An Open and Integrated Management Platform for Wireless Sensor Networks

M. Kalochristianakis, V. Gkamas, G. Mylonas, S. Nikoletseas, E. Varvarigos
Research Academic Computer Technology Institute,
Department of Computer Engineering and Informatics,
University of Patras, Rio, Greece
{kalohr, vgkamas, mylonasg, nikole, manos}@cti.gr

J. Rolim
Department of Computer Science,
University of Geneva, Switzerland
rolim@cui.unige.ch

***Abstract*— We present the conceptual basis and the initial planning for an open source management architecture for wireless sensor networks (WSN). Although there is an abundance of open source tools serving the administrative needs of WSN deployments, there is a lack of tools or platforms for high level integrated WSN management. This is because of a variety of factors, including the lack of open source management tools, the immaturity of tools that offer manageability for WSNs, the limited high level management capabilities of sensor devices and architectures, and the lack of standardization. The current work is, to our knowledge, the first effort to conceptualize, formalize and design a remote, integrated management platform for the support of WSN research laboratories. The platform is based on the integration and extension of two innovative platforms: jWebDust, a WSN operation and management platform, and OpenRSM, an open source integrated remote systems and network management platform. The proposed system architecture can support several levels of integration (infrastructure management, functionality integration, firmware management), corresponding to different use-cases and application settings.**

Keywords- remote management, wireless sensor networks, pervasive communications, open source.

I. INTRODUCTION

Wireless sensor networks typically consist of spatially distributed autonomous devices that use sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion, acceleration or pollutants, at different locations. They are typically being used in the scientific, medical, commercial, and military domains in order to perform sensing and monitoring in the physical world, such as habitat monitoring, object tracking industrial automation, fire detection, traffic monitoring, etc. The effective and flexible integration of sensory systems with interconnection networks and grid technologies is a critical step towards developing major pervasive computing and communication services and applications. Because of sensor network inherent characteristics such as heterogeneity, limited bandwidth and energy constraints, WSN management and monitoring architectures need to:

- provide a wide range of services so as to cover the whole spectrum of supported WSN architectures
- minimize the overall implementation effort and manage WSNs uniformly to the degree possible
- reduce the needs for network administration and sensor node software management
- expose web interfaces for WSNs management

Most management tools for WSNs provide network management functionality; they primarily support visualization based on collecting data from the network of sensors using data logging on a designated WSN gateway node. The nodes poll their sensors at a user-configurable sampling rate and send them to the gateway using multi-hop protocols; readings from the network are typically stored in a relational database for further processing. Most tools that rely on mechanisms such as the above must be considered monitoring or supervisory tools rather than management tools, since they lack functionality for sensor state change or WSN configuration. Moreover, such tools are typically installed and configured to monitor a single WSN installation.

A lot of administrative and management tasks that are typically supported by standard management tools for workstations would be valuable to WSNs operators and administrators. Such tasks are:

- multiple WSNs management and monitoring
- web-based control
- remote command execution or remote configuration
- software / firmware upgrade
- reporting
- grouping

An integrated management platform for WSN must include all the above tasks. The scenery of open source tools for WSN management is analogous to the one for remote management of general systems, in terms of product maturity and market fragmentation. There are very few integrated management systems offered with open source licensing schemes and none of them offers WSN interfacing.

This paper presents an architecture that is capable of delivering remote management functionality to wireless sensor networks. The idea is based on the combination of jWebDust

[1], a software environment that allows the implementation of customized applications for wireless sensor networks, and OpenRSM [2], one of the very few open source remote management platforms. In order to bring management to WSN, it is essential to automate routine WSN practice. jWebDust provides WSN management functionality, including querying, monitoring, data logging and visualization. jWebDust employs an extendable architecture and provides easy interfacing and API. OpenRSM provides the implemented distributed logic suitable to deliver services, such as inventory and asset management, software delivery, remote control and network monitoring, all integrated in one environment.

The paper is organized as follows. Section II describes the related state of the art regarding monitoring and administration tools for wireless sensor networks. Section III presents an overview of the jWebDust and OpenRSM architectures. The proposed integrated architecture and its applicability to relevant use-case scenarios are presented in Section IV. Finally, conclusions and future work are discussed in Section V.

II. Related Work

Software environments that provide the necessary tools and operations to allow the monitoring and administration of a wide range of WSN's applications are relatively few and to our best knowledge, none of them integrates all the desired management functionality in a single environment.

TinyDB [3] is an example of an application that allows multiple concurrent queries, event-based queries and time synchronization through an extensible framework that supports adding new sensor types and event types. The central idea of TinyDB is to provide an SQL-like interface to the programmer that makes the wireless sensor network look like an RDBMS (Relational Database Management System). Tiny Application Sensor Kit (TASK) [4] is built on top of TinyDB in order to further simplify application deployment and development, and to provide additional management capabilities. TinyDB is distributed with TinyOS [5], the de-facto operating system used in WSNs so far.

MoteWorks [6] is a commercial WSN management product offered by Crossbow. MoteWorks is built on n-tier architecture model and offers a number of standard WSN functionalities, along with APIs for easy interfacing and integration with other software. It also supports numerous sensor node hardware platforms. Mote-VIEW [7], a part of MoteWorks, is a platform that provides visualization tools to the user, combined with a data logger that runs on the sensor network gateway. The logger listens to readings arriving from the network through a control center attached to the gateway and stores them in a relational database.

ArchRock [8] is an another example of a company which produces products that offer WSN management capabilities. Like MoteWorks, it offers sensing functionalities, and APIs to interface with. It also offers integration of the sensor network with IP networks, using a 6LoWPAN network stack inside the sensor network, making it easier to interface with the nodes.

TWIST [9] and MoteLab [10] are examples of testbed deployment management environments, targeted toward research teams. They provide capabilities as job scheduling (i.e., binary code updates from different users), network activity sniffing, etc. However, these tools are designed for specific-purpose environments that focus on the testing methods of WSN applications. Moreover, special hardware and network topologies are required, making TWIST and MoteLab not well-suited for general network management.

MANA [11] is an example of a other management architecture for WSNs. It provides functional, information, and physical management architectures that take into account specific characteristics of a WSN network. Some of them restrict physical resources such as energy and computing power, frequent reconfiguration and adaptation, and faults caused by unavailable nodes.

GSN [12] is another middleware architecture (extendible software infrastructure) for rapid deployment and integration of heterogeneous wireless sensor networks. GSN is tested with Mica2, Mica2Dot, TinyNodes, Wisenode, Wired & Wireless cameras, several RFID readers.

Finally, Hourglass [13] is an Internet-based infrastructure for connecting a wide range of sensors, services, and applications in a robust fashion. In Hourglass, streams of data elements generated from sensor networks are routed to one or more applications. The Hourglass infrastructure consists of an overlay network of well-connected dedicated machines that provides service registration, discovery, and routing of data streams from sensors to client applications.

The above platforms and tools provide specific administrative functionality, mostly in terms of development, and do not give an integrated remote management WSN environment, under which the overall administration and monitoring of the wireless sensor network can performed both in low and high level.

III. Overview of jWebDust and OpenRSM

In this section an overview of the jWebDust and OpenRSM platforms is given. Subsection A describes the jWebDust platform, while subsection B describes the OpenRSM platform.

A. *jWebdust overview*

jWebDust [1] differentiates the system into two main groups: the networked sensor devices that operate using TinyOS [5], and the rest of the network (e.g. control centers, database server, etc.) that is capable of executing Java code. Both system groups use an open architecture implementing a component-based architecture. The component interface and the exchange of data over broadly used protocols provide increased portability. This implies that the system can be used over different machine architectures as well as operating system and server technologies.

From a high-level perspective, the components that make up jWebDust are organized using the n-tier application model. We distinguish the following five tiers:

- the Sensor Tier that consists of one or more wireless sensor networks deployed to areas of interest,

- the Control Tier that corresponds to the control centers where the wireless sensor networks report the realization of events,
- the Data Tier responsible for storing the information extracted from the wireless sensor network(s),
- the Middle Tier that is responsible for processing the data to generate statistics and other meaningful information and
- the Presentation Tier that interfaces the information with the final user in an easy way based on the capabilities of the user's machine.

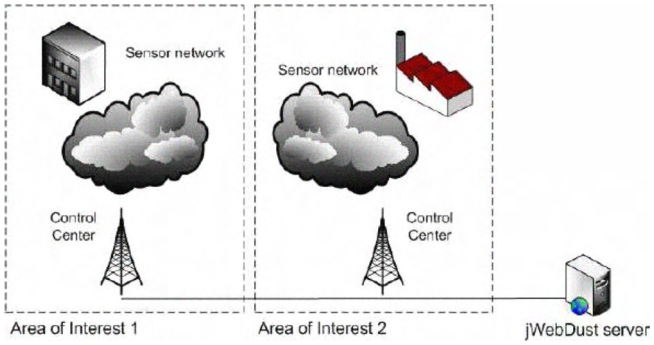

Figure 1. Management of multiple WSNs from a single, unified, virtual sensor network using jWebDust

The component-based architecture enables the jWebDust system to gain control over all critical resources, required by the implemented functionality (connection to database, communication with a server, code execution). The autonomy of the components makes the system independent of the machine architecture, allowing it to be executed over any favored machine architectures using best-of-breed server technologies.

jWebDust uses a simple Discovery Service [1] in order to keep track of the sensor nodes that participate in the wireless sensor network and their technical characteristics (e.g., type of sensors attached to each device, available power, etc) [14].

A distinct feature of jWebDust is its ability to manage multiple wireless sensor networks, each with a different control center, under a common installation (Figure 1). This is done by introducing virtual sensor networks that hide the actual topology and allow users to control the sensor nodes as if they were deployed under a single, unified, sensor network. This abstraction significantly reduces the overhead of administering multiple networks. Furthermore, the idea of a unified, virtual sensor network allows the integration of totally heterogeneous sensor networks, i.e. not only regarding different kinds of sensors attached to the sensor nodes of the network, but also different kinds of CPU architectures.

B. OpenRSM overview

The Open Source Remote System Management (OpenRSM) [2] is a pioneering initiative in the lightweight management software area of open source enterprise management (EMS) tools [15]. The initiative has been based on the observation that most of the components comprising an open source EMS tool can be developed by extending existing rated components offered by the open source community [16], [17].

In principle, OpenRSM needs to be simple and lightweight [18] so that it can be used by naive end users, not specialized in the use of management tools or asset reporting tools. OpenRSM was designed for fast and automated deployment, in order to cover the needs of administrators who manage very dynamic environments. The development model adopted for the OpenRSM system is open source development, in order to exploit the dynamics of projects that relate to management technologies [19], [20], [21], [22] and gain value from integration. The open source community has been put under scrutiny [23] in order to recover the open source management initiatives suitable for the purposes of OpenRSM [24], [25]. The architecture of the OpenRSM platform has been chosen to be modular in order to follow the logical categorization of entities involved [16] and to provide integration with other open source management tools. The implemented framework was based on the agent-server model [26], [27].

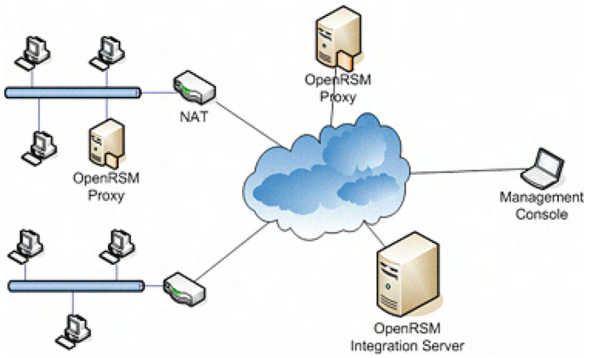

Figure 2. The OpenRSM system architecture

The components of the OpenRSM platform are the agent that makes target stations manageable, the graphical management console presented to the user and the server where processing logic is implemented. The OpenRSM agents are abstract, multiplatform, manageable entities that convey administrative actions coming from the OpenRSM server. Administrative actions originate from user actions on the OpenRSM management console, the component exposed to the end-users and the administrators of the service. Management commands are conveyed to the OpenRSM server and then to the agents. The architecture of the OpenRSM system is presented in Figure 2.

OpenRSM utilizes the power of third party tools that have been enhanced and integrated in order to deliver the following services:

- inventory and asset management
- software delivery
- remote desktop control
- network monitoring

Emphasis was given to system integration, extending from database backend migration, server and web content porting, and agent logic concatenation. The combined subsystems have been modified so that information can be shared among them. At the same time, the overall system has been designed so that it is subject to the minimum possible set of limitations. Its capabilities extend to managing any stations reachable through standard IP connectivity in a secure manner. The architecture at the server tier has been kept open and thus adaptable to any specific needs and business models [17]. Care has been taken so that OpenRSM can manage stations hidden behind the Network Address Translation (NAT) protocol, using a proxy server developed for that reason. The system also supports multiplatform systems management, and provides a multilingual user interface. The server has been put under stress and performance tests [2].

IV. THE CONCEPT OF INTEGRATED WSN MANAGEMENT

jWebDust and OpenRSM can complement each other in order to deliver integrated management to WSNs. With respect to the level of integration we distinguish the following management layers:

- WSN server / gateway infrastructure management
- WSN server functionality support, application programming interface (API) integration
- firmware management

A. Infrastructure-level management

OpenRSM can be primarily used to delivery infrastructure management to WSN servers that is, bring inventory and assets management, network monitoring, software delivery, and remote desktop control to laboratory servers. This level of integration is achieved by incorporating the OpenRSM agent within a jWebDust installation.

B. Functionality integration

OpenRSM can be further integrated with jWebDust. In particular, we propose an extension that allows to support jWebDust services, APIs and tools and to expose them as special-purpose remote management services. The OpenRSM will then be capable to schedule and synchronize the execution of jWebDust-specific jobs and provide users with the capability to define custom jWebDust-specific jobs. In OpenRSM terminology, jobs are entities abstracted in accordance to object oriented design principles. They are designed to be abstract system tasks (e.g., inventory, remote control, remote command, etc) or reusable user-created objects and play a central role in terms of usability, design efficiency and system scalability. Jobs can themselves be managed by administrators/users, decoupling their creation and execution stages. They can be grouped or dynamically created based on the attributes of managed systems. Each jWebDust-specific job will correspond to a distinct administrative task manageable within the jWebDust platform. OpenRSM will embody typical jWebDust template jobs. Users can utilize the template jobs to create custom ones. Jobs can be correlated with machines and then be submitted to the OpenRSM integration server; the server will forward them to the agent running at WSN's server/gateway where jobs will be executed as jWebDust procedures. The architecture of the integrated system is presented in Figure 3.

C. Firmware level management

In order to cover advanced needs for WSN management such as firmware development and deployment, a higher level of integration is required. In order to enable firmware construction, deployment and discovery of individual sensor nodes the integration must reach the firmware level. This will enable the management platform to deliver identifiable sensor nodes, capable to dynamically change their state and runtime environment in order to test new ideas, protocols and implementations. In order to achieve the above it is necessary for the management platform to:

- support the underlying development environment.
- design a framework for firmware construction that will support the desired functionality and integration characteristics.

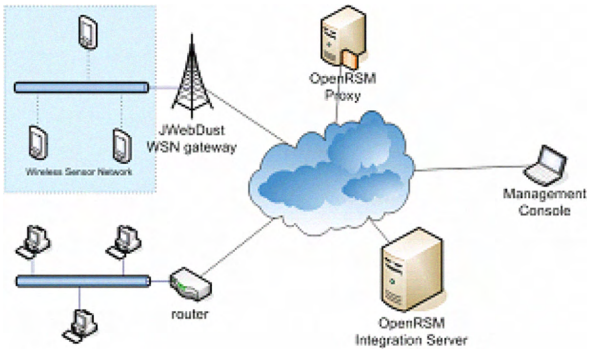

Figure 3. The proposed system architecture

The underlying development environment for sensor node firmware, TinyOS, offers development tools that can be integrated into custom OpenRSM jobs. TinyOS functionality can be used by OpenRSM via TinyOS job templates, disposable to users for customization or immediate usage. It can be disputed whether TinyOS jobs can be generic, or specialized; generic jobs offer users the freedom to fully exploit the underlying functionality, while specialized ones can be formalized and their operation can be guaranteed. For the purposes of remotely managing sensor nodes a conservative approach must be followed. The compilation and deployment tasks can be formalized to produce accurate and safe results. The more generic the supported TinyOS functionality, the wider the space of cases the overlying firmware framework must support. The management system will thus support selected TinyOS functionality. It will formalize the development and the deployment tasks so as to allow specific changes disallowing the creation of modules that may result to fundamental malfunction of the WSN, such as incompatibilities in the discovery or firmware update services. The output of the TinyOS jobs will be registered and stored.

The jWebDust layer uses the underlying TinyOS infrastructure and provides the higher layers of the sensor nodes protocol stack. The integrated management platform can

support a safe and formalized subset of the API, tools and functionality provided by jWebDust in the same manner as TinyOS jobs. The challenge is to create an OpenRSM-compatible branch of jWebDust firmware, considering the restrictions in terms of sensor node memory and processing power, so as to include minimal implementations of OpenRSM inventory and discovery modules.

Currently jWebDust supports sensor networks based on TinyOS. In order to provide heterogeneous sensor networks management, as in the case where individual managed networks are based on other architectures (eg ContikiOS [28]), the platform can be extended in order to support the additional infrastructures. Firstly, firmware management functionality can easily be mapped to OpenRSM jobs using the graphical interface of the OpenRSM. jWebDust must then interface with the sensor architecture communication API in order to establish connection with the sensors. In the case where the sensor architecture supports IP-enabled networks, customized sensor management can be achieved by incorporating the OpenRSM agent in the code executed on the sensor.

D. *Use-case scenarios*

We mention here some use-case scenarios, in order to show the conceptual similarity of usual management tasks in integrated WSN management tools, and further justify our conceptual approach. As WSNs tend to come closer to IP networks (e.g., 6LoWPAN), it is our belief that such similarities will only become more. In the case of IP-based sensor networks where the sensors are individually addressed, they can also be managed by remote management systems such as OpenRSM. Management can be achieved by integrating the lightweight agent version in the sensor firmware. In that case the OpenRSM will have to extend its server functionality in order to support sensor agents.

Regarding inventory and asset management, which is a standard feature in OpenRSM and similar management tools, it is also a necessary feature in WSN as well. Although the initial concept for WSN was that they should be data-centric, i.e., only the information matters and not the source (node) it came from, this trend has shifted to more node-centric approaches. Very-large-scale WSNs, at least on a single location, are yet extremely rare, and with new standards and interoperability between different WSN hardware, heterogeneous WSNs are the current and future trend. The administrator of a WSN must be able to see at any instance the capabilities of the nodes, the hardware and software associated with each one of them. A dynamic directory of such "services" is necessary to take full advantage of WSN's capabilities.

Also, software delivery is important in WSNs, even more so than in regular networks. This is because even moderate-size WSNs include more than 50 nodes, each one usually running the same software and difficult to interface with and program. This is a feature that has attracted much attention (e.g.[26]) in WSNs and we plan to take advantage of such previous work. For example, a research lab should be able to schedule jobs related to the software executed in its WSN and reprogram it dynamically and on-the-air (as opposed to manual reprogramming).

Regarding remote control, this is an inherent feature of most WSN software, since the resources of such a network are its sensors, actuators, communication subsystems, and, if available, file storage. WSN administrators need a way to control these resources in a way that resembles the management of regular networks, i.e., on groups of resources, that are built based on the dynamic directory mentioned above.

Finally, network monitoring is also an important activity in WSN. In general, such networks are prone to failures and topologies change quite often. Statistics like the energy available, package failures and retransmissions are useful in evaluating the overall performance of the network and the software used, e.g., of the routing protocol used, and directly linked to the overall functionality offered by the WSN.

E. *Integration Methodology*

The integration of the two platforms will be done in accordance with the integration levels, mentioned previously.

The first objective is to set the jWebDust under OpenRSM management by producing an agent module for WSN gateways. The agent module can be integrated into the codebase of the jWebDust platform or it can be deployed as an external module.

The next milestone is the development of high level system management functionality for the specific needs of WSN gateways, such as customized reports and assets management, into the agent module. Integration will proceed further into producing connectors for the jWebDust API interface. At this point the remote management agent will be capable to provide for the execution of all WSN management functionalities supported by the jWebDust API.

After this point the integration will affect all OpenRSM levels and will reach the management console where the user will be presented with visual representations of the managed boards and even sensors. After this point the integration of jWebDust and OpenRSM may even consider branching from OpenRSM and jWebDust, since it will have the potential needed for building end-user services for WSN management.

One of the most interesting and challenging services will be the support of TinyOS jobs that will foster development and maintenance actions for jWebDust managed laboratories. The cornerstone of these services will be the firmware management service that will formulate and support the remote firmware update.

V. CONCLUSIONS AND FUTURE WORK

This paper presents the conceptual design of an integrated management environment enhanced with WSN management functionality. The design is based on the integration of the WSN monitoring platform jWebDust with the OpenRSM systems and network management system. The derived platform is an integrated remote management WSN environment, under which the overall administration and monitoring of the wireless sensor network can performed both in low and high level. Future work includes the gradual integration of the two platforms in accordance with the integration levels, as mentioned in Section IV, and the

evaluation of its feasibility and performance (implementation overhead, evaluation of performance metrics).

ACKNOWLEGMENT

This work has been partially supported by the IST Programme of the European Union under contract number IST-2005-15964 (AEOLUS).

REFERENCES

[1] I. Chatzigiannakis, G. Mylonas, and S. Nikoletseas, jWebDust : A java-based generic application environment for wireless sensor networks, In the proceedings of the first International Conference on Distributed Computing in Sensor Systems (DCOSS '05), 2005, pp. 376–386. Also, in the International Journal of Distributed Sensor Networks (IJDSN), Taylor and Francis, accepted, to appear in 2008.

[2] I. Karalis, M. Kalochristianakis, P. Kokkinos, E. Varvarigos – OpenRSM: a lightweight integrated open source remote management solution, Intl Journal of Network Management 2008

[3] TinyDB: A declarative database for sensor networks, http://telegraph.cs.berkeley.edu/tinydb/

[4] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden, TASK: Sensor network in a box, In the Proceedings of the 2nd European Workshop on Sensor Networks, 2005.

[5] P Levis, S Madden, J Polastre, R Szewczyk, K Whitehouse, A Woo, D Gay, J Hill, M Welsh, E Brewer, D Culler - TinyOS: An Operating System for Sensor Networks. In Ambient Intelligence, Springer Berlin Heidelberg, (2005), pp. 115-148

[6] Mote-Works monitoring software, Crossbow Technology Inc., http://www.xbow.com/

[7] Mote-VIEW monitoring software, Crossbow Technology Inc., http://www.xbow.com/

[8] ArchRock, http://www.archrock.com/

[9] TWIST Community Web Site, http://www.twist.tu-berlin.de/wiki

[10] MoteLab, Harvard Experimental Wireless Sensor Network Testbed, http://motelab.eecs.harvard.edu/.

[11] L. Ruiz, J. Nogueira, A. Loureiro – MANNA, a management architecutre for wireless sensor networks. In Communications Magazine, IEEE, vol 41, issue 2, Feb 2003

[12] K. Aberer, M. Hauswirth, A. Salehi – A middleware for fast and flexible sensor network deployment. Proceedings of the 32nd international conference on Very large data base, p1199-1202, 2006

[13] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, M. Welsh - Hourglass: An Infrastructure for Connecting Sensor Networks and Applications *Harvard Technical Report TR-21-04*

[14] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas, Power conservation schemes for energy efficient data propagation in heterogeneous wireless sensor networks, 38th Annual ACM/IEEE Simulation Symposium (ANSS 2005), IEEE Computer Society Press, pp. 60-71, 2005.

[15] D Kakadia, T. Thomas, S. Vembu, J. Ramasamy, "Enterprise management systems part I: architectures and standards", Sun Microsustems, Inc.

[16] A, Westerinen, W Bumpus, "The continuing evolution of distributed systems management", IEICE Trans. Inf. & Syst., vol. E86-D, no. 11, November 2003.

[17] M. Sale, "IT service management and IT governance: review, comparative analysis and their impact on utility computing", HP invent research labs, 2004

[18] M. Dekhil, V. Machiraju, K. Wurster, M. Griss, Remote management services over the web, Software Technology Lab, HP, May 2000

[19] OpenAudit: http://sourceforge.net/projects/openaudit/

[20] UltraVNCm http://www.uvnc.com/

[21] Windows-get: http://windows-get.sourceforge.net/

[22] Nino network management system: http://sourceforge.net/projects/nino/

[23] A. Hochstein, R. Zarnekow, W. Brenner, Evaluation of service-oriented IT management in practice, In Proceedings of the International Conference on Services Systems and Services Management, 2005, Vol. 1, pp. 80-84.

[24] S. Lee, M. Choi, S. Yoo, J. Hong, H. Cho, C. Ahn, S. Jung, Design of a wbem-based management system for ubiquitous computing servers, http://www.dmtf.org/education/academicalliance/, accessed on January 2008.

[25] K. Carey, F. Reilly, Integrating CIM/WBEM with the Java enterprise model, http://www.dmtf.org/education/academicalliance/, accessed on January 2008.

[26] M Wren, J. Gutierrez, Agent and web-based technologies in network management, In Proceedings of the Global Telecommunications Confernece (GLOBECOM), 1999, Vol. 3, pp. 1877-1881.

[27] I. Bailey, A simple guide to enterprise architecture, Model Futures TM white paper, 2006, http://www.modelfutures.com/file_download/4/SimpleGuideToEA.pdf, accessed on January 2008.

[28] A. Dunkels, B. Gronvall, and T. Voigt. Contiki – a lightweight and flexible operating system for tiny networked sensors. In Workshop on Embedded Networked Sensors, Tampa, Florida, USA, Nov. 2004.