Simpleware Device Surrogates

Enabling high-level description & interaction with resource constrained devices

(Presented at IE10, Kuala Lumpur, Malaysia 19-21 July 2010)

James Dooley
Vic Callaghan
Hani Hagras

Essex Universty
Colchester
Essex
jpdool@essex.ac.uk

***Abstract* —** *As the Home Area Network (HAN) evolves, there is an increase in both the number and diversity of device deployment. This includes embedded devices whose resource constraints do not permit the efficient performance of high level middleware functionality. We herein present the functionality and knowledge representations required to enable such "simpleware" devices to be dynamically represented by proxy within our Nexus middleware framework. We also present a use case to illustrate the proposed solution.*

Keywords: Sensor Network; Home Area Network; Middleware;

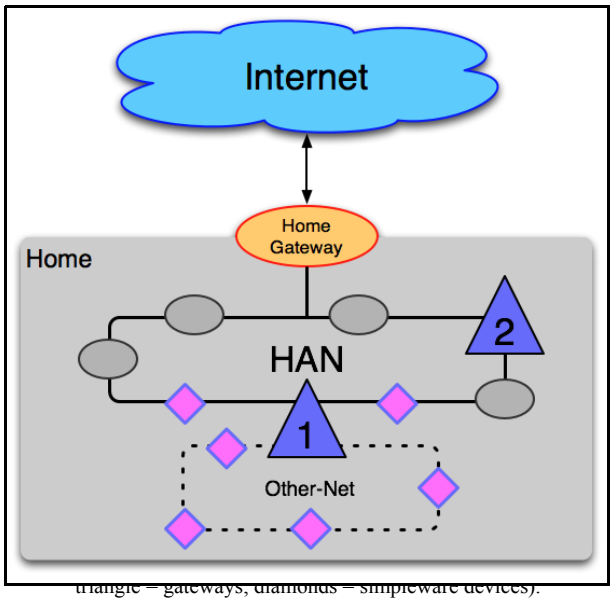
I. Introduction

As a subset of the wider pervasive / ubiquitous computing research context [1][2], the *"smart home"* vision seeks to enable the deployment and intelligent automation of digital resources in the home [3][4]. These resources reside on tangible computing devices which are physically distributed throughout the home and interconnected by a common network infrastructure we call the *"Home Area Network"* (HAN) [5][6][7] as shown in Figure 1.

A. Embedded devices in the Home Area Network

The cutting edge of computing technology has traditionally been viewed as that of computational throughput. In todays world, there are now other criteria (such as physical size and operational power consumption) by which computing technology is measured. This is driven by the need for small and efficient electronic devices on which advanced software can be executed. These *"embedded devices"* are deployed into an ever increasing number of physical objects that surround us in our everyday lives, thus validating Mark Weisers vision of ubiquitous technology *"receding into the background"* [1]. These devices join the traditional Desktop PC and games consoles on the HAN to result in a dynamic and distributed computing system that is feature rich and multi-purpose.

As with distributed systems in general [8], heterogeneity is a key property of the HAN device set (i.e devices in the home are different in terms of their hardware and software). This includes resource constrained devices that are unable to execute complex software (e.g middleware). Such devices are almost exclusively used for sensing and acting purposes due to low computational performance and power requirements (in some cases harvesting power from their surroundings or taking several years to exhaust their batteries).



triangle = gateways, diamonds = simpleware devices).

B. The Digital Divide Problem

Figure 1 shows the *"home gateway"* which can permit regulated external access to / from the *"connected home"*[2] and its digital resources (known as *"entities"* in our Nexus framework). These entities are hosted by devices that populate the HAN (shown as ovals and triangles).

As with our previous Nexus publications [6][7][9], the HAN is shown in Figure 1 without any topology details (switches, routers, access points, ethernet, wi-fi, etc.). This reflects our perspective of the device interconnection at the transport layer (layer 4) of the well known OSI model. That is, we do not care so much on how the lower level technologies deliver data between devices, we care only that it is achieved. Building on this; communications in the Nexus framework are established with any entity in the same home deployment through several well defined protocols (including discovery, presence, query, knowledge retrieval, invocations and eventing) that form a functional middleware.

1. This phrase is often used to describe the divide between people that are able and unable to use desktop computer technology [10].

2. A home to which internet access is available by some means.

It is, however, inevitable that some devices are unable to fulfill the execution requirements imposed by the Nexus middleware (as indicated by diamonds in Figure 1). The reasons for this are varied but include :

- **Incapacity :** The device may be unable to execute the necessary software due to a lack of some resource such as processing capability, memory, storage, etc.
- **Designed Inability :** The device is capable of executing the Nexus middleware, but does not as the result of some design decision.
- **Isolation :** The communications capabilities available to the device (e.g USB) are unable to route messages to the HAN. Thus communication is not possible either to or from other Nexus participants.

The resulting *"digital divide"* provides a separation in which certain devices are currently excluded from the Nexus sphere of communication. We herein label these sub-middleware capable devices as having *"simpleware"*. Solving this problem requires a solution that :

> *"Enables simpleware devices to publish fully functional entities that exhibit behaviour and structure as defined by the Entity abstract data type of the Nexus framework"*

It is implicit (by the very nature of simpleware devices) that the solution need not implement a full middleware stack.

C. This Paper

Motivated by increasing the range of embedded devices that are eligible for inclusion in Nexus deployments; The purpose of this paper is to propose, describe and demonstrate a solution to the previously described *"digital divide"* problem within the context of the Nexus framework.

We begin by examining some related work in the area of sensor networks to tap into existing knowledge concerning the enablement of small resource limited devices that need to communicate their data to consumers. The discourse continues to explore device descriptions which examine what information is communicated and the form (structure) in which that information is represented.

Following the related work section, we provide an overview of the relevant Nexus framework features. This is intended to prime the reader and set the scene for the finer details of our proposed solution which is then presented.

Finally, as an aide to validation and depth of understanding, we provide the description of a real use case and a conclusion section that summarises the consequences / shortcomings of the proposed solution.

II. Related Work

A. Sensor Networks

Motivated by a great many applications (including habitat monitoring [11]), much research in the area of sensor networks has focused on the low level communication of both raw and processed data within the sensor network infrastructure (for example by using multi-hop routing [12]). In support of this, it has also been recognised that middleware offers *"a novel approach to fully meeting the design and implementation challenges of wireless sensor network technologies"* [13].

Existing middleware solutions for sensor networks adopt a wide array of approaches and entire detailed surveys exist that compare them [13][14][15]. For example, some approaches treat the entire sensor network as a virtual database [16][17][18], while others task (and optimise) the sensor network specific to individual application requirements. In one case [19] the sensor network appears as a unified Java virtual machine (called a Single System Image) in which an object-oriented java program can be efficiently distrinuted and executed. Showing some similarities, EnviroTrack [20] provides a novel abstraction layer that hides the complexities of managing the sensor network and exposes objects which applications can handle conveniently and natively.

A common pattern (Figure 2) has emerged (especially where the network in which the application resides is logically separate to the sensor network) in which a "gateway" (a.k.a *"proxy"*, a.k.a *"base station"*) acts as an interface for access from higher level infrastructure[3] [21][22][23].

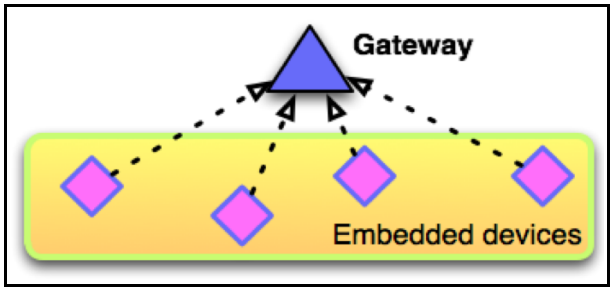


Figure 2: The gateway pattern (triangle = gateway; diamonds = simpleware devices)

B. Describing Devices

There have been many attempts to codify, classify or otherwise label groups of devices. These have been presented in the form of taxonomies, ontologies or general descriptions. Some focus on one set of criterion in establishing their classification scheme, such as the work reported by Weiser in one of the baptising publications [1] of ubiquitous computing. Therein, a very simple classification of three device types was provided based on size : tabs (inch scale), pads (foot scale) and boards (yard scale). Each of these classifications had some implied usability in the environment (for example, pads were described as being *"scrap computers ... that can be grabbed and used anywhere; they have no individualized identity or importance"*) that Weiser used as a descriptive aide (it is therefore neither exhaustive, nor complete).

In the years since Weisers classification, more formal ontologies have emerged for expressing device description in a machine readable way [24][25][26][27]. Notably, [26] extends the frame based FIPA device ontology [27] and provides a description that is split into five logical parts : device, hardware and software descriptions, device status and service.

3. Outside the academic sphere, the award winning *"pachube"* project deserves recognition here for internet scale, multi-site sensor monitoring.

Suitable to the smart home vision, the AMIGO project [28] has produced a consumer electronic device ontology based around a concept of device types (for example *"audio device"*), this imposes a certain level of *"ontological commitment"* [29], that our Nexus framework seeks to avoid.

III. An overview of the Nexus Framework

Nexus is our information centric, distributed HAN middleware that builds on an abstraction of entities and facets.

A. Entities

An entity is a self describing, type independent (physical, virtual, conceptual, etc.) abstract representation of an *"object"* that can be dynamically discovered and interacted with (using action invocation, event notification and facet retrieval). Each entity has a unique and immutable ID that is URN encoded and namespace qualified, where the namespace provides a hint as to the broad type of the entity (e.g the identity *"urn:nexus:user:james"* indicates the *"james"*[4] user entity).

B. Facets

An individual facet is a mutable XML document that has a URN encoded name unique in the entity facet set (e.g. exactly zero or one facets named *"urn:nexus:facet:basic"*, may exist per entity). Each facet document has a common root element (named *"Facet"*) with two attributes : parent entity ID and facet name. Any content is permitted in this element allowing existing ontologies and XML schemas to be reused in our framework. The set of facets that an entity possesses collectively form a novel self descriptive information space.

C. Actions

Our framework shares a similar invokable action concept to other middleware (such as UPnP). Action definitions (in the *"entity"* facet) declare named and typed (integer, boolean, string, etc.) input and output variables. Invocation is achieved over the network with all input variable values set, and returns a completed set of output variable values (or an error).

D. Events

An individual event is URN named and consists of a parameter set where each parameter is named and typed. Events are sent asynchronously through URN named event channels (an entity may have zero or more event channels) that are described in the *"entity"* facet and can be subscribed across the network.

E. Classification

The general mantra concerning the classification of entities in the Nexus framework is to avoid the static definition of types / classes and allow observers to dynamically specify their own criteria. That is, an entity is neither required nor inhibited from declaring any form of *"type" / "class"* (the previously described entity ID namespace is an exception).

Consequently; It is necessary for a dynamic *"classification operation"* to exist whereby classification rules are executed against the information space (facets) of an entity. If all the classification rules are satisfied by an entity, the entity is said to be of that class. More formally :

$$\forall (r \in R : match(r, e)) \Leftrightarrow C(e, R)$$

where :

r	: A single classification rule,
R	: A set of classification rules,
match(e, r)	: The entity **e** satisfies the classification rule **r**,
C(e, R)	: The entity e satisfies the classification rule set **R** and is therefore of the associated *"class"*.

Using this, the following inclusion map would derive a sub-set of members (**RE**) from an entity set (**E**), where all the entity members $(e \in \mathrm{RE})$ are of the same class (according to the classification rule set **R**) :

$$C : \mathrm{RE} \rightarrow E, \quad C(e, R) = e$$

IV. Our Method

Similar to other related work, our method uses a gateway proxy approach. Specific to our solution, the proxy communicates with simpleware devices using messages through the multi-transport iris component of our framework. As a result, the proxy creates, configures and publishes surrogate entities to the HAN as shown in Figure 3.

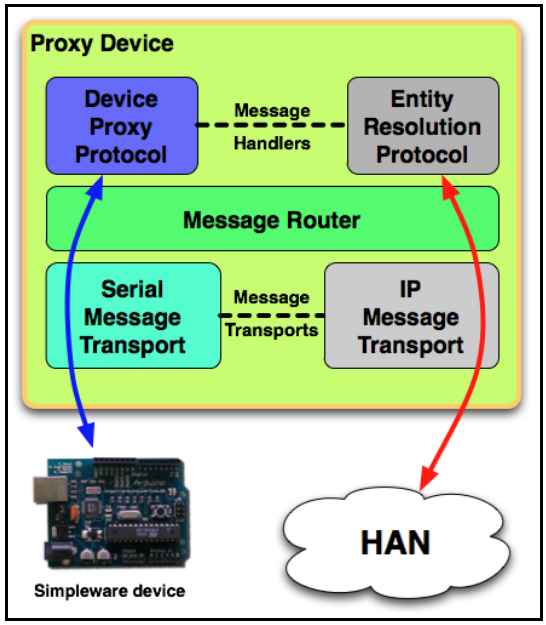


Figure 3: Iris components in the Proxy device.

A. The Device Proxy Protocol (DPP)

The simpleware devices implement and behave according to a finite state machine as shown in Figure 4, where the states are described as :

- **Orphaned :** The initial state in which the device is not associated with any proxy and broadcasts regular

4. In reality, a unique value such as a UUID value would be used here.

PROXY_DISCOVERY messages. Upon recieving a response, the simpleware device will request service by sending a directed **PROXY_REQUEST** message (that includes the id of the entity to create). Only if the proxy accepts (by a **PROXY_ASSOCIATED** message), will the simpleware device change state to Associated:Configuration. A simpleware device can return to this state at any time due to failure or a **RESET** message from the proxy.

- **Associated:Configuration :** In this state, the simpleware device can configure the structure (facets, event channels and actions) of its surrogate entity. This must be done in an unpublished state to avoid state synchronisation errors or a backlog of requests which may stress the overall system (for example, in our experimentation, the relatively slow process of retrieving a facet template from EEPROM and then uploading it to the proxy caused considerable action invocation latency, thus reducing application QoS).
- **Associated:Operational :** In this state the surrogate entity is fully functional and published to the HAN. The proxy maintains a queue from which action invocations are sequentially forwarded to the simpleware device (our experimentation shows that without this, the device can become overwhelmed). All other HAN functionality is internally handled by the proxy. At any time (unless occupied by an action invocation) the simpleware device can update a facet held by the proxy using xquery (for example to reflect a change in sensor value) or generate an event for distribution by the proxy to event subscribers.

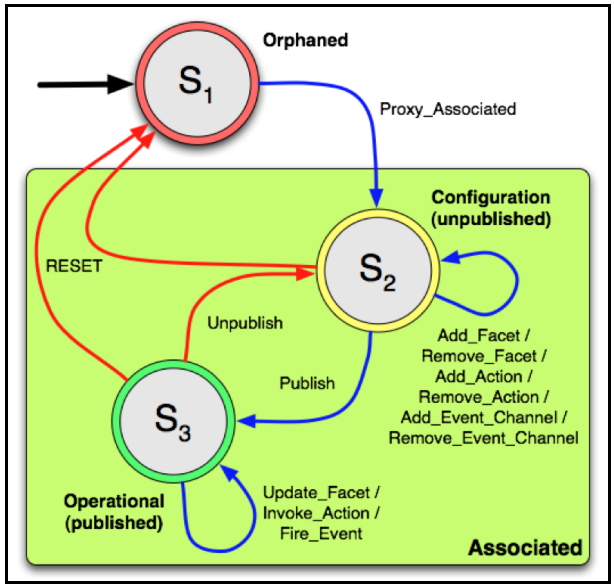


Figure 4: Finite State Machine of an embedded device.

B. *The Nexus Device Ontology*

Every surrogate entity that is created by a proxy for a simpleware device is tagged with a *"device"* facet (named *"urn:nexus:facet:device"*) to indicate what the entity represents (as opposed to any other kind of entity). The use of this facet is not exclusive to simpleware devices and can be used in any entity that represents a computational device.

As previously discussed; Nexus does not seek to establish static / finite taxonomies of entities. Therefore our device ontology seeks to be descriptive, allowing applications to carry out dynamic classification against its content. Figure 5 shows a partially completed example device facet for an entity.

```
<Facet    facet-name="urn:nexus:facet:device"
          parent-id="urn:nexus:device:1bd3cd77-be94-4d0e-
                                    9d2a-7a1d86dfc80f">
 <DeviceDescription>
   <DeviceInfo>
     <ProductName>
       Arduino Duemilanove
     </ProductName>
     <Manufacturer>tinker.it</Manufacturer>
     <ManufacturerUrl>
       http://www.tinker.it/
     </ManufacturerUrl>
     <Version>
       <MajorVersion>1</MajorVersion>
       <MinorVersion>0</MinorVersion>
     </Version>
   </DeviceInfo>
   <PowerSaving> … </PowerSaving>
   <Hardware> … </Hardware>
 </DeviceDescription>
</Facet>
```

Figure 5: A partial device facet example (xml namespaces removed for clarity).

Within the root *"Facet"* element of the document, the single *"DeviceDescription"* element contains our ontology and has three sub-elements :

- ***"DeviceInfo" :*** Contains a descriptive account of the device intended for viewing by human users. A name, general descriptive sentence, icon and graphic elements are missing here as they all appear elsewhere in another generic entity facet (the *"basic"* facet).
- ***"PowerSaving" :*** Reserved for future use. We intend to describe wake-on-lan like behaviour so devices can be put to sleep and awoken, thus making the home more energy efficient (a pervasive topic in todays world).
- ***"Hardware" :*** Describes the core hardware of the device[5] (CPU, RAM, etc.), but not capabilities (such as display, audio, etc.) which should be described in a more abstract way in other facets (e.g a *"media-playback"* facet, or *"user-interface"* facet).

Driven by *ad-hoc* application developments that need some information and thus ask the question *"in which facet does that belong?"*, we anticipate that this ontology will expand. Care must be taken that the information to be included is appropriate. For example; physical size, is not exclusive to a computational device, and therefore belongs in another facet.

5. Removed due to space constraints.

V. Use Case

A. Overview of the Use Case

The purpose of this use case is to demonstrate a user initiated asynchronous event that is generated from a simple sensor device (included in the Nexus sphere of communications by proxy), to which an application reacts. As a whole this use case demonstrates many features of the Nexus framework (including actions, eventing, discovery, and facets), but we would like to emphasise the concept of *"Recombinant applications"* which can be composed from multiple re-usable functional blocks that are discovered and bound at runtime. The use-case components are :

1. **RF-ID reader :** This simpleware device is composed of an 8-bit micro-controller with and RF-ID reader module. Communications are achieved with the proxy device via USB.
2. **Alphanumeric LCD display :** Another USB simpleware device that has a "setText" action.
3. **Proxy Device :** This is the higher capability device that implements the proxy side of the DPP over USB.
4. **Controller Application :** Subscribes to the RF-ID reader and has in built logic to control other HAN entities in reaction to RF-ID tag events.
5. **Music Source Entity :** An entity representing a music source (MP3 file), complete with *"media facet"* describing the encoded music and an alias entry in the *"Basic facet"* that matches an RF-ID tag.
6. **Speaker Entity :** An entity that streams an audio source over the network, decodes it and renders it as human audible sound.

B. Component Configuration

In this use case there are two simpleware devices (RF-ID reader and a text display) that require a proxy to participate in the Nexus deployment. They both communicate over USB and successfully follow the Device Proxy Protocol. This results in the proxy publishing a surrogate of each to the HAN.

When the control application starts, it searches for RFID readers and subscribes to them (for event notifications). It may also search the HAN for the textual LCD and speakers, but the Nexus framework performs this process exceptionally fast and so it can be performed as needed in later steps.

C. Use Case Execution[6]

When a tag is passed in front of the RF-ID reader, it notifies the application controller (that has previously subscribed to the reader) with an event that contains, among other things, the unique tag ID. This is labelled (1) in Figure 6.

Being stateless, the controller does not internally know what entity the tag ID relates to. Therefore, the controller must submit a query to the HAN using the Nexus discovery protocols to find the entity to which the tag is associated, labelled (2) in Figure 6. For simplicity in our use case, the tag ID matches exactly one entity which is an MP3 encoded audio source (i.e music file).

The application searches the HAN for a suitable *"speaker"* entity (or uses the result of a previous search) and invokes the *"setSource"* action (with the URL extracted from the *"media facet"* of the entity identified in the previous step) to tell the speakers what to render. The *"play"* action is then invoked on the same speaker entity to start streaming and rendering the media, labelled (3) in Figure 6.

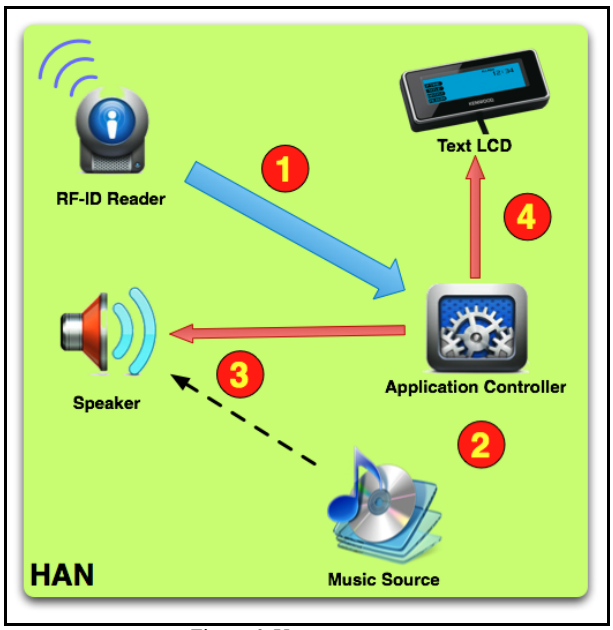


Figure 6: Use case process.

If a search of the HAN also yields an appropriate textual display, then the artist name and track name are extracted from the media facet of the entity identified in step (2) and sent to the found display using its *"setText"* action. This is labelled (4) in Figure 6.

VI. Conclusions

Motivated by the inability of certain simpleware devices to fulfill the information and functional needs of high-level software within the smart home context. This paper has proposed a methodology for certain devices to act as proxy and provide surrogates for the resources that simpleware devices possess.

Previous to the solution proposed in this paper, any device that wished to participate in the Nexus framework would need to fulfill the requirements of executing high-level middleware. Experience has shown those requirements to be between 100 and 200Mhz with ~32Mb of RAM and a few hundred megabytes of storage space (for an embedded linux OS, java virtual machine, middleware, supporting libraries and then application code). Our proposed solution now reduces that requirement to any device that can implement the DPP state machine and protocol. Experimentation reveals that inexpensive devices with 8-bit micro-controllers (such as the

6. Those readers familiar with the UPnP A/V specification will notice a similar pattern in this use case.

open source *"Arduino"* range) are more than capable of these requirements. This means that deployments can have a higher granularity (more devices individually providing small functional contributions) and be cheaper to deploy (cheaper hardware) / maintain (lower power requirements).

There is however a price to pay for the drastic reduction in required device functionality; Although proxies are dynamically discovered at runtime, the solution relies on a client-server model (between simpleware devices and a proxy). There are two primary problems that could exist with this centralised model :

1. If the proxy device fails for some reason, then all the connected simpleware devices that it is responsible for disappear from the Nexus deployment. The DPP state machine has built in failure recovery for this and will attempt to find an alternative proxy if available.
2. The proxy device must be suitably capable of providing enough surrogates for the intended number of simpleware devices. In some deployments this number may raise to a point where the proxy device is overwhelmed and fails (i.e scalability limits).

The novelty of this paper and the solution it presents lie in the specific way that simpleware devices are transposed by proxies into the HAN as configurable surrogate entities that present their own descriptions as component facets.

References

[1] M. Weiser. *"The Computer for the Twenty-First Century,"* Scientific American, pp. 94-10, September 1991.

[2] T. Hoare and R. Milner. *"Grand Challenges for Computing Research"*. The Computer Journal. Vol. 48, no. 1, pp. 49–52. The British Computer Society: London. 2005.

[3] W.K. Edwards and R. Grinter. *"At Home with Ubiquitous Computing: Seven Challenges"*. Proceedings of the Conference on Ubiquitous Computing (Ubicomp 2001). Atlanta, GA. 2001.

[4] S. Helal, W. Mannm H. El-Zabadani, J. King, Y. Kaddoura and E. Jansen. *"The Gator Tech Smart House: A Programmable Pervasive Space"*. Computer. IEEE Computer society press. Vol.38, Issue 3, pp.50-60. 2005.

[5] W. Treese. *"Putting it Together : The Home Area Network"*. Networker. 2000.

[6] J. Dooley, V. Callaghan, H. Hagras and P. Bull. *"Discovering the Home"*. 5th International Conference on Intelligent Environments (IE'09). Barcelona, July 2009.

[7] J. Dooley, V. Callaghan, H. Hagras and P. Bull.. *"Discovering the Home : Advanced Concepts"*. 2nd International Conference on the Applications of Digital Information and Web Technologies (ICADIWT). London, August 2009.

[8] G. Coulouris, J. Dollimore and T. Kindberg. *"Distributed Systems : Concepts and Design"* third edition. Addison Wesley. 2001.

[9] J. Dooley, V. Callaghan, H. Hagras and P. Bull. *"Resource Discovery in the Home Area Network"*. International Journal of Information studies, vol. 1, no. 4, pp. 251-262. October 2009.

[10] P. Cochrane. *"Peter Cochrane's Blog: Digital divide? What digital divide?,"* Silicon.com. November 2006.

[11] A. Cerp, J. Elson, D. Estrin, L. Girod, M. Hamilton and J. Zhao. *"Habitat Monitoring: Application Driver for Wireless Communication Technology,"* Proc. ACM SIGCOMM Workshop Data Comm., ACM Press, pp. 20-41. 2001.

[12] A. Woo, T. Tong and D. Culler. *"Taming the underlying challenges of reliable multihop routing in sensor networks"*. In Proceedings of the 1st Intl. Conf. on Embedded Networked Sensor Systems. SenSys '03. pp. 14-27. ACM, New York, NY. November, 2003.

[13] S. Hadim and N. Mohamed, *"Middleware: Middleware Challenges and Approaches for Wireless Sensor Networks,"* IEEE Distributed Systems Online, vol. 7, no.3, pp.1. March, 2006.

[14] M.M. Wang, J.N. Cao, J. Li and and S.K. Das. *"Middleware for wireless sensor networks: A survey"*. Journal of Computer Science and Technology. Vol. 23, no. 3, pp. 305–326. May 2008.

[15] K. Henricksen and R. Robinson. *"A survey of middleware for sensor networks: state-of-the-art and future directions"*. In Proceedings of the international Workshop on Middleware For Sensor Networks (MidSens '06). vol. 218, pp. 60-65. ACM, New York, NY. November, 2006.

[16] P. Bonnet, J. Gehrke and P. Seshadri. *"Towards Sensor Database Systems"*. In Proceedings of the Second international Conference on Mobile Data Management. Lecture Notes In Computer Science, vol. 1987. pp. 3-14. Springer-Verlag. January, 2001.

[17] S.R. Madden, M.J Franklin, J.M. Hellerstein and W. Hong. *"TinyDB: an acquisitional query processing system for sensor networks"*. ACM Trans. Database Syst. Vol. 30, no.1, pp. 122-173. March 2005.

[18] C. Srisathapornphat, C. Jaikaeo and C. Shen. *"Sensor Information Networking Architecture and Applications"*. IEEE Personal Communications, Vol. 8. pp. 52-59. 2001.

[19] R. Barr, J.C Bicket, D.S Dantas, B. Du, T.W. Kim, B. Zhou, and E.G Sirer. *"On the need for system-level support for ad hoc and sensor networks"*. SIGOPS Operating Systems Review. Vol.36, no.2, pp. 1-5. April 2002.

[20] T. Abdelzaher, B. Blum, Q. Cao, D. Evans, J. George, S. George, T. He, L. Luo, S. Son, R. Stoleru, J. Stankovic and A. Wood. *"EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks"*, IEEE International Conference on Distributed Computing Systems. March 2004.

[21] K. Aberer, M. Hauswirth, and A. Salehi. *"Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks"*. In Proceedings of the 2007 international Conference on Mobile Data Management (MDM'07). IEEE Computer Society, pp. 198-205. May 2007.

[22] H. Song. *"Implementing a wireless base station for a sensor network"*, Thesis (M. Eng.), Massachusetts Institute of Technology, Dept. of Civil and Environmental Engineering, 2004.

[23] V. Trifa, S. Wieland, D. Guinard and T.M. Bohnert. *"Design and Implementation of a Gateway for Web-based Interaction and Management of Embedded Devices"*, Proceedings of the 2nd International Workshop on Sensor Network Engineering (IWSNE'09). June, 2009.

[24] C. Chrysoulas, G. Koumoutsos, S. Denazis, K. Thramboulidis and O. Koufopavlou. *"Dynamic Service Deployment using an Ontologybased Description of Devices and Services,"* International Conference on Networking and Services (ICNS '07), pp. 80 2007.

[25] M. Sarnovsky, P. Kostelnik, J. Hreno, P. Butka. *"Device Description in HYDRA Middleware"*, In Proceedings of the 2nd Workshop on Intelligent and Knowledge oriented Technologies 2007 (WIKT'07), pp.71-74. November 2007 .

[26] A. Bandara, T.R. Payne, D. de Roure, and G. Clemo. *"An Ontological Framework for Semantic Description of Devices",* International Semantic Web Conference (ISWC). 2004.

[27] *"FIPA Device Ontology Specification"*, Foundation For Intelligent Physical Agents (FIPA). Last retrieved on 20/02/2010. http://www.fipa.org/specs/fipa00091/

[28] N. Georgantas, et al. *"Amigo middleware core: Prototype implementation and documentation"*, IST Amigo Project, deliverable 3.2. Technical report, IST-2004-004182 (2006).

[29] R. Davis, H. Shrobe, and P. Szolovits. *"What is a Knowledge Representation?"*. AI Magazine, vol. 14, no. 1, pp. 17-33. 1993.