Semantic Network Management for Next Generation Networks

Christopher J. Matheus, Aidan Boran
Bell Labs, Blanchardstown, Ireland

Dominic Carr
School of Computing, National College of Ireland, Dublin, Ireland

Rem W. Collier, Barnard Kroon, Olga Murdoch
David Lillis, Michael J. O'Grady, Gregory M. P. O'Hare
School of Computer Science, University College Dublin, Ireland

Abstract

To accommodate the proliferation of heterogeneous network models and protocols, the use of semantic technologies to enable an abstract treatment of networks is proposed. Network adapters are employed to lift network specific data into a semantic representation. Semantic reasoning integrates the disparate network models and protocols into a common data model by making intelligent inferences from low-level network and device details. Automatic discovery of new devices, monitoring of device state and invocation of device actions in a generic fashion that is agnostic of network types is enabled. A prototype system called SNoMAC is described that employs the proposed approach operating over UPnP, TR-069 and heterogeneous sensors. These sensors are integrated by means of a sensor middleware named SIXTH that augments the capabilities of SNoMAC to allow for intelligent management and configuration of a wide variety of sensor devices. A major benefit of this approach is that the addition of new models, protocols or sensor types merely involves the development of a new network adapter based on an ontology. Additionally, the semantic representation of the network and associated data allows for a variety of client interfaces to facilitate human input to the management and monitoring of the system.

Keywords: semantic computing, sensing web, network monitoring, home area networks

1 Introduction

By the year 2021, the Internet of Things (IoT) is expected to encompass 3.5 times as many connected devices as there are people on earth [1]. In addition to the sheer volume of devices, there is the added complexity of dealing with the unchecked proliferation of new network data models and protocols. To deal with these issues, network and active media applications will not only require high performance and scalability but will also need the means for quickly and dynamically evolving to accommodate the changing universe of devices. Doing this effectively necessitates a new approach for integrating network models and protocols that facilitates the intelligent management of devices across layers and at various levels of abstraction. This allows devices to be handled generically as collections while maintaining the specifics necessary to monitor and control them individually. This paper advocates an approach to solving this problem that leverages the benefits afforded by semantic web technologies, combined with the advantages of intelligent middleware. This automates much of the organisation and integration of heterogeneous devices and provides a platform upon which a wide variety of applications can be built.

Semantic web technologies enable the definition of formal data models called "ontologies" that provide a number of conceptual and computational benefits. This includes data model alignment, heterogeneous data integration, built-in data abstraction mechanisms, automated inferencing, dynamic meta-modelling and automatic consistency checking. These ontologies form the base upon which semantic tools such as SPARQL [2], SWRL [3] and OWL [4], or a hybrid of these, can make intelligent inferences about both devices themselves and their network topology. This is achieved by means of techniques such as inheritance and property reasoning, in addition to using chainable or transitive properties to infer entire network topologies from individual descriptions of internet-based devices and their connections. For systems involving heterogeneous network monitoring and control, such intelligent inference systems are crucial in capturing essential domain knowledge from the underlying ontologies.

Intelligent middleware is essential when a developer or service provider is tasked with dealing with a variety of heterogeneous devices. In addition to concentrating on those network-enabled devices that make up the IoT, other categories of devices provide sensing capabilities either as individual sensors, wireless sensor networks, legacy sensor networks or sensor-equipped mobile devices. An intelligent middleware automatically handles the discovery and management of such sensors, while providing a unified interface through which they can be configured and accessed. SIXTH is an intelligent middleware system for the Sensor Web that incorporates not only physical sensors but also provides access to web-based data sources through "cyber sensors" [5]. Embedded intelligence allows for the autonomic management of sensors to deal with issues such as power and bandwidth restrictions, and data management.

This paper provides an overview of the SNoMAC approach to semantic web management. It also describes a functioning prototype that detects, monitors and controls devices in a home area network that involves UPnP [6], TR069 [7] and sensors that are incorporated using SIXTH. This integration of SNoMAC and SIXTH results in a “system of systems” that enables the automatic discovery of the configuration, state and capability data of both internet-enabled devices and sensors. This then provides an intuitive interface through which a network of devices can be monitored and managed. In so doing, it removes the burden from users of configuring devices for use within the system. More generally, the SNoMAC architecture facilitates intelligent reasoning systems to co-operate at different levels, from in-network intelligence operating on individual network devices to intelligent data analysis and monitoring implemented within the middleware. Since the primary feature of SNoMAC is as a service producing semantic descriptions of networked devices and sensors upon which other applications can be developed, it also facilitates the integration of an intelligent management level in the future. This hybrid approach to the integration of various intelligence technologies has great potential in the area of network automation, for example for within a home area network. It also allows for the addition of other services such as intelligent user interfaces and personalisation and so facilitates the rapid development of human-centric computing applications. The work presented in this paper is an extension of that described in [8].

The primary novelty of the SNoMAC approach is the ability to “lift” device data into a semantic representation automatically. This facilitates more complex intelligent reasoning about devices in the system while requiring minimal user input. Unlike similar systems (discussed in Section 4), this does not depend on the prior existence of the semantics.

This paper is structured as follows. Section 2 begins by presenting an illustrative example of the SNoMAC system, in addition to outlining how it generates ontologies and how this can be leveraged by a variety of semantic technologies. Following this, Section 3 describes a further all-level prototype that has been created in the area of Home Area Network management. This prototype gives particular focus to how it integrates with the SIXTH sensor middleware platform to integrate both physical and cyber sensors into the system. The combination of SIXTH and SNoMAC offers a full platform from low-level device integration to a high-level user interface. A discussion of related work is presented in Section 4. Finally, this is followed in Section 5 by details of evaluations that have been carried out on the system’s components thus far, as well as future work directions. Section 6 concludes the paper.

2 SNoMAC Overview

SNoMAC is a research prototype from Bell Labs that demonstrates the use of semantic approaches to enable dynamic monitoring, analysis and control of heterogeneous network devices. It allows applications be written against SNoMAC’s formal semantic API, which removes the need to deal with lower-level details that may change as standards evolve and new standards are integrated into the system. To demonstrate this concept, Bell Labs worked with researchers at University College Dublin (UCD) to integrate SNoMAC with their SIXTH sensor middleware.

SNoMAC is designed with a number of specific aims in mind:

- Facilitate the automatic discovery of new devices that are contactable via a network.
- Allow for the configuration of the functionality of such devices by enabling the remote setting of state variables associated with them.
- Where these devices are capable of executing commands, route these from client applications to appropriate devices for execution.
- Perform basic network analysis to ascertain the topology and configuration of the network.

An overview of the SNoMAC concept and architecture is depicted in Figure 1. This architecture consists of a number of devices that are to be discovered and managed, the SNoMAC server itself, and a client interface that allows users to interact with the server in order to control and manage connected devices.

Network Adapters operate in a two-way fashion, to interface with the network to "lift" device data into a semantic representation before sending it to the server and also to invoke device-specific commands coming from the client via the server. They also have a responsibility to listen for user-defined events on the network. Separate network adaptors are required for each supported protocol. Currently, SNoMAC supports UPnP and TR069, in addition to interfacing with the SIXTH sensor middleware (discussed in detail in Section 3.1).

These network adapters interface with the SNoMAC server through two simple REST-based web service APIs supported by the server's Listener-Controller Interface. The API implemented by the network adapter is for processing requests issued by the server, such as to get/set state variables associated with devices (e.g. the rate at which data is sent to the server), or invoke commands on them. The other API is implemented by the server to receive pushed updates from the adapters as the devices in the network change. The information that is exchanged is in the form of RDF annotations based on a network-specific ontology that is aligned with the NetCore ontology (i.e. all network ontology classes and properties are sub-classes or sub-properties of entities defined in NetCore). The NetCore ontology is discussed in Section 2.1.

The SNoMAC server initiates connections with available network adapters by requesting information for all known devices and issuing a request to be informed of subsequent device changes (i.e. node additions, deletions and state updates). Information about devices is stored in the Semantic Layer which includes an RDF data store plus the means by which to intelligently infer new information about the devices based upon meta data stored in their corresponding ontologies.

A significant challenge that must be addressed is the heterogeneous nature of the devices that can be integrated into such a system. To overcome this, the Data Lifting Layer converts data from each device to RDF using an XML

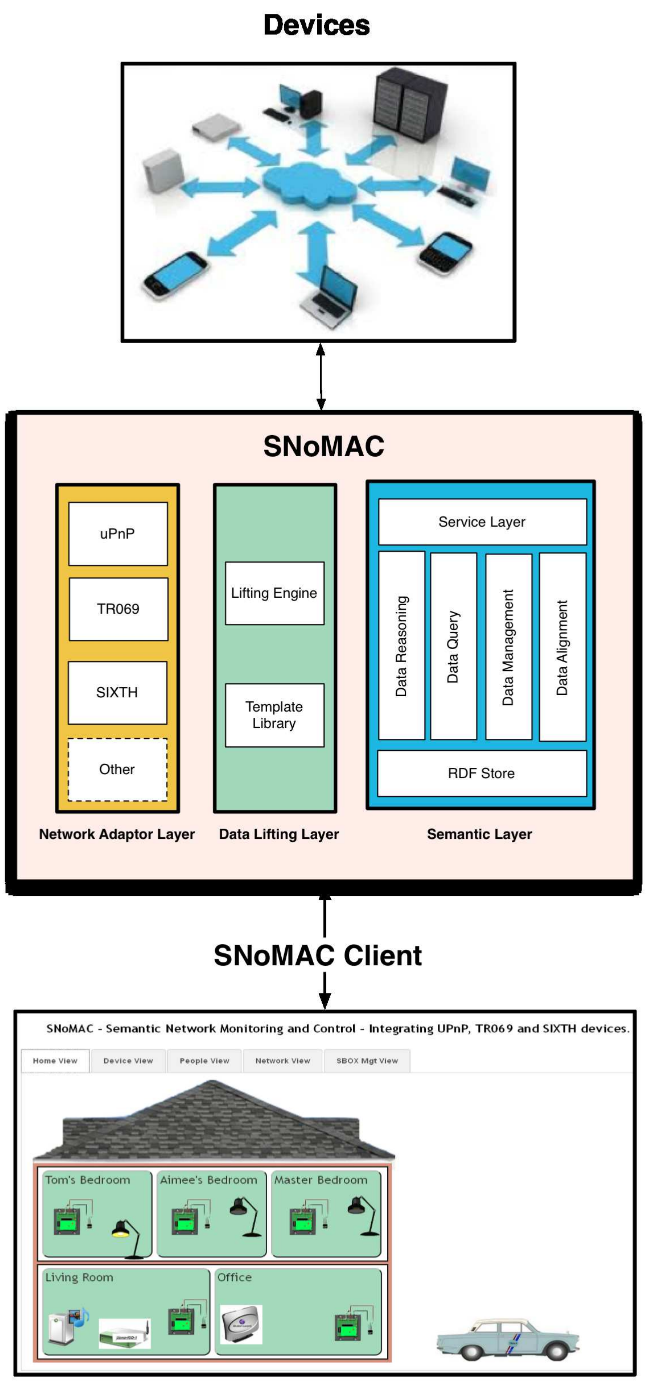


Figure 1: Conceptual Overview of SNoMAC.

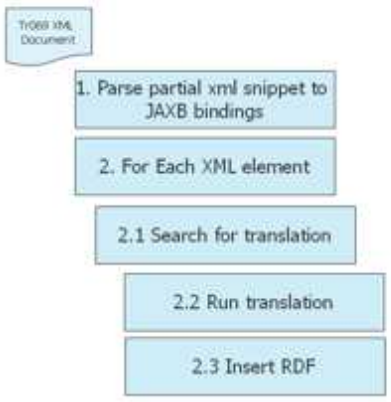


Figure 2: Data Lifting Approach.

binding and visitor design pattern approach (illustrated in Figure 2). The lifted data is inserted into an RDF triple store in the Semantic Layer. This provides a fast and flexible approach for system designers to create translations from native device models to RDF.

This data lifting process involves identifying the entities as they are described in their supported protocol and matching this against an appropriate template to facilitate translation to RDF. Once this translation has occurred, it can be passed to the Semantic Layer for storage.

The lifted data can now be aligned with the NetCore model (discussed in Section 2.1). Alignments are expressed as SPARQL queries, which construct NetCore instances from the underlying RDF device data. Web-based clients connect to the SNoMAC server through the Client Manager. Communication from the server to the client includes device information, action results and event notifications triggered by changes in the network. The clients communicate back to the server using “action requests” representing device-specific commands that are automatically propagated to the designated device, where they are executed. Users interact with devices by clicking on their graphical representations in the client interface to bring up a menu of available commands. These commands include getting and setting device variables (e.g. volume levels) and device-appropriate actions (e.g. increase/decrease volume).

2.1 Ontologies

In Computer Science, an ontology is described as “a specification of a conceptualization” [9]. Within the context of this paper, the term is used more specifically to refer to RDFS [10] and OWL [4] data models. These formally define the classes, properties, individuals and their interrelationships relevant to a particular problem domain (e.g. networks in an IoT scenario).

SNoMAC is built around the NetCore ontology shown in Figure 3. This formal OWL 2 ontology is designed to describe arbitrary networks at the highest level of abstraction, and as such consists of a relatively small number of primary classes: Network, Node, Link, State and Action.

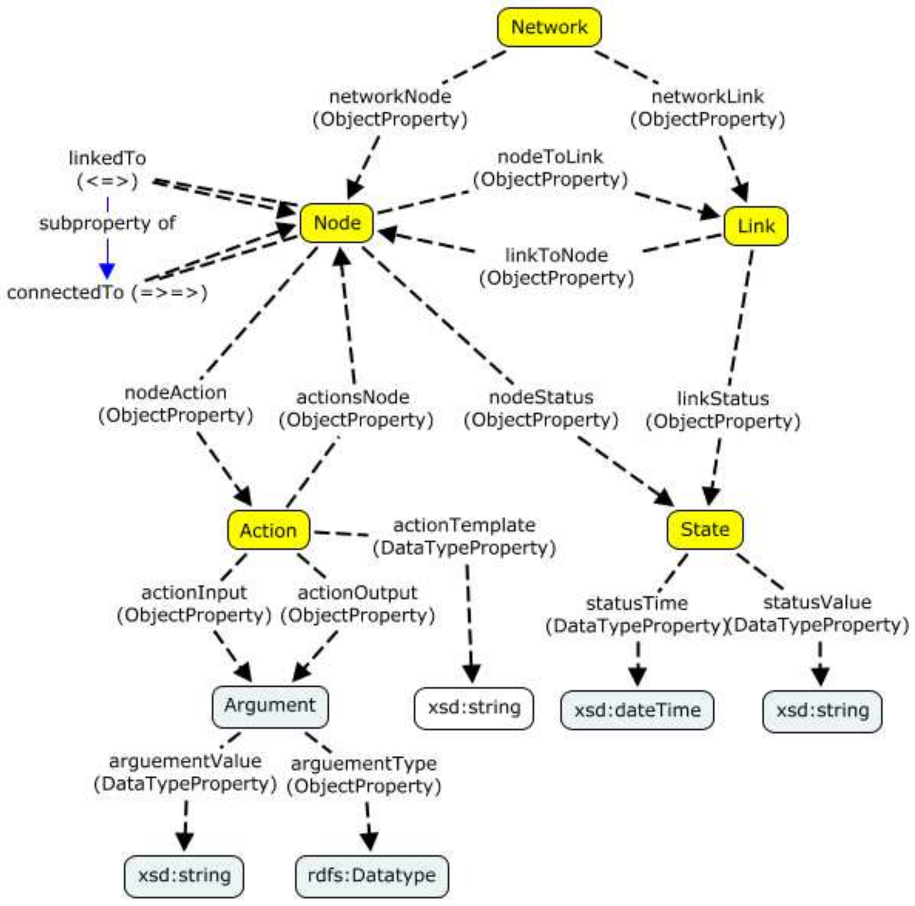

Network
networkNode (ObjectProperty)
networkLink (ObjectProperty)
linkedTo (<=>)
subproperty of
connectedTo (=>=>)
nodeToLink (ObjectProperty)
Node
Link
linkToNode (ObjectProperty)
nodeAction (ObjectProperty)
actionsNode (ObjectProperty)
nodeStatus (ObjectProperty)
linkStatus (ObjectProperty)
Action
actionTemplate (DataTypeProperty)
State
actionInput (ObjectProperty)
actionOutput (ObjectProperty)
statusTime (DataTypeProperty)
statusValue (DataTypeProperty)
Argument
xsd:string
xsd:dateTime
xsd:string
arguementValue (DataTypeProperty)
arguementType (ObjectProperty)
xsd:string
rdfs:Datatype

Figure 3: Base Classes and Properties of the NetCore OWL Ontology.

The *Node* and *Link* classes make up the essential core of the ontology and are used to represent individuals that constitute network nodes and connections between them, respectively. Individual *Node* and *Link* instances are identified as members of one or more *Networks* through the *networkNode* and *networkLink* properties. Links can be represented as instances of the *Link* class, which allows them to be annotated with additional information (e.g., *bandwidth*, *isActive*). Alternatively, they can also be represented using the symmetric property *linkedTo*, which allows them to be treated as OWL properties and thus can leverage optional property characteristics (e.g., symmetry, propertyChains). The *linkedTo* property is defined as a sub-property of *connectedTo*, which means that a node that is *linkedTo* another node is also *connectedTo* that node. Unlike *linkedTo*, the *connectedTo* property is transitive, meaning that if *n1* is *connectedTo n2* and *n2* is *connectedTo n3* then it can be inferred that *n1* is *connectedTo n3*. This captures the distinction between a direct link between nodes and a path between nodes that may be routed through intervening linked nodes.

To go beyond basic network topology, the ontology includes *State* and *Action* classes that represent state variables (for *Nodes* or *Links*) and executable actions (for *Nodes*), respectively. State variables may be read-only or both readable and writable (i.e. configuration variables). Actions represent parameterised functions that can be executed on a node, with the *actionTemplate* representing the functions call signature.

The NetCore ontology is intended to be extended and specialised to encompass various telecommunication networks. It is the inheritance afforded by the alignment of the upper-level NetCore ontology and the lower level telecommunications ontologies that permits applications to deal with specific telecommunication devices in a generic fashion, ignoring low-level details until they are required to invoke some action. The details of the low level models and protocols are handled by the specific network adapters that employ their own ontologies. As an example, the base classes and properties for the partial TR069 ontology developed at Bell Labs for SNoMAC is shown in Figure 4. The UPnP ontology used in SNoMAC comes from [11], which has been aligned with the NetCore ontology. An ontology has also been developed for SIXTH, which is discussed in Section 3.1.3.

2.2 Benefits of a Semantic Approach

Formal ontologies are used in SNoMAC to provide a number of benefits, which are illustrated in the sections that follow:

- a formal API that can help ensure that programmers interpret the model consistently through automated consistency checking and a precise definition of what can be inferred from data;
- hierarchical abstractions that can more easily hide or reveal low-level modelling and implementation details;
- automated inferencing of class and property membership that makes it trivial to inherit higher abstractions;
- ability to encode certain types of axioms that automatically detect specific conditions.

2.2.1 Abstraction and Specialisation

Inheritance is the cornerstone of semantic reasoning and is heavily leveraged in SNoMAC. Simply put, it involves automatically inferring an individual's class or

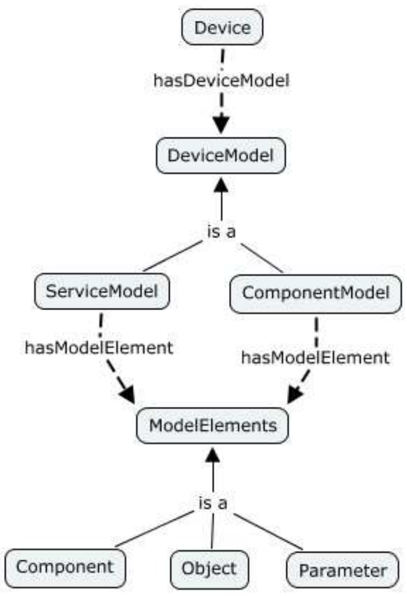


Figure 4: Base Classes and Properties of the TR-069 Ontology.

property membership based on an ontology's class and property hierarchies. For example, an ontology may state that *FemtoCell* is a subclass of *TR069:Device* and *TR069:Device* is a subclass of *NetCore:Node*. If we then state that *femtocell1* is a member of the class of *FemtoCell* then it can automatically be inferred that *femtocell1* is both a *TR069:Device* and a *NetCore:Node*, even if this information is not explicitly stated in the data description of *femtocell1*. This kind of *subsumption reasoning* can also be used to infer property membership as in the example described in Section 2.1 of how all *linkedTo* relationships imply a similar *connectedTo* relationship.

This form of *inheritance reasoning* provides the primary mechanism for supporting abstraction and data integration in SNoMAC. By aligning the NetCore class and property hierarchies with those of the network protocol specific ontologies, all devices can be treated as NetCore devices regardless of which network protocol they use. The fact that a low-level device is in fact an abstract NetCore device is automatically inferred by the system through the alignment of NetCore with the specified network ontologies.

2.2.2 Transitivity

As described in the *connectedTo* example in Section 2.1, a transitive property allows a reasoner to infer a property relationship between two entities (e.g., *connectedTo*) if they can be related through intermediary entities using the same property. The transitive characteristic of the *connectedTo* property allows for minimal specification of node connectivity in the data (specifically only the direct connections need be provided by means of *linkedTo* relationships) while permitting the automatic inferencing of total network connectivity if and when needed.

2.2.3 Property Chaining

Similar to transitive properties, property chains allow properties to be defined by defining a path of property connections chained from one entity to another.

The classic example of this is the definition of the relationship *uncleOf* as the chaining of the *brotherOf* and *parentOf* properties: if *Bob* is the *brotherOf John* and *John* is the parent of *Sue* then *Bob* is the *uncleOf Sue*. In NetCore, a property chain is used to relate the property *linkedTo* to the chained properties *nodeToLink* and *linkToNode*.[1]

Accordingly, two nodes *n1* and *n2* are automatically inferred to be *linkedTo* each other *l* that connects the two by the relationships *n1 nodeToLink l* and *l linkToNode n2*. Because *linkedTo* is a sub-property of *connectedTo*, this property chain also allows the automatic inferencing of full network connectivity (as defined by *connectedTo*) without ever mentioning either *linkedTo* or *connectedTo* in the data representation of the network.

2.2.4 Complex Axioms

It is possible to define complex axioms in OWL from which more interesting and useful relationships can be inferred. In the following example, a technique referred to as "man-man" [12] is used to automatically infer actual functional connectivity (as opposed to the logically stated connectivity of the network design). Such a relationship can be instrumental in identifying and isolating network faults.

$$\begin{array}{c}
ActiveThing \equiv\ isActive\ some\ true \qquad (1)\\
ActiveThing\ \subseteq\ isActiveSelf\ some\ self \qquad (2)\\
connectedTo\ \circ\ isActiveSelf\ \circ\ connectedTo \subseteq\ activelyConnected \qquad (3)\\
n2\ isActive\ true \qquad (4)\\
n1\ connectedTo\ n2 \qquad (5)\\
n2\ connectedTo\ n3 \qquad (6)\\
\hline
n1\ activelyConnectedTo\ n3 \qquad (7)
\end{array}$$

Figure 5: Axiom Inferences

[1]Although not graphically depicted in Figure 3 this property chain is defined in the NetCore OWL ontology.

The axioms shown in Figure 5 specify that two nodes are *activelyConnected* (line 7) if all of the intervening nodes along the logical path that connects them (i.e., the *connectedTo* path) are actually active (line 3). This is accomplished by creating a property called *isActiveSelf* that takes on the value of an individual node in the event that the node is an *ActiveThing* (line 2). An *ActiveThing* is any thing (such as a node) that has the property *isActive* equal to true (line 1). This *isActiveSelf* property is an *ObjectProperty* and as such can be used in a property chain with *connectedTo* to define what it means for two nodes to be *activelyConnected* (line 3).

Then, whenever a node *n2* is know to be active (line 4) and the logical connections between *n1* & *n2* and *n2* & *n3* are declared (lines 5 and 6) it can be inferred that *n1* and *n3* are *activelyConnected*.

2.3 Device Integration using SNoMAC

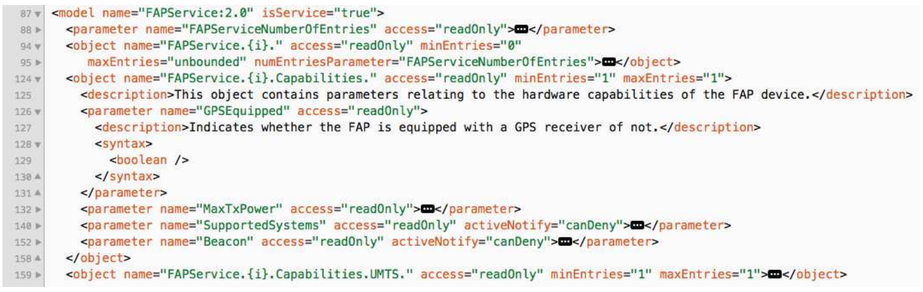
To demonstrate the flexibility of the SNoMAC system in incorporating a variety of devices, the management of two radically different TR069 devices was incorporated into the system. TR069 is a Broadband Forum (formerly known as DSL Forum) standard for the management of remote devices. The two TR069 devices used in this illustration were a Femto Cellular base station (femtocell[2]) and a remote-controlled car. Femtocells were designed for use in a home or small business to improve localised cellular service and offload bandwidth usage from macrocells (i.e. traditional cell towers). Both of these devices made use of the same NetCore abstraction.

TR069 provides a framework to describe devices in terms of "managed objects". Managed objects are logical representations of physical aspects of the device serialised in XML. Figure 6 shows a snippet of XML data for a femtocell for the Femto Access Service. This illustrates how capabilities of a TR069 device may be defined. In this instance, the *GPSEquipped* parameter is used to indicate whether or not the Femtocell Access Point (FAP) is equipped with a GPS receiver. In this example, a number of object and parameter definitions have been hidden from view. The *GPSEquipped* parameter is expanded to illustrate its definition in its entirety.

There are three major parts to this definition that should be noted: the object description and attributes within the "object" element, the parameters associated with this objects within the "parameter" element and finally the "activeNotify" attribute. The latter specifies whether this device will raise events when the parameter value changes. The data lifting layer is responsible for translating the incoming XML data into a TR069 ontology using the XML bindings approach.

A second TR069 device was implemented to demonstrate the ability to carry out actions on the device. A model remote control car had a TR069 model implemented to allow remote management of its electric motor and lights. These were presented as "Motor", "Brake" and "Lights" managed objects with a TR069

[2]`http://networks.nokia.com/products/small-cells`



```
<model name="FAPService:2.0" isService="true">
  <parameter name="FAPServiceNumberOfEntries" access="readOnly">⋯</parameter>
  <object name="FAPService.{i}." access="readOnly" minEntries="0"
     maxEntries="unbounded" numEntriesParameter="FAPServiceNumberOfEntries">⋯</object>
  <object name="FAPService.{i}.Capabilities." access="readOnly" minEntries="1" maxEntries="1">
    <description>This object contains parameters relating to the hardware capabilities of the FAP device.</description>
    <parameter name="GPSEquipped" access="readOnly">
      <description>Indicates whether the FAP is equipped with a GPS receiver of not.</description>
      <syntax>
        <boolean />
      </syntax>
    </parameter>
    <parameter name="MaxTxPower" access="readOnly">⋯</parameter>
    <parameter name="SupportedSystems" access="readOnly" activeNotify="canDeny">⋯</parameter>
    <parameter name="Beacon" access="readOnly" activeNotify="canDeny">⋯</parameter>
  </object>
  <object name="FAPService.{i}.Capabilities.UMTS." access="readOnly" minEntries="1" maxEntries="1">⋯</object>
```

Figure 6: Snippet from Femto TR069 Specification (TR196).

agent software running on an Arduino controller to interface with the motor and lighting circuitry on the remote control car. In this case, the car is represented as a NetCore node, with actions for the motor and lights. As shown in Figure 7, the state of the car itself can either be *on* or *off*, while actions allow the car to be driven forward or backwards and allow lights and brakes to be turned on or off. Through the SNoMAC server, a client application can be informed of changes in the state of the car, in addition to invoking any of the available actions. A particular implementation of a web-based client application is discussed in Section 3.

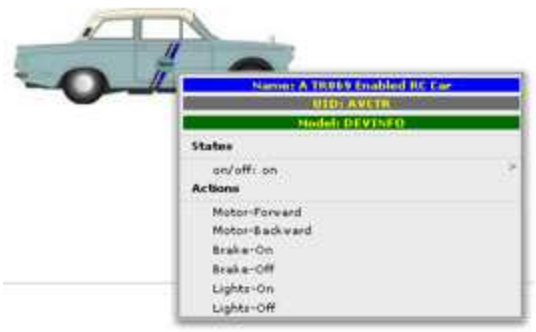


Figure 7: Simple TR069 Device Management.

2.4 Intelligent Performance Data Processing for Femtocells

The integration of a femtocell into the prototype allows for the illustration of the intelligent data processing that is fundamental to SNoMAC. During operation, femtocells (as well as most other network elements) produce many low-level performance metrics (e.g. number of successful handovers, number of call initiation attempts) across a range of performance categories (e.g. packet data performance, handover performance) [13, 14]. This Performance Management (PM) data is periodically captured and stored as XML, either temporarily on the femtocell or to a network management application for subsequent analysis. In the case of the femtocell test bed used in the prototype, an XML document representing a single recording on a femtocell contains 128 numeric values with a mixture of integer and floating-point numbers.

The "health" status of network nodes (femtocells) can be intelligently inferred from the low level performance management schema in an automated manner. This is intended to facilitate a scenario whereby network managers or applications programmers wish to develop simple abstractions of complex underlying network data collected from femtocells. Figure 8 illustrates this process. Firstly, the raw PM data (expressed in XML) is "lifted" into an ontology-based RDF representation so it can be processed by other semantic tools (step 1). Based on the lifted metrics, the Key Performance Indicators (KPIs) of the femtocells must be calculated (step 2). Finally, the health class of the femtocells themselves can be inferred from this data (step 3).

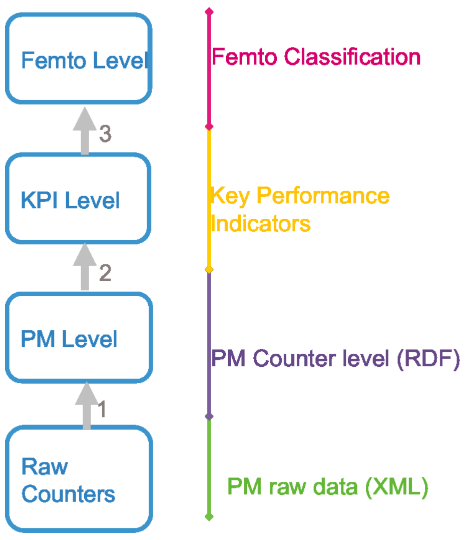


Figure 8: Femtocell PM Data Processing in a Semantic Context.

Because of the representation as RDF, the latter step can be carried out using OWL axioms, SPARQL queries or SWRL rules. SPARQL queries have been found to process this type of classification more efficiently [15].

As an example, a number of range-based classification functions are defined that determine the health of a femtocell. Figure 9 shows a definition of a HighHealth class (similar classes are possible for MediumHealth and LowHealth). For simplicity, the names of the KPIs have been anonymised.

A SPARQL query for the HighHealth class can be seen in Figure 10. This consists of three parts: a construct template containing the inferred rdf:type

```
HighHealth(x) = true if ((80 <= KPI1(x) < 100) and
(80 <= KPI2(x) < 100) and
(80 <= KPI3(x) < 100) and
(80 <= KPI4(x) < 100) and
(80 <= KPI5(x) < 100) and
(80 <= KPI6(x) < 100))
false otherwise
```

Figure 9: Definition of the HighHealth classification.

triple representing the new classification information, a basic graph pattern that selects femtocells along with the appropriate KPI data values and a filter expression to filter only those femtocells that can be classified as HighHealth.

```
CONSTRUCT { ?femto a :HighHealth . }
WHERE { ?femto :KPI1 ?v1 ; :KPI2 ?v2 ; :KPI3 ?v3 ;
:KPI4 ?v4 ; :KPI5 ?v5 ; :KPI6 ?v6 .
FILTER(  80 <= ?v1 && ?v1 < 100 &&
80 <= ?v2 && ?v2 < 100 &&
80 <= ?v3 && ?v3 < 100 &&
80 <= ?v4 && ?v4 < 100 &&
80 <= ?v5 && ?v5 < 100 &&
80 <= ?v6 && ?v6 < 100) }
```

Figure 10: HighHealth represented as a SPARQL query

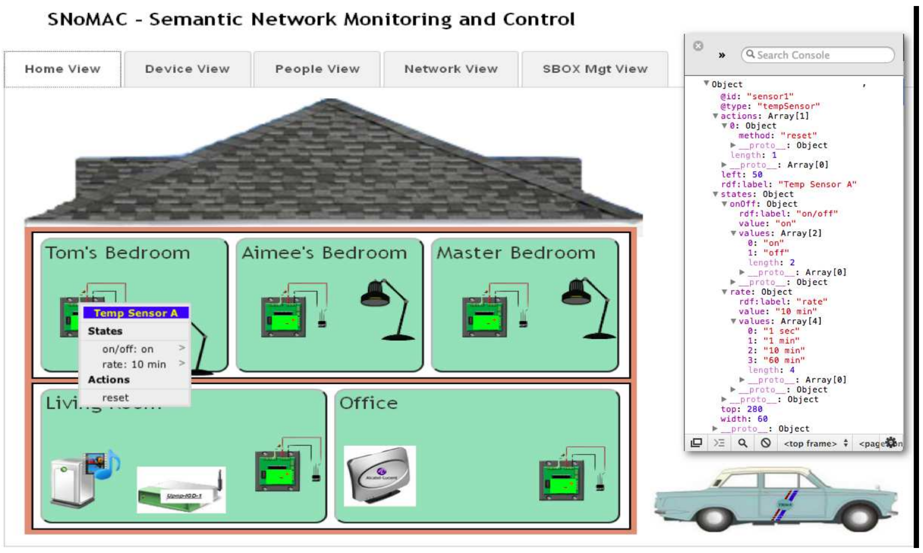
If the data values of a femtocell are in the specified ranges, then a new triple, classifying the femtocell as HighHealth, is constructed. The fact that the RDF representation of the underlying devices supports a variety of semantic tools allows for a wide range of reasoning to occur about the devices, their network topology and their states, even when such information is not readily available from the raw data.

3 The SNoMAC Prototype

A SNoMAC prototype has been developed that targets the task of monitoring and controlling a Home Area Network that includes UPnP devices, TR069 devices and an array of heterogeneous sensors managed by SIXTH (discussed in Section 3.1 below). This prototype is a stepping stone towards achieving the vision of an Ambient Assisted Living (AAL) environment, where intelligent decision-making can be carried out based on standardised access to information about the home [16]. This information can come either directly from UPnP or TR069-enabled devices or via sensors that are either physically embedded within the home or gather information from online sources. A user is provided with a client that automatically reflects the availability of devices, people and other entities within a home, and that facilitates action requests being sent to appropriate devices so as to control them.

At the lower level, specific use cases that the system can handle include discovering when new devices are added or removed, remotely getting and setting device state variables, issuing commands to specific devices followed by processing their results and seamlessly handling the addition of new network types through the introduction of additional network adapters. The core of the prototype has been implemented as a node.js[3] application that connects to independent network adapters through a Web service API using JSON-LD[4] to exchange serialised RDF data about the networked devices. Clients connect to the server via websockets and pass event-driven messages using JSON-LD.

An RDF datastore and SPARQL engine (based on the node.js module rdfstore[5]) has been installed in the prototype and loaded with the NetCore ontology. OWL 2 inference rules have been implemented as SPARQL queries to enable the reasoning required to support semantics built into NetCore.



```
▼Object
   @id: "sensor1"
   @type: "tempSensor"
  ▼actions: Array[1]
    ▼0: Object
       method: "reset"
     ▶__proto__: Object
     length: 1
    ▶__proto__: Array[0]
   left: 50
   rdf:label: "Temp Sensor A"
  ▼states: Object
    ▼onOff: Object
       rdf:label: "on/off"
       value: "on"
      ▼values: Array[2]
         0: "on"
         1: "off"
         length: 2
       ▶__proto__: Array[0]
     ▶__proto__: Object
    ▼rate: Object
       rdf:label: "rate"
       value: "10 min"
      ▼values: Array[4]
         0: "1 sec"
         1: "1 min"
         2: "10 min"
         3: "60 min"
         length: 4
       ▶__proto__: Array[0]
     ▶__proto__: Object
    ▶__proto__: Object
   top: 280
   width: 60
  ▶__proto__: Object
```

Figure 11: SNoMAC Prototype Client: Home View.

A desktop web interface for the prototype has been implemented using HTML5 and JavaScript. The interface has been designed to provide four views on the network (Home, Device, Network and People). The Home View, which is shown in Figure 11, depicts the devices distributed across the rooms of a house. The set of devices depicted by the interface currently includes temperature sensors, lamps, femtocells, WiFi router, a media server, computers, printers and the car discussed in Section 2.3. The image contains an inset showing the SNoMAC console that is used to display the JSON-LD RDF objects relating to the various devices.

[3]`http://nodejs.org`
[4]`http://json-ld.org`
[5]`https://github.com/antoniogarrote/rdfstore-js`

A context menu is activated by mousing over a device. It permits the viewing and setting of state variables as well as the invocation of device specific actions. This can be seen in Figure 11 where a context menu has been activated for a temperature sensor located in “Tom’s Bedroom”. This indicates two state variables for the device: one indicates whether the sensor is turned on or off, while the other shows the rate at which the sensor captures data. Changing either of these through the submenu in the GUI (indicated by the arrows) will result in a command being sent back to the server where it is forwarded to the appropriate network adaptor. In the case of a temperature sensor, this will ultimately trigger a (re)tasking message to be sent through SIXTH, which will cause the state of the device to change. An action message can also be triggered to reset the device, which is also routed through SIXTH in this case.

For other devices connected via different network adaptors, state changes and action messages are sent through the SNoMAC server directly, as illustrated in the remote control car example from Section 2.3. In each case, when the state of the device changes the network adapter pushes the updated state back to the server and it is passed on to the client where the state of the device changes accordingly (e.g. in the case of turning off a lamp, it turns the state variable to “off” and the icon goes dark).

The other views are intended to allow client access to other entities. The Network View shows the topology of the network, so that the connections between entities known to the SNoMAC server can be visualised. The People View is used to indicate the presence of people in various locations. This view and its implementation is discussed in more detail in Section 3.1 to focus on how SIXTH augments the capabilities of SNoMAC.

The use of SNoMAC in such a prototype allows for the intelligent reasoning about the presence and connectedness of entities to be handled automatically without the need for user interaction or configuration. Adding and removing new devices, sensors or people of interest causes the interface to update automatically. Similarly, the framework allows for client applications to easily route action messages and device state change requests to the appropriate location based on its ontological specification in the underlying triple store.

3.1 SIXTH Sensor Middleware

SIXTH is an extensible, scalable and intelligent sensor middleware based upon the Open Services Gateway initiative framework (OSGi)[6] [5]. SIXTH is targeted toward dynamic and reconfigurable sensor networks. It provides connection into both physical (e.g. SunSPOT, Shimmer) and cyber (e.g. Twitter [7], Foursquare [8], Xively[9]) sensing networks. These sources can be mined in concert to create a wider context for detected activity. SIXTH has previously been utilised in

[6]http://www.osgi.org/

[7]http://www.twitter.com

[8]http://www.foursquare.com

[9]Provides a web-based gateway to access IoT devices: http://xively.com

a variety of application domains, including mobile sensing with Augmented Reality [17] and personal health monitoring [18].

The primary SIXTH components include:

- A properties-based sensor model for control of and access to sensing devices.
- Adaptors that provide sensor specific implementations to a standardised interface.
- Higher level APIs for access to SIXTH's core functionality covering data access.
- Retasking, notification, security and discovery services.
- Data processing service layers which build upon the APIs.
- Integrated multi-agent system to support in-situ reasoning and wireless sensor network management.

Within SIXTH, extensibility is provided by allowing for the dynamic addition of new components to the middleware during runtime. Such components may include a new adaptor layer. Adaptors are responsible for providing the communication and translation mechanisms for connection with heterogeneous data sources. In a similar way to how SNoMAC adopts the adaptor design pattern for internet protocols, SIXTH integrates various sensors and sensor platforms using dedicated adaptors also. Other functionality may be injected at run-time such as a custom sensor data retention policy (to indicate for how long and under what circumstances historical sensor data should be retained), or a new service (e.g. to allow JmDNS-based communication across SIXTH deployments).

SIXTH encompasses the sensing of data from online resources (cyber-sensing) as a first-class citizen alongside direct sensing of physical world phenomena. This provides a more unified view of Sensor Web resources that encompasses data from multiple disparate sources. Cyber-sensing streams can be dynamically configured and the details and differences between these and their physical sensing counterparts are hidden from the end user. SIXTH is primarily a gateway-side middleware (software resides on a host machine). This decision was made to easily support heterogeneous hardware. As such, even in the case where the network has been programmed independently, within SIXTH an adaptor can be defined for that network.

To augment these abilities, efforts have been invested in augmenting SIXTH's core benefits with intelligent, agent-driven, sensor network deployments. Existing work includes running Agent Factory Micro Edition (AFME) directly on sensor motes to provide in-network intelligence [19], using AFME on the middleware gateway to react to sensor data [20] and also in the use of intelligent agents for the autonomic management of the sensor middleware itself [21]. In some cases, it is important for intelligence to be incorporated towards the edges of the network, on or near the nodes themselves. This includes handling issues such as coping with limited bandwidth, battery preservation and dealing with computationally restricted devices [22]. This approach allows for the embedding of intelligent agents into all levels of the system, which is consistent with the vision of SNoMAC.

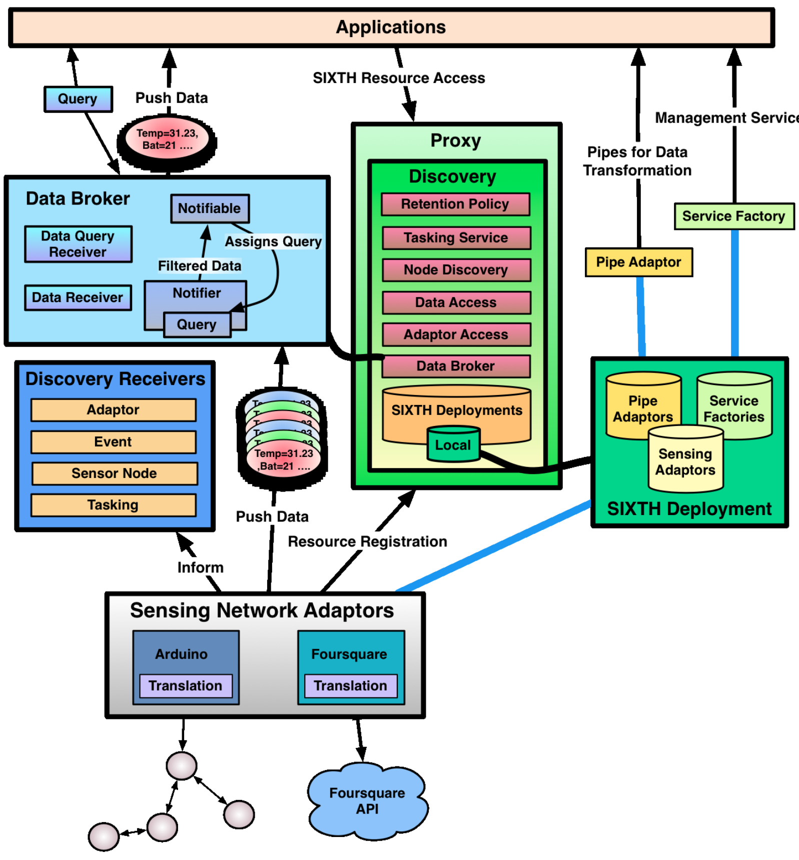


Figure 12: The SIXTH Sensor Middleware System Architecture.

The SIXTH system architecture can be seen in Figure 12. Within this high level architecture several important concepts are to be seen. The Data Broker module is collectively responsible for distribution and management of sensor data. The discovery sub-system provides notification of resource status (e.g. node timeout, adaptor creation etc.) and management notice dissemination based upon credentialed security policies. The service registry provide access to service modules (e.g. sensor transformation services, aggregation services etc.).

Whereas SNoMAC supports devices that communicate using various IoT communications protocols, SIXTH augments these capabilities with a unified mechanism by which a multitude of heterogeneous sensing devices, both physical and cyber, can be utilised. SIXTH can be deployed in a distributed fashion on a wide variety of devices, and so provide access to built-in sensors (in the case of mobile devices), sensors attached via USB, wireless sensor networks accessible through a base-station node, sensors that communicate using other protocols (e.g. Bluetooth) and web-based resources that are made available through cyber sensors using HTTP.

3.1.1 SIXTH Integration with SNoMAC

To aid with the development of the SNoMAC prototype, it was necessary to use Arduino and Foursquare adaptors to connect networks of these sensor types to the SIXTH middleware. Foursquare was used specifically for the "People View" component of the prototype, which can be seen in Figure 13. This informs the system of where relevant users are through data from real-time Foursquare user sensors. Foursquare users of interest to the system are monitored by these cyber sensors, which report updated user and location information for new 'check-ins'. For the Home View of the SNoMAC prototype discussed in Section 3, the representation of lights is implemented by means of Arduino devices equipped with light sensors. A further Arduino device was used for to enable discovery and control of the remote controlled car.

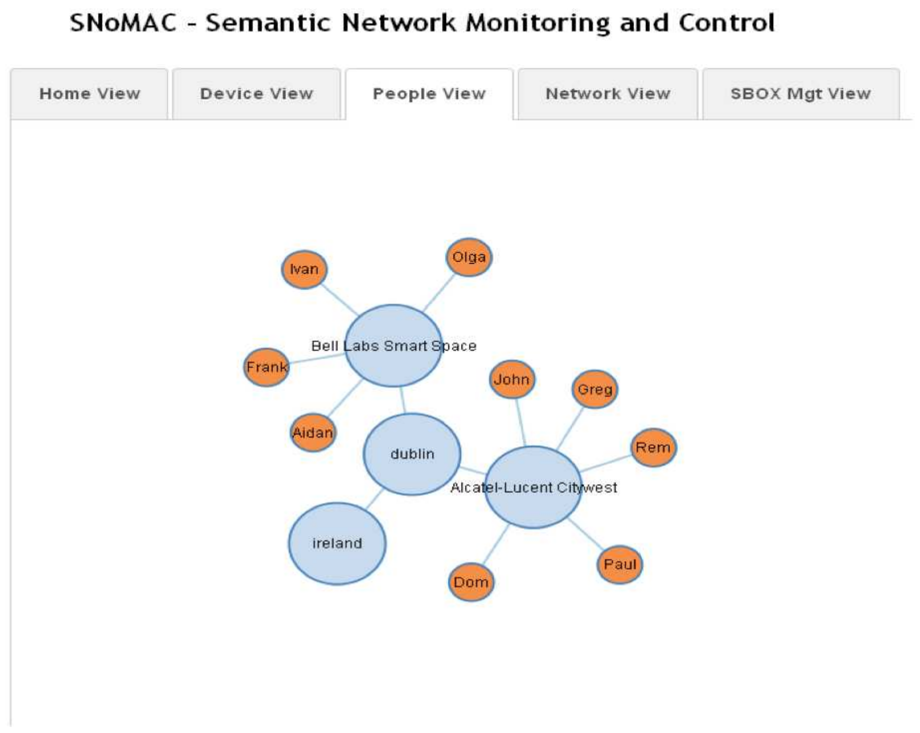


Figure 13: SNoMAC Prototype Client: People View.

3.1.2 SIXTH/SNoMAC Communication Bridge

To create a communication bridge between SIXTH and SNoMAC, a software component was developed conforming to the SNoMAC Network Adapter API. The Restlet[10] engine for Java 5 was used to enable RESTful communication.

```
GET /nodes
returns an collection of nodes with 1) ids, 2) their RDF
types (as defined by a specific network ontology) and 3) the
current value of the node's "status" variable (as defined
by NetCore)
e.g. [
{"@id":"n1","@type":"upnp:mediaServer","ncStatus":"yellow"},
{"@id":"n2","@type":"upnp:light","ncStatus":"green"},
{"@id":"n3","@type":"upnp:light","ncStatus":"green"}
]
```

Figure 14: Snippet of the SNoMAC RESTful API.

Figure 14 shows an extract from the SNoMAC RESTful API which specifies the JSON to be returned to show all nodes which are held by a SIXTH deployment acting as a SNoMAC network adaptor. Listing 1 shows the SIXTH implementation of this method within the Restlet component. References to the current SIXTH adaptor set are retrieved and using the *NodeDiscovery* component all the Sensor Nodes are extracted and passed out for abstraction and serialization into JSON format. This helps to illustrate how the programming framework associated with SIXTH facilitates the rapid development of client systems to access its components, in this case sensor nodes.

Listing 1: Accessing the Tasking Service

```java
public class NodesResource extends ServerResource {

@Get("text|json")
public String represent() {
IDiscovery discovery = SIXTHMonitor.getDiscovery(
new Credentials("SnoMac", "SnoMac", UUID.randomUUID()));

ISixthDeployment locDep = discovery.getLocalDeployment();

List<ISensorNetworkAdaptor> adap = locDep.getSensorAdaptors();

NodeDiscovery nodeDiscovery = new NodeDiscovery(adap);

return SIXTHSerializer.toJSON(nodeDiscovery.getSensorNodes());
}
}
```

[10]http://www.restlet.org/

3.1.3 SIXTH Ontology

A SIXTH sensor device ontology was developed in alignment with NetCore, which can be seen in Figure 15. In this model, both queries and sensors are defined as having optional "Modality" and "Location" properties. For a sensor, this represents the sensing modality capability of the sensor and the location of the sensed modality respectively. When used as properties of a query, these represent the requirements imposed on the requested sensor data, in terms of their modality and location. Similarly, these requirements can specify a particular source sensor also, which indicates that the sensor is capable of providing data that will satisfy the query. This property is inferred using a custom SPARQL rule.

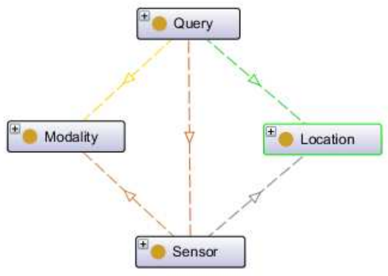


Figure 15: SIXTH Sensor Ontology.

Since SIXTH treats all sensors as equal citizens, regardless of origin and form, the same ontology facilitates the use of both physical and cyber sensors. The initial development effort took less than three days and resulted in a complete working system capable of passing the entire suite of API tests except action commands. While this work represents just one case study, it does provide support for the claim that the addition of new network models/protocols to SNoMAC, thereby leveraging the semantic benefits that accrue, is a relatively simple and straightforward process.

A significant advantage associated with using SIXTH alongside SNoMAC is that it provides access to a wide variety of heterogeneous, commodity sensing platforms in a uniform and consistent manner. The use of a Foursquare cyber sensor illustrates the added capability of accessing data made available through web APIs, which is not a feature native to SNoMAC. By providing the communications bridge between the SNoMAC prototype server and SIXTH, it provides access to a sensor network that can be distributed over a wide area, connecting to a multitude of sensing capabilities. Finally, since SIXTH supports intelligent agents at several levels, this opens the possibility of supporting autonomic reconfiguration of the sensor network, thus augmenting the intelligent reasoning capabilities of SNoMAC itself.

4 Related Work

The utility of ontologies within sensor-based information systems was established by Liu and Zhao [23]. In this work, an ontological abstraction is used to facilitate the optimisation of collecting, storing and processing data. It also allows for resource-aware execution of service composition and the building of a sensor information hierarchy. Further work by Whitehouse et al. [24] augments this by allowing end users to pose declarative queries that can be applied to semantic interpretations of sensor data, thus creating much more powerful and flexible access to data above working merely with raw streams.

A number of projects have carried out various types of reasoning using semantic descriptions of devices expressed in various languages. The work of Tran et al. [25] focuses on the dynamic composition of sensors and sensor data transformation services to create workflows that transform raw sensor data into data specified by user requests. This work focuses on the underlying semantic ontology [26], which acts as the main reasoning engine for enabling the dynamic composition of the various resources. OWL ontologies are combined with SPARQL and description logic to achieve the dynamic composition.

Bröring et al. [27] have undertaken similar work from the perspective of the Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) initiative [28] and present a framework that uses semantically-enabled matchmaking to allow plug-and-play usage of sensors within a network using SensorML standards. To enable this they have focused on the semantic matchmaking functionality and offer a system to facilitate the declarative description of the sensor interfaces, which allows the advertised and requested characteristics of the system to be specified in a formal language supporting inferencing.

Further initiatives have been documented within the context of OGC standards. OpenSensorHub[11] is an open source software stack built on SWE and IoT standards. It promotes the seamless integration of all kinds of sensors into sensor webs while providing a suite of common services for sensor management, for example, discovery, visualisation and so forth. Cloud-Hosted Real-time Data Services for the Geosciences (CHORDS) [29] is a project of the NSF EarthCube initiative: this project seeks to deliver a real-time cyber-infrastructure that enables the seamless acquisition and distribution of sensor data via the cloud. Standardised data and metadata formats, including XML and JSON, are essential to achieving the CHORDS vision of broadening access to real-time data for the geosciences community.

It is also important to note that these semantic technologies are seen by many researchers as key enablers for the Internet of Things (IoT). Soldatos et al. propose a semantic IoT platform built on the SSN for the semantic interoperability it enables [30], and this approach is echoed in the advocacy of Sheth et al. [31]. Desai et al. continue this trend by using semantic annotation, built on the SSN ontology, for interoperability and automated conversion between different messaging protocols such as XMPP, CoAP and MQTT [32].

[11]https://opensensorhub.org/

Maarala et al. demonstrate the use of semantic IoT data for reasoning on actionable knowledge [33]; this research is of particular relevance to this discussion on SNoMAC in that it considers characteristic IoT-specific challenges of resource constraints, dynamics, scalability, and real-time processing.

Similarly, Wu et al. propose a semantic Web of Things (WoT) framework for Cyber Physical Systems (CPS) to address the complex relationships between devices where dynamic composition and collaboration are the norm [34]. In each of these situations, ontology-based semantic reasoning is seen as the means to securing interoperability between diverse devices by facilitating intelligent reasoning about an underlying network to achieve higher-level application goals.

Although SNoMAC leverages a similar semantic approach (using NetCore ontologies, RDF stores and SPARQL), what sets it apart and marks it as novel is the aim to present the relevant semantic information to the end user through the interface as per Figure 11. This allows the end user to navigate the network at the appropriate level of abstraction, and allows them to make well informed changes at the best possible level. In addition, SNoMAC does not assume pre-existing ontologies that have been manually created. Instead, the Data Lifting Layer discussed in Section 2 uplifts the known information about the devices into RDF, after which it is further aligned with the NetCore model.

In terms of its outlook and aims, perhaps the most similar work to that presented in this paper is that of Aloulou et al. [35]. They also present an OSGi-based framework that is focused on enabling plug and play sensor integration with a semantic reasoning engine. In this case, the usefulness of the system is demonstrated within an Ambient Assisted Living scenario. This framework is designed to operate in a dynamic environment, where new sensors and devices can be added as the scenario evolves with the patient's changing needs, and mainly focuses on the reasoning engine supporting this dynamic environment in a seamless manner. As with the above-referenced works, this also depends on a pre-existing ontology being present (represented in "Notation 3" (N3)).

While the above has focused on more general technological research, the true benefits of these semantic approaches comes from their wide applicability in a variety of smart scenarios. Case studies include parking garage management [24, 23], oil spills [36], real-time geoscience [29] and ambient assisted living [35].

At the building level, semantic technologies have been leveraged to allow sensor data from various sources to be integrated, ingested, and reasoned about for automated building monitoring [37]. Similarly, Ahmadi-Karvigh et al. harness semantic reasoning for activity monitoring, so as to lead to greater energy efficiency within buildings [38].

Smart cities also provide a platform to demonstrate the potential of semantic technologies. Bischof et al. use the Semantic Sensor Network (SSN) ontology to promote interoperability in this context [39]. Also within the smart cities domain, D'Aniello et al. present a semantic-driven architecture for supporting cross-domain decision-making at operational, tactical and strategic levels [40].

SIXTH can be considered to be a gateway-side middleware platform. This means that it does not run on sensor nodes themselves, but instead communicates with individual sensors and devices through gateways into wireless sensor networks. Numerous sensor middleware platforms exist, and in-depth discussions of these can be found in [41, 42].

As discussed in Section 3.1, one particular feature that sets SIXTH apart from the majority of sensor middleware platforms is its ability to provide intelligent reasoning through integration with agent programming languages. Some other middleware offerings also support intelligent reasoning. Agilla [43] and In-Motes [44] both allow software agents to be injected into the sensor network to run the nodes on behalf of the user. These agents provide autonomous control of the network by behavioural evaluation in response to shifting constraints (e.g. lessening battery life).

Generic Role Assignment (GRA) targets the issue of network configuration, particularly as it pertains to the large scale [45]. Within such networks the user cannot realistically assign tasks to each node. Nodes in the GRA systems tune their behavioural set autonomously, informed partly by the status and capabilities of neighbouring nodes. To accomplish this, nodes expose their capabilities and remaining resources to aid co-ordination in accomplishing the application goal.

While agent technologies have been used to at the fringes of the sensor network, higher level frameworks may also utilise such technology to provide control and co-ordination. One example of this is IrisNet [46], within which the use of agents is two-fold. One set of agents manage the collection of sensor data while another set is responsible for the storage and organisation of data across a distributed database.

Clearly, intelligence is seen as a desirable property for many middleware platforms. The principal feature of SIXTH that sets it apart from these offerings is that it is intended to support heterogenous sensor types, and also be extensible. This extensibility allows for the addition of new adaptors at runtime that can dynamically add new sensor types to the system without requiring a shutdown.

A related approach to intelligent agents is the smart objects framework described by López et al. [47]. This paradigm has much in common with agent-based reasoning. The key features of a smart object lie in their ability to make decisions regarding their own actions and communicate collaboratively with other smart objects [48].

5 Evaluation and Future Work

Recent work has focused on augmenting SIXTH in terms of its core functionality so as to deliver the practical but seamless approach to heterogeneous device discovery envisaged by SNoMAC. Three broad categories of evaluation have resulted from this endeavour. Firstly, its versatility and extensibility are illustrated by extensions that have been made to the core platform. An extension to SIXTH to develop cyber adaptors for the gathering of data from online resources such as Twitter and Facebook is described in [49]. SIXTH has also been extended and ported to Android devices, where it operates as an in-network middleware [50]. This approach is similar to the usage of GSN [51] as the basis for the MOSDEN [52] middleware for Android. In addition to its integration with SNoMAC, SIXTH has also been linked with an intelligent Agent-Oriented Programming (AOP) runtime platform [21]. In this scenario the data gathered by SIXTH-connected devices provides the intelligent software agents with an abstraction of their physical environment. In each of these situations, SIXTH has been demonstrated to adapt easily to new data sources and additional subsystems.

Secondly, SIXTH has been deployed in several application scenarios, which demonstrates its capability of servicing real-time, real-world applications. One such scenario is WAIST [53], which utilises Wireless Sensor Network technology in the detection of illegal waste disposal during transit. Herein, SIXTH acts as the data provider for the reasoning engine, which detects anomalous movement of waste materials to provide an indication of improper disposal. SIXTH has also been used as a data provider and reconfiguration engine in two home energy management systems [54, 55]. Additionally, it has been used as a framework for citizen sensing, leveraging the ubiquity of modern smart phones and wearables to provide detailed observations of remote physical locations [56].

The third evaluation focus is on code quality metrics and user evaluations [41]. The static analysis of the code base yielded favourable results when compared to that of two similar middleware offerings: GSN [51] and WSNWare [57]. The metrics utilised are long established and specified in [58]. A user evaluation study using the System Usability Scale (SUS) also resulted in SIXTH obtaining positive scores when evaluated in accordance with work done by Bangor et al. [59] and by Sauro [60].

In terms of future work, it is planned to continue developing the UPnP and TR069 ontologies to increase the level of coverage over the functionality of the devices based on those models/protocols. It is also intended to further explore the extent to which the SNoMAC approach can be seamlessly applied to other network models/protocols (e.g. ZigBee, Bluetooth, Bonjour and OMA-D). This future work will include experiments to help quantify the level of effort required to add new network adapters to SNoMAC. Network analysis using semantic techniques will also be explored more deeply.

The prototype presented in this paper featured a web-based client to allow a user to interact with the system. However, since this client interacts with the SNoMAC prototype server through a RESTful interface, this architecture is extensible to a wide variety of user interface paradigms to facilitate human control over a device network. Further investigation is required as to the form these interfaces may take, including the possibility for mobile/tablet and augmented reality interfaces. A further extension is the integration of an intelligent management system to implement techniques to relieve the user of the responsibility of direct control over devices. In this way, we move from a system-centric paradigm to a human-centric paradigm. This type of system is a natural extension of SNoMAC, as the semantic descriptions of the configuration, state and capabilities of heterogeneous devices are presented in a consistent, uniform manner.

6 Conclusion

The rapid evolution of communication networks presents challenges that require a new approach to network monitoring and control. Semantic technologies offer the means to encapsulate network details at various levels of abstraction making it easier to develop solutions that can adapt to changing data models and protocols. Ontological descriptions of data sources and devices have shown great promise in the literature in allowing systems to reason about their resources to satisfy user needs. With appropriate ontologies, such reasoning can be intelligently performed within the systems themselves, without the requirement for direct human intervention. SNoMAC, and the NetCore ontology on which it is predicated, provide an example of how this can be effectively achieved today, at least in the domain of home area networks. Furthermore, the uplift approach taken eliminates the need for the prior existence of semantic descriptors, as these are generated when needed, and can then be further enhanced by loading existing semantic resources associated with these descriptors.

Incorporating the SIXTH sensor middleware widens the range of available devices yet further, beyond telecommunication devices and the IoT into the realm of physical sensing devices and cyber sensors. Furthermore, the combination of the intelligent capabilities of both SNoMAC and SIXTH provides a powerful basis upon which a variety of next generation human-centric applications can be developed. In combination, they provide an end-to-end platform that encompasses the ability to gather data from heterogeneous devices that can be added and removed at runtime; the automated uplifting of device information into a queryable ontology that reflects hierarchical abstractions; the provision of intelligent reasoning capabilities through agent programming languages; and the offering of a user interface to allow users to explore and manage all aspects of their systems.

ACKNOWLEDGMENT

This work was part funded by Science Foundation Ireland under grant number 07/CE/I1147 and also by the Industrial Development Authority of Ireland.

References

[1] Cisco Systems . Cisco Visual Networking Index: Forecast and Methodology, 2016-2021 2017.

[2] Harris S, Seaborne A, Prud'hommeaux E. SPARQL 1.1 Query Language http://www.w3.org/TR/sparql11-query/ 2013.

[3] Horrocks I, Patel-Schneider PF, Boley H, Said T, Grosof B, Dean M. SWRL: A Semantic Web Rule Language Combining OWL and RuleML http://www.w3.org/Submission/SWRL/ 2004.

[4] W3C OWL Working Group . OWL 2 Web Ontology Language Document Overview (Second Edition) https://www.w3.org/TR/owl2-overview/ 2012.

[5] O'Hare GMP, Muldoon C, O'Grady MJ, Collier RW, Murdoch O, Carr D. Sensor Web Interaction *International Journal on Artificial Intelligence Tools.* 2012;21:1240006.

[6] Contributing Members of the UPnP Forum . UPnP Device Architecture 1.0 http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf 2008.

[7] Cruz T, Simoes P, Batista P, Almeida J, Monteiro E, Bastos F. CWMP Extensions for Enhanced Management of Domestic Network Services in *Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks*LCN '10(Washington, DC, USA):180–183IEEE Computer Society 2010.

[8] Matheus CJ, Boran A, Carr D, et al. Semantic Network Monitoring and Control over Heterogeneous Network Models and Protocols *Active Media Technology.* 2012;7669:433–444.

[9] Gruber TR. A translation approach to portable ontology specifications *Knowledge Acquisition.* 1993;5:199–220.

[10] Brickley D, Guha RV. RDF Vocabulary Description Language 1.0: RDF Schema http://www.w3.org/TR/rdf-schema/ 2004.

[11] Togias K, Goumopoulos C, Kameas A. Ontology-Based Representation of UPnP Devices and Services for Dynamic Context-Aware Ubiquitous Computing Applications *International Conference on Communication Theory, Reliability, and Quality of Service.* 2010;0:220–225.

[12] Tsarkov D, Sattler U, Stevens M, Stevens R. A solution for the Man-Man problem in the Family History Knowledge Base in *Proceedings of the 6th International Conference on OWL: Experiences and Directions-Volume 529*:69–78 2009.

[13] Kreher R. *UMTS Performance Measurement: a Practical Guide to KPIs for the UTRAN Environment.* Wiley 2006.

[14] Technical Standard 3GPP 32.403. Telecommunication Management. Performance Management (PM). UMTS Performance Measurements and Combined UMTS/GSM http://www.3gpp.org/ftp/Specs/html-info/32403.htm 2005.

[15] Boran A, Bedini I, Matheus CJ, Patel-Schneider PF, Bischof S. An Empirical Analysis of Semantic Techniques Applied to a Network Management Classification Problem in *Proceedings of the 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*:90–96 IEEE Computer Society 2012.

[16] Van Den Broek G, Cavallo F, Wehrmann C. *AALIANCE Ambient Assisted Living Roadmap.* Amsterdam, The Netherlands: IOS Press 2010.

[17] Kroon B, Görgü L, Russell S, et al. SIXTH: Cupid for the Sensor Web in *Proceedings of the Seventh Annual Irish Human Computer Interaction Conference (iHCI 2013)* 2013.

[18] Carr D, O'Grady MJ, O'Hare GMP, Collier RW. SIXTH: A Middleware for Supporting Ubiquitous Sensing in Personal Health Monitoring in *Proceedings of the 3rd International Conference on Wireless Mobile Communication and Healthcare (MobiHealth 2012)*(Paris, France):421–428 2013.

[19] Tynan R, Muldoon C, O'Hare GMP, O'Grady MJ. Coordinated Intelligent Power Management and the Heterogeneous Sensing Coverage Problem *Comput. J..* 2011;54:490–502.

[20] Lillis D, O'Sullivan T, Holz T, Muldoon C, O'Grady MJ, O'Hare GMP. Smart Home Energy Management in *Recent Advances in Ambient Intelligent and Context-Aware Computing* (Curran K. , ed.):155–168 IGI Global 2015.

[21] Lillis D, Russell S, Carr D, Collier RW, O'Hare GMP. Intelligent Decision-Making in the Physical Environment in *Ambient Intelligence* (Augusto JC, Wichert R, Collier R, Keyson D, Salah AA, Tan AH. , eds.);8309 of *Lecture Notes in Computer Science*:235-240Springer International Publishing 2013.

[22] O'Grady MJ, O'Hare GMP, Chen J, Phelan D. Distributed network intelligence: A prerequisite for adaptive and personalised service delivery *Information Systems Frontiers.* 2008;11:61–73.

[23] Liu J, Zhao F. Towards semantic services for sensor-rich information systems in *2nd International Conference on Broadband Networks (BroadNets 2005)*:967–974 2005.

[24] Whitehouse K, Zhao F, Liu J. Semantic streams: A framework for declarative queries and automatic data interpretation *Microsoft Research.* 2005.

[25] Tran KN, Compton M, Wu J, Gore R. Short Paper: Semantic Sensor Composition in *Proceedings of the 3rd International Workshop on Semantic Sensor Networks* 2010.

[26] Compton M, Neuhaus H, Taylor K, Tran KN. Reasoning about Sensors and Compositions in *Second International Semantic Sensor Networks Workshop*:33–48 2009.

[27] Bröring A, Maué P, Janowicz K, Nüst D, Malewski C. Semantically-Enabled Sensor Plug and Play for the Sensor Web *Sensors.* 2011;11:7568–7605.

[28] Botts M, Percivall G, Reed C, Davidson J. OGC Sensor Web Enablement: Overview and High Level Architecture in *GeoSensor Networks* (Nittel S, Labrinidis A, Stefanidis A. , eds.);4540 of *Lecture Notes in Computer Science*:175-190Springer Berlin / Heidelberg 2008.

[29] Kerkez B, Daniels M, Graves S, et al. Cloud Hosted Real-time Data Services for the Geosciences (CHORDS) *Geoscience Data Journal.* 2016;3:4–8.

[30] Soldatos J, Kefalakis N, Hauswirth M, et al. Openiot: Open source internet-of-things in the cloud in *Interoperability and open-source solutions for the internet of things*:13–25Springer 2015.

[31] Sheth A. Internet of things to smart iot through semantic, cognitive, and perceptual computing *IEEE Intelligent Systems.* 2016;31:108–112.

[32] Desai P, Sheth A, Anantharam P. Semantic gateway as a service architecture for iot interoperability in *Mobile Services (MS), 2015 IEEE International Conference on*:313–319IEEE 2015.

[33] Maarala AI, Su X, Riekki J. Semantic reasoning for context-aware Internet of Things applications *IEEE Internet of Things Journal.* 2017;4:461–473.

[34] Wu Z, Xu Y, Yang Y, Zhang C, Zhu X, Ji Y. Towards a Semantic Web of Things: A Hybrid Semantic Annotation, Extraction, and Reasoning Framework for Cyber-Physical System *Sensors.* 2017;17:403.

[35] Aloulou H, Mokhtari M, Tiberghien T, Biswas J, Kenneth L. A Semantic Plug&Play Based Framework for Ambient Assisted Living in *Impact Analysis of Solutions for Chronic Disease Prevention and Management* (Donnelly M, Paggetti C, Nugent C, Mokhtari M. , eds.);7251 of *Lecture Notes in Computer Science*:165-172Springer Berlin / Heidelberg 2012.

[36] Bröring A, Echterhoff J, Jirka S, et al. New Generation Sensor Web Enablement *Sensors.* 2011;11:2652–2699.

[37] Ploennigs J, Schumann A, Lécué F. Adapting semantic sensor networks for smart building diagnosis in *International Semantic Web Conference*:308–323Springer 2014.

[38] Ahmadi-Karvigh S, Ghahramani A, Becerik-Gerber B, Soibelman L. Real-time activity recognition for energy efficiency in buildings *Applied Energy.* 2018;211:146–160.

[39] Bischof S, Karapantelakis A, Nechifor CS, Sheth AP, Mileo A, Barnaghi P. Semantic modelling of smart city data in *Proceedings of the W3C Workshop on the Web of Things* 2014.

[40] D'Aniello G, Gaeta M, Orciuoli F. An approach based on semantic stream reasoning to support decision processes in smart cities *Telematics and Informatics.* 2018;35:68–81.

[41] Carr D. *The SIXTH Middleware: sensible sensing for the sensor web.* PhD thesisUniversity College Dublin (Ireland) 2015.

[42] Ngu AH, Gutierrez M, Metsis V, Nepal S, Sheng QZ. IoT middleware: A survey on issues and enabling technologies *IEEE Internet of Things Journal.* 2017;4:1–20.

[43] Fok CL, Roman GC, Lu C. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks *ACM Transactions on Autonomous and Adaptive Systems (TAAS).* 2009;4:16.

[44] Georgoulas D, Blow K. In-motes: an intelligent agent based middleware for wireless sensor networks in *Proceedings of the 5th WSEAS International Conference on Application of Electrical Engineering*:225–231 2006.

[45] Römer K, Frank C, Marrón PJ, Becker C. Generic role assignment for wireless sensor networks in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*:2ACM 2004.

[46] Gibbons PB, Karp B, Ke Y, Nath S, Seshan S. Irisnet: An architecture for a worldwide sensor web *Pervasive Computing, IEEE.* 2003;2:22–33.

[47] López TS, Ranasinghe DC, Harrison M, McFarlane D. Adding sense to the Internet of Things *Personal and Ubiquitous Computing.* 2012;16:291–308.

[48] López TS, Ranasinghe DC, Patkai B, McFarlane D. Taxonomy, technology and applications of smart objects *Information Systems Frontiers.* 2011;13:281–300.

[49] Murdoch O. *Middleware and Programming Support Bridging the Cyber-Physical-Social Divide.* PhD thesisUniversity College Dublin 2014.

[50] Gorgu L, Kroon B, Campbell AG, OHare GMP. Enabling a Mobile, Dynamic and Heterogeneous Discovery Service in a Sensor Web by Using AndroSIXTH in *Ambient Intelligence* (Augusto JC, Wichert R, Collier R, Keyson D, Salah A, Tan AH. , eds.);8309 of *Lecture Notes in Computer Science*:287-292Springer International Publishing 2013.

[51] Aberer K, Hauswirth M, Salehi A. Global Sensor Networks in *School Comput. Commun. Sci., Ecole Polytechnique Federale de Lausanne*Lausanne, Switzerland: EPFL 2006.

[52] Perera C, Jayaraman PP, Zaslavsky A, Christen P, Georgakopoulos D. Mosden: An internet of things middleware for resource constrained mobile devices in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*:1053–1062IEEE 2014.

[53] Russell S. *Real-time monitoring and validation of waste transportation using intelligent agents and pattern recognition.* PhD thesisUniversity College Dublin 2014.

[54] O'Sullivan T, Muldoon C, Xu L, O'Grady M, O'Hare GMP. Deployment of an autonomic home energy management system in *18th IEEE International Symposium on Consumer Electronics (ISCE 2014)* 2014.

[55] Kazmi AH, O'Grady MJ, O'Hare GMP. Energy Management in the Smart Home in *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*:480–486IEEE 2013.

[56] O'Grady MJ, Muldoon C, Carr D, Wan J, Kroon B, OHare GMP. Intelligent Sensing for Citizen Science *Mobile Networks and Applications.* 2016;21:375–385.

[57] Viani F, Robol F, Polo A, Rocca P, Oliveri G, Massa A. Wireless architectures for heterogeneous sensing in smart home applications: Concepts and real implementation *Proceedings of the IEEE.* 2013;101:2381–2396.

[58] Martin R. OO design quality metrics *An analysis of dependencies.* 1994.

[59] Bangor A, Kortum PT, Miller JT. An Empirical Evaluation of the System Usability Scale *International Journal of Human–Computer Interaction.* 2008;24:574–594.

[60] Sauro J. Measuring usability with the system usability scale (SUS) 2011.