2009 International Conference on Network-Based Information Systems

H-WSNMS: A Web-Based Heterogeneous Wireless Sensor Networks Management System Architecture

Wei Zhao, Yao Liang, Qun Yu, and Yan Sui
Department of Computer and Information Science
Indiana University - Purdue University Indianapolis
yliang@cs.iupui.edu, {zhao9, qunyu, yansui}@iupui.edu

***Abstract*—**As heterogeneous wireless sensor networks (WSNs) are being widely deployed for various applications, we are faced with a new challenge of network management for heterogeneous WSNs. On one hand, current available WSN management tools are either application specific, or platform specific, thus suffering from the lack of reusability in heterogeneous WSNs management environment. On the other hand, to develop a new WSN management system for heterogeneous WSNs from scratch is time consuming and may not be feasible. In this paper, motivated by such a challenge, we propose a novel WSN management system architecture targeted for heterogeneous WSNs (H-WSNMS). By introducing the concept of Virtual Command Set, H-WSNMS decouples management functions for specific WSN applications from the individual WSN platforms. Therefore, H-WSNMS facilitates the reuse of each individual WSN's preliminary management command service as much as possible, and at the same time, presents to users a unified management interface across multiple WSNs. We demonstrate H-WSNMS architecture though design and implementation of a prototype system.

Keywords: management system; heterogeneous WSN; component

I. INTRODUCTION

As wireless sensor network (WSN) technology is maturing, it has become an increasingly important and irreplaceable approach for data collection and processing of wide range applications, including spatial and geographical surveillance, environmental monitoring, machinery performance and malfunction monitoring, habitat monitoring, medical care and battle field surveillance. Usually battery-powered, sensor nodes are inherently power constrained and error prone, especially in harsh and dynamic environments. It is a challenge to make sure that sensor nodes perform correctly, and for some advance applications, work cooperatively with balanced workloads. Moreover, the job of WSN's daily operation, maintenance and the changes of application tasks should also be conducted easily and efficiently. Thus, it is necessary and important to have an effective WSN management system in deployment to address the above issues. Currently, some WSN management systems are available with some commercial WSN products, such as MoteView [1] for TinyOS [2] based motes. However, the functionality of such WSN management products is quite limited, and may not be adequate for given applications. For richer functionality, researchers usually choose to develop their own application-specific management systems, like [3], [4], [5] dedicated for power management and [6], [7], [8], [9] for faulty detection, or management system implemented on specific system software, like SenOS [10], RMTool [11], designed for SenOS and RNTOS, respectively. While this approach can meet the users own application-specific management requirements, the management system's development is time consuming and not cost-effective. Moreover, WSN applications are becoming more diverse, and accordingly, management functions are also getting increasingly complex in order to meet different requirements of diverse WSN applications. To address those problems, in this paper, we propose a web-based Heterogeneous Wireless Sensor Networks Management System architecture (H-WSNMS). Our H-WSNMS is a systematic framework of WSN management aimed at the following goals: (1) decoupling WSN management functions from WSN's applications, so that existing WSN management systems (e.g., MoteView) can be used and extended seamlessly to adapt various and dynamic application requirements with minimal effort and to hide those extension details from end-users; and (2) facilitating future *heterogeneous* WSNs environment with a *unified* management system for users where each individual WSN may adopt different platforms, network protocols, and gateway technologies. With H-WSNMS, it would be basically unnecessary to develop a new management system for a given specific WSN application from scratch, if one can apply some available WSN management system, combined with necessary extension through H-WSNMS architecture.

Currently, as the development of WSN management systems is not standardized, their functionality is usually limited and specific to given applications. For example, SNMS [12] provides support for collection of network information, but lacks on-line reconfiguration function; TinyCubus [13] includes a configuration engine and data management but no network monitoring; [14] provides data acquisition with support of a local database, but without reconfiguration. Moreover, none of the above WSN management systems supports heterogeneous WSNs, in which individual sensor networks use different operating systems (for example, TinyOS and MANTIS [15]), network protocols and gateway technologies. Due to its flexible and component-based structure, H-WSNMS can directly support network management for heterogeneous WSNs, and provide rich (or tailored) functionality ranging from Sensor Network Monitoring and Reconfiguration, to Data Query, each working as an independent component. In the following sections, we present the architecture of H-WSNMS in Section II. We present a WSN management design example to demonstrate the proposed H-WSNMS in Section III. We present the Data Query component of H-WSNMS in Section IV. Finally, conclusions and future work are given in Section V.

II. OVERALL ARCHITECTURE OF H-WSNMS SYSTEM

A key concept introduced in H-WSNMS is its Virtual Commands Set (VCS). By VCS, each management function is deemed to be realized by a Virtual Command or a sequence of Virtual Commands from the VCS. On the other hand, each individual Virtual Command could be either partially or completely mapped to a combination of some existing Command Services under the given WSN gateway (with its preliminary management Command Services), as shown by Fig. 1.

In Fig.1, each plane presents a concrete WSN gateway Command Service, to which H-WSNMS maps some Virtual Commands, indicated by black nodes. The grey circles drawn inside the VCS represent those commands without counter parts available in the given WSN gateway system. Therefore, the WSN gateway Command Services need to be extended, which will be discussed in Section IV. To realize the mapping from a subset of Virtual Command Set to a concrete WSN gateway Command Service, H-WSNMS adopts a client-server architecture with three tiers (see Fig.1). The top/client tier is the composition of different WSN management components, each of which is tailored for application requirements and independently performs some specific functions that clients define. The bottom/gateway tier consists of, in general, multiple heterogeneous WSN gateways associated with their preliminary management Command Services. The middle/agent tier is the core of our proposed H-WSNMS architecture that is responsible for interpreting and mapping each Virtual Command from VCS into a concrete WSN gateway Command Service(s). The agent tier contains a

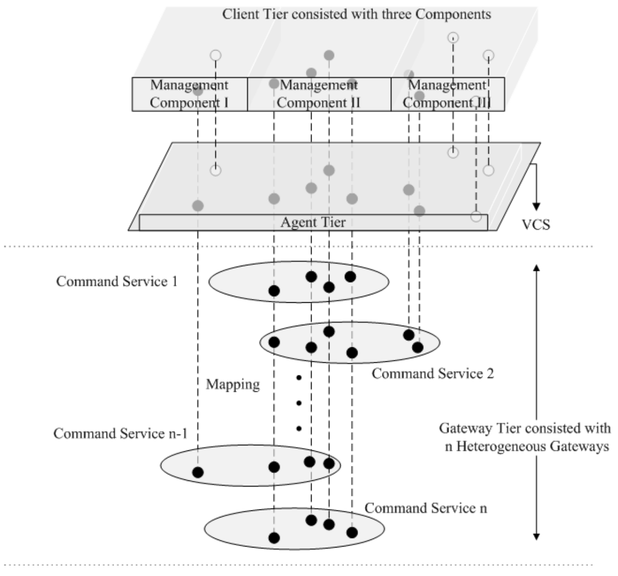


Figure 1. VCS mapping to multiple existing Command.

group of Agent Servers, each of them provides a communication channel between a management component at the client tier and a WSN gateway at the gateway tier. At the same time, each Agent Server is a command interpreter for a subset of VCS. This way, the agent tier in our proposed H-WSNMS architecture works as an *extensible* interface between management components and concrete WSN gateway(s), and thus *decouples* network application-specific management functionalities residing at the client tier from concrete WSN gateways residing at the gateway tier, as shown in Fig. 1. Through this agent tier and its VCS, H-WSNMS can make management components more reusable across heterogeneous WSN platforms and also easier to develop, because developers can create management components based on predefined VCS and be free from handling the details early on with the variety of WSN platforms.

After a Virtual Command is analyzed by an Agent Server and received by the client tier, an appropriate command mapping will be established for the selected underlying WSN gateway with third-part Command Services or some command extensions. With such three-tier architecture, the client tier and gateway tier are independent of each other based on the agent tier, and thus achieve the scalability from both client and gateway tiers point of view. To illustrate the H-WSNMS architecture, in the next two sections, we will present a design example of one WSN gateway instantiation using Crossbow's TinyOS based WSN Xserve [16] gateway. Also, note that because the functionality of the Data Query in the H-WSNMS system has its relatively independent structure, we will present the Data Query separately in Section V.

III. A WSN MANAGEMENT DESIGN EXAMPLE USING H-WSNMS

In this design example, Crossbow's Xserve is used as the WSN gateway technology at the gateway tier in the H-WSNMS architecture as shown in Fig. 2. We demonstrate the design idea through developing two management components in the client tier: Monitoring and Reconfiguration. From the client tier point of view, those two management components are executed by a subset of VCS and the execution details are hidden from users. In general, partial functionality of commands required by Monitoring and Reconfiguration can be mapped to the Command Service already provided by Xserve and the rest functionality has to be implemented through the Xserve command extension. We will focus on the management functions that are able be mapped to Xserve first in section III. A, and present the implementation of Xserve command extension in Section III. B. The general control and status information flow in H-WSNMS system is illustrated, in principle, in Fig. 3 for both Monitoring and Reconfiguration.

A. WSN gateway mapping

In the current H-WSNMS system, we map/implement all the Monitoring functions through Xserve. The collected status information includes basic information, for example the node ID, neighbor information, voltage values, and so on. To tailor the application to maximize the performance according to practical requirements, users may need to configure some parameters such as sampling rate, sensor id or working mode of mote sensors. For example, we may need to adjust the workload according to their current remaining power levels, or, in another case, may need to reset some nodes sending outragous measurements. Supported by Xserve, there are mainly two types of commands available: Mote Configuration and Network Power Management.

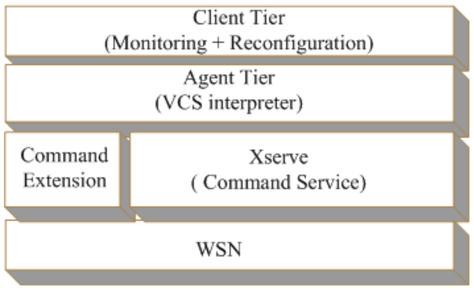


Figure 2. Three-tier architecture instantiated by Crossbow's Xserve WSN gateway.

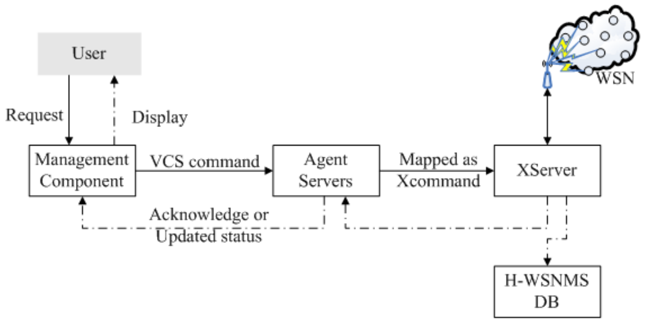


Figure 3. General control and status information flow for Monitoring and Reconfiguration .

Table 1 lists the Virtual Commands implemented. Partial functionality of our required reconfiguration commands is not supported by Xserve and thus indicated with grey color. In, Fig.4, the snapshot of Reconfiguration UI, the extended commands are indicated with red rectangular. As illustrated in Fig.3, each command responds back with an acknowledgement to confirm the correct acceptance of a command. The only exceptions are actuation commands. They do not send an acknowledgement response.

For most real-world WSN applications, retasking is necessary and thus sensor nodes should be remotely reprogramming over a wireless multi-hop network after being deployed in the field. This ability will be an important function for management systems since sensor networks

TABLE I. Configuration Commands of VCS

Category	VCS	XCommand	Function
Reconfiguration	H_SpRate	SET_RATE	Set new Sampling Rate
	H_NID	SET_NODEID	Assign Node ID
	H_GID	SET_GROUP	Assign a Node to new group
		…….	
	H_CRate	*Unavailable*	Set collection rate
	H_ECollect	*Unavailable*	Immediately perform a data collection from WSN and store it to DB
Power Management	H_RESET	RESET	
	H_SLEEP	SLEEP	
	H_WAKEUP	WAKEUP	

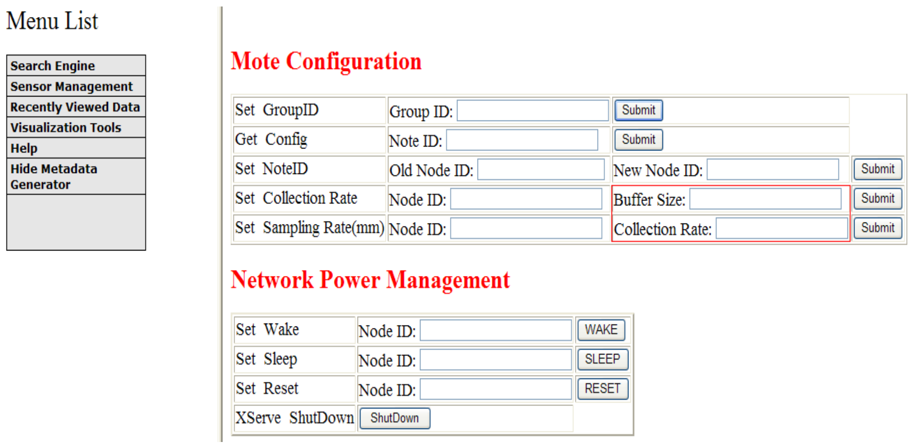


Figure 4. SnapShot of Reconfiguration UI.

may be deployed in inaccessible areas and may scale to thousands of nodes so that there is no need for on-field sensor reprogramming, which sometimes is impossible.

B. WSN gateway extensions

In Section III. A, we have presented the functions implemented by Xserve, which are relatively straightforward by deploying corresponding mapping Agent Servers. In this section, we will present the command *extensions* for those functions that are not directly supported by existing Command Service of Xserve. As we have discussed above, taking advantage of the three-tier structure of H-WSNMS and its unified interface, we can achieve more flexibility and reduce dramatically the amount of work during the development of new function components required by WSN applications. In the following, we first briefly review NesC and TinyOS for our implementation of command extensions, and then present our approach of command extensions in detail.

1) Introduction to NesC& TinyOS

NesC is a new language for programming structured component-based applications [17]. The nesC language has a C-like syntax, but provides some new features, like concurrency model support, component concept including its naming and linking mechanisms and event driven structure. TinyOS is an open-source operating system programmed in nesC by U.C. Berkeley for embedded platform. Benefitting from its component-based and event-driven structure, TinyOS is designed to cope with very limited on-chip resource, for example, in a WSN application. To further understand TinyOS and nesC, there are some important concepts we need to be clarified.

Component: NesC applications are built out of components. Based on the difference of functionality, there are two types of components:

- Modules: Implement the application behavior
- Configurations: Wires components together

Interface: Interfaces attached to each component can be considered as bidirectional "gates", the only way of communication between components. To insure runtime efficiency, components are statically wired together through their interfaces. Components **use** and **provide** interfaces.

Events: Events are time critical, running in response to a hardware interrupt or signaled by a component. Events can preempt one another and follow last-in first-out semantics.

Tasks: Tasks are typically posted in response to an event to handle long background processing jobs. Unlike events, they are atomic with respect to other tasks (single threaded).

Command: Interfaces consist of commands and events. Those commands are called by the user of the interface and implemented by the provider of the interface.

2) Command Extensions

Let us consider how the two new functions, not originally supported by Xserve, could be added to the H-WSNMS management system. The key point to illustrate in this section is that, with H-WSNMS system, such function extension is not visible from the client tier.

XServe provides the only entry (i.e., port 9003) for XCommand inputs. After a command is wrapped in a packet and injected to wireless mesh network (Xmesh), it will follow the flow shown in Fig.5. In Fig. 5 there are mainly two parts of localization information for a XCommand packet to arrive at the correct mote and the command handler program branch to executed properly: the first part is to address information including which group the target node belongs to and its node address; the second part is the program entry information for a local application to find the correct program entry in the existing table. Group ID is an 8 bit value specified in <tos>/apps/Makelocal, which actually creates a virtual sensor network. The node address is a 16-

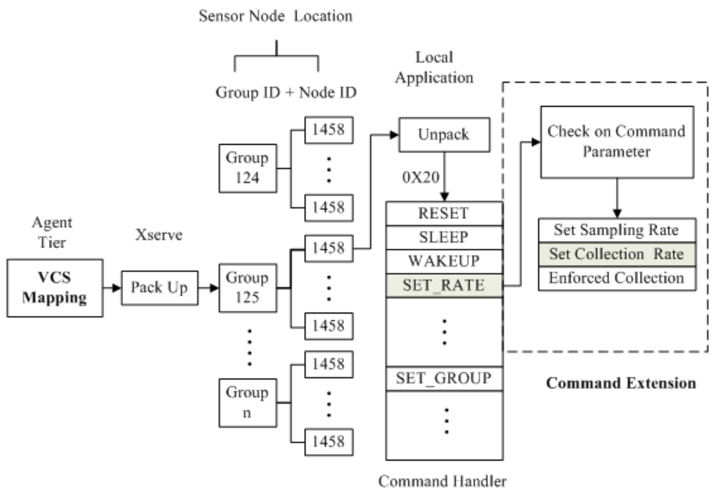


Figure 5. XCommand flow illustration.

bit value specified by the make command "make install <address> micaz" [18]. There are usually three kinds of node addresses:

- Reserved addresses:
  - 0x007E-UART (TOS_UART_ADDR)
  - 0xFFFF-Broadcast (TOS_BCAST_ADDR)
- Local address: TOS_LOCAL_ADDRESS

There are two parts of information as payload of the XCommand packets: the corresponding entry number of the command and the parameter inputs (for Fig. 5, it is the sampling rate designated by the user). In our design example, there exist two command tables based on such two parts of delivered information on each sensing mote. When the XCommand packet arrives at its destination, the first part of the information, the entry number of the XCommand, will be extracted and used to locally search against the XCommand table on the mote, for example 0x20 for "XCOMMAND_SET_RATE". In the next step, the local application will run an additional check on the parameter inputs, the second part of the information from the XCommand packet, such as to locate the correct entry of second command table. As shown in Fig.5, the command for configuring sampling rate will be finally executed and the acknowledge packet will be sent back to base station. In our application, except for periodic data collection on each mote, the following functions are needed from the client side with XCommand:

- Set Collection Rate: push sampled data to a queue structure and collect all sampled data at the end of each automatic collection period;
- Enforced Collection: send back all existing data samples in the queue.

The first function is obviously practical to adapt to different sensing environment or sensing tasks for a mote or a group of motes. After a data item is sampled, it is wise for a deployed WSN to first store it in mote locally than just to send it back to the data sink immediately to improve power savings for each mote. We applied a circle queue as temporal storage space for collected data. Once the queue is full or its size reaches some threshold based on the pre-configured collection period, the mote will send all the data samples currently in that queue back to data sink. In some cases, a user may need an up-to-date sampling data, so as there is a need to manually collect all the data in the queue back in an enforced manner even though the queue is not full. As we know, TinyOS applies event driven structure, and thus each function to complete needs to be triggered by some event as shown below:

- Time.fired->sample
- Sensor.dataready->push
- next; Send.sendDone->pop next
- XCommand.received-> entry searching

In Fig.5, to configure the sampling rate of the sensor mote (group125, node1458), a corresponding command, "XCommand_Set_Rate" is injected to Xmesh through XServe. According to the entry sequent number extracted from the command packet, a correct program entry is found from the table and then the component TIMER will be set to new rate.

```
timer = opcode->param.newrate;
call Timer.stop();
call Timer.start(TIMER_REPEAT,timer);
```

In this design example, the sequent number space for Xserve's XCommand is *refined* so that we can extend the two newly added operations, Set Collection Rate and Enforced Collection, under the same entry as the existing "XCommand_Set_Rate". For these two newly added commands, they will follow the exactly same flow as shown in Fig. 5. When a new sample is collected, a "data-ready" event for the corresponding sensor board will occur, and then the queue will execute a "push" operation. Similarly, during pop-out period, any time a "sendDone" event occurs, sampled data is popped out.

```
event result_t Send.sendDone(TOS_MsgPtr msg, result_t
success) {
   call Leds.greenToggle();
   atomic sending_packet = FALSE;
           atomic counter--;
           dequeue_next++;
dequeue_next %= MESSAGE_QUEUE_SIZE;
  atomic  post QueueServiceTask();
   return SUCCESS;
 }
```

After a command packet arrives at a destination mote, an "XCommand.received" event occurs, then the entry searching process will begin.

IV. DATA QUERY IN H-WSNMS SYSTEM

The data query component in the H-WSNMS collects and stores all sensed data from multiple heterogeneous WSNs, and supports data query and retrieval using unified H-WSNMS Web interface in a real-time fashion. While it is convenient to store all sensed data in H-WSNMS system's central database, referred to as H-WSNMS database, it is also possible that the sensed data collected from each individual WSN is stored into its own local database, referred to as remote WSN database in this paper. For this reason, H-WSNMS also supports distributed data query and access from multiple remote WSN databases.

In order to do so, each remote WSN database is abstracted as a data source. We use a metadata-driven approach. An XML-based metadata wrapper is created for each individual data source to describe the data source's various aspects including its access method, specific query form and interface, and format. Moreover, TSA-Data Node tree, a data integration model for heterogeneous data sources developed in [19], is adopted to improve the performance of data query and retrieval and support the scalability of H-WSNMS when the number of heterogeneous WSNs increases. In the following, we briefly describe query engine and access engine of the data query component of the H-WSNMS.

- Query Engine

This module handles the query evaluations in the TSA-DataNode tree. Based on the query posed by the user, the engine determines appropriate TSA-DataNodes with the help of the search engine and performs the query on the TSA-DataNode.

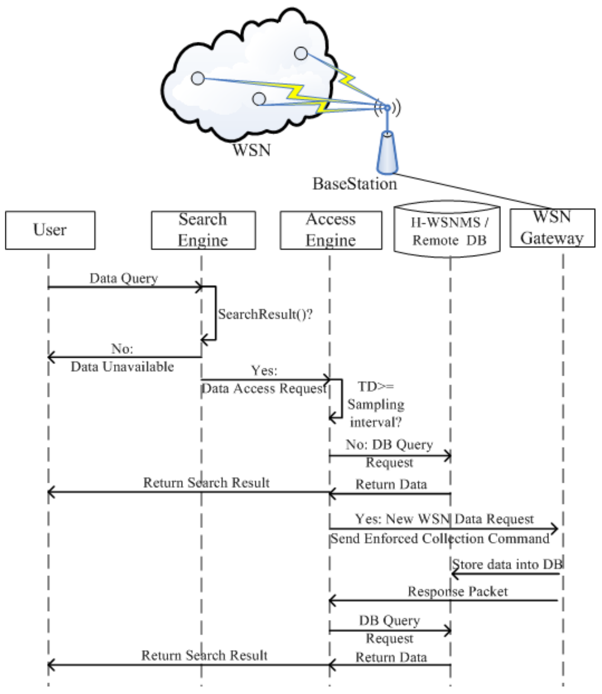
- Access Engine

Access Engine acts as a point of contact for data sources to the H-WSNMMS system. It creates, and maintains network connections with data sources and retrieves datasets. The necessary information to define an access point is clearly defined in our metadata standards.

The process of data query in the H-WSNMS is illustrated in Fig. 6. When an end-user issues WSN data query, two scenarios need to be considered: (1) all data requested are already collected and stored in H-WSNMS database or a remote database, and (2) all or part of the data requested still reside at motes. In the first case, it is just a database access. However, in the second case, we need to use our extended 'XCommand' *Enforced Collection* described in Section III. B.

V. CONCLUSIONS AND FUTURE WORK

To cope with the heterogeneous WSNs and facilitate command extension over existing command services, we have proposed a web-based management system architectur-



TD: The time difference between the latest sensor data collection and the rightmost point of time segment specified by Data Query

Figure 6. Data query process in H-WSNMS.

e, H-WSNMS. The underlying idea of H-WSNMS is to decouple the development of application-specific management functions from deployed heterogeneous WSN platforms and gateway technologies including their associated preliminary management command services. This is achieved by introducing a critical "middle/agent" tier characterized by VCS, in the client-server structure of three tiers in the H-WSNMS. H-WSNMS architecture not only directly supports network management for heterogeneous WSNs, but also facilitates the reuse of each individual WSN's preliminary management tool as much as possible, and at the same time, presents to users a unified interface across multiple WSNs. This unified management and data interface would be able to greatly simplify heterogeneous WSNs' daily operations and their data access. We illustrated the H-WSNMS using Xserve mote network through mapping H-WSNMS' Virtual Commands to both the existing Xserve Command Service and the newly extended Xserve Command Service to realize the full VCS of the H-WSNMS. Based on those discussions, we can see that the Agent Servers embedded responsible for mapping of VCS would enable the logical independence between the client tier WSN applications, and individual concrete WSN platforms and technologies made by different venders and/or research communities. With the extensible VCS mapping, we can first select a basic set of Virtual Commands based on one available WSN management tool and gradually extend the VCS under the same structure and interface as more WSNs are deployed. From the perspective of end-users, the management system works as one unified entity and hides all the WSN platform heterogeneity details and the implementation of mapping and dynamic command extensions. H-WSNMS also provides distributed data query functionality with a unified data portal across all heterogeneous participating WSNs through its Data Query component. Our current H-WSNMS prototype is developed in Java. We plan to extend our prototype system to include two different WSN platforms and gateway technologies at the gateway tier to further study and verify our proposed new H-WSNMS architecture. We also plan to create more sophisticated management functionality at the client tier by the composition of a sequence of Virtual Commands.

ACKNOWLEDGMENT

This work is supported in part by National Science Foundation under grant CNS-0758372.

REFERENCES

[1] M.Turon, "Mote-View: A Sensor Network Monitoring and Management Tool," in *Proc. of IEEE EMNET-II Workshop*, May 2005, pp.11-18.

[2] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An Operating System for Wireless Sensor Networks, " In Ambient Intelligence, Springer-Verlag, 2005.

[3] R. Tynan, D. Marsh, D. Okane, and G.M.P. Ohare, "Agents for Wireless Sensor Network Power Management," In Proc. of IEEE ICPPW Conf., June 2005.

[4] N. Ramanathan and M. Yarvis, "A Stream-oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications," in *Proc. IEEE EMNET-II Workshop,* May 2005.

[5] A.Boulis and M.B. Srivastava, "Node-level Energy Management for Sensor Networks in the Presence of multiple Applications," In *Proc. IEEE PERCOM Conf.,* Mar. 2003.

[6] L.B. Ruiz, I.G. Siqueira, L.B. e Oliveira, H.C. Wong, J.M.S. Nogueira, and A.A.F. Loureiro, "Fault Management in Event-Driven Wireless Sensor Networks," In *Proc. ACM MSWIM Conf.,* Oct. 2004.

[7] W. L. Lee, A. Datta, and R. Cardell-Oliver, "WinMS: Wireless Sensor Network Management System, an Adaptive Policy-based management for Wireless Sensor Networks," Tech. Rep. UWA-CSSE-06-001, The University of Western Australia, June 2006.

[8] C. Hsin and M. Liu, "A Two-Phase Self-Monitoring Mechanism for Wireless Sensor Networks," *Journal of Computer Communications special issue on Sensor Networks*, vol.29, no. 4, 2006, pp. 462-476.

[9] N. Ramanathan, E. Kohler, and D. Estrin, "Towards a Debugging System for Sensor Networks," *International Journal for Network Management*, vol.15, no. 4, 2005, pp. 223-234.

[10] T.H. Kim and S. Hong, "Sensor Network Management Protocol for State-Driven Execution Environment," In *Proc. ICUC Conf.,* Oct. 2003.

[11] H. Cha and I. Jung, "RMTool: Component-Based Network Management System for Wireless Sensor Networks" In *Proc. 4th Consumer Communications and Networking Conf.,* 2007, pp.614-618.

[12] G. Tolle and D.Culler, "Design of an Application-Cooperative Management System for Wireless Sensor Networks," In *Proc. 2th European Workshop on Wireless Sensor Networks (EWSN)*, Istanbul, Turkey, January, 2005.

[13] P.J. Marrón, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, and K. Rothermel, "TinyCubus: A Flexible and Adaptive Framework for Sensor Networks," In *Proc. 2th Europ. Workshop on Wireless Sensor Networks*, 2005, pp.278-289.

[14] W.S. Jang, W.M. Healy, M.J. Skibniewski, "Wireless Sensor Networks as Part of a Web-Based Building Environmental Monitoring System," Automation in Construction 17, 2008, pp. 729-736.

[15] S.Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, R. Han, "Mantis OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," *ACM/Kluwer Mobile Networks &*

Applications Special Issue on Wireless Sensor Networks, vol. 10, no.4, Aug. 2005.

[16] Xserve User Manual, CrossBow Technology Inc., Available online at http://www.xbow.com/Support/Support_pdf_files/Xserve Users Manual.pdf, accessed in March, 2009.

[17] D. Gay, P. Levis, R.V. Behren, M. Welsh, E. Brewer, D. Culler, "The NesC Language: A Holistic Approach to Networked Embedded Systems," In Proc. ACM SIGPLAN Conf. on Programming Language Design and Implementation, June, 2003.

[18] MicaZ DataSheet, CrossBow Technology Inc., Available online at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf, accessed in March, 2009.

[19] N. Ravindran, Y. Liang, "HIDE – A Web-based Hydrological Integrated Data Environment," Proceedings of the SNPD 2007 (8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing), Vol. 3, pp. 143 – 148, July 30 – Aug. 1, 2007.