield, Dublin 4, Ireland*
*michael.j.ogrady@ucd.ie*

REM W. COLLIER
*CLARITY: The Centre for Sensor Web Technologies,*
*School of Computer Science and Informatics,*
*University College Dublin, Belfield, Dublin 4, Ireland*
*rem.collier@ucd.ie*

OLGA MURDOCH
*CLARITY: The Centre for Sensor Web Technologies,*
*School of Computer Science and Informatics,*
*University College Dublin, Belfield, Dublin 4, Ireland*
*olga.murdoch@ucdconnect.ie*

DOMINIC CARR
*CLARITY: The Centre for Sensor Web Technologies,*
*School of Computer Science and Informatics,*
*University College Dublin, Belfield, Dublin 4, Ireland*
*dominic.carr@ucdconnect.ie*

Ubiquitous sensing fuses the concepts of intelligent systems with ubiquitous computing in the development of novel sensor web applications, whereby the interaction of multiple disparate autonomous artefacts is a key requirement. In this paper, we present SIXTH, which is a middleware infrastructure for Ubiquitous Sensing that facilitates, and supports, the development and deployment of Sensor Web applications. SIXTH has been designed to be extensible, with provisions for user definable data retention policies, custom sensor data representations, and custom sensor node representations, whilst still

providing a rich set of default behaviours. Within SIXTH, support is provided for the development and interaction of applications that incorporate both physical and cyber (virtual server side) sensors. With a view to supporting intelligent, in network, interaction policies, whereby sensor nodes must negotiate and coordinate their behaviour, the system has been designed to operate in conjunction with Agent Factory Micro Edition (AFME). AFME is a minimised footprint intelligent agent platform designed for resource constrained devices. It is based on the standard Agent Factory platform, which was developed for desktop machines, and is representative of a class of agent systems, which are referred to as Agent Oriented Programming frameworks. The paper discusses a ubiquitous mapping application that was developed using the middleware.

*Keywords*: Sensor Web, Wireless Sensor Networks, Multi-Agent Systems

## 1. Introduction

Sensor Web technologies represent a key enabler of pervasive computing. Though the original pervasive computing vision was originally proposed over 20 years ago, realising a sensor fabric that would conform to the exacting requirements of miniaturisation, power-efficiency, and robustness amongst others, has proved a formidable technical challenge, and one that remains tantalisingly beyond the horizon in a number of aspects. The Sensor Web offers a construct by which this sensor fabric can be conceptualised, realised, and accessed. A conventional view of the Sensor Web focuses on its potential as an integrated extension of geospatial sensor networks [30,3]. Though a significant development in its own right, and one that is increasingly being manifested in practice at present, this view is narrowly focused and circumscribes what the sensor web could become. For the purposes of this discussion, it is envisaged that the sensor web will provide a shared uniform construct for accessing sensor data of all hues, in much the same way as the standard WWW provides a ubiquitous interface for multimedia documents and other dynamic data sources from all over the world.

Supporting interaction with the sensor web is essential; however, how best to enable interaction remains to be seen. Ambient Intelligence (AmI) represents a more human-centric interpretation of pervasive computing [21,1]. Specifically, it acknowledges the need for supporting interaction, and proposes the adoption of Intelligent User Interfaces (IUIs) [11] as the means by which this is achieved. The manner wherein the sensor layer and application layer interact, however, is not defined. Indeed, the lack of an agreed approach, in terms of standardised protocols and ontologies, for enabling such interaction, is a key inhibitor of pervasive access to the sensor web. Likewise, the potential of IUIs and ultimately, AmI, is seriously compromised. In this paper, a framework for bridging this gap between applications and the sensor web is proposed.

The remainder of this paper is organised as follows. This is followed, in Section 3, with a discussion on various interaction modalities of the sensor web. Section 4 provides an overview of the SIXTH middleware architecture. Embedded Intelligence is delivered in SIXTH through its interaction with AFME. An overview of AFME is provided in Section 5. A discussion of a SIXTH application, namely Ubiquitous

Mapping, is provided in Section 6. In Section 7, we discuss potential directions for future research and give some concluding remarks.

## 2. Challenges in Delivering the Sensor Web

A number of key challenges exist that need to be addressed before the vision of the Sensor Web can be realised. These include:

- *Deliver Cyber-Physical Systems*: It is necessary that the sensor web embraces the richness of the term sensor providing a seamless topology of sensors be they physical or web-based [12]. Sensor networks will be comprised of a diffuse range of sensors including physically deployed sensors, web based sensors like those of RSS feeds or twitter streams, historical data akin to linked data together with opportunistic crowd-sourced data. The challenge is to weave this into a unified collage of sensor data performing data fusion and data enrichment. Such heterogeneity poses considerable difficulties.
- *Introduce Intelligence: The Autonomic Sensor Network*: Sensor Web nodes operate as part of a collective and as such the activities of one sensor necessarily has an impact on those around it. Sensor Networks must operate not as a collection of individual sensors but rather as a collective which works collaboratively and in concert in the delivery of defined Quality of Service constraints. Autonomic Sensor networks [15] make provision for opportunistic collaboration between sensors. One approach to realising such functionality has been through the use of embedded agents [20].
- *Preserving System Longevity*: Invariably, the lifetime of a physical sensor is dictated by the battery power source. While this is not a constraint for web based sensors or higher end physical sensors with a dedicated power supply it nevertheless compromises those computational tasks that are undertaken on the sensors themselves. As such an energy aware deductive regime needs to be put in place.
- *Support Sensor (Re)tasking* : While capture and harvesting of sensed data is crucial, so to is the possibility to task or retask the sensor network. The Sensor Web must support bi-directional communication. This communication must offer more than the mere passive collection of data. In the context of a network that needs to conserve energy consumption and in so doing may utilise intelligent algorithms to dynamically adapt sensor(s) operation by for example degrading sampling frequency or by increasing the rigour of reporting thresholds.
- *Accommodate Co-existing Applications*: The Sensor Web infrastructure is such that it must represent and deliver a single infrastructure, which supports a potentially infinite set of users and associated applications. This Sensor Web would need to offer core functionality for sensor interaction exhibiting resonances with the manner in which the internet offers a core enabling infrastructure for data access and information sharing.

- *Facilitate Ambient Interaction*: The ubiquitous nature of the Sensor Web will demand that interaction with it is *ambient, intuitive*, and *non intrusive* in the conduct of our everyday lives. Interaction will demand support for: sensor discovery, interoperability, personalisation of information filters, management of protocol diversity through standardisation adoption. It is this latter challenge that forms the focus of this paper.

## 3. Sensor Web Interaction Modalities

The concept of the Sensor Web envisages a complex, spatially-distributed, heterogeneous, networked infrastructure, yet one that is inherently singular in how it is perceived, accessed, and controlled. In this paper, the sensor web is viewed of as comprising of a collection of subnets, each designed for a specialised domain; for example, surveillance or environmental monitoring. Interaction within the Sensor Web is primarily governed by the architecture of its constituent subnets. Many WSN applications are being realised through a suite of technologies that lack interoperability and support diverse interaction modalities. As noted by Broering, et al.[4], a key contributor to this situation is that both the Sensor Web and sensor networks remain distinct layers, each comprising distinct islands of technologies. Whilst there have been efforts towards realising the Sensor Web through standard WWW technologies and protocols, sensor network applications, typically, adopt different protocols and have different semantics, many standardised but some propriety. As an example of this standardisation effort, the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) initiative [5] has developed a suite of specifications for sensor models and interfaces, as well as for web services. In practice, most sensor platforms adhere to popular communication protocols, including Zigbee and Bluetooth, that are not interoperable. This demands that software or hardware bridges [2] be constructed for new sensor network deployments or new services being deployed on existing sensor network installations. This incompatibility between layers represents a significant constraint on sensor network interoperability and also restricts how interaction with the network can occur.

Presently, the predominant approach to accessing WSN data has been one of centralised access and visualisation. Data is sensed, processed, and tagged with appropriate metadata prior to be being stored in a repository for subsequent access. The classic web portal construct has been adopted as the primary interface for such repositories in many instances. SensorMap [22] and Sensorpedia [9] represent two exemplar cases of this category of approach. The public can upload, query, and access data as desired; in some cases, they can even register their own sensors - Weather Underground[a], for example, allows registered public members to provide data streams from their own Personal Weather Stations (PWSs). In essence, sensor portals support conventional interaction modalities.

[a]http://www.wunderground.com

In mobile computing scenarios, interaction with data, provided it is a accessible via an wireless internet connection , is relatively straightforward. As the geographic position of the user is often available, the possibility for accessing data from sensors in the user's immediate vicinity provides a useful filter for minimising the search space. Foerster, et al.[7] demonstrate a methodology for enabling remote access to sensor data, using air quality as a case study. A more challenging scenario arises if interaction with sensor subnets, and individual sensor platforms, is required. A standardised approach for such interaction is not yet available, though there have been prototypes demonstrated in the literature, for example, Tricorder [13] and the Sensor Browser [31].

The lack of interoperability between the physical sensor and sensor web layers represents a key deficiency in many sensor web application scenarios, where internationally accepted standards exist for both layers, but where there are no standards that define how the different layers should interoperate. One proposed solution to this problem is the construction of an intermediate layer, an approach adopted by SensorBus [4]. In essence, this represents a middleware solution that harnesses Twitter as the enabling technology. This represents a viable approach for ubiquitous data collection, but assumes that the sensor network is an instantiation of the Web of Things (WoT), a similar approach is adopted by SemSense [16]. The WoT assumption, however, is not valid for the majority of sensor network applications. In the following section, a middleware framework for ubiquitous sensing that is capable of operating in conjunction with an intelligent agent platform is proposed as a means of addressing the physical network - sensor web divide.

## 4. Enabling Sensor Web Interaction

In this section, we describe the SIXTH sensor web middleware. SIXTH combines physical and cyber sensing technologies within a single unified framework, which promotes a consistent view of Sensor Web resources. The primary goal of SIXTH is to support the development and deployment of ubiquitous sensing applications. SIXTH enables dynamic re-tasking of sensors to suit dynamic application demands. The middleware has been designed as a distributed system, whereby various components of the system are distributed over a network, but are accessed as though operating locally. Figure 1 provides an overview of the SIXTH architecture.

Within SIXTH, a common core uniform representation is provided for both sensor nodes and sensor data, which enables applications that use the sensor data and nodes to operate consistently and transparently across different node or data sources, without the need to modify the code of the user applications. The uniform representation can be viewed as the lowest common denominator of the functionality and attributes of all sensor network applications that are supported by the middleware.

SIXTH is extensible and application developers build upon, and extend, the common core in the development of the requisite application specific functionality.

To this end, SIXTH supports the provision of user definable data retention policies, custom sensor data representations, and custom sensor node representations.

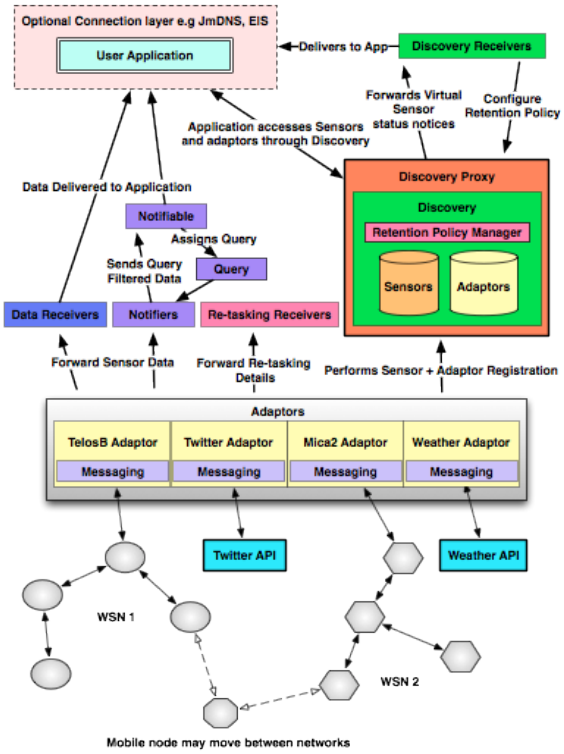


Fig. 1. SIXTH architecture

SIXTH is built on top of the Open Service Gateway initiative[28] component framework. OSGi facilitates the creation of dynamic, modular service platforms. A high level of fault tolerance is ensured as the failure or removal of one unit of functionality does not effect the rest of the system. Bundles can be installed, started, stopped, updated and uninstalled without requiring a restart of the whole system. SIXTH facilitates the collection of sensor data from heterogeneous sources using adapters. An adaptor provides a means to collect sensor data from a producer or intermediaries. Adaptors can be injected into a SIXTH instance through the use of OSGi.

Adaptors transform heterogeneous data formats into the SIXTH message interface format, facilitating data independence from sensor platform implementation. To limit the workload necessary to create an adaptor, a default implementation has been developed. This implementation handles discovery service registration and maintenance, the creation and management of virtual sensor objects, data access,

and sensor data forwarding. The discovery service is a key component of the SIXTH software architecture. It is responsible for providing notifications regarding sensor status to event subscribers. Such status changes include sensor arrivals and time-outs.

For a simple WSN adaptor, the software converts the received sensed data to the SIXTH data format and forwards it to the receivers. Adaptors also provide the functionality to reconfigure the network, for example, to change the sampling frequency. Within SIXTH, receivers are the simplest form of sensor data consumers. The receiver mechanism fulfils the subscriber role of the pub/sub model, wherein the sensors are the publishers.

SIXTH supports the development of both physical and cyber adaptors. Physical WSN adaptors provide a connection to a WSN through a gateway node. Messages received by the gateway node are translated to the correct format for SIXTH. For each sensing platform, custom code is written to allow for re-tasking in line with SIXTH formats. To date adaptors have been developed for WSN networks running on the Sun SPOT, TelosB, Mica2, and Tyndall physical sensing platforms. Cyber adaptors can be developed for software sensors that operate on server side machines. For example, a cyber sensor has been developed to monitor Twitter feeds.

The data retention policy, within SIXTH, enables autonomous control over the retention/removal of stored data. Primarily, this concerns the sensor data objects stored in virtual sensors. Some user applications implement their own data retention policy to manage the holding of data in line with their own needs. For instance, some applications retain GPS data for much longer than temperature data. The default policy employed works via a query on the time stamp of the data, removing data older than one minute.

Programming sensor networks in imperative languages is a time consuming and complex task. Furthermore, many potential users of sensor networks will not be from a Computer Science background and will not have knowledge of procedural programming. To remedy the situation, SIXTH enables networks to be programmed using a declarative language with a SQL-like syntax. Users or applications issue queries to the network and in order to realise these queries, the system determines efficient implementation plans on a per query basis. The system draws from prior work on acquisitional query processing in sensor networks [14].

## 5. Agent Factory Micro Edition for Ubiquitous Sensing

The SIXTH middleware infrastructure has been designed to operate in conjunction with Agent Factory Micro Edition (AFME) [20,18]. AFME provides SIXTH with a suite of pre-existing algorithms for the management and control of WSN applications, such as coordinated intelligent power management [29]. In this section, we provide an overview of AFME and discuss some agent platform services that enable it to operate in sensor network applications.

Ubiquitous Sensing fuses the concepts of intelligent systems with ubiquitous

computing in the development of intelligent electronic infrastructures that sense the environment and act in a proactive manner in delivering novel applications that go well beyond the mandate of the desktop computing era. Such environments are dynamic and, as such, the system must be capable of dealing with uncertain and inaccurate information, whereby the environment, along with user's requirements, change over time. Intelligent agents offer an attractive metaphor for dealing with uncertainty and change in that agents do not commit to a particular course of action forever, but revise their commitments over time as circumstances change [8]. AFME is an intelligent agent framework, which has been specifically designed for use with resource constrained ubiquitous devices. It is based on a declarative agent programming language, which, in a similar vein to other intelligent agent platforms, is used in conjunction with imperative components. These imperative components imbue agents with mechanisms that enable them to interact with their environment; agents perceive and act upon the environment through perceptors and actuators respectively[b].

Perceptors and actuators represent the interface between the agent and the environment and are implemented in Java. This interface acts as an enabling mechanism through which the agents are situated. AFME incorporates a variant of the Agent Factory Agent Programming Language (AFAPL) [6] specifically designed for resource constrained devices. AFAPL is a declarative language; it is based on a logical formalism of belief and commitment and forms part of the Agent Factory Framework [6,23], which is an open source collection of tools, platforms, and languages that support the development and deployment of multi-agent systems. Agent Factory and AFME are representative of a class of intelligent agent platforms, which are referred to as Agent Oriented Programming frameworks. Agent Oriented Programming stems from Shoham's seminal work in this area [26], whereby agents communicate using speech acts, such as inform and request. In its latest incarnation, the Agent Factory Framework has been restructured to facilitate the deployment of applications that employ a diverse range of agent architectures. As such, the framework has become an enabling middleware layer that can be extended and adapted for different application domains. The framework is broadly split into two parts:

- support for deploying agents on laptops, desktops, and servers;
- support for deploying agents on constrained devices such as mobile phones and WSN nodes.

AFME represents the latter, whereas the former is delivered through Agent Factory Standard Edition (AFSE). In the remainder of this paper, we shall only consider AFME. In AFME, commitment rules that define the conditions under which commitments are adopted are used to encode an agent's behaviour. A commitment represents an intended course of action. Commitments are useful because they have

[b]The word perceptor is used rather than sensor to distinguish the software component from hardware sensors.

a stabilising effect on system behaviour. Consider the situation in which an agent is situated in a highly dynamic environment and an event occurs. What should an agent do? replan or continue operating? If the agent always replans, the system will become unstable; but if it never replans, the system will not be adaptive [25]. In AFME, agents commit to actions, but they revise their commitments at various points throughout execution. If a commitment is no longer relevant, it is dropped. When a commitment is adopted, it represents either a primitive action or a plan. Plans are ultimately executed as a series of primitive actions. When a primitive action is executed, an actuator is fired. The following is an example of an AFME commitment rule:

```
message(request, ?sender, data), informTime(?t), data(?d) > ?sender, ?t,
                    true, inform(?sender, data(?d));
```

In the above rule, the terms to the left of the implication (the > symbol) represent beliefs and form a belief sentence; those to the right represent arguments to a commitment. The arguments to a commitment represent to whom the commitment is made, the time at which the plan or primitive action of the commitment should be executed, the maintenance condition of the commitment, and either a trigger for a primitive action or a plan operator, which incorporates a number of primitive action triggers. The truth of a belief sentence is evaluated using the agents belief set. The result of this query process is either failure, in which case the belief sentence is evaluated to false, or to a set of bindings in which belief sentence is evaluated to true. In AFAPL, the ? symbol represents a variable. In this example, if the agent has adopted a belief that it has received a message from another agent to send data and the agent has beliefs in relation to the data and the transmission time, it adopts a commitment to send data back to the requesting agent. At an imperative level, a preceptor monitors the message transport service, which contains a server thread that receives incoming messages. Once a message is received, it is added to a buffer in the service. Subsequently, the perceptor adds a belief to the agent's belief set. The interpreter periodically evaluates the belief set. If the conditions for a commitment are satisfied (that is, all of the beliefs prior to the > symbol in the rule have been adopted), either a plan is executed to achieve the commitment or a primitive action or actuator is fired[c]. When an actuator is created, it is associated with a symbolic trigger. In this case, an inform actuator, written in Java, is associated with the trigger string inform(?sender,data(?d)). Once the agent has adopted the commitment, it will wait until time ?t and then pass the arguments ?sender and data(?d) to the actuator. The actuator then executes the imperative code for informing the sender of the request of the data value. Structuring agents in this manner is useful in that it enables their behaviour to be altered at a symbolic level rather than having to modify the imperative code.

[c]In this paper, we shall only consider primitive actions.

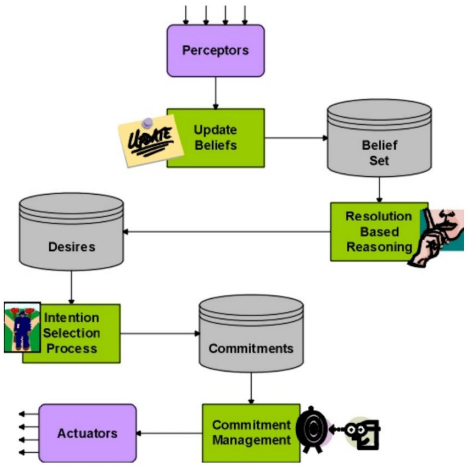


Fig. 2. AFME Control Process

Figure 2 illustrates the AFME control process. In AFME, agents follow a sense-deliberate-act process. Initially, perceptors are fired and agents update their belief set. Based on their current set of beliefs, resolution based reasoning is performed to indentify a set of desired states. The agents desired states act as input to an intention selection process[d]. Once intentions have been identified, a commitment management process commences. Finally, depending on the nature of the commitments adopted, various actuators are fired[e].

In order to facilitate communication between AFME agents in WSN applications, a Wireless Message Transport Service has been developed that can be controlled and monitored through the use of actuators and perceptors. The Wireless Message Transport Sevice has been designed forthe Java-enabled Sun SPOT mote, which uses the Squawk Java Virtual Machine[27] (JVM). The Sun SPOT motes communicate using the IEEE 802.15.4 standard. The Wireless Message Transport Service facilitates peer to peer communication between agents and is based on the Sun SPOT radiogram protocol rather than TCP/IP, which is used for agents deployed on mobile phones or PDAs that have a 3G or GPRS connection. The radiogram protocol uses datagrams to facilitate communication between motes. With the Sun SPOT radiogram protocol, the connections operating over a single hop have different semantics to those operating over multiple hops. This is due to a performance optimisation. When datagrams are sent over more than one hop, there are no guarantees about delivery or ordering. In such cases, datagrams will sometimes be silently lost, be delivered more than once, and be delivered out of sequence. When datagrams

[d]In situations where resource bounded reasoning is not being used and the utilities of desired states are not specified, an agent's desired states will be its intentions.
[e]Additional features are supported in Agent Factory and AFME[17,19], such as dynamic role adoption, but go beyond the scope of this paper.

are sent over a single hop, they will not be silently lost or delivered out of sequence, but sometimes they will be delivered more than once.

The radiogram protocol operates in a client server manner. When the message transport service is created, a server thread is created to receive incoming messages. When a message is received it is added to an internal buffer within the service. An agent will subsequently perceive messages through the use of a perceptor. When an agent is sending a message, it attempts to open a datagram client connection. The datagram server connection must be open at the destination. With datagrams a connection opened with a particular address can only be used to send to that address. The wireless message transport service only allows agents to send messages of a maximum size. If the content of the message is greater than the limit, it is first split into a number of sub messages within an actuator and each sub message is then sent using the message transport service. When all sub messages have been received, the entire message is reconstructed within a perceptor and then added to the belief set of the agent.

One of the core features of AFME is the support for agent migration. For the Sun SPOT platform, this support is delivered through the AFME Wireless Migration Service. Agent migration is often classified as either strong or weak. This classification is related to the amount of information transferred when an agent moves. The more information transferred the stronger the mobility. Within AFME, support is only provided for the transfer of the agent's mental state. Any classes required by the agent must already be present at the destination. The reason for this is that the Java platform AFME has been developed for, namely the Java Micro Edition (JME) Constrained Limited Device Configuration (CLDC), does not contain an API for dynamically loading foreign classes. Only classes contained, and preverified, in the deployed Java ARchive (JAR) file can be dynamically loaded through the use of the Class.forName method. This is also one of the reasons why component deployment frameworks, such as OSGi, cannot be used for CLDC applications. In the Squawk JVM, which operates on Sun SPOTs, it is possible to migrate an application to another Squawk enabled device. Squawk implements an isolate mechanism, which can be used can for a type of code migration. Isolate migration is not used in AFME. The reason for this is that isolate migration is dependent on internal details of the JVM and is therefore not really platform independent in the sense that an isolate can only be transferred to another Squawk JVM. It could not be used to transfer an application to a C or C++ CLDC JVM written for a mobile phone JVMs, for instance. Additionally, with isolates, it is necessary to migrate the entire application or platform, rather than just a single agent.

This AFME Migration Service uses both the Sun SPOT radiogram protocol and the radiostream protocol. The radiostream protocol operates in a similar manner to TCP/IP sockets. It provides reliable, buffered, stream-based communication between motes. This, however, comes at a cost in terms of power usage. The reason this approach is adopted for agent migration is that we wish to ensure that agent does not become corrupt or lost due to the migration process. If a message is lost or

corrupt, the system can recover by resending the message. If an agent is lost or corrupt, it cannot be recovered without duplication or redundancy, which would also use up resources and would become complex to manage as agent artefacts would be scattered throughout the network.

The problem with the radiostream protocol, however, is that both the target platform and the source platform must know each others MAC address before a connection can be established. That is, it does not adopt a client server approach or operate in a similar manner to the radiogram protocol. In a dynamic mobile agent setting, it is unlikely that the addresses of the platforms of all source agents will be known *a priori* at compile time. To get around this problem, when an agent wishes to migrate to a particular platform, initial communication is facilitated through the use of datagrams. Using datagrams, the platforms exchange address and port information and subsequently construct a radiostream. Once the radiostream is established, the agent is transferred through the reliable connection and then terminiated at the source. Subsequently, the stream connection is closed. At the destination, the platform creates and starts the agent.

Through the use of AFME, SIXTH developers gain access to technology, such as mobile agents and distributed optimisation algorithms [29], which would not otherwise be readily be available, or would have to be developed from scratch.

## 6. Case Study

This section provides a discussion of an application that incorporates the use of a diverse set of cyber and physical sensors. Specifically, it discusses the ubiquitous mapping application, which displays readings from sensors that were configured by a user through a graphical user interface (see Figure 3). The interface allows a user to (1) select the types of environment/network they want to monitor and (2) configure sensors in the selected environment. The ubiquitous mapping application sends requests to SIXTH, which in turn creates instances of cyber sensors, along with adaptors for physical sensors, that perform the necessary work to stream information to the application, providing near real time data, which is displayed on a map.

### 6.1. *Ubiquitous mapping: An application for mapping and configuring diverse cyber and physical sensors*

Ubiquitous mapping provides a client side user interface for the production, configuration, and visualisation of diverse cyber, and physical, sensors in real time. The application uses Open Street Map [10] to provide a visualisation of the world. The application been built as an Eclipse plugin. In order to ensure a diverse data set, three different types of scenario were considered: event driven, user driven, and crowdsourced physically sensed data.

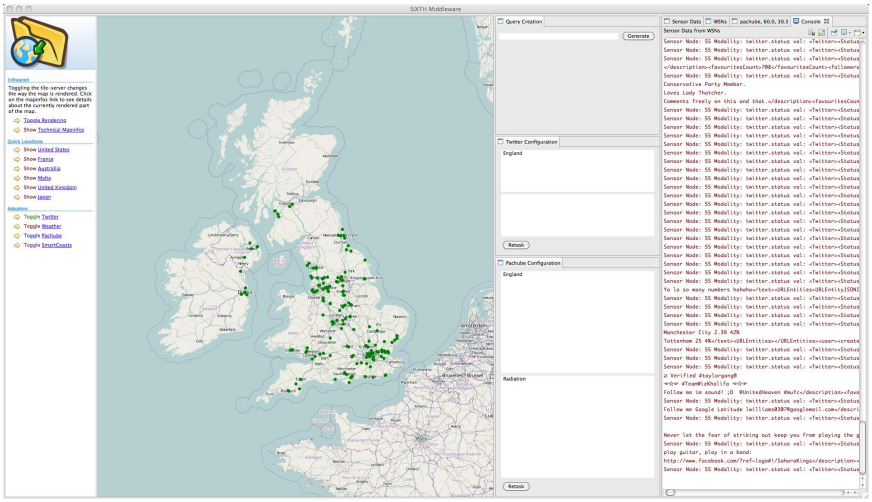

Fig. 3. Ubiquitous mapping interface

### 6.2. *Event Driven*

Event driven systems often monitor phenomena that are difficult to predict, or which, at times, exhibit sporadic behaviour. In this scenario, we consider weather sources. With weather sources, information is often updated at regular intervals of a predefined length. When an unsual weather event occurs, however, the sources provide updates more frequently as events occur and/or are predicted in real time. When monitoring a cyber environment that produces weather information, a sensor can be produced for every location that the source allows to be queried. This means that an application can have a large number of sensors running to monitor a large spatial region and it will, thus, be necessary to filter responses from weather sources to identify weather events that match a particular set of requirements. By combining location based queries with keyword filtering, it is possible to produce weather cyber sensors that monitor a web-based weather source by events and/or location. Such cyber sensors often produce useful information for context driven applications and services [24].

The particular weather source considered in this application is the Yahoo! Weather API, which a SIXTH adaptor has been developed for. The API allows RSS requests to be made for geographical locations and returns XML responses containing metadata elements describing forecast information for the locations. In the weather adaptor, functionality is provided that allows queries to be made based on a predefined list of weather keywords. Functionality is provided to enable locations to be configured in terms of their lat/long positions, the location name, or the Where On Earth IDentifier (WOEID), which is a reference assigned by Yahoo! to identify features on Earth. The adaptor manages the conversion of the former two to the latter when required. Functionality has also been included that allows a user

to monitor weather for their current location by translating their IP address into an approximate geographical location.

Since the number of locations possible to sense weather is often quite large, the application makes uses of a database of known locations to display weather sensors on the map, prior to the configuration by a user. When a user selects a node on the map, a request is sent to SIXTH to produce a virtual sensor that monitors weather information at that location at a configurable frequency. The SIXTH adaptor then streams the relevant information to the client application making it viewable to the user.

### 6.3. *User Driven*

A SIXTH adaptor has been developed for microblogging website Twitter. Twitter provides a real time API for the querying and streaming of user generated data, i.e., tweets. The API provides functionality allowing queries to be made based on a number of attributes, including location and keywords. The streaming API, however, does not allow for the combination of location and keywords in a search (rather it responds with tweets that satisfy either constraint), but this has been accounted for in the SIXTH twitter adaptor, which allows queries by location, keyword, or a combination of both. This facilitates the production of virtual sensors that monitor the tweets provided by users in a specific location relating to a specific topic or event. Locations can be specified by users as names that SIXTH converts into bounding box coordinates required by the Twitter Streaming API.

When a new Twitter sensor is configured in the application, the SIXTH adaptor produces a virtual sensor that performs the required queries and filtering to retrieve the appropriate data. The data is then streamed back to the application, where nodes dynamically appear on the map in real time.

### 6.4. *Crowdsourced physical sensor data*

An Internet of Things cyber environment is as a web-based source for the publication and retrieval of personal sensor data. These environments provide APIs that facilitate the streaming of physical sensor data from personal deployments. These data streams can be made either public or private and are accessible to consumers via API's or HTTP requests. While the original source of data maintained in these environments is most often physical sensors, it is also the case that cyber sensor data is provided. An adaptor for this type of data source has been developed for SIXTH and incorporated in to the ubiquitous mapping application.

IoT cyber environments and physical sensor networks deployments differ in relation to the semantic enhancement of the data and access mechanisms. An IoT data provider sends the raw sensor data in tagged semi-structured (usually XML) feeds. Since the data can be acquired through a web-based interface, a SIXTH cyber adaptor manages API call limits, rather than low level connectivity issues inherent with SIXTH adaptors for direct physical sensor networks.

## 7. Future Work and Conclusion

This paper discussed the SIXTH middleware architecture, which is a cohesive framework that supports the development and deployment of Sensor Web applications. Specifically, SIXTH supports the development of applications that incorporate both cyber and physical sensors. It makes use of standard industrial technologies to ease deployment issues and facilitate interoperability. Applications in SIXTH are packaged as OSGi bundles. This enables cyber sensors and server side code to be deployed in a seamless manner. The SIXTH architecture has been designed to be capable of operating in conjunction with the AFME platform, a reduced footprint variant of the Agent Factory platform. Agent Factory and AFME are representative of a class of intelligent agent platforms, referred to as Agent Oriented Programming frameworks, whereby agents adopt beliefs and commitments and communicate using speech acts, such as request and inform. One of the core features of Agent Factory/AFME is agent migration, which enables agents to dynamically move to both cyber and physical sensor locations at run time. The paper discussed a ubiquitous mapping application that was developed using the middleware. The application enables a user to select the type of environment they wish to monitor and configure sensors in the selected environment in real time.

One of the problems with the current SIXTH architecture is that, since it is based around the OSGi framework, it cannot be deployed on Java-based CLDC WSN motes. At present AFME agents are capable of operating on the motes and communicating with SIXTH applications deployed on the base station. Since CLDC does not have an API for dynamically loading foreign classes, however, it is not possible to deploy imperative code to the motes at run time, only mobile agents. Another problem with deploying SIXTH on the motes is that it has dependencies on standard Java classes not contained in CLDC and, thus, the code must be ported for these types of device. Future work will investigate the potential of using over the air programming to enable applications to be dynamically deployed to the network, the porting SIXTH to CLDC, along with the development of an array of Sensor Web applications. Additionally, we will investigate the use of Bayesian Machine Learning algorithms, within SIXTH, with regard to making inferences with the uncertain and noisy data typical of sensor network sources.

## Acknowledgments

This work is supported by Science Foundation Ireland (SFI) under grant 07/CE/I1147. The authors would also like to acknowledge the support of the Irish Research Council for Science, Engineering and Technology and the European Commission Marie Curie Actions.

## References

1. Hamid Aghajan, Juan Carlos Augusto, and Ramon Lopez-Cozar Delgado. *Human-Centric Interfaces for Ambient Intelligence*. Academic Press, 2009.

2. P. Angove, M. O'Grady, J. Hayes, B. O'Flynn, G.M.P. O'Hare, and D. Diamond. A mobile gateway for remote interaction with wireless sensor networks. *IEEE Sensors Journal*, 11(12):3309 –3310, dec. 2011.
3. M. Botts, G. Percivall, C. Reed, and J. Davidson. Ogc® sensor web enablement: Overview and high level architecture. *GeoSensor networks*, pages 175–190, 2008.
4. Arne Broering, Theodor Foerster, Simon Jirka, and Carsten Priess. Sensor bus: an intermediary layer for linking geosensors and the sensor web. In *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application*, COM.Geo '10, pages 12:1–12:8, New York, NY, USA, 2010. ACM.
5. Arne Brring, Johannes Echterhoff, Simon Jirka, Ingo Simonis, Thomas Everding, Christoph Stasch, Steve Liang, and Rob Lemmens. New generation sensor web enablement. *Sensors*, 11(3):2652–2699, 2011.
6. Rem Collier, Gregory O'Hare, Terry Lowen, and Colm Rooney. Beyond prototyping in the factory of agents. In *Proceedings of the 3rd Central and Eastern European conference on Multi-agent systems*, CEEMAS'03, pages 383–393, Berlin, Heidelberg, 2003. Springer-Verlag.
7. Theodor Foerster, Daniel Nst, Arne Brring, and Simon Jirka. Discovering the sensor web through mobile applications. In Georg Gartner and Felix Ortag, editors, *Advances in Location-Based Services*, Lecture Notes in Geoinformation and Cartography, pages 211–224. Springer Berlin Heidelberg, 2012.
8. M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The belief-desire-intention model of agency. *Intelligent Agents V: Agents Theories, Architectures, and Languages*, pages 1–10, 1999.
9. B.L. Gorman, D.R. Resseguie, and C. Tomkins-Tinch. Sensorpedia: Information sharing across incompatible sensor systems. In *International Symposium on Collaborative Technologies and Systems (CTS '09)*, pages 448 –454, may 2009.
10. M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.
11. V.L. Jaquero, F. Montero, JP Molina, and P. Gonz´ lez. Intelligent user interfaces: Past, present and future. *Engineering the User Interface*, pages 1–12, 2009.
12. E.A. Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008.
13. Joshua Lifton, Manas Mittal, Michael Lapinski, and Joseph A. Paradiso. Tricorder: A mobile sensor network browser. In *Proceedings of the ACM CHI 2007 Conference - Mobile Spatial Interaction Workshop*, April 2007.
14. S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.
15. D. Marsh, R. Tynan, D. OKane, and G.M. P OHare. Autonomic wireless sensor networks. *Engineering Applications of Artificial Intelligence*, 17(7):741–748, 2004.
16. A. Moraru, D. Mladenic, M. Vucnik, M. Porcius, C. Fortuna, and M. Mohorcic. Exposing real world information for the web of things. In *Proceedings of the 8th International Workshop on Information Integration on the Web: in conjunction with WWW 2011*, page 6. ACM, 2011.
17. C. Muldoon, G.M.P. OHare, R.W. Collier, and M.J. OGrady. Towards pervasive intelligence: reflections on the evolution of the agent factory framework. *Multi-Agent Programming:*, pages 187–212, 2009.
18. C. Muldoon, G.M.P. O'Hare, M.J. O'Grady, and R. Tynan. Agent migration and communication in wsns. In *Ninth International Conference on Parallel and Distributed*

*Computing, Applications, and Technologies (PDCAT)*, pages 425–430. IEEE, 2008.
19. Conor Muldoon. *An agent framework for ubiquitous services*. Ph.d. dissertation, School of Computer Science and Informatics - University College Dublin (UCD), Dublin, Ireland, 2007.
20. Conor Muldoon, Gregory M. P. O'Hare, and John F. Bradley. Towards reflective mobile agents for resource-constrained mobile devices. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS'07, pages 141:1–141:3, New York, NY, USA, 2007. ACM.
21. Hideyuki Nakashima, Hamid Aghajan, and Juan Carlos Augusto. *Handbook of Ambient Intelligence and Smart Environments*. Springer Publishing Company, Incorporated, 1st edition, 2009.
22. S. Nath, J. Liu, and F. Zhao. Sensormap for wide-area sensor webs. *Computer*, 40(7):90–93, 2007.
23. GMP O'Hare, R. Collier, J. Conlon, and S. Abbas. Agent factory: An environment for constructing and visualising agent communities. In *Proc. AICS98*. Citeseer, 1998.
24. GMP O'Hare, PT O'Hare, and TD Lowen. Far and a way: Context sensitive service delivery through mobile lightweight pda hosted agents. In *Proceedings of 15th International Florida Artificial Intelligence (FLAIRS) Conference*, 2002.
25. A.S. Rao, M.P. Georgeff, et al. Bdi agents: From theory to practice. In *Proceedings of the first international conference on multi-agent systems (ICMAS-95)*, pages 312–319. San Francisco, 1995.
26. Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, March 1993.
27. D. Simon, C. Cifuentes, D. Cleal, J. Daniels, and D. White. Java on the bare metal of wireless sensor devices: the squawk java virtual machine. In *Proceedings of the 2nd international conference on Virtual execution environments*, pages 78–88. ACM, 2006.
28. The OSGi Alliance. OSGi service platform core specification, release 4.1, 2007. http://www.osgi.org/Specifications.
29. R. Tynan, C. Muldoon, G. O'Hare, and M. O'Grady. Coordinated intelligent power management and the heterogeneous sensing coverage problem. *The Computer Journal*, 54(3):490, 2011.
30. TL Van Zyl, I. Simonis, and G. McFerren. The sensor web: systems of sensor systems. *International Journal of Digital Earth*, 2(1):16–30, 2009.
31. J. Wan, G.M.P O'Hare., and M.J. O'Grady. Sensing the sensor web. In *Proceedings of the 10th IEEE Pervasive Computing and Communication Conference (Percom2012)*, March 2012.