UPnP-Based Sensor Network Management Architecture

Hyungjoo Song, Daeyoung Kim, Kangwoo Lee, Jongwoo Sung

Real-time and Embedded Systems Lab
Information and Communications University
{iamhjoo, kimd, kangn2, jwsung}@icu.ac.kr

ABSTRACT

Abstract— We propose a management framework for sensor networks based on the standard service discovery protocol, UPnP. Using UPnP, we can achieve easy deployment of sensor network services and provide zero-configuration sensor networking. Since sensor nodes are highly resource limited, it is not feasible to accommodate UPnP protocol in each sensor node. Thus, the framework includes an UPnP agent, called BOSS (Bridge Of the SensorS), which is implemented in the base station and lies between the UPnP controllers and the non-UPnP sensor nodes to be managed. In addition to the BOSS agent, UPnP device and service descriptions for sensor network services and management are documented in XML. We implemented the framework using an xscale-based base station and Berkeley's sensor node, MICA2.

Keywords: sensor network, management, UPnP, middleware

1 INTRODUCTION

Over the past few years, the sensor network has been used mainly in the military and science fields. But recently, sensor network technologies are being applied to more places such as the smart home, offices and universities, and are considered a main component of ubiquitous computing. As the utilization of the sensor network is increasing, sensor network users face problems related to the configuration and control of the many sensor devices. Since there is still not standardization of sensor network management technology, the establishment of general sensor network management tools has become a necessity.

UPnP architecture can be a good candidate to solve this problem. UPnP allows zero-configuration networking and automatic discovery of devices; it also leverages TCP/IP and the Web to enable seamless proximity networking, in addition to control and data transferring among networked devices in the home, office, and others [1]. As well, UPnP devices can provide available services to UPnP control point devices like PDAs, notebooks, etc. And, if we use UPnP, other network applications like smart home applications based on PLC, Wireless LAN, and IEEE 1394 are easily interoperable with sensor networks.

However, UPnP is not suitable for tiny sensor devices because it is operated on top of a TCP/IP. Applying UPnP to the device requires computational power and memory space to process UPnP protocol. However, tiny sensors have very limited physical resources, including low-end CPU, small memory space and limited power.

To overcome this limitation and apply UPnP to the sensor network, we propose UPnP-based sensor network management architecture. The basic concept of this architecture is to allow non-UPnP tiny sensor devices with limited resources to use UPnP through the implementation of bridge architecture between the UPnP control point and sensor nodes. We have named this base node *BOSS (Bridge Of the SensorS)*.

In this architecture, BOSS plays an important role since it contains the services of each sensor to provide them with a control point. As well, BOSS is able to interpret and transfer messages between the sensor network and the control point since UPnP protocol using an XML message format is different from a sensor network specific network message format. In addition, BOSS provides overall sensor network management services, such as localization, power management, context awareness, etc.

This paper is organized as follows. Section 2 introduces the related work of our paper. In Section 3, we present the UPnP sensor network architecture and explain the necessary components. Section 4 explains BOSS architecture and the sensor network management services it provides. The design and implementation of our architecture are discussed in Section 5 and 6. And finally, we present our conclusions in Section 7.

2 RELATED WORK

Currently, researches on sensor network management are actively progressing. In [2], a web-based sensor network gateway is used to manage the sensor network. This has some advantages in that the Internet has flexibility, facility of development, and convenience of access. But when users using this architecture want to add new sensor nodes with a new sensing facility, they must reconfigure the sensor nodes manually and modify the management program.

Another attempt to apply UPnP to sensor networks [3] can be found in Sindrion architecture. Sindrion's basic idea is that complex operations like UPnP control and eventing are sourced out from the sensors to the device in order to secure more computing power, called terminal. In ther words, the Sindrion Transceiver, which is a small RF transceiver with integrated microcontroller attached to embedded sensors or actuators, do not implement UPnP control and eventing capabilities. The Sindrion Transceiver

only supports UPnP discovery, description and presentation, which need much less computational effort than UPnP control and eventing. In the Sindrion system, the Java applet is included in the presentation page downloaded from the Sindrion Transceiver for the terminal to process UPnP control and event [3]. In this architecture, each sensor node has to operate based on the TCP/IP to support UPnP addressing and discovery. However, sensor nodes have too limited resources to process the UPnP protocol. It is infeasible to apply Sindrion architecture to a resource limited sensor. As such, we utilize the base node as the bridge to connect the sensor network and the UPnP network.

3 SENSOR NETWORK MANAGEMENT ARCHITECTURE

Fig. 1 represents sensor network management architecture based on the UPnP. This architecture is composed of three parts: UPnP control point, BOSS, and non-UPnP sensor nodes.

In UPnP technology, the control point is the entity in the network that works with the services provided by UPnP devices. In our architecture, the control point is a device to control and manage a sensor network using the services provided by BOSS, which is also an UPnP device that has enough powerful resources to compute UPnP protocol. This base node uses bridge architecture for communication between the sensor network and control point. Finally, each sensor node is a non-UPnP device which has sensing capabilities. But the current sensor device does not have sufficient resources to be able to process the UPnP protocol. In our architecture, the control point and BOSS communicate with each other using UPnP protocol. On the other hand, non-UPnP sensor devices and BOSS use sensor network specific protocol for communication.

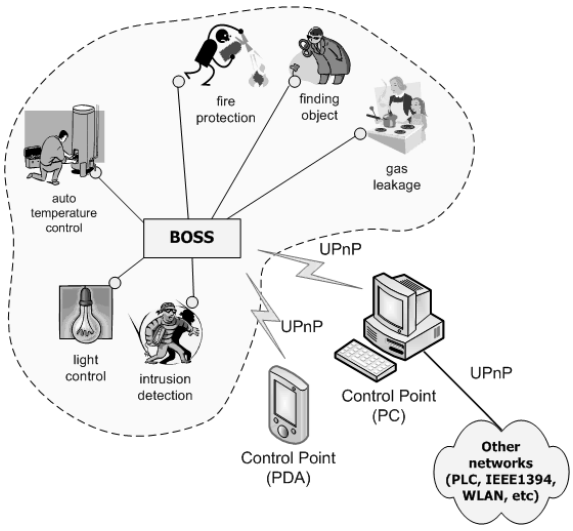

Fig. 1. UPnP-based sensor network management architecture

4 BOSS : Bridge Of SensorS

4.1 Architecture

Fig. 2 shows BOSS architecture. BOSS has five components: service manager, control manager, event manager, service table, and sensor network management service.

The main functionalities of UPnP are automatic device and its service discovery, device control, and event processing. UPnP uses XML-based messages such as SSDP (Simple Service Discovery Protocol), SOAP (Simple Object Access Protocol), and GENA (Generic Event Notification Architecture) to support them. If a device wants to support UPnP, it needs to implement these UPnP protocols. However, as previously mentioned, UPnP protocols can not be operated on tiny sensors. Thus, BOSS applies to the manager to interpret and transfer UPnP messages, instead of non-UPnP sensor devices. The service table is the data structure to store the services of the sensor nodes under BOSS. In addition, BOSS provides sensor network management services which are required to manage the sensor network.

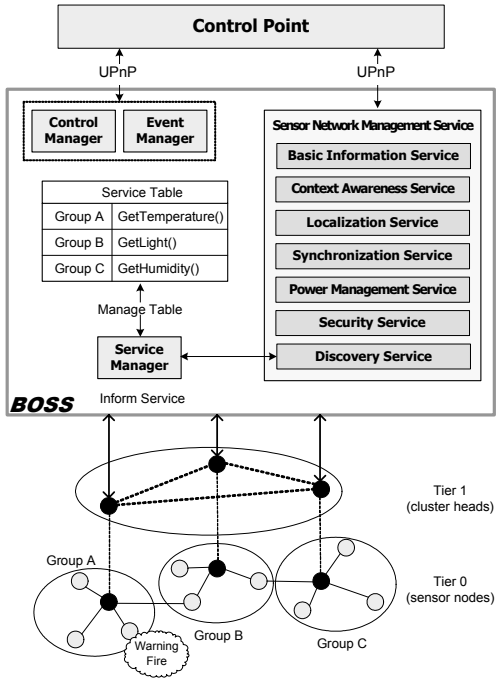

Fig. 2. BOSS Architecture

4.2 Service Manager

When an UPnP device is added to the network, it advertises its service to the control point. However, non-UPnP sensor devices do not have this capability so that they

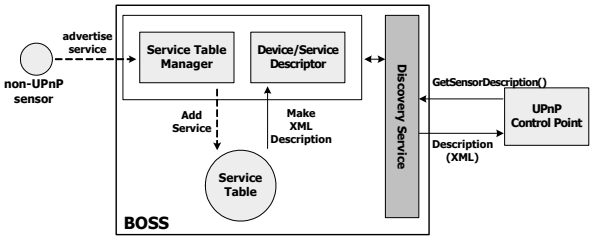

Fig. 3. Service Manager internal structure

can not be connected to the control point directly through the UPnP network. In our architecture, the *Service Manager* and discovery service of BOSS do this work on behalf of the sensors. BOSS has the ability to know the list of services of the sensors through communicating with sensor nodes using sensor network proprietary protocol.

Fig. 3 shows the internal structure of the Service Manager and work flow. The process is as follows: if the sensor is added in the sensor network, it sends a service advertisement message to the service table manager. The service table manager adds the entry into the service table with a received message from the non-UPnP sensor devices. At this time, the control point receives an event notification message about the device addition through UPnP eventing. The control point can call the *GetSensorDescription()* action, one of the discovery services provided by BOSS, in order to get the description message related with the sensors. When *GetSensorDescription()* is called by the control point, the device/service descriptor makes a sensor description message based on XML, and sends the created message to the control point. In addition, when a control point is added to the networks, the control point can request the description document for the nodes in the sensor network in which it has participated using the discovery service.

4.3 Control Manager

After the control points get the descriptions of the devices, they can invoke any of the actions included in the device services. In UPnP, this process is called *control*. As we mentioned, UPnP uses SOAP to deliver control messages to devices and returns the result to the control points. While SOAP is an XML-based protocol operated on top of a TCP/IP, non-UPnP sensor devices are not able to implement SOAP protocol. As such, our architecture provides the *Control Manager* with control points in order

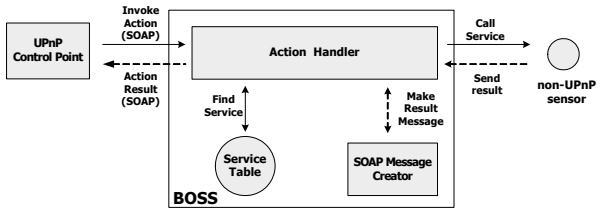

Fig.4. Control Manager internal structure

to use the services of non-UPnP sensor devices by using SOAP.

Fig. 4 shows the internal architecture of the Control Manager and work flow. After service discovery of the sensors through the *GetSensorDescription()* action, it has the ability to know the list of services of the sensors. When the control point invokes the service of a sensor in which it is interested, it sends the Control Manager a SOAP message for the service. The Action Handler of the Control Manager then parses the SOAP message and sends a sensor network message to the node which has the service mapped with the invoked action. When the Action Handler responds to the result from the node, it translates the message into a SOAP reply message through the SOAP Message Creator and transfers the created message to the control point.

4.4 Event Manager

Since the control point has discovered UPnP devices and retrieved a description of the devices and their services, it can use UPnP *eventing*. UPnP eventing allows the control points to receive information regarding device state changes. UPnP eventing uses the publisher/subscriber model. The control point is the subscriber of events while the UPnP device is the publisher of events. As such, the control point must subscribe to the event of the device in which it is interested before using UPnP eventing. When the state variables of the UPnP device change, it notifies the event to the control point subscribed to the device's service. In UPnP eventing, a GENA (General Event Notification Architecture) protocol, which is formatted using XML, is used to subscribe and notify the event. BOSS provides the *Event Manager* with the ability to process the event of non-UPnP sensors whose resources are too limited to implement UPnP eventing.

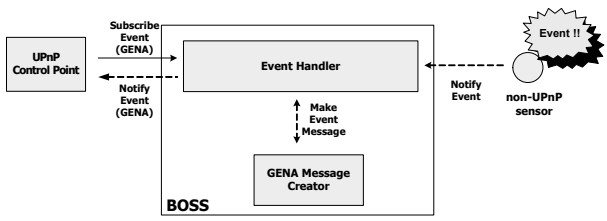

Fig. 5. Event Manager internal structure

Fig. 5 presents the internal structure of the Event Manager and work flow. As mentioned above, the control point sends the subscription request to the Event Handler to subscribe the event. And then, when some event happens in the non-UPnP sensor, the Event Handler receives the event notify message from the sensor node. The Event handler converts that message into an UPnP event notify message based on GENA through the GENA Message Creator, and transfers the converted message to the control point.

4.5 Sensor Network Management Service

BOSS is also an UPnP device with UPnP-based sensor network management services. The control point can manage the sensor network in which it is contained using these services as in Table 1.

TABLE. 1
SERNSOR NETWORK MANAGEMENT SERVICES

Service	***Description***
Basic Information	Using the basic information service, we can get basic information from the sensor network.. (Ex. the device description, the number and topology of the sensor nodes.)
Context Awareness	UPnP eventing happens only when state valuables already defined change in each device. This is simple and provides poor information to the user. But if we use the concept of context awareness, the sensor network becomes more intelligent. Context is any information that can be used to characterize the situation of an entity such as a person, place, or object [5].
Localization	Sensor node positions are considered necessary information in many sensor network applications. Using the localization service, we can find the position o f each sensor node using our localization algorithm. (Ex., start localization algorithm.)
Synchronization	Clock synchronization among nodes in the sensor network is important. When a user configures the sensor network or a sensor node is added, we can synchronize nodes using this service.
Power Management	Power is a very important factor in embedded devices like sensor nodes. Users can manage the power of the sensor nodes using this service. (Ex. check the remaining battery, change operation mode of sensors.)
Discovery	The control point has the ability to get a list of services of the sensor nodes under BOSS using this service.
Security	Using the security service, only an authenticated user is allowed to use the sensor network.

5 DESIGN DEVICE/SERVICE DESCRIPTION

5.1 Specification of UPnP Description

UPnP devices use a simple XML document to describe the device information and services which they provide. Thus, UPnP description is divided into two parts: device description and service description. We must use the XML syntax defined by the UPnP Forum to create device and service descriptions.

Device descriptions contain the vendor name, device information, a list of services and the URL for control and eventing. Table 2 shows the required field of UPnP device description. In addition, we may define optional elements like *manufacturerURL, modelNumber, deviceList,* etc. [4]. The service description contains more detailed information about the provided services than the contents described in the *serviceList* field in the device description. Service description defines actions and their arguments; state variables and their data type; and event characteristics [4].

TABLE. 2
UPNP DEVICE DESCRIPTION REQUIRED FIELD

Element	***Description***
deviceType	UPnP device type
friendlyName	Short text description of the device for end user
manufacturer	Manufacturer's name
modelName	Device's model name
UDN	Unique Device Name; universally-unique identifier for the device.
serviceList	serviceType, serviceID, SCPDURL, controlURL, eventURL

5.2 Implementation of BOSS Device Description

The UPnP device must define the device description document as explained before. However, it is inefficient, in our architecture, to define the description documents for all sensor nodes and transfer them to the control point. To solve this problem, we divide the description into two parts: BOSS description part and sensor nodes description part. Using this approach, BOSS description uses the UPnP description mechanism, but the descriptions for the sensor nodes use new description methods to protect against duplication for the same kind of sensor nodes. These sensor node descriptions are created by the Service Manager of BOSS and provided to the control point through a discovery service provided by BOSS. Currently, there is no standard for sensor device description in UPnP. To implement BOSS, we use the Basic Device 1.0 standard suggested in the UPnP Forum, that provides a mechanism for products that wish to use UPnP, but for which there is not yet an appropriate standard base device type [6]. Its implementation is shown in the Appendix.

A sensor description made by the Service Manager is similar to the UPnP device description. Its implementation for Fig. 2 is as follows. Most fields, with the exception of <*sensorID*>, are similar to those used in UPnP device description. We have defined the <*sensorID*> field to describe the same sensor devices. *Group A* in Fig. 2 are the same sensor devices with temperature sensing functionality so that the Service Manager only inserts the sensor node ID in the <*sensorID*> field like **`node1:node2:node3`**, instead of defining each device description respectively. *node1, node2* and *node3* refer to the node identifier, and ':'

means the id classifier. After receiving this document, the control point has the ability to know the device description for the sensor nodes by parsing it.

```
<sensorList>
  <sensor>
    <sensorType>
       urn:schemas-resl-icu-ac-kr:device:
            TemperatureSensor:1</sensorType>
    <manufacturer>RESL</manufacturer>
    <sensorID>node1:node2:node3</sensorID>
    <service>
      <serviceType>
          urn:schemas-resl-icu-ac-kr:service:
              GetTemperature:1
       </serviceType>
      <argument>
        <name>temperature</name>
        <direction>out</direction>
        <dataType>float</dataType>
      </argument>
    </service>
  </sensor>
  <sensor> .. Group B description .. </sensor>
  <sensor> .. Group C description .. </sensor>
</sensorList>
```

Fig. 6. Sensor nodes description example

5.3 Implementation of BOSS Service Description

The BOSS service description is divided into two parts: 1) sensor network management services provided by the BOSS; and 2) a list of service descriptions of non-UPnP sensor devices included in the device description made from the service manager of BOSS. A sample of a BOSS service description can be found in the Appendix.

6 IMPLEMENTATION

Fig. 7 is the captured image represented that control point discovered BOSS device in our architecture using UPnP discovery. iPAQ 3870 used control point which is running PocketPC 2003 and the Jeode VM[8] has an Intel StrongARM 1110(206MHz), 32Mbytes flash memory, 64Mbyte RAM. The control point application was implemented by using Siemens Java SDK v1.01 for UPnP[9] because this supports Personal Java, Java1.1 or Java 2. We chose the EMPOS Ⅱ [10] as the BOSS platform. EMPOS Ⅱ running linux (kernel 2.4.19) and Blackdown Java 1.3[11] has an Intel XScale PXA-255(400MHz), 32Mbytes Flash memory, and 128Mbytes SDRAM. This specification is suitable for processing UPnP protocol. We also used Siemens UPnP SDK to implement BOSS architecture. MICA2 is used as the non-UPnP sensor node in our architecture. Moreover, to transfer messages between BOSS and a set of nodes, MICA2 as the root node is connected to the EMPOS through serial port. MICA2 uses an Atmel ATMEGA 128L (7MHz) 8-bit microcontroller and has a 128Kbyte flash memory and a 4Kbtye static RAM. MICA2 runs the TinyOS [11] and is suited for a wireless sensor network. Message Handler processes Active Message, the basic networking primitive of TinyOS, for the communication with the sensor nodes.

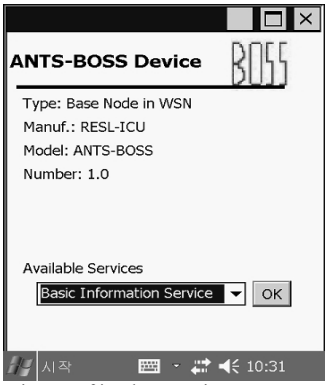

Fig. 7. Capture image of implementation program

7 CONCLUSION

We presented an UPnP sensor network management architecture based on BOSS for non-UPnP sensor devices. BOSS is not only the bridge between the sensor network and the UPnP control point, but is also an UPnP device that provides sensor network management services. Although a resource limited sensor node is not able to support UPnP, we can handle non-UPnP sensor nodes like an UPnP device by using BOSS. In addition, the user using the sensor network in which our architecture is applied can manage the sensor network easily through various services provided by BOSS. Moreover, the sensor network in which our architecture is applied can be easily connected to other networks (WLAN, IEEE 1394, PLC, etc) through UPnP so that the utilization of the sensor network can be maximized.

REFERENCES

[1] UPnP Forum, http://www.upnp.org

[2] K.I.Hwang, J.S.In, N.K.Park, D.S. Eom, " A Design and Implementation of Wireless Sensor Gateway for Efficient Querying and Managing through World Wide Web" , IEEE Transactions on Consumer Electronics, Vol. 49, Issue:4, Nov 2003 Pages:1090 - 1097

[3] Yvonne Gsottberger, Xiaolei Shi, Guido Stromberg, et al, " Embedding Low-Cost Wireless Sensors into Universal Plug and Play Environments" , EWSN 2004, LNCS 2920, pp.291-306, Springer-Verlag Heidelberg 2004

[4] Universal Plug and Play Device Architecture Reference Specification Version 1.0, Microsoft Corporation, June 2000, http:/www.upnp.org

[5] A.K.Dey, G.D.Abowd, " Toward a better understanding of context and context-awareness." , GVU Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology

[6] UPnP Basic Device 1.0 standard, UPnP Forum, Available: http://www.upnp.org/standardizeddcps/default.asp

[7] iPAQ 3870, http://www.hp.com
[8] Insignia Jeode VM, http://www.insignia.com
[9] Siemens Java SDK v1.01 for UPnP, http://www.plug-n-play-technologies.com
[10] EMPOS Ⅱ, Hanback Electronics, http://www.hanback.co.kr
[11] Blackdown Java, http://www.blackdown.org
[12] TinyOS, http://www.tinyos.net

APENDIX

1. Simplified BOSS Device Description Example

```
<?xml version="1.0" ?>
 <root xmlns="urn:resl-icu-ac-kr:device-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <URLBase>http://resl.icu.ac.kr:5431</URLBase>
  <device>
   <deviceType>
    urn:schemas-resl-ac-kr:device:BOSS:1
   </deviceType>
   <friendlyName>ANTS BOSS</friendlyName>
   <manufacturer>RESL</manufacturer>
   <manufacturerURL>http://resl.icu.ac.kr
   </manufacturerURL>
   <modelDescription>ANTS BOSS 1.0</modelDescription>
   <modelName>ANTS-BOSS</modelName>
   <UDN>uuid:RESL_ANTS-BOSS-1_12356789</UDN>
   <serviceList>
    <service>
     <serviceType>
       urn:schemas-RESL-ANTS:service:BasicInfo:1
     </serviceType>
     <serviceId>
       urn:schemas-RESL-ANTS:serviceId:BasicInfo
     </serviceId>
     <SCPDURL>/BOSS/BasicInfo.xml</SCPDURL>
     <controlURL>/BOSS/control/basicinfo</controlURL>
     <eventSubURL>/BOSS/event/basicinfo</eventSubURL>
    </service>
    <service>
     <serviceType>
       urn:schemas-RESL-ANTS:service:ContextAware:1
     </serviceType>
     <serviceId>
       urn:schemas-RESL-ANTS:serviceId:ContextAware
     </serviceId>
     <SCPDURL>/BOSS/PowerManage.xml</SCPDURL>
     <controlURL>/BOSS/control/powermanage</controlURL>
     <eventSubURL>/BOSS/event/powermanage<eventSubURL>
    </service>
    <service> Localization Service</service>
    <service> Synchronization Service </service>
    <service> Power Management Service </service>
    <service> Discovery Service </service>
    <service> Security Service </service>
   </serviceList>
  </device>
</root>
```

2. Partial BOSS Basice Information Service Description (BasicInfo.xml)

```
<?xml version="1.0" ?>
 <scpd xmlns="urn:schemas-resl-ac-kr:BOSS-service-1-0">
  <specVersion>
   <major>1</major>
   <minor>0</minor>
  </specVersion>
  <actionList>
   <action>
    <name>GetTotalSensors</name>
    <argumentList>
     <argument>
       <name>totalSensors</name>
       <direction>out</direction>
       <relatedStateVariable>
         totalSensors
       </relatedStateVariable>
     </argument>
    </argumentList>
   </action>
   <action>
    <name>GetNodeInfo</name>
    <argumentList>
      <argument>
        <name>sensorID</name>
        <direction>in</direction>
        <relatedStateVariable>
          A_ARG_sensorID
        </relatedStateVariable>
      </argument>
      <argument>
         <name>nodeInfo</name>
         <direction>out</direction>
         <relatedStateVariable>
           nodeInfo
         </relatedStateVariable>
      </argument>
    </argumentList>
  </action>
             ...
 </actionList>
 <serviceStateTable>
   <stateVariable sendEvents="yes">
      <name>totalSensors</name>
      <dataType>ui2</dataType>
   </stateVariable>
   <stateVariable sendEvents="no">
      <name>A_ARG_SensorID</name>
      <dataType>string</dataType>
   </stateVariable>
   <stateVariable sendEvents="no">
      <name>nodeInfo</name>
      <dataType>string</dataType>
   </stateVariable>
            ...
 </serviceStateTable>
</scpd>
```