IEICE TRANS. COMMUN., VOL.E90–B, NO.11 NOVEMBER 2007

PAPER Special Section on Next Generation Network Management

Policy-Based Management for Self-Managing Wireless Sensor Networks

Si-Ho CHA[†], **Jong-Eon LEE**[††], **Minho JO**[†††a)], **Hee Yong YOUN**[†††], ***Members***,
Seokjoong KANG[††], ***and*** **Kuk-Hyun CHO**[††], ***Nonmembers***

SUMMARY In a wireless sensor network (WSN), a large number of sensor nodes are deployed over a wide area and multi-hop communications are required between the nodes. Managing numerous sensor nodes is a very complicated task, especially when the energy issue is involved. Even though a number of ad-hoc management and network structuring approaches for WSNs have been proposed, a management framework covering the entire network management infrastructure from the messaging protocol to the network structuring algorithm has not yet been proposed. In this paper we introduce a management framework for WSNs called SNOWMAN (SeNsOr netWork MANagement) framework. It employs the policy-based management approach for letting the sensor nodes autonomously organize and manage themselves. Moreover, a new light-weight policy distribution protocol called TinyCOPS-PR and policy information base (PIB) are also developed. To facilitate scalable and localized management of sensor networks, the proposed SNOWMAN constructs a 3-tier hierarchy of regions, clusters, and sensor nodes. The effectiveness of the proposed framework is validated through actual implementation and simulation using ns-2. The simulation results reveal that the proposed framework allows smaller energy consumption for network management and longer network lifetime than the existing schemes such as LEACH and LEACH-C for practical size networks.

key words: *wireless sensor network, PBM, self-management, WSN PIB, TinyCOPS-PR, clustering*

1. Introduction

Wireless sensor network (WSN) consists of a large number of small sensor nodes having sensing, data processing, communication, and ad-hoc functions. WSNs are characterized by densely deployed nodes, frequently changing network topology, variable traffic, and unstable sensor nodes of very low power, computation, and memory constraints. Theses unique characteristics and constraints present numerous challenging problems in the management of WSNs which are not encountered in the traditional wireless networks [1], [2]. Among them, energy-efficient management for prolonging the network lifetime needs a special attention.

The management of WSNs [3], [4] requires to be lightweight, autonomous, intelligent, and robust. A network management system needs to handle a certain amount of control messages to cope with various conditions of the network. With WSNs, it is extremely important to minimize the signaling overhead since the sensor nodes have limited battery life, storage, and processing capability. Given the dynamic nature of WSNs, an adaptive management framework that autonomously reacts to the changes in the network condition is required. The diverse and hostile environments of WSNs also require the network management system to be robust. Note that even a single faulty node in a traditional wireless network may harm the operation of the entire network. This is not the case with the WSN. Since several sensor nodes in a WSN sense the event simultaneously, a single faulty node does not affect the operation of the whole network. Therefore, it is not energy-efficient to directly manage all the individual sensor nodes.

Ruiz [5] proposed WSN models for guiding the management activities using the correlation between them. Younis [6] proposed to monitor and manage the sensor network using some agent sensors among the sensor nodes. An intelligent self-organized management mechanism was also suggested by Lee [7] using a hierarchical management structure. In addition to these schemes, a number of ad-hoc management and network structuring approaches for WSNs have been reported. However, a management framework covering the entire network management infrastructure from the messaging protocol to the network structuring algorithm has not yet been proposed. Moreover, few actual implementations of the proposed schemes have been reported.

In this paper, therefore, we introduce a management framework for WSNs. The proposed SNOWMAN (SeNsOr netWork MANagement) framework employs the policy-based management (PBM) [8], [9] approach for letting the sensor nodes autonomously organize and manage themselves. The PBM approach adopted in this paper is to dynamically apply the changes in the management requirement to the sensor nodes by modifying only the high-level policies and let the sensor nodes make local decisions by themselves using the policies. The PBM approach so far has been applied to the traditional TCP/IP network, but not yet to the WSN. Here XML schema is used to define high-level policies, and a new light-weight policy distribution protocol called TinyCOPS-PR is developed. The policy information base (PIB) for WSNs is also designed to be used by the sensor nodes. To facilitate scalable and localized management of sensor networks, the proposed SNOWMAN constructs a 3-tier hierarchy of regions, clusters, and sensor nodes based

Manuscript received March 19, 2007.
Manuscript revised June 13, 2007.

[†]The author is with the Department of Information and Communication Engineering, Sejong University, Korea.
[††]The authors are with the Department of Computer Science and Engineering, Kwangwoon University, Korea.
[†††]The authors are with the School of Information and Communication Engineering, SungKyunKwan University, Korea.
a) E-mail: minhojo@skku.edu

DOI: 10.1093/ietcom/e90–b.11.3024

on SNOWCLUSTER clustering algorithm. The effectiveness of the proposed framework is validated through actual implementation and simulation using ns-2. Compared to the existing schemes such as LEACH [12] and LEACH-C [13], the simulation results reveal that the proposed framework displays smaller energy consumption for network management and longer network lifetime than the existing schemes for practical size networks.

The rest of the paper is organized as follows. Section 2 discusses the related work, and Sect. 3 introduces the proposed framework. It is implemented and evaluated in Sect. 4. Finally, in Sect. 5, conclusions are made including the future research.

2. Related Works

The role of PBM is to manage the network using high-level policies independently of network devices. A policy is not applied to a specific device but various types of resources. By modifying only the high-level policies, PBM is able to implement the management dynamically applicable to all the devices. It is thus energy efficient. The IETF defined a policy framework [8] consisting of four basic elements: policy management tool (PMT), policy repository (PR), policy decision point (PDP), and policy enforcement point (PEP). The PMT is used by the administrator to input different policies. The PDP is responsible for interpreting the policies stored in the repository and sending them to the PEP. The PEP executes the policies. The PR is a storage where the policies are inserted by the PMT and retrieved by the PDPs. The IETF also proposed COPS/COPS-PR [10], [11] as the protocol for policy distribution. The COPS-PR is a protocol for exchanging information between the PDP and PEP in the format of PIB (Policy Information Base) tree, which consists of PRC (Provisioning Class) and PRI (Provisioning Instance) while conforming to ASN.1 standard. A PIB tree is a policy containing various management data. The COPS-PR, however, was designed for TCP/IP networks and thus it is too heavy to be applied to WSNs. We therefore propose a new management information exchange protocol, TinyCOPS-PR, particularly fitted to WSNs.

Ruiz [5] designed the MANNA architecture for WSNs, which considers three management dimensions: functional areas, management levels, and WSN functionalities. The proposed WSN models guide the management activities using the correlation between them. Here the conceptual view of the distribution of management functionalities among the manager and agent is described. Younis [6] proposed the architecture for monitoring and managing sensor networks. He suggested agent sensors that relay messages to and from unreachable sensors and groups of sensors around them while considering the load on each agent. Lee [7] proposed an intelligent self-organized management mechanism for sensor networks. The nodes are classified into three levels according to their functionality. The nodes in the lower-level are managed by those in the higher-level while they form a hierarchical management structure. His work indicates how high-level nodes form a cluster with low-level nodes through a contest.

When a sensor network is first activated, the neighboring nodes may wish to organize themselves into clusters so that redundant sensing can be avoided and the resources may be reused across non-overlapping clusters. In the clustered system, the data gathered by the respective sensor nodes are transmitted to the data processing server through a hierarchy of cluster heads. To improve the clustering efficiency, several clustering algorithms have been proposed such as LEACH [12] and LEACH-C [13]. LEACH is a self-organizing, adaptive clustering protocol which distributes the load evenly among the sensors in the network. It adopts randomized rotation of the cluster-head among the sensors in order not to drain the battery of a single sensor. It leads to balanced energy consumption of the nodes and hence a longer lifetime of the network. However, since LEACH does not consider the nodes' current energy level in selecting the cluster-head, it may not be effective when the nodes have different battery power as expected in reality. An improved version of LEACH called LEACH-C forms clusters at the beginning of each round using a centralized algorithm executed by the base station. It determines cluster heads based on the nodesĺocation information and energy level. This allows organizing robust clusters, but frequent communications between the base station and sensor nodes consume a significant amount of energy. Clusters are built in a different way from LEACH and LEACH-C in PEGASIS [14]. Here sensor nodes are connected by a chain and each node sends data only to the neighboring node. A node selected at random in the chain sends data to the base station. For comprehensive comparison on these algorithms, refer to Muruganathan, et al. [15]. The more detailed explanation is given in the following sections.

3. The Proposed Framework

3.1 The Overview of the Architecture

The proposed SNOWMAN framework includes a policy manager (PM), one or more policy server (PS), and a large number of policy agents (PAs) as depicted in Fig. 1.

The PM is used by the administrator to input various policies, and it is located in the manager node. A policy in this context is a set of rules assigning management tasks to the sensor nodes. The PS and PA reside in the base station and sensor node, respectively. The PS is responsible for interpreting the policies and sending them to the PA. The enforcement of the rules in the sensor nodes is handled by the PA. In a WSN, individual nodes will not be able to maintain a global view of the network. Such task well suites a machine which is not constrained by battery or memory. This is the reason for locating the PS in the base station. The job of the PS is to maintain a global view required to be capable of reacting to larger scale changes in the network such as replacement of old policies with the new ones. If the state of a node is changed or the current state matches preset rules, the

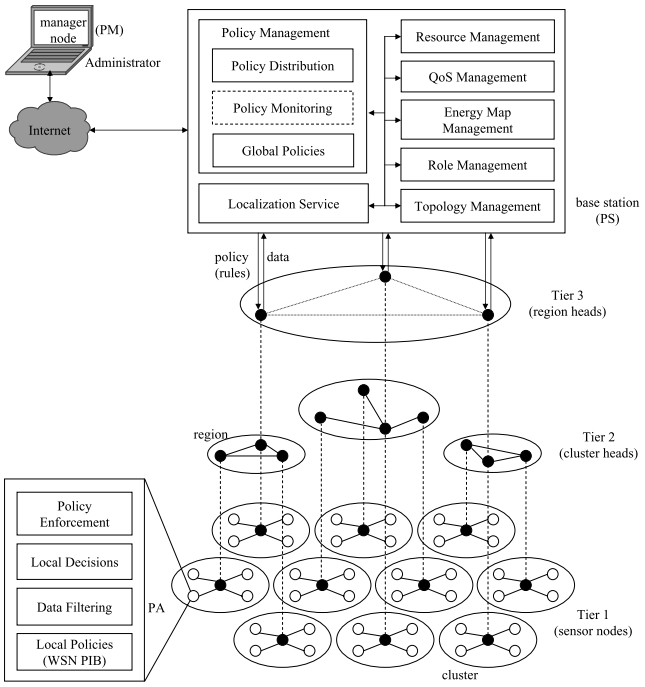

Fig. 1 The proposed SNOWMAN framework.

PA makes decisions based on local rules rather than sending the sensed data to the base station. This approach thus saves the resources of the sensor node, i.e., resulting in saved energy. In the proposed SNOWMAN framework, the PS is equivalent to the PDP and PA to the PEP of IETF framework, respectively.

To facilitate scalable and localized management of sensor networks, the SNOWMAN constructs a 3-tier hierarchy of regions, clusters, and sensor nodes as shown in Fig. 1. Here a WSN is comprised of regions. A region covers several clusters. The sensor nodes are aggregated to form clusters based on their power level and proximity. A subset of sensor nodes are elected as cluster heads. In the proposed 3-tier hierarchical architecture of SNOWMAN, the cluster heads constitute the routing infrastructure for handling the aggregated, fused, and filtered data from their neighboring sensor nodes. The PS can deploy specific policies in particular regions to manage them independently. Therefore, the SNOWMAN framework will be effective to provide optimized management to respective regions of the sensor network.

3.2 The Procedure of Management

The autonomic management of a WSN through the SNOWMAN framework is done by the following six steps (Step 1 and Step 2 are done by the SNOWCLUSTER and more detailed explanation is in Sect. 3.7):

1. Cluster-head Selection: the sensor nodes build clusters and elect a head node by themselves based on their power level.
2. Region-head Selection: each cluster head informs the PS in the base station of its power level and the information on its cluster, and then the PS selects region heads among the cluster heads using the information. The cluster heads and region heads are selected periodically to balance the power consumption of the whole network.
3. Service Discovery: the services which the WSN can provide are discovered along with other information such as sensor type, energy level, and so on.
4. Policy Definition: high-level policies are defined in XML by the services discovered in Step 3.
5. Policy Validation, Translation, and Distribution: the policies defined in Step 4 are verified and transformed into low-level policies and then distributed to the PAs of the sensor nodes.
6. Policy Enforcement: low-level policies distributed from the PS to the PAs are implemented which allows autonomic WSN management.

The proposed SNOWCLUSTER consists of 3 tiers compared to the 2-tier LEACH-C algorithm. The top tier called region head tier is adopted for sensor network management which LEACH-C does not have. Clustering in the top tier will increase the lifetime by minimizing the traffic among the nodes with efficient autonomous network management according to the policies, i.e., the sensor node applications (vigilance, data gathering, and so on) and sensor types (acoustic, seismic, and so on). The policy-based autonomous network management is explained in detail in the next sections.

3.3 The Functional Components

The PS consists of several functional components: policy distribution, policy monitoring, resource management, energy map management, QoS management, topology management, role management, and localization service. Region-wise localized service is achieved via role management and topology management. Global policies are specified by the network administrator in a logically centralized fashion and are expected to be static.

Policy distribution is the first essential task to ensure that the nodes are managed consistently with the defined policies. We developed and implemented the TinyCOPS-PR protocol that is a light weight version of COPS-PR protocol to deploy the policies in the sensor nodes.

The PS communicates with the PA using the TinyCOPS-PR protocol to distribute the policy, which allows asynchronous communication between the PS and PAs using notifications (reports, changes in policies, etc.) generated only when required. In order to provide robust management of the network, it is desirable to have an independent policy monitoring process to ensure that the deployed policies behave properly as defined. Policy monitoring occurs considering the limited network resources. Energy map management continuously updates the residual energy level of each sensor node, especially the cluster heads and re-

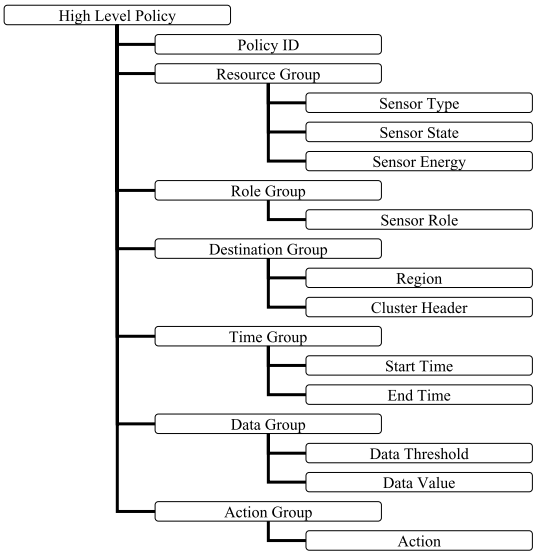

Fig. 2 The structure of high-level policy.

gion heads. It is achieved via topology management process, which consists of topology discovery, resource discovery, and role discovery. The resource management and role management handle the detected resources and roles (sensor node, cluster head, or region head), respectively. QoS management is a part of policy management based on the QoS policies such as bandwidth allocation for emergency. Energy map management goes through the aggregation and fusion phase when the energy information collected are merged and fused into energy contours by the cluster heads.

The PA enforces local policies assigned by the PS to make local decisions and filter out unessential redundant sensing data. For this, the PA consists of the functions of policy enforcement, local decision, data filtering, and local policies (WSN PIBs).

3.4 The High-Level Policy

The WSN management policies consist of two parts: high-level policy and low-level policy. The high-level policies are defined by the network administrator while the low-level policies are for device organization and sensor control. The conversion from high-level policies to low-level policies is made by the PS.

The structure of high-level policy is shown in Fig. 2, which contains Policy ID, Resource Group, Role Group, Destination Group, Time Group, Data Group, and Action Group, State, and Sensor Energy. The Role Group represents the information of the role of the sensors. Sensor Type is about the type of sensing such as seismic, acoustic, humidity, and so on. Sensor State is classified into Available, Unavailable, or Non-operational. Destination Group indicates the places where the policies are executed: region or cluster heads. The start time and ending time of a policy is determined by Time Group. The Data Group provides the threshold values on sensed data. The actions taking place in a sensor node according to the policy are contained in the Action Group.

Table 1 An XML schema for a WSN XML policy.

```
<?xml version=‘‘1.0’’?>
<xsd:schema xmlns:xsd=‘‘http://www.w3.org/2001/XMLSchema’’>
<xsd:element name=‘‘WSNPolicy’’ type=‘‘WSNPolicy’’ />
 <xsd:complexType name=‘‘WSNPolicy’’>
  <xsd:sequence>
   <xsd:element name=‘‘policyID’’ type=‘‘ID’’ />
   <xsd:element name=‘‘resourceGroup’’
                type=‘‘resourceGroup’’ />
   <xsd:element name=‘‘roleGroup’’ type=‘‘roleGroup’’ />
   <xsd:element name=‘‘destinationGroup’’
                type=‘‘destinationGroup’’ />
   <xsd:element name=‘‘timeGroup’’ type=‘‘timeGroup’’ />
   <xsd:element name=‘‘dataGroup’’ type=‘‘dataGroup’’ />
   <xsd:element name=‘‘actionGroup’’ type=‘‘actionGroup’’ />
  </xsd:sequence>
  </xsd:complexType>
 <xsd:complexType name=‘‘resourceGroup’’>
  <xsd:sequence>
   <xsd:element name=‘‘sensorType’’ type=‘‘xsd:string’’ />
   <xsd:element name=‘‘sensorState’’ type=‘‘xsd:string’’ />
   <xsd:element name=‘‘sensorEnergy’’
                type=‘‘xsd:decimal’’ />
   <xsd:element name=‘‘destinationGroup’’
  </xsd:sequence>
 </xsd:complexType>
  ...
 <xsd:complexType name=‘‘dataGroup’’>
  <xsd:sequence>
   <xsd:element name=‘‘dataThreshold’’ type=‘‘xsd:string’’/>
   <xsd:element name=‘‘dataValue’’ type=‘‘xsd:float’’/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name=‘‘actionGroup’’>
  <xsd:sequence>
   <xsd:element name=‘‘action’’ minOccurs=‘‘1’’
                maxOccurs=‘‘unbounded’’>
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name=‘‘sensorAction’’
                type=‘‘xsd:string’’/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

An XML policy document defined by the PM is transferred to the PS via SOAP protocol. After the PS verifies the received XML document using the XML schema, it is distributed to the sensor nodes via the TinyCOPS-PR protocol to apply the policy.

3.5 The TinyCOPS-PR and WSN PIB

IEEE 802.15.4 is appropriate for asynchronous data communication of low speed and volume with the WSN, which defines the MAC layer and physical layer. Therefore,the followings are the requirements in designing TinyCOPS-PR and PIB for WSNs.

- Asynchronous communication: while the existing COPS-PR protocol is for TCP-based synchronous communication, WSN requires asynchronous commu-

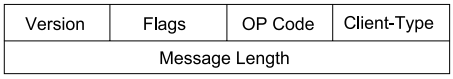

Fig. 3 The header of the TinyCOPS-PR protocol.

Table 2 The TinyCOPS-PR common header.

Name	Description	etc
Version	Protocol Version = 1	
Flags	Type of Commands combined with Op-Code	
OP Code	1 = Request (REQ)	PS ← PA
	2 = Decision (DEC)	PS → PA
	3 = Report State (RPT)	PS ← PA
Client-Type	1 = Use authentication processes	
	2 = DiffServ PIB	
	5 = WSN PIB	
	0x4000 ∼ 0x7FFF = private use	
	0x8000 ∼ 0xFFFF = enterprise use managed by IANA	
Message Length	Message in bytes	

nication allowing the sensor nodes to communicate independent from each other due to mainly limited power problem. Because the TinyCOPS-PR protocol introduced in this paper adopts IEEE 802.15.4, it also allows asynchronous communication such that the PA, PS, and PM are independently activated and communicate with each other.

- Light weight: WSNs restrict the volume of data per transmission as well as frequency of transmissions due to limited power. The proposed TinyCOPS-PR protocol adopts only the core operations of the COPS-PR protocol such as REQ, DEC, and RPT for this reason.
- Securing high-level policies: the definitions of WSN PIB should be consistent with the high-level policies.

3.5.1 The TinyCOPS-PR Protocol Suit

The TinyCOPS-PR protocol was designed by reforming the COPS-PR protocol to be fitted to the WSNs. Its header is compatible with the COPS headers, and an example is shown in Fig. 3. The only basic and core operations of the COPS-PR protocol are available in the TinyCOPS-PR protocol to minimize the communication cost as mentioned above.

Here, 'Flags' are combination of OP code, identifying the type of three operations, Request (REQ), Decision (DEC), and Report State (RPT). Note that the existing COPS-PR protocol has 10 operations. 'Client-Type' is used to represent the clients. The command header of the TinyCOPS-PR protocol is summarized in Table 2.

- Request (REQ): REQ is the 'Configuration Request' message transmitted from the PA to the PS. While the PEP sends REQ to the PDP whenever a policy is needed to be made, REQ is used only when a WSN is formed in the TinyCOPS-PR protocol. REQ is a message indicating completion of clustering and readiness of sensors to install the policies. 0s are inserted in the Flags field.
- Decision (DEC): This is a message sent from the PS to the PA for sending, installing, and deleting the policies. The policies are PRC and PIB in the PRI form. The PRC can be selectively installed and deleted. The value of Flags is 0 (Null Decision), 1 (Install policies), 2 (Remove policies).
- Report State (RTP): RTP is used by the PA in order to inform the PS of successful install of the policies. Flags can be 0 (Reboot), 1 (Success), 2 (Failure), or 3 (Emergency).
- Policy Provisioning Object Format (PPOF): PPOF is used for sending actual value of PIB, that is, PRID (Complete Provision Instance Identifier) or EPD (Encoded Provision Instance Data). An example of PRID message using an ID and an example of EPD message using the name of an object is shown in Fig. 4 and Fig. 5, respectively. PPOF can be used with DEC or RTP.

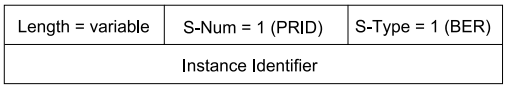

Fig. 4 Complete Provision Instance Identifier (PRID).

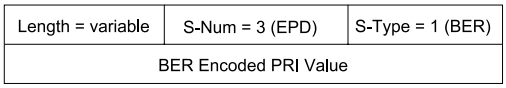

Fig. 5 Encoded Provision Instance Data (EPD).

3.5.2 The WSN PIB

The PIB of WSN is classified into Table 3 representing the condition and action of the sensors. An action of a sensor node is a combination of the functions embedded in the sensor node.

WSN PIB SMI is defined by entSensorConditionsTable representing sensor condition and action as illustrated in Fig. 6.

3.6 An Example of TinyCOPS-PR Message

Assume that we want to manage the WSN according to the following policy: filter out all sensing data of humidity below 0.8 from 10:00 a.m. to 1:00 p.m. in area R2. Then the PM creates a policy in XML as shown in Table 4, which will be verified by the predefined XML policy schema.

The created policy is sent in an SOAP message from the PM to the PS. The PS creates a TinyCOPS-PR message of the high-level policy received from the PM based on the PIB SMI structure. An example of DEC message installing the policy in the PA is given in Fig. 7. PRID 1.5.1 indicates entSensorConditionsTable and PRID 1.5.2 does the action. Sub-instances of the 2 PRIDs are represented in the EPD

Table 3 The classification of WSN PIB.

Name		Syntax	Description
Sensor Condition	Sensor Type	INTEGER (0..9)	Any(0), Temp(1), Light(2) Humidity(3), Infrared(4), Gas(5), Ultrasonic(6), Seismic(7), Acoustic(8), etc(9)
	Sensor State	INTEGER (1..3)	OK(1), Unavailable(2), Nonoperational(3)
	Sensor Energy	REAL	Values for current sensor energy status
	Sensor Role	INTEGER (0..3)	Any(0), Region(1), Cluster head(2), Common(3)
	Start Time	INTEGER	Policy start time
	End Time	INTEGER	Policy end time
	Sensor Data Thres-holds	INTEGER (1..6)	lessThan(1), lessOrEqual(2), greaterThan(3), greaterOrEqual(4), equalTo(5), notEqualTo(6)
	Sensor Data Value	REAL	Values for current sensing data value
Sensor Action		INTEGER (0..6)	Null(0), Filtering(1), Merging(2), Actuation(3), EnergySaving(4), Clustering(5), Notification(6)

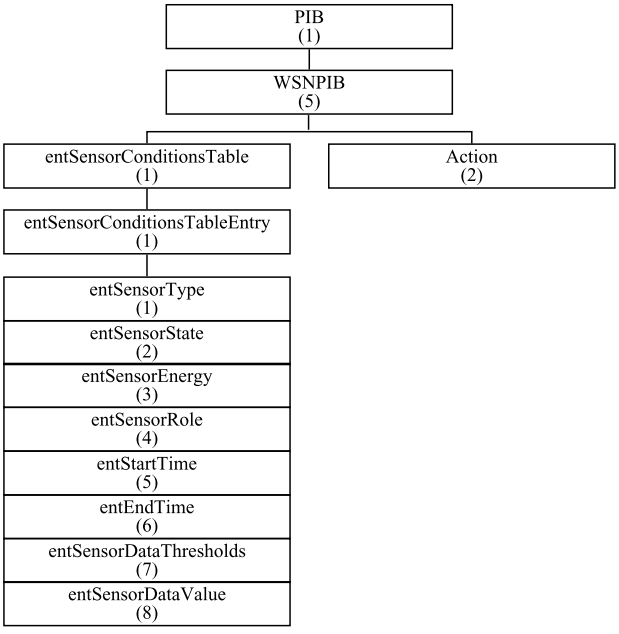

Fig. 6 The WSN PIB tree.

format. Each of the EPD values is also encoded to be sent to the PA.

When a PA receives a TinyCOPS-PR DEC message, it creates an actual policy action code (for low-level policy) based on the table values and action values. If the conditions of a sensor match the policy action code, the policy is actually applied. Table 5 shows the conceptual representation of the low-level policy in a PA.

Table 4 An example of high-level policy.

```
<?xml version=‘‘1.0’’ encoding=‘‘UTF 8’’ standalone=
‘‘yes’’?>
<SNOWMANPolicy>
 <PolicyID>1</PolicyID>
 <ResourceGroup>
  <SensorType>Humidity</SensorType>
 </ResourceGroup>
 <RoleGroup>
  <Role>
   <SensorRole>ANY</SensorRole>
  </Role>
 </RoleGroup>
 <DestinationGroup>
  <Destination>
   <RegionID>2</RegionID>
   <ClusterHeadID>0</ClusterHeadID>
  </Destination>
 </DestinationGroup>
 <TimeGroup>
  <StartTime>10:00</StartTime>
  <EndTime>13:00</EndTime>
 </TimeGroup>
 <DataGroup>
  <DataThreshold>lessOrEqual</DataThreshold>
  <DataValue>0.8</DataValue>
 </DataGroup>
 <ActionGroup>
  <Action>
   <SensorAction>Filtering</SensorAction>
  </Action>
 </ActionGroup>
</SNOWMANPolicy>
```

+Common Header (DEC)				
	+Context: configuration request			(Install)
	+Decision Object: Named Decision Object			
		+PRID	1.5.1 {entSensorConditionsEntry}	
		+EPD	entSensorType	Humidity(3)
			entSensorRole	Any(0)
			StartTime	1000
			EndTime	1300
			entSensorDataThresholds	lessOrEqual(2)
			entSensorDataValue	0.8
		+PRID	1.5.2	
		+EPD	Action	Fitering(1)

Fig. 7 An example of TinyCOPS-PR DEC message.

Table 5 An example of low-level policy action code.

```
IF (entSensorType == 3) {
   IF (isInTime()) {
      IF (SensorDataValue <= 0.8) {
         filteringData();
      }
   }
}
```

3.7 The Clustering Algorithm

The SNOWMAN constructs a hierarchical cluster-based senor network using SNOWCLUSTER clustering algorithm

Table 6 The SNOWCLUSTER algorithm.

```
// CLUSTER HEAD SELECTION
1. For All node(x), where x is # of nodes
2.   node(x).role ← cluster_head
3.   node(x).cluster_id ← node(x).node_id
4.   node(x).broadcast(discovery_msg)
5.   if node(i).hears_from(node(j))
6.   if node(i).energy_level < node(j).energy_level
7.     node(i).request_join(node(j))
8.   if node(j).role ≠ cluster_head
9.     node(j).reject_join(node(i))
10.  else
11.    node(j).confirm_join(node(i))
12.  if node(i).receive_confirm(node(j))
13.    node(i).role ← cluster_member
14.    node(i).cluster_id ← node(j).node_id

// REGION HEAD SELECTION
1. For All node(x), where x is # of nodes
2.   if node(x).role = cluster_head
3.     node(x).broadcast(cluster_info_msg)
4.   if PS.receive(cluster_info_msg)
5.     PS.assign(region_heads)
6.     PS.broadcast(region_decision_msg)
7.   if node(k).receive(region_decision_msg)
8.     if node(k).role = cluster_head
9.       if node(k).node_id
             = region_decision_msg.region_id
10.        node(k).role ← region_node
11.        node(k).region_id ← node(k).node_id
12.      else if node(k).node_id
             ∈ region_decision_msg.region_list
13.        node(k).region_id
             ← region_decision_msg.region_id
14.  node(k).broadcast(region_conf_msg)
```

[16] shown in Table 6.

The SNOWCLUSTER algorithm consists of two main procedures: cluster head selection and region head selection. The selection of cluster heads is made by the sensor nodes while region heads are selected by the PS. We assume that all sensor nodes are stationary, and they have knowledge on their locations.

A cluster is defined as a subset of nodes that are mutually reachable within two hops. Each cluster is identified by the cluster head which can reach all nodes in the cluster within one hop. In order to select a cluster head, each node periodically broadcasts a discovery message that contains its node ID, cluster ID, and remaining energy level. Each node sets the cluster ID(*cluster_id*) to be the node ID (*node_id*) of its cluster head (*cluster_head*). If Node *i* hears from Node *j* of a larger residual energy level (*energy_level*) than itself, it sends a message to Node *j* requesting to join the cluster of Node *j*. If Node *j* is not a cluster head, it returns a rejection message. Otherwise, it returns a confirmation message. When Node *i* receives a confirmation, it sets its cluster ID to Node *j*'s ID. When the cluster head selection process is completed, the entire network is divided into a number of clusters.

After cluster heads are selected, the PS selects region heads among the cluster heads. The PS receives the cluster information messages (*cluster_info_msgs*) containing cluster ID, the list of nodes in the cluster, residual energy level, and location data from all cluster heads. The PS selects region heads according to the residual energy level and location of cluster heads. If a cluster head receives a message on region (*region_decision_msgs*) from the PS, it compares its node ID with the region ID (*region_id*) in the message. If they match, it declares itself as a region head (*region_head*) and sets its region ID to its node ID. Otherwise, if its node ID is listed in another region (*region_list*), it sets its region ID to that region ID. The region head selection process is completed as region confirmation messages (*region_conf_msgs*) are broadcasted from all the cluster heads.

The role of cluster head and region head are rotated among the nodes for load balancing and thereby extending the lifetime of every sensor node. For this, SNOWCLUSTER periodically selects cluster heads having larger residual energy compared to other nodes according to the clustering policy assigned by the administrator.

4. Implementation and Evaluation

In this section the proposed framework is implemented and evaluated by ns-2 simulator for practical size networks.

4.1 The WSN Testbed

The proposed scheme is validated by implementing the components of the framework on Nano-24 [17] platform using the TinyOS programming suite. The Nano-24 uses Chipcon CC4220 RF for transmission and supports 2.4 GHz, Zigbee. The sensor nodes use atmega 128L CPU with 32 kbytes main memory and 512 kbytes flash memory. The Nano-24 also supports Qplus-N sensor network development environment. We organized a testbed network using 10 Nano-24 nodes. Each sensor node contains a PA to support policy-based management. In the testbed, the sensor nodes are configured hierarchically according to the SNOWCLUSTER clustering mechanism.

4.2 SNOWMAN

The PM and PS of SNOWMAN framework are implemented on the Windows XP platform using pure JAVA. The PA is implemented on TinyOS in the Nano-24 nodes using gcc. Figure 8 shows an example of an XML policy created from high-level policy mentioned in Sect. 3.6. We use XML to define and handle global policies. Since many useful XML-based parsers and validators exist, the efforts needed for developing a PBM system can be reduced. JAXB (Java API for XML Binding) [18] is used for SNOMAN PM, which provides a standard of mapping between XML and Java code. The XML policy schema defined by JAXB is created in the Java class form. The created Java class is converted into an XML policy using the Marshalling method.

The defined policies are stored locally in the PM and also remotely in the PS.

Fig. 8 A view of created XML policy.

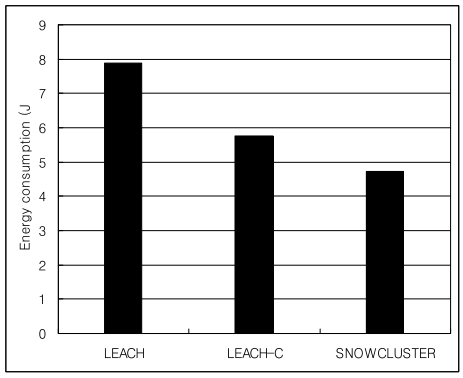

Fig. 9 The amount of energy consumed for managing.

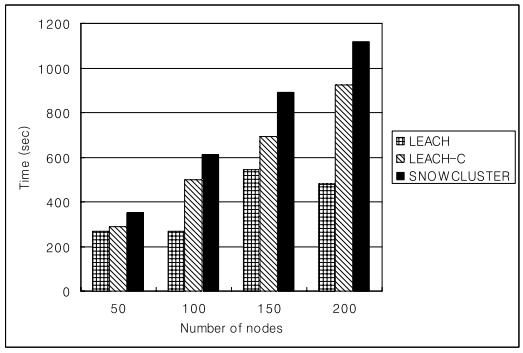

Fig. 10 The comparison of network lifetimes.

4.3 Performance Evaluation

For the performance evaluation of the proposed clustering algorithm, the ns-2 [19] network simulation tool installed with Red Hat Linux 9.0 was used. The elements adopted for establishing a virtual experimental environment are as follows:

- Sensor network topology of 50, 100, 150, 200 nodes.
- The area of 100 m x 100 m
- Transmission speed of 1 Mbps
- Wireless transmission delay of 1ps
- Radio speed of 3 x 108 m/s
- Omni-directional antenna
- Lucent WaveLAN DSSS (Direct-Sequence Spread-Spectrum) wireless network interface of 914 MHz
- Use of DSDV (Destination Sequenced Distance Vector) for routing protocol

Experiment was conducted with LEACH, LEACH-C, and the proposed SNOWCLUSTER. The energy consumed for handling the management messages were included in the experiment while the processing inside the sensor nodes were not considered since it relatively insignificant.

Figure 9 is compares the amount of energy consumed during transmission of management messages from the base station to the sensor nodes after organizing three clusters in the network of 200 nodes.

The location of the nodes is not considered in the selection of cluster heads in the LEACH. As a result, the energy consumed for constructing the routing path between the nodes is greater than the other two clustering methods. SNOWCLUSTER requires the least energy among the three due the 3-tier approach of using region heads and cluster heads.

Figure 10 shows the network lifetime when six clusters are formed in a network of 50, 100, 150, and 200 sensor nodes, respectively. Notice that the proposed SNOWCLUSTER shows network lifetime about 20% longer than that of LEACH-C due to energy saving achieved through the 3-tier structure, while improvement compared to LEACH is much more significant.

5. Conclusion

In this paper a policy-based management (PBM) framework for autonomous sensor management of the WSNs has been proposed. A hierarchical clustering approach was also developed in order to minimize the management cost of the framework. The effectiveness of the proposed scheme has been validated through actual implementation and simulation. The TinyCOPS-PR protocol which was developed for the policy distribution and the definition of WSN PIB were also introduced in this paper for policy-based management of the WSN. They were designed to be light-weight to be particularly fitted to the WSNs requiring energy efficiency. Compared to the existing clustering schemes, the simulation results reveal that the proposed framework displays smaller energy consumption for network management and longer network lifetime than the existing schemes for practical size network. More detailed specifications and the management framework for dynamic WSNs will be developed as a future

work.

References

[1] R.A.F. Mini, A.A.F. Loureiro, and B. Nath, "The distinctive design characteristic of a wireless sensor network: the energy map," Comput. Commun., vol.27, no.10, pp.935–945, Feb. 2004.
[2] F. Zhao and L. Guibas, Wireless Sensor Networks: An Information Processing Approach, Morgan Kaufman, 2004.
[3] W. Heinzelman, Application-Specific Protocol Architectures for Wireless Networks, PhD Thesis, Massachusetts Inst. of Technology, June 2000.
[4] K.S. Phanse, L.A. DaSilva, and S.F. Midkiff, "Design and demonstration of policy-based management in a multi-hop ad hoc network," Ad Hoc Networks, vol.3, no.3, pp.389–401, May 2005.
[5] L.B. Ruiz, J.M. Nogueira, and A.A.F. Loureiro, "MANNA: A management architecture for wireless sensor networks," IEEE Commun. Mag., vol.41, no.2, pp.116–125, Feb. 2003.
[6] M. Younis, P. Munshi, and E. Al-Shaer, "Architecture for efficient monitoring and management of sensor networks," IFIP/IEEE E2EMON, LNCS 2839, pp.488–502, Sept. 2003.
[7] C.-A. Lee, Y.-C. Chang, and J.-L. Chen, "Intelligent self-organization management mechanism for wireless sensor networks," Proc. 6th International Conference on Wireless Communications, pp.47–54, July 2004.
[8] R. Yavatkar, D. Pendarakis, and R. Guerin, "A framework for policy-based admission control," IETF RFC 2753, Jan. 2000.
[9] D.C. Verma, Policy-Based Networking: Architecture and Algorithms, New Riders, 2000.
[10] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, "The COPS (Common Open Policy Service) protocol," IETF RFC 2748, Jan. 2000.
[11] K. Chen, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith, "COPS usage for policy provisioning (COPS-PR)," IETF RFC 3084, March 2001.
[12] M.J. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," IEEE International Conference on Mobile and Wireless Communications Networks (MWCN 02), pp.368–372, Sept. 2002.
[13] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," IEEE Trans. Wireless Communications, vol.1, no.4, pp.660–670, Oct. 2002.
[14] S. Lindsey, C.S. Raghavendra, and K. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," IEEE Trans. Parallel Distrib. Syst., vol.13, no.9, pp.924–935, Sept. 2002.
[15] S.D. Muruganathan, D.C.F. Ma, R.I. Bhasin, and A.O. Fapojuwo, "A centralized energy-efficient routing protocol for wireless sensor networks," IEEE Radio Communications, vol.43, no.3, pp.8–13, March 2005.
[16] S.-H. Cha and M. Jo, "An energy efficient clustering algorithm for large scale wireless sensor networks," International Conference on Grid and Pervasive Computing 2007 (GPC 2007), LNCS 4459, pp.436–446, May 2007.
[17] Nano-24: Sensor Network, Octacomm Inc., http://www.octacomm.net
[18] Java Architecture for XML Binding, http://java.sun.com/webservices/jaxb/index.jsp
[19] The VINT Project, The network simulator - ns-2, http://www.isi.edu/nsnam/ns/

Si-Ho Cha received the B.S. degree in Computer Science, Sunchon National University, in 1995. He received the M.S. and the Ph.D. degree in Computer Science from Kwangwoon University, Seoul, Korea in 1997 and 2004, respectively. From 1997 to 2000, he worked for Daewoo Telecom as a Senior Researcher. He had been an Adjunct Professor in the Department of Computer Engineering, Sejong University until 2006. He is now a Research Professor (BK21) of Sejong University. His research interests include Advanced Network Management Technology, Wireless Sensor Networks, Wireless Mesh Networks, and Policy-based Networking.

Jong-Eon Lee received his B.S. and M.S. degrees in Computer Science from Kwangwoon University, in 2001 and 2003, respectively. He is currently working on Ph.D. degree in Department of Computer Science, Kwangwoon University, Seoul, Korea. His research interests include Wireless Mobile Networks, Computer Networks, and Ubiquitous Computing.

Minho Jo received B.S. degree in Industrial Engineering, Chosun Univ., South Korea and his Ph.D. in Computer Networks from Lehigh University, Bethlehem, PA, USA in 1994. He worked as a staff researcher with Samsung Electronics, South Korea and was a professor of School of Ubiquitous Computing and Systems, Sejong Cyber University, Seoul, South Korea. He is now a Research Professor, School of Information and Communication Engineering, Sungkyunkwan Univ., South Korea. He is a member of IEEE Communications Society and Computer Society, respectively. He is Chairman of Ubiquitous Research Group of IEEK and Chairman of International Ubiquitous Conference 2007. He is Technical Program Committee of IEEE International Conference on Communications 2008. He is Associate Editor of Security and Communication Networks and Referee of IEEE Transactions on Vehicular Technology. His interesting area lies in WSN, RFID, security, and ubiquitous computing.

Hee Yong Youn received the B.S. and the M.S. degrees in Electrical Engineering, Seoul National University, in 1977 and 1979, respectively. He received the Ph.D. degree from Computer Science and Engineering, University of Massachusetts at Amherst in 1988. He had been a professor of Univ. of Texas at Arlington until 1999. He is now a Professor of School of Information and Communication Engineering, Sungkyunkwan University, South Korea, and Director of Ubiquitous computing Technology Research Institute. His research topics include ubiquitous computing, WSN, middleware, and has published more than 200 papers. Among them, he received best paper award from 1988 IEEE Int'l Symp. on Distributed Computing and 1992 Supercomputing, respectively. He is a senior member of IEEE.

Seokjoong Kang received his B.S. and M.S. degrees from the Computer Science Department at Indiana University in 1988 and 1991 respectively and a Ph.D. degree from the Electrical Engineering and Computer Science Department at University of California, Irvine (UCI) in 2003. He worked as a lecturer and research staff at UCI and worked as a senior researcher at Korea Institute of Defense Analyses in Korea. He also worked as a principle researcher at Samsung Electronics. He is now an Assistant Professor of Department of Computer Science and Engineering, Kwangwoon University, Seoul, Korea. His research interests include Embedded System, Software engineering, Real-time system and Distributed systems.

Kuk-Hyun Cho received his B.S. degree in Electronic Engineering from Hanyang University, Seoul, Korea in 1977 and his M.S. and Ph.D. degrees from Tohoku University, Sendai, Japan in 1981 and 1984, respectively. Since 1984, he has been a Professor in the Department of Computer Science and Engineering, Kwangwoon University, Seoul, Korea. From 1998 to 2000, he was a President of Open Standards and Internet Association (OSIA). His research interests include Network and Service Management, Policy-based Networking, and Internet QoS Management.