SNMP-Based Management Architecture for Wireless Sensor Networks

Seonghun Jang, Daeyoung Kim, Hyoung-soon Kim, and Su-Chong Joo

Abstract. We propose a management system for wireless sensor networks based on Simple Network Management Protocol (SNMP). SNMP is the standard network management protocol that facilitates the exchange of management information between network devices to monitor managed devices, and is widely deployed and available in almost all network devices. Using SNMP to manage WSNs, we can get an integrated management view of multiple WSNs that have heterogeneous systems and protocols. Because a sensor node is incapable of running an SNMP agent due to lack of resources, we propose an approach that uses the base station as an SNMP agent representing other sensor nodes.

Keywords: Wireless sensor networks, management, SNMP.

1 Introduction

Wireless sensor networks (WSNs) are growing in terms of quality and quantity. Over the past half decade, a considerable number of studies have been conducted on WSNs which are now being applied not only to the military and science fields but also to the smart home, offices, manufacturing facilities, and etc. As the various applications of WSNs appear, there is no single best sensor network protocol that covers all requirements of each WSN. They use different MAC protocols, routing algorithms, management protocols and application-specific message formats to accommodate heterogeneous applications. The question now arises: What if one organization has multiple WSNs for different applications? Should the organization have one management tool for each application? Of course, one management tool can cover several WSNs that have same management protocols and message formats. What if two or more organizations that use totally different protocols and tools to manage
their WSNs are merged? The problem which we have to consider is the lack of a standard way to manage WSNs.

Simple Network Management Protocol (SNMP) [1] can be a good solution to manage WSNs. SNMP is a standard network management protocol that facilitates the exchange of management information between network devices to monitor managed devices, and is widely deployed and available in almost all network devices. It provides an easy and simple management method of network resources for the network administrator. By using SNMP, the network administrator can access the set of parameters which can be read from or written to an element. This set of parameters is called as Management Information Base (MIB). Many vendors define their own MIB to add network-management functions to their products. SNMP is popular because of this flexibility. It can be easily extended to applications, and the MIB can be defined privately. Through the use of SNMP, we can make WSNs interoperate with existing network devices allowing the administrator to have an integrated view of the network.

Several studies have been made on applying SNMP to ad hoc network management by running an SNMP agent on the ad hoc nodes [2],[3]. ANMP [2] is Ad hoc Network Management Protocol which is fully compatible with SNMP. It uses SNMP PDU to gather the management information, and uses cluster-based approach to manage the nodes. One representative node, called a cluster head, in a cluster manages the nodes belonging to the cluster. All nodes support ANMP agent, which is a variation of SNMP agent, and have ANMP MIB that extends SNMP MIB. [3] proposes a management scheme for ad hoc networks called Guerrilla management architecture. Similar to ANMP, Guerrilla management architecture is compatible with SNMP and uses GMIB that extends SNMP MIB as the management information base. To envision networks to be self-organized and self-managed, it proposes an approach that allows a node to select its own role according to its functionality and capability. Both [2] and [3] take advantage of SNMP by having compatibility with SNMP. They, however, assume that all nodes in a network have SNMP agent. Unfortunately, this assumption is not always true for wireless sensor networks. An SNMP agent requires additional memory space and computing power to run on the devices but a sensor node that makes up a WSN does not support TCP/IP stack, has very limited memory space, and is equipped with low-end processor.

To solve this problem, we propose an SNMP-based management architecture for WSNs. Our architecture uses SNMP to manage the WSN and lets the administrator use an existing SNMP manager tool or SNMP-based Network Management System (NMS). The basic idea is to run an SNMP agent on the base station, not on individual sensor nodes. The base station acts as a bridge or gateway between sensor nodes and TCP/IP network including NMS. In our architecture, a base station interprets SNMP message into a local message format and stores management information from sensor nodes in the form of MIB for the NMS. Our architecture can be implemented without altering existing local protocol for sensor nodes.

This paper is organized as follows. Section 2 provides an overview of our approach and Section 3 describes each component of a base station. Section 4 describes the implementation of our architecture. Conclusions are presented in Section 5.
2 WSN Management Architecture

In this paper we propose the management architecture for WSNs using SNMP. A typical SNMP architecture consists of three components: managed devices, SNMP agents, and one or more NMS. An SNMP agent is typically software that resides on a managed device. In WSNs, sensor nodes are incapable of running SNMP agents. The basic idea of our approach is to separate SNMP agents from managed devices (sensor nodes). While removed from managed devices, the SNMP agent is located on the base stations that have more resources than sensor nodes do. The base stations are capable of running SNMP agents and act as the local management system in our architecture.

Fig. 1 depicts our management architecture. Our architecture has a hierarchical three level structure that consists of NMS, base stations, and sensor nodes. One WSN for the specific application consists of one or more base stations and a number of tiny sensor nodes. Because our main goal is to manage multiple WSNs that have different protocols, operating systems and applications with one or more NMS, the architecture should support diverse forms of WSNs. We divide the forms of WSNs into three groups according to the number of base stations and cluster heads. The first type of WSNs, which is simplest form, has only one cluster head and one base station. In this type of WSNs, all sensor nodes transmit the collected data to a base station and there is actually no cluster, so the base station acts as a cluster head of sensor nodes and a bridge to the TCP/IP network. In other words, the base station and cluster head are implemented on a device that has plenty of power, rich in resources, and have full functionalities. The second type consists of one base station and multiple cluster heads. A cluster consists of a partial set of adjacent sensor nodes and a cluster head. While the cluster head, which performs data aggregation and computation and transmits the result to the base station, does more energy-consuming job than other sensor nodes do, overall network efficiency is higher than the first type of WSNs in terms of time and energy. The third type of WSNs has multiple base stations and
cluster heads. Having plural base stations gives added advantage of security, fault
tolerance and scalability [6]. We assume that each base station in the same WSN has
the same functionalities.

In our architecture, the base station communicates with sensor nodes using WSN
specific protocol. For example, in WSN A, sensor nodes that run TinyOS as their
operating system communicate with each other using Protocol A, while sensor nodes
of WSN B do using Protocol B. On the other hand, a NMS communicates with each
base station using SNMP messages: GET, SET, and TRAP. GET and SET messages
for the management of WSN are sent from the NMS to the base station, and TRAP
messages are issued by the base station to inform the NMS that some events happened
in the WSN. Even if each WSN has different specification, allowing base stations to
have same interfaces to NMS makes the administrator be able to manage multiple
WSNs using a management tool.

3 Base Station

Fig. 2 shows the architecture of a base station that plays the role of an agent for sensor
nodes. The base station has mainly three components: SNMP Agent, Local Manager,
and MIB cache. The SNMP agent in the base station is extended to meet the
requirements of WSNs. It interprets and transfers SNMP message from NMS to Local
Manager. Local Manager, which provides a set of WSN specific management
modules, has interfaces to interact with SNMP Agent in the base station. Each module
in Local Manager can be selected from among existing management modules by
WSN developers. The management information obtained by Local Manager will be
stored in MIB cache that is referred by SNMP Agent. The operation of each
component and their interactions are described in detail in the following sub-sections.

![Base Station Diagram](image)

Fig. 2. The structure of a base station.
3.1 SNMP Agent

This component is an extended version of the typical SNMP agent. It has the role of processing SNMP Protocol Data Unit (PDU). PDU defined in RFC1157 is a message format for SNMP operation. SNMP agent should be able to interpret and generate PDU for communication with NMS. There are three sub-components of the agent: Request Handler, MIB manager, and Trap Generator.

**Request Handler.** NMS is able to get responses from and set variables in agents with GET and SET messages respectively in our architecture. Request Handler takes GET and SET message type of PDU from NMS. Fig. 3 shows how Request Handler works.

![Fig. 3. Operation of Request Handler.](image)

Request Handler receives requests from NMS. The type of requests can be GET or SET. In GET’s case, Request Handler checks MIB cache for the management information that is stored in the base station. If there is a cache hit, Request Handler makes PDU for response and transfers it to NMS. Local Manager and sensor nodes do not participate in this case. If there is a cache miss, Request Handler calls Local Manager to retrieve the management information from sensor nodes. Local Manager will communicate with sensor nodes and return the expected value. In the case of SET, there is no step for searching MIB cache, and the request will be forwarded directly to Local Manager to be translated into the WSN specific message.

**MIB Manager.** MIB is a database of objects for management information of managed devices. Each object in MIB has an object identifier (OID) which is a sequence of integers used to locate the object within MIB. MIB Manager retrieves objects from MIB cache with their OID, and updates MIB cache according to commands from Request Handler.

**Trap Generator.** SNMP traps are issued by SNMP agents to notify SNMP NMS of asynchronous events. Trap Generator allows NMS to receive information regarding WSN state changes. Fig. 4 presents the interactions of Trap Generator.

When some events happen in sensor nodes, the base station receives the event notification messages from sensor nodes. Trap Generator is called by Local Manager when it receives an event notification and it refers the trap registration table that contains the information about trap recipients (NMS). The trap registration table tells Trap Generator which NMS is interested in the event now occurring.
3.2 MIB Cache

Typical SNMP agent has MIB for its own management information. In our architecture, MIB stored in the base station is not only for the base station. It also contains management information for sensor nodes belonging to the base station. Each information element in MIB should be updated properly. If the update is too late, the administrator may get old value that cannot represent current status of the sensor network. If the update is too frequent, the sensor nodes may suffer reduced battery life because of producing numerous, redundant management data. To solve this problem, we categorize the management information into three groups according to the update property (see Table 1).

The first group is management information that is continuously changed, or dynamic. Remaining battery power of a node is a good example of this kind of information. This kind of information is stored for a specific time (TTL) in MIB cache when the agent refers that information. Although TTL can vary from the application or the environment, we suggest the sensing period as a default TTL value. Fig. 5 shows the interaction in the base station when dynamic information is queried by NMS.

<table>
<thead>
<tr>
<th>Type of information</th>
<th>Example</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic</td>
<td>Remaining battery level, usage of processor, usage of memory, the number of issued packets … Information related to network topology (node join, node failure), node status, sensing data…</td>
<td>Stored in MIB cache with specific TTL (time-to-live) Asynchronous update and retrieval</td>
</tr>
<tr>
<td>Event-driven</td>
<td>Sensing period, application specific values …</td>
<td>Converted into WSN specific messages</td>
</tr>
</tbody>
</table>

Table 1. Information classification according to update property
The second group is the information that is updated by events from sensor nodes. Network topology is one of them. Node join or node failure can be recognized as an event in WSN that should be informed to the base station. This kind of information is updated only when an event occurs. The third group of the management information is configurable information. Sensing period can be a candidate for this kind of information if the WSN supports the change of sensing period by the administrator. The information in this group is updated only by the administrator.

To meet the requirement of network management for WSNs, we define new MIB that extends original MIB based on MANNA’s information architecture [7]. Fig. 6 shows part of MIB tree structure for WSNs (the equipment information defines more detail MIB, but omitted in Fig. 6). The senNetwork group of MIB has the information about a sensor network, and the senNode group has the information about the specification of each sensor node that makes up the sensor network. We are currently working on defining more MIB for sensor nodes that are rapidly developing with newer technologies.

Fig. 5. Interaction for exchange of dynamic management information.

Fig. 6. Management Information Base for Wireless Sensor Networks.
3.3 Local Manager

Local Manager can be viewed as having three main functions: translation, management, and notification.

Translation. Local Manager takes a role of converting SNMP command into WSN specific command. If there are SET messages coming from NMS, SNMP agent lets Local Manager convert these messages into WSN specific messages and send them to sensor nodes. If there are GET messages coming from NMS and there is no MIB cache entry that contains the management information required by NMS, SNMP agent calls Local Manager for that management information with OID as a parameter. Local Manager translates OID into WSN specific object identifier and sends a query to sensor nodes. When a response message that contains the management information is delivered from sensor nodes to the base station, Local Manager interprets the message and returns the management information to SNMP agent that will make an SNMP PDU according to the return value. Table 2 shows the example of translation by Local Manager. Depending on the application, a local message from sensor nodes can be converted into multiple MIB objects. Local Manager also updates MIB cache when it takes new management information from sensor nodes.

Table 2. Translation example

<table>
<thead>
<tr>
<th>Local message</th>
<th>WSN Application</th>
<th>SNMP MIB (translated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7E 00 0A 7D 1A 01 00 28 00 01 00 96 03 97 03 ...</td>
<td>TinyOS Oscilloscope Application</td>
<td>MIB=senMIB.senNode.equipment.sensor.light Sensor.value.1.0 Value = 918</td>
</tr>
<tr>
<td>#1,759,5550,93,20,2 500,236,515,499$</td>
<td>ANTS Climate Monitoring Application</td>
<td>MIB=senMIB.senNode.equipment.sensor.light Sensor.value.1.0 Value = 881</td>
</tr>
</tbody>
</table>

Management. Local Manager provides a set of WSN specific management modules. WSN developers can either choose each module in Local Manager among the existing modules or make new management module to meet the requirements of their own application. For example, WSN developers can choose TopDisc [8] as a topology discovery algorithm while using their own power management module.

Notification. When events occur in WSNs Local Manager should notify SNMP agent of what happens. When Local Manager receives event messages from sensor nodes, it finds appropriate MIB to store the value that represents the event, and calls Trap Generator of SNMP agent with the MIB as a parameter. The event notification from Local Manager will be converted into a TRAP message by Trap Generator.
4 Implementation

Our implementation is divided into three parts: 1) GUI for the administrators; 2) base station application; and 3) sensor node programming. Fig. 7 is the captured image of simple GUI program for monitoring network topology and battery level of each node. This program is implemented in PC as NMS. An embedded Linux board, ANTS-H4 [4] (Intel XScale PXA270 ARM RISC chip, 64Mbyte SDRAM), is used for the base station (see Fig. 8 (a)). The specification of the board is enough to run SNMP agent and to have MIB cache for sensor nodes. The operating system of the base station is ARM Linux (kernel 2.6), and the application is implemented using Java. Free open source SNMP API for Java, SNMP4J, is used in both the base station and NMS. ANTS-H2, which is shown in Fig. 8 (b), is used as the sensor node in our implementation. This sensor node has an Atmel ATmega 128L 8-bit microcontroller and CC1100 RF transceiver. Each sensor node runs ANTS-EOS [5] as its operating system. In our implementation, the sensor nodes are going to be deployed for monitoring weather and climate in the mountains. They use a string routing, so each node has one parent and one child (an end node has no child). The message format for communication between the base station and sensor nodes is based on the sequence of integers like #3,2,0,2,4$ where # and $ mean the start and the end of message respectively. The base station interprets the messages from sensor nodes and makes SNMP PDU to be sent to NMS. The application on NMS that receives SNMP PDU from the base station displays the management information.

Fig. 7. Screenshot of implementation program.

Fig. 8. Hardware for implementation.
5 Conclusion

In this paper, we have described a network management architecture based on SNMP for WSNs. To apply SNMP to management of WSN, we proposed an approach that allows the base station to process and translate SNMP messages for sensor nodes that are incapable of handling SNMP. Thus, the network administrators can take advantage of SNMP to manage WSN while negating the need to change existing WSN specific protocol. We also proposed MIB caching policy on the base station to prevent sensor nodes from wasting energy as a result of excessive polling. Management information is categorized into three groups according to update property and the information of each group is handled in different way when it is cached on the base station. With our architecture, the network administrator can achieve an integrated management view of networks, including not only SNMP-supported network devices but also tiny sensor nodes, with reducing development time and minimizing the amount of additional code required in the management tool.

Acknowledgments. This research has been supported by the National Computerization Agency and the Ministry of Information and Communication in Korea. This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) for Healthcare Technology Development in 2006.

References