Send Orders for Reprints to reprints@benthamscience.ae

The Open Cybernetics & Systemics Journal, 2015, *9,* 1019-1024

Open Access

Applying SNMP Technology to Manage the Sensors in Internet of Things

Huang Hui-Ping*, Xiao Shi-De, Meng Xiang-Yin

Electromechanical and Control Department, Mechanical Engineering College, Southwest Jiaotong University, Chengdu, Sichuan, 610031, P.R. China

Abstract: Nowadays, the IoT is largely dependent on sensors. The IoT devices are embedded with sensors and have the ability to communicate. A variety of sensors play a key role in networked devices in IoT. In order to facilitate the management of such sensors, this paper investigates how to use SNMP protocol, which is widely used in network device management, to implement sensors information management of IoT system. The principles and implement details to setup the MIB file, agent and manager application are discussed. A prototype system is setup to validate our methods. The test results show that because of its easy use and strong expansibility, SNMP is suitable and a bright way for sensors information management of IoT system.

Keywords: Agent, internet of Things, manager, MIB, sensors, SNMP.

1. INTRODUCTION

The Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices. Each thing is uniquely identifiable through its embedded computing system but is able to interoperate within the existing Internet infrastructure [1]. The Internet of Things (IoT) transforms the everyday physical objects that surround us into an ecosystem of information that will enrich our lives. The IoT is largely dependent on sensors. Sensors detect and measure changes in position, temperature, light, etc. and they are necessary to turn billions of objects into data-generating "things" that can report on their status, and in some cases, interact with their environment. For example, product and shelf sensors collect data throughout the entire supply chain. In the gaming industry, companies use tracking sensors to transfer the movements of users onto the screen and into the action. Internet-connected smart meters can measure power usage and provide feedback to the power consumer, sometimes automatically adjusting the system's parameters. All of these facts show that sensors play a critical role in the IoT. So it's important to ensure the normal operation of the sensors.

SNMP [2] is one of the most commonly used technologies when it comes to network monitoring. Anuj Sehgal [3] has researched how to apply SNMP protocol to implement management of resource constrained devices in the Internet of Things. Deng Hubin [4] discussed the analysis and implementation of embedded SNMP agent. Hillbrecht [5] has investigated how to setup a SNMP-based virtual machines management interface. Dow Chyi-Ren [6] has researched on adaptive SWE and SNMP-based sensor management for environmental monitoring.

Because of its simplicity, we choose SNMP technology to help us manage the sensors in IoTs. This paper describes how to use SNMP technology to facilitate management of sensors in IoT. This paper is organized as follows: In section II, the SNMP technology is introduced. In section III, we explain some of the implementation details of applying SNMP technology to manage the sensors of IoT devices, including the design details of MIB, SNMP manager and Agent. In section IV, we present the results of a prototype system which can manage some sensors information. In section V, we make a conclusion of this paper and propose the future research directions.

2. SNMP INTRODUCTION

The Simple Network Management Protocol (SNMP) is a component of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite as defined by the Internet Engineering Task Force (IETF). Introduced in 1988 to provide management capability for TCP/IP-based networks, SNMP rapidly became the most widely used standardized network management tool. It is a UDP-based network protocol. As an application layer protocol, it is used widely in network management systems to facilitate the exchange of management information between different network-attached devices. It also helps the network administrators to monitor the conditions of network devices for attention.

SNMP is based on the manager/agent model. An SNMP-managed network has some key components: an SNMP manager, managed SNMP devices, an SNMP agent, a database of management information (MIB), and the network management protocol [7].

- The SNMP manager acts as an interface between the network administrator and the management system.

1874-110X/15

- The managed SNMP device is a network node that can be any type of hardware device such as computer hosts, routers, access servers, and printers that are connected to network.
- The SNMP agent is a network-management software module that resides on a managed device. It provides the interface between the manager and the physical managed device(s). It collects and stores management information received by the managed devices.
- The SNMP MIB is a collection of managed objects residing in a virtual database used to manage the devices in a communication network. The database is organized in a tree structure and entries are addressed through object identifiers (OID).

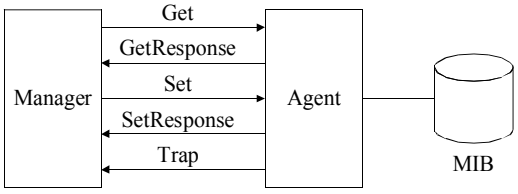

Fig. (1). Communication between the SNMP manager and agent.

As illustrated in Fig. (**1**), the SNMP manager and agent use a MIB and five basic commands (GET, GET-NEXT, GET-RESPONSE, SET, and TRAP) to exchange information [8]. The following Table **1** is a list of these commands.

Security has been the biggest weakness of SNMP since the beginning. SNMPv1/v2 can neither authenticate the source of a management message nor provide encryption. Without authentication, it is possible for nonauthorized users to exercise SNMP network management functions. It is also possible for nonauthorized users to eavesdrop on management information as it passes from managed systems to the management system. To correct the security deficiencies of SNMPv1/v2, SNMPv3 was issued as a set of Proposed Standards in January 1998. SNMPv3 addresses the security problems that have plagued both SNMPv1 and SNMPv2. So, by using SNMPv3 technology, we can not only manage the sensors in IoT, but also ensure the security and privacy of sensor data.

3. KEY PROBLEMS AND IMPLEMENT DETAILS

We choose SNMP to manage the sensors in IoT because SNMP provides a bare-bones set of functions, and it is indeed easy to implement, install, and use. As mentioned above, SNMP is based on three basic concepts: managers, agents, and the Management Information Base (MIB). In our IoT sensors management system, one manager node runs SNMP management software. The networked devices equipped with sensors to be managed are installed with an agent software module. This SNMP agent monitors sensor data. The management agent has data stored in memory locations. Polling the SNMP agent from a remote SNMP manager *via* the SNMP protocol can access the data of sensors embedded in IoT devices. The database within the manageable networked devices in IoT is called a MIB (management information base.) The MIB contains real-time status data of various sensors. So, the key problems in applying SNMP to manage the sensors in IoT involve the design of MIB, development of manager and agent software.

3.1. Design of MIB

MIB stands for Management Information Base and is a collection of information which is organized hierarchically. The SNMP MIB is conceptually a tree structure. The SNMP-related branches of the MIB tree are located in the internet branch, which contains two main types of branches:

- Public branches (mgmt=2), which are defined by the Internet Engineering Task Force (IETF) RFCs, are the same for all SNMP-managed devices.
- Private branches (private=4), which are assigned by the Internet Assigned Numbers Authority (IANA), are defined by the companies and organizations to which these branches are assigned.

The following Fig. (**2**) shows the structure of the SNMP MIB tree. There are no limits on the width and depth of the MIB tree.

Immediately beneath the root of the MIB tree, International Organization for Standardization (iso), is the Organization (org) branch, followed by Department of Defense (dod), and then Internet (internet). Management (mgmt), the main public branch, defines network management parameters common to devices from all vendors. Underneath the Management branch is MIB-II (mib-2), and beneath this are branches for common management functions such as system management, printers, host resources, and interfaces.

The private branch of the MIB tree contains branches for large organizations, organized under the enterprises branch.

Table 1. Five basic commands of SNMP.

Command	Operation
GET	The SNMP manager sends an SNMP get request to the specified node in the MIB tree to obtain one or more variables from an SNMP agent.
GET-NEXT	The SNMP manager sends an SNMP get next request to the next specified node in the MIB tree from an SNMP agent.
GET-RESPONSE	The SNMP agent responds with a GET-RESPONSE message with either the information requested or an error indication as to why the request cannot be processed.
SET	The SNMP manager uses the SNMP SET command to set or change the value of one or more variables in an SNMP agent.
TRAP	The SNMP agent sends a TRAP message to report the SNMP manager of extraordinary events.

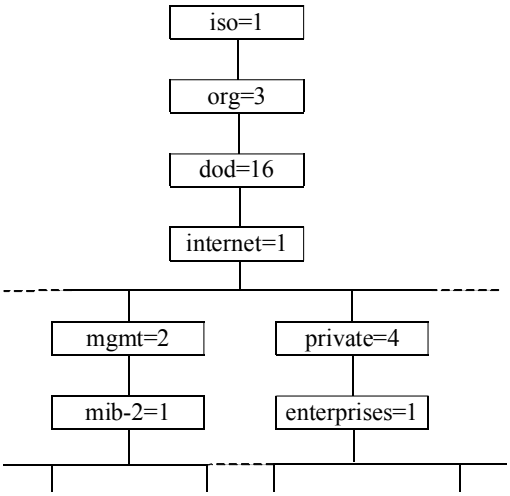

Fig. (2). Structure of the SNMP MIB tree.

Each organization has a root branch node under this object. Each organization creates its own subset of MIB branches and objects, which must comply with a common definition of SNMP information known as Structure of Management Information (SMI). SMI defines the allowed data types for MIB objects.

There are two types of MIBs: scalar ones and tabular ones. Scalar objects define a single object instance whereas tabular objects define multiple related object instances grouped in MIB tables. At the programmatic level, the definition of each MIB object that an SNMP agent manages includes the following elements:

- The object name and object identifier (also known as an OID).
- A text description of the object.
- The object's data-type definition (such as counter, string, gauge, or address).
- The index for objects that are assigned complex data types. The index specifies the key field for the table — that is, the field that can be used to identify a row.
- The only complex SNMP data type that is allowed is a table, and tables cannot be nested.
- The level of access to the object (such as read or read/write) that is allowed.
- Size restrictions.
- Range information.

Because the various sensors embedded in networked devices in IoT are non-standard managed objects, we must define private MIB objects. The MIB tree for managing sensors in IoT is actually a private MIB tree. Private MIB objects are defined under the node {1.3.6.1.4.1}, allowing individual users to expand their MIB objects.

SNMP references each MIB variable by using its unique object identifier, which identifies the location of a given managed object within the MIB namespace. The object identifier reflects the object's position within the hierarchy of the MIB tree, containing a sequence of subidentifiers that begin at the root of the MIB tree and end at the object (leaf node). Subidentifiers are separated with a period.

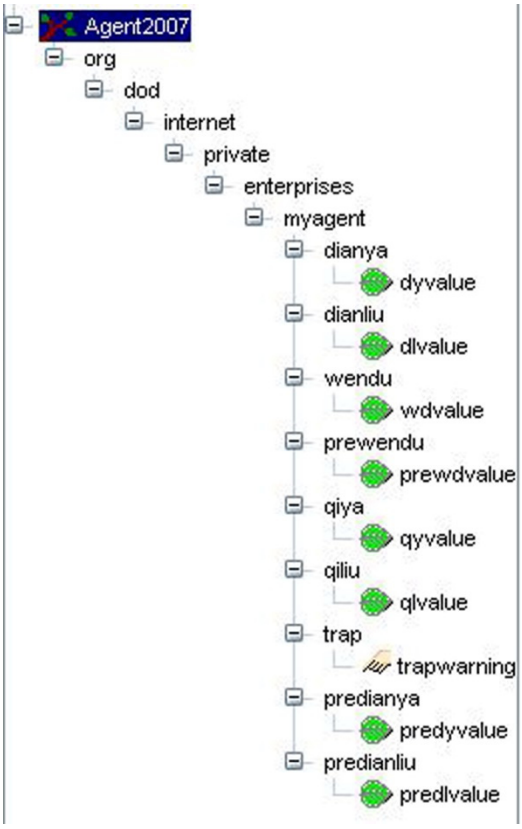

Fig. (3). An example of our designed MIB tree for managing some sensors information.

There are two methods to define a MIB. One is to write MIB text file directly in a text editor in accordance with SMI syntax. The other is to use some graphical tools to create a MIB tree. In this paper, we choose MIB Editor sub-module of Agent Toolkit C Edition to produce a MIB file. The MIB file defines the voltage, current and temperature variables of sensors. This MIB tree is shown in Fig. (**3**). A piece of code defining a voltage variable is shown as follows.

```
dyvalue    OBJECT-TYPE
    SYNTAX          DisplayString
    MAX-ACCESS      read-write
    STATUS          current
    DESCRIPTION     "Description :this variable de-
scripts the value of voltage"
    DEFVAL          { "220.0" }
    ::= { dianya 1 }
……
```

3.2. Design of Agent

SNMP agents do the bulk of the work. They are responsible for gathering information about the sensors in IoT and storing them in MIB. As mentioned above, the MIB is a hi-

erarchical, pre-defined structure that stores sensors information that can be queried. In a managed IoT devices which has some sensors, the specialized agent software accesses information about the sensors and make it available to the SNMP Manager. Managed sensors maintain values for a number of variables and report those, as required, to the manager. For example, an agent might report such data as the voltage of a sensor, temperature value monitored by a temperature sensor.

SNMP agents respond to most of the commands defined by the protocol. These include GetRequest, GetNextRequest, GetBulkRequest, SetRequest and InformRequest. In addition, an agent is designed to send Trap messages.

When the MIB file has been created, we can program the agent software using Agent Toolkit C Edition Tool. AdventNet Agent Toolkit C Edition is a rapid prototyping and development tool to build SNMP (SNMP v1, SNMP v2c and SNMP V3) agent [9]. It is a platform independent Java-based toolkit that can be installed in any operating system with JVM support. It offers a comprehensive and complete development environment with agent development tools for agent information-defining editors, Compilers for development and testing tools. Agent development tools such as MIB Editor, TL1 Message Builder, CLI Editor, MIB compiler, TL1 complier, CLI compiler reduce agent development complexity and enhance productivity. The main function of the agent module include managing agent MIB file, listening on the network port, SNMP packet decoding and encoding, and processing manager's request.

Take the MIB tree shown in Fig. (**3**) for example, if we want to read a sensor variable value "dianliu", we should add some code like this:

```
U_CHAR *GetDlvalue(INT32 *varValLen, U_CHAR
*status)
{
DEBUGMSG1("\n\t@@@@@ Inside GetDlvalue()
@@@@@\n");
CHECK_FOR_NULL(gv_dlvalue);
* Please provide your code to instrument "dlvalue" here
*/
printf("there is a getting requestment for the variable of
dianliu.\n");
printf("the value of dianliu is:%s A\n\n",gv_dlvalue);
*varValLen = __Strlen((CHAR *)gv_dlvalue);
return (U_CHAR *)gv_dlvalue;
}
```

3.3. Design of Manager

An SNMP manager is a component that is configured to poll SNMP agent for information. The management component, when only discussing its core functionality, is actually a lot less complex than the agent configuration, because the management component simply requests data.

The manager can be any machine that can send query requests to SNMP agents with the correct credentials. Sometimes, this is implemented as part of a monitoring suite, while other times this is an administrator using some simple utilities to craft a quick request.

Almost all of the commands defined in the SNMP protocol are designed to be sent by a manager component. These include GetRequest, GetNextRequest, GetBulkRequest, SetRequest, InformRequest, and Response. In addition to these, a manager is also designed to respond to Trap, and Response messages.

In our design, WinSNMP [10] is used to develop the SNMP manager. Using WinSNMP is as simple as including the correct header file as opposed to downloading and linking to an outside library. WinSNMP provides a single interface to which application developers can program and multiple SNMP software vendors can conform. This specification thus defines the procedure calls, data types, data structures, and associated semantics to which an application developer can program and which an SNMP software vendor can implement.

In general, the SNMP manager software include two important threads, sending thread and receiving thread, which has several implement steps. We can use the relevant WinSNMP API functions to code these steps so as to produce SNMP manager software module.

Sending thread include the following steps:

(1) Create the managed object's OID

(2) Create a variable binding table VBL

(3) Generate PDU and assign its content

(4) Send the messages

There are some API functions to support every step.

(1) Receive the messages

(2) Extract VBL from PDU, record the OID and value of every variable in VBL

(3) Convert the format and show people the get value in manager application.

A WinSNMP application must call the SnmpSendMsg function to request that the Microsoft WinSNMP implementation transmit the PDU, with the other required message elements defined by the relevant RFC. In addition to the destination entity, the application must specify the source entity and a context for the request. The SnmpSendMsg function executes asynchronously.

The WinSNMP application must call the SnmpRecvMsg function to retrieve the response to a SnmpSendMsg request. The implementation verifies the validity of the PDU and the other message elements when an application calls SnmpSendMsg.

The SnmpCreateSession function passes an application window handle and a notification message identifier to the Microsoft WinSNMP implementation. When the application window receives this message, it signals the application to call the SnmpRecvMsg function using the session handle returned by SnmpCreateSession.

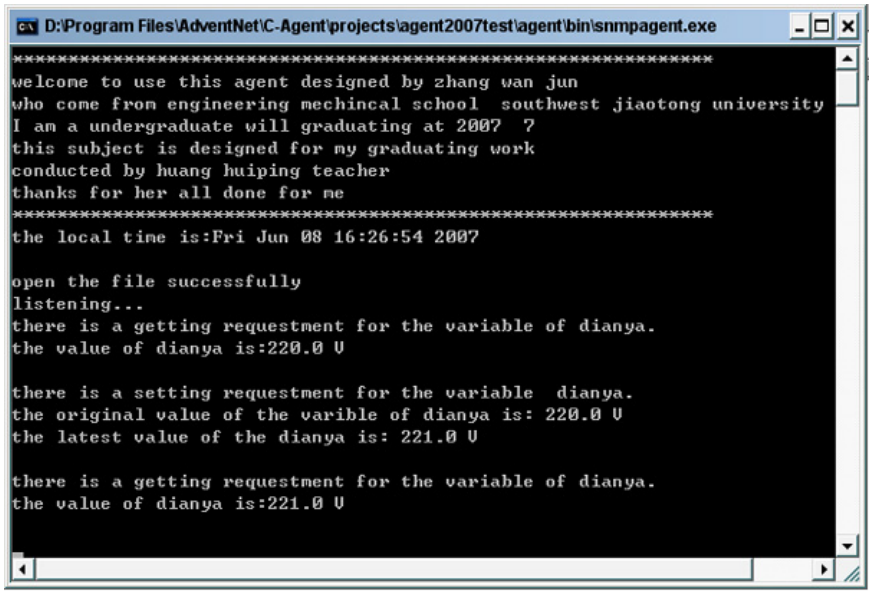

Fig. (4). Test results of our prototype system.

The SnmpRecvMsg function returns two entity handles, a context handle, and the handle to a PDU. It is recommended that the WinSNMP application free these resources using the SnmpFreeEntity, SnmpFreeContext, and SnmpFreePdu functions.

Several important programming function examples are listed as follows:

SnmpMgrOpen(

IN LPSTR lpAgentAddress, // Name/address of target agent

IN LPSTR lpAgentCommunity, // Community for target agent

IN INT nTimeOut, // Comm time-out in milliseconds

IN INT nRetries // Comm time-out/retry count

);

SnmpMgrRequest(

IN LPSNMP_MGR_SESSION session, // SNMP session pointer

IN BYTE requestType, // Get, GetNext, or Set

IN OUT RFC1157VarBindList *variableBindings, // Varible bindings

OUT AsnInteger *errorStatus, // Result error status

OUT AsnInteger *errorIndex // Result error index

);

4. TEST RESULTS OF A PROTOTYPE SYSTEM

We have developed a prototype system to validate the methods proposed in this paper. We use the MagicARM2200-A hardware platform, which has several sensors embed in it, to emulate the networked device in IoT. According to the implement details and steps described above, we develop a MIB file and an SNMP agent application with the AdventNet Agent Toolkit c edition software development kit. Also, we develop a SNMP manger with WinSNMP. The MIB file stores the information of the sensors embedded in MagicARM2200-A hardware platform. The agent application is aimed to manage the MIB file and responds to manager requests, such as get, getnext and set commands etc. Besides, the agent can store all the operation and the time when the operation carry out, so that the manager can query it when the device runs down, and yet tried the trap function. The manager can receive and display the context of the sensor information on the screen. The results of our prototype system in shown in Fig. (**4**). The results show that the SNMP-based methods proposed in this paper can facilitate management of sensors in IoT.

CONCLUSION

Now, more and more objects are becoming embedded with sensors and gaining the ability to communicate. Many IoT devices have sensors that can register changes in temperature, light, pressure, sound and motion. A variety of sensors play a key role in networked devices in IoT. To facilitate the management of such sensors, this paper presents a method based on SNMP technology and the implement details to setup the MIB file, agent and manager application. The agent stores all of its data into a tree of Management Information Bases(MIBs), which contain variables that hold the values representing sensors information. A manager application can send a message to the agent of the managed IoT devices embedded with sensors and get the value of the variables stored in MIB to know the sensor information. For example, when people want to know the relevant information about a temperature sensor in IoT, he can use the manager application to send a get request to agent, then receive the corresponding information stored in MIB. We have setup a prototype system to validate the methods presented in this paper. The test results show that the prototype system can work successfully and reliably. By applying SNMP technol-

ogy combined with the sensors in IoT devices, people will be able to easily collect, store, and view sensors information. In future, we plan to promote this method and prototype system to actual IoT systems and products so as to facilitate the management of sensors embedded in IoT devices.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

We acknowledge that our work is supported by the Sichuan Application Fundamental Research Funds (No. 2014JY0212).

REFERENCES

[1] QIAN Zhi-hong,WANG Yi-jun, "IoT technology and Application", Acta Electronica Sinica, vol. 40, no. 5, pp. 1023-1028, May 2012.

[2] J. Schönwälder and V. Marinov, "On the Impact of Security Protocols on the Performance of SNMP," IEEE Trans. Network and Service Management, vol. 8, no. 1, pp. 52-64. Mar. 2011,

[3] Sehgal, Anuj; Perelman Vladislav;and etc., "Management of resource constrained devices in the internet of things". IEEE Communications Magazine, vol. 50, No. 12, pp. 144-149, May 2012

[4] Deng, Hubin; Liu, Guiyuan; Zhang, Lei. "Analysis and implementation of embedded SNMP agent", Computer and Computing Technologies in Agriculture IV - 4th IFIP TC 12 Conference, Nanchang, China. 2010, pp. 96-102

[5] Hillbrecht, Ricardo, De Bona, Luis Carlos. "SNMP-based virtual machines management interface". Proceedings - 2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC 2012, Chicago, Illinois, USA, 2012, pp. 279-286

[6] Dow, Chyi-Ren; Lee, Yu-Hong; Hu Ruei, Yu. "Adaptive SWE and SNMP-based sensor management for environmental monitoring", International Journal of Communication Networks and Distributed Systems, vol. 13, pp. 314-334, January 2014.

[7] LIU Su-ping, DING Yong-sheng1,"A Scalable Policy and SNMP Based Network Management Framework", Journal of Donghua University(English Edition), vol. 26, PP. 143-146, February 2009.

[8] Li Mingjiang, "Simple Network Management Protocol", Beijing, Electronic Industry Press, 2008, pp. 186-190.

[9] Jiang Fei, Shi Hao-shan, Xu Zhi-yan, Dong Xiang-jun, "Novel hierarchical framework for the network management system based on multi-agent collaboration", Journal of Xidian University, vol. 36, pp. 366-372. Apr. 2009.

[10] Wang Dong, "The Design and Realization of Network Managment Software Based on SNMP". M.S. thesis, Beijing University of Posts and Telecommunications, Beijing, 2012.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015