Ashdin Publishing
International Journal of Swarm Intelligence and Evolutionary Computation
Vol. 1 (2012), Article ID 235560, 12 pages
doi:10.4303/ijsiec/235560

*Research Article*

# Response Threshold Model-Based Device Assignment for Cooperative Resource Sharing in a WSAN

**Takuya Iwai, Naoki Wakamiya, and Masayuki Murata**

*Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565-0871, Japan*
*Address correspondence to Takuya Iwai, t-iwai@ist.osaka-u.ac.jp*

Received 10 March 2012; Revised 15 April 2012; Accepted 21 April 2012

**Abstract** Many researchers have been attracted by a wireless sensor and actuator network (WSAN) for its wide range of applications. In a WSAN, embedded sensors detect and conjecture environmental and personal conditions and actuators provide users with information services and environmental control which are suited for time, place, occasion, and people. Since it apparently is wasteful and redundant to deploy an independent WSAN for each of envisioned applications, building a multi-purpose WSAN consisting of heterogeneous sensors and actuators and sharing them among applications are considered promising. However, we need a mechanism to effectively share available resources among concurrent applications while taking into account application requirements and resources. Although there are several proposals on centralized or deterministic device assignment mechanisms, they suffer from difficulty in designing an appropriate set of rules with fine-tuned parameters. In this paper, we propose a fully distributed and self-organizing device assignment mechanism by adopting a response threshold model, which imitates division of labors in a colony of social insects. Our proposal does not require deterministic and complicated rules and appropriate device assignment emerges as a consequence of autonomous decision of individual nodes. Through simulation experiments, we confirmed the our proposal accomplishes as effective device assignment as an existing deterministic mechanism and our proposal is less sensitive to parameter setting errors.

**Keywords** wireless sensor and actuator network; device assignment; resource sharing; response threshold model

## 1 Introduction

In recent years, many researchers have been actively working in the field of wireless sensor and actuator networks (WSANs) [1]. A WSAN consists of embedded sensors, e.g. thermometer, hygrometer, and motion sensor, that detect and obtain environmental and personal conditions and actuators, e.g. heater, cooler, buzzer, light, and switch, that control environment and machinery. By distributing nodes with appropriate sensors and/or actuators at appropriate locations in an area, e.g. field, building, and room, and organizing a network by wireless multi-hop communication, a variety of applications can be provided in the area. We hereafter call sensors and actuators "*devices*" and a "*node*" corresponds to an equipment with CPU, memory, wireless transceiver, and one or more devices.

In general, WSANs are constructed and managed in an application-oriented manner to answer specific requirements of an individual application. Therefore, nodes are deployed for a specific application and they are not shared with others. For example, both of WSANs for illumination control and intrusion detection employ nodes with a motion sensor to detect location of people and nodes with a switch to turn on or off a light. Although applications use the same kind of devices in the same way, their WSANs are made of dedicated nodes and independent from each other with current form of deployment. It is apparently redundant and wasteful. Furthermore, an application-oriented deployment requires previous knowledge about the operational environment and careful planning of types and locations of nodes to place. However, it is impossible to predict all events that may occur in the area and make WSANs well prepared for unpredictable events.

Considering above-mentioned issues, interests of researchers are shifting from a *special purpose WSAN* to a *multi-purpose WSAN* where multiple concurrent applications are running over a single WSAN [18]. In a multi-purpose WSAN, heterogeneous nodes are deployed in the area and applications employ those nodes with desired devices. The first challenge exists in the heterogeneity in node architecture [2, 14], which makes application implementation and interoperation of nodes difficult. As an example of solution of the challenge, SOA (Service Oriented Architecture) provides an application with a common interface with nodes having different architecture [2, 16]. Once heterogeneous nodes can be handled through the common interface, another challenge arises in selection

of nodes and devices [15,20]. For example, in starting an intrusion detection application in the area where an illumination control application already exists, is it better to use the node with a motion sensor that the illumination control application is using? If they share the node, other nodes with a motion sensor can sleep and save energy and network bandwidth. A decision on device assignment must be made taking into account a variety of conditions, e.g. the degree of device sharing and the amount of residual energy, and it is not trivial. For this purpose, there are several proposals on dynamic device assignment [8,9], but they usually employ rule-based mechanisms. As such, as a WSAN becomes large and the number and heterogeneity of applications increase, they will suffer from difficulty in making an appropriate set of rules without contradictions.

In [12], to effectively share available resources among concurrent applications while taking into account application requirements and resources, we presented a basic idea of fully-distributed and self-organizing device assignment mechanism where each node determines whether to offer its own devices to an application or not. Results of preliminary simulation experiments showed that nodes and devices were appropriately selected taking into account the amount of residual energy and the degree of contribution to applications. In this paper, we improved our mechanism by incorporating with SPAN [6] to efficiently share nodes engaged in message relaying among applications and simplifying a decision making algorithm to make parameter setting easier. In our proposal, the minimum connectivity is maintained by SPAN, where a set of coordinator nodes constructs a forwarding backbone. Once a need for device assignment occurs, a request message is disseminated from a request node of an application to all nodes through a forwarding backbone. On receiving the request, each node determines whether to offer its devices to the application or not. The decision is sent back to the request node through the forwarding backbone.

For autonomous decision making without deterministic if-then type of rules, we adopt a response threshold model [4], which imitates a mechanism of division of labors in a colony of social insects. In a colony, each individual decides to be engaged in a task without any centralized control and the number of workers is dynamically adapted in accordance with the demand of the task. In our proposal, a request message advertised by a request node expresses the demand intensity to stimulate nodes to offer their devices. A request node does not appoint nodes to offer their devices as existing mechanisms do. Instead, each node has has the right to make a decision of device assignment in our proposal. Furthermore, device assignment is performed stochastically at a node. Therefore, our proposal is not deterministic. As such, as will be verified in the paper, our proposal is less sensitive to parameter setting than the existing mechanism. It implies that our proposal can be used in the area where a variety of applications emerge and their requirements dynamically change.

The remainder of this paper is organized as follows. First, in Section 2, we describe related work. Next, in Section 3, we describe application scenario that our proposal assumes. Then, in Section 4, we propose a mechanism for a node to autonomously decide whether it assigns its devices to an application or not. In Section 5, we show results of simulation to evaluate our proposal and compare it with an existing mechanism. Finally, in Section 6, we provide concluding remarks and future work.

## 2 Related work

The heterogeneity of node architecture makes application implementation and node management difficult. There are several proposals to deal with heterogeneous nodes through the common interface [2,14]. For example, TinySOA, which is based on the concept of SOA [16], allows application developers to write application programs without concerning differences in node architecture, OS, and programming languages.

To share sensors among multiple applications, TinyONet is proposed in [13]. The main focus of TinyONet is reuse of sensing data gathered at a sink. When a sink receive a request for assignment of sensors from an application, it organizes a slice, i.e. a group of virtual sensors. A virtual sensor is a representative of a physical node and it provides an application with cached sensing data. From a viewpoint of an application, a dedicated sensor network is tailored over heterogeneous sensors, from which it can collect required sensing data at the desired frequency. Since a virtual sensor is a cache, it is possible to accommodate multiple applications without putting extra load on a physical sensor network. However, TinyONet assumes that sensing data are collected from all sensor nodes at regular intervals, which should be the minimum among all applications. That is, it is energy and bandwidth consuming. Furthermore, it does not consider actuators.

VSN (Virtual Sensor Network) is another example of a mechanism to overlay application-oriented virtual networks over physical WSANs [3]. In their proposal, a VSN can consist of nodes belonging to different WSANs. A VSN is a single network of tree topology and all messages are exchanged over the tree. Although VSN realizes service-oriented and inter-WSAN overlay networking, it is inefficient to concentrate all traffic at a single tree. Furthermore, when there are multiple concurrent applications, there exist multiple and independent VSN trees in the area.

Regarding on-demand selection of nodes which offer devices or functions to an application, mechanisms for generic role assignment are proposed in [8,9]. In their framework, an application developer injects a role

specification to a WSAN through a gateway. A role specification defines roles and rules to assign roles to nodes. Rules are in the form of Boolean expression, i.e. if-the-else statement. A specification is disseminated in a WSAN and a node receiving it decides whether to play a role or not in accordance with the specified rules and its properties. As far as rules are well defined, roles are assigned to appropriate nodes. However, it is not trivial to define an appropriate set of consistent rules to appoint the necessary and sufficient number of nodes with desired properties taking into account a variety of conditions.

Our proposal can assign an appropriate set of nodes and their devices to an application taking into account application's requirements and multiple conditions, i.e. residual energy and resource sharing, without centralized control or deterministic rules.

## 3 Application scenario

There are applications operating in the area or being introduced on demand. Each application has one or more application servers or control units, such as a home server of a home automation system, which manages the application. However, a server does not have the complete knowledge of the whole WSAN, e.g. type and location of nodes and their devices. We assume that an application consists of a series of *processes*, such as turn on or off the light, and a process is realized by devices embedded in the area. For example, a lighting control application (1) senses user presence and illuminance and (2) turns a room light on or off depending on the situation.

Sharing a sensor among multiple applications is not harmful as far as the sensor can provide them with requested sensor data at the desired precision and frequency. On the other hand, sharing an actuator among multiple applications sometimes causes a problem, which we call "actuator contention." An actuator usually has multiple operations that cannot be performed simultaneously, e.g. turn on and off a light. To solve competition for a device among applications, we assume that each process of an application has a priority value. A priority value is predetermined at implementation and deployment, but it can dynamically change in response to environmental conditions. A process with a higher priority takes precedence over a process with a lower priority. When there is a tie, a decision making algorithm of a WSAN selects a process to assign a device. An assigned device operates as requested by the designated application.

## 4 Our proposal

The proposal adopts a response threshold model of division of labors in a colony of social insects [4] to accomplish autonomous and fully distributed decision making of nodes on whether to assign embedded devices to an application.

### 4.1 Service network

An application is realized by devices which are selected by our decision making algorithm. We call a network consisting of nodes contributing to an application a "service network," which is logically laid on a physical WSAN. Nodes constituting a service network are a "*request node*" that initiates organization of the service network, "*member nodes*" that are equipped with devices which can satisfy application requirements, and "*relay nodes*" that deliver messages among a request node and member nodes. In addition, there are two types of member nodes, i.e. "*active member nodes*" and "*idle member nodes*." A role of a node is determined per application and changes in the course of operation. For example, a node is an active member node of application A, a relay node of application B, and a non-member node of application C.

An active member node assigns devices to one or more applications. We call a device which provides a sensing or actuation function to an application an "*active device*." On the contrary, an idle member node is equipped with devices which can answer application requirements, but it does not assign them to any application. We call an unassigned device an "*idle device*." An active member node has one or more active devices and some idle devices, but an idle member node has only idle devices. A decision on whether to become an active or not is made by a node taking into account several conditions such as application requirements, the degree that devices are shared among applications, and the residual energy, to efficiently share active member nodes among applications and balance energy consumption of member nodes.

When there is no operating application, no node is active in a WSAN. Device assignment is initiated by a request node, e.g. an application server or a getaway node between a WSAN and an outside application server. We should note here that our proposal can be applied to both of a static and dynamic application. In the case of a static application, a request node is a sink of data of periodic monitoring, for example. In the case of a dynamic application, a node detecting an event becomes a request node, for example.

### 4.2 Basic behavior

A request node first disseminates a request message which specifies necessary devices and their desired operational mode to all nodes (step 1 in Figure 1). The minimum connectivity of a WSAN is maintained by SPAN [6]. SPAN forms the forwarding backbone, which consists of coordinator nodes. A coordinator node is a node which stays awake to maintain connectivity of neighbor nodes. Nodes which are not a coordinator can sleep and communicate with each other through the forwarding backbone when needed. Decision to become a coordinator is made locally by a node. A request message is sent to all nodes in the

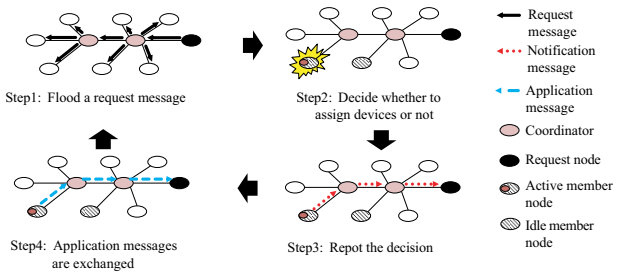



**Figure 1:** Overview of device assignment.

whole area or nodes in the specified area of interest when location information is available, through the forwarding backbone.

When a node receives a request message, it first examines whether its devices can answer the request. If the node is equipped with such devices, it becomes a member node. Next, a member node decides whether to assign devices to a requesting application or not by using the response threshold model-based decision making algorithm (step 2). Then active member nodes report the decision to the request node by sending a notification message. Nodes where notification messages traverse become a relay node and they adjust the sleep scheduling if necessary (step 3). A member node of a certain application can be a relay node of the same application. A coordinator node is likely to become a relay node. Finally, application messages including sensing and control information are exchanged among active member nodes and a request node through relay nodes until the next timing of periodic dissemination of a request message (step 4).

Above-mentioned steps are repeated while an application is running. A request node can change contents of a request message to perform a different process of the application. It is also possible for a member node to issue a new request message for the application. At the end of an application, a request node stops sending request messages. Those internal values that a node holds for the application is removed when a timer expires without receiving a request message for a predetermined duration.

### 4.3 Internal values of nodes

In our proposal, a node maintains a set of information summarized in Table 1. Details of each information is given in the followings.

A node is equipped with a set $\mathbf{D}$ of devices. A node also has a set $\mathbf{O_j}$ of operational modes of device $j \in \mathbf{D}$. A set $\mathbf{O_j}$ is represented by the following expression, where $n = |\mathbf{O_j}|$, i.e. the number of operational modes.

$$\mathbf{O_j} = \{\text{mode}_1, \ldots, \text{mode}_{n-1}, \text{mode}_n\}.$$

A device cannot operate in different operational modes simultaneously. "$\text{mode}_n$" is a "*default mode*" of a device and an idle device is in $\text{mode}_n$. When device $j$ is a sensor, a typical set is $\mathbf{O_j} = \{\text{sensing}, \text{sleep}\}$. In the case of an ON/OFF switch, $\mathbf{O_j} = \{\text{ON}, \text{OFF}\}$. In general, a default mode is an operational mode where a device and a facility can save energy.

**Table 1:** Internal values of a node.

| Notation | Default | Description |
|---|---|---|
| $\mathbf{D}$ | $\phi$ | set of devices |
| $\mathbf{O_j}$ | | set of possible operational modes of device $j$ |
| $\mathbf{S}$ | $\phi$ | set of requirements of applications |
| $\mathbf{X}$ | $\phi$ | set of $X_{i,j}$ |
| $\mathbf{Y}$ | $\phi$ | set of $Y_j$ |
| $\mathbf{\Theta}$ | $\phi$ | set of $\theta_{i,j}$ |
| $X_{i,j}$ | *false* | boolean flag of assignment of device $j$ to application $i$ |
| $Y_j$ | default mode | operational mode of device $j$ |
| $\theta_{i,j}$ | 5 | threshold of assignment of device $j$ to application $i$ |

A node also maintains a set $\mathbf{X}$ of $X_{i,j}$, a set $\mathbf{Y}$ of $Y_j$, and a set $\mathbf{\Theta}$ of $\theta_{i,j}$ for application $i$ and device $j$ ($i \in \mathbf{I}$ and $j \in \mathbf{D}$), which are used by the response threshold model-based decision making algorithm. $\mathbf{I}$ is a set of identifiers of application for which a node received a request message. $X_{i,j} \in \{true, false\}$ represents whether device $j$ is assigned to application $i$ ($X_{i,j} = true$) or not ($X_{i,j} = false$). $Y_j \in \mathbf{O_j}$ represents the current operational mode of device $j$. $\theta_{i,j}$ ($0 < \theta_{i,j} \leq \theta_{\max}$) is a threshold representing hesitation of node in assigning device $j$ to application $i$.

A node maintains a set $\mathbf{S}$ of 7-tuples $(i, j, m, k, h, s_i(t), r_i)$. These values are updated on receiving the $t$th request message of application $i \in \mathbf{I}$. The identifier $i$ which is unique in the whole network can be generated as concatenation of a node identifier and a sequence number of application it initiates. $j$ is an identifier of a device which application $i$ requires. When application $i$ requires multiple devices, the tuple is generated for each of devices. $m$ is an operational mode which application $i$ request to device $j$. $k$ is a sequence number of the last request message. $h$ is an identifier of a neighbor node where it received the request message. $s_i(t)$ is the demand intensity representing the degree that the request node wants its request to be satisfied. $s_i(t)$ is calculated by the request node in accordance with the number of devices assigned to application $i$. Finally, $r_i$ is the priority of application $i$ or its process.

### 4.4 Node behavior

A request node sends a request message at regular intervals of $I_{demand}$ s. We call an interval between successive emissions of a request message a "*round*." As a simple example, assume a process for periodic data gathering which requires a motion sensor to report the condition at coordinates $(x, y)$ every $I_{data}$ s. In this case, a request

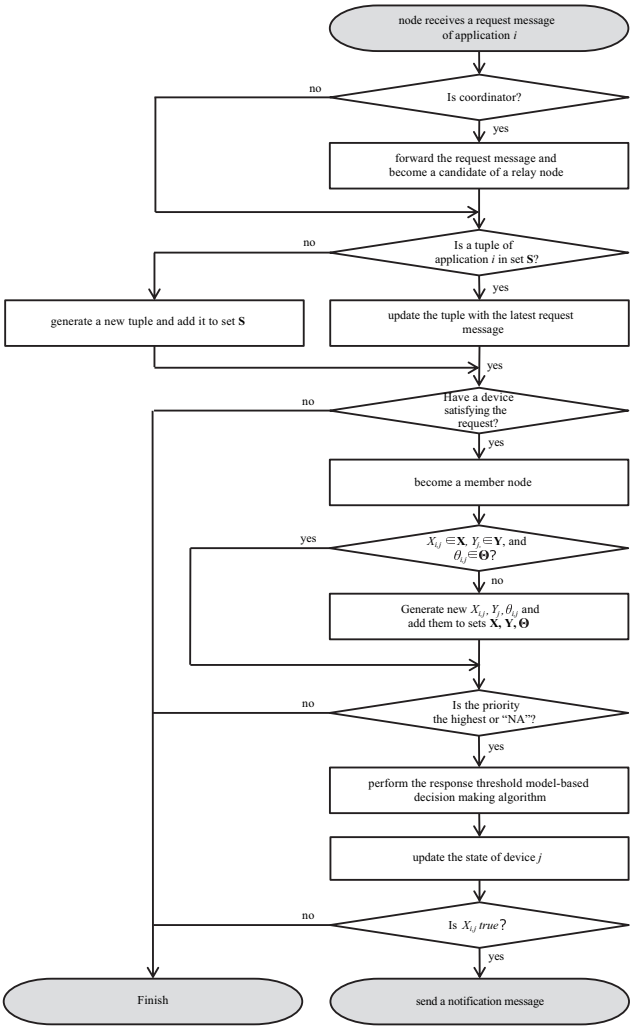



**Figure 2:** Behavior of a node on receiving a request message.

message emitted at the $t$th round is a pair of attributes in the form $(i, k, s_i(t), r_i)$ and a request body in the form $(\text{motion sensor}, \text{sensing}, (x, y), I_{data})$. The content of a request message can be extended by using an XML-based method [10].

When a node other than a request node of an application receives a request message, it behaves following a flow chart shown in Figure 2. First, if a node is a coordinator of SPAN, it forwards the request message to neighbor nodes and it becomes a candidate of a relay node. Next, if it does not have an element of application $i$ in set **S**, it generates a new 7-tuple element. If the corresponding tuple exists, it is updated. Then, a node examines whether it has a device which satisfy the request. If it has, the node becomes a member node. A member node initializes elements $X_{i,j}$, $Y_j$, and $\theta_{i,j}$ of application $i$ in sets **X**, **Y**, and **Θ**, respectively, if not exist. If the priority of the application is the highest in all applications in set **S** or the requested devices are not assigned, values $X_{i,j}$, $Y_j$, and $\theta_{i,j}$ are updated by a decision making algorithm explained in Section 4.5. Then, if the node assigns device $j$ to application $i$, the decision is reported to the request node by sending a notification message which contains an identifier of the active member node, an identifier of the application and $X_{i,j}$. A notification message is sent to neighbor $h$, from which a node received the corresponding request message. Following a reverse path, a notification message reaches the request node.

An assigned device operates in the decided operational mode. In the above example, an active member node sends sensing data of a point $(x, y)$ obtained by a motion sensor to the request node at regular intervals of $I_{data}$ s. Data messages are sent to the request node through the forwarding backbone of SPAN. Every time a member node receives a request message, the above steps are conducted. If a member node does not receive a request message for $E_i$ s, it considers the corresponding application terminates and it removes corresponding information from the memory.

A request node receives notification messages from active member nodes. In the proposal, a request node uses a scalar value, called the demand intensity, to control the number of active member nodes while leaving decision making to nodes. The demand intensity at the beginning of the $(t+1)$th round is calculated from the number of notification messages by the following equation, where the initial demand intensity $s_i(0)$ is set at 0.

$$s_i(t+1) = s_i(t) + \delta_i - N_i(t). \quad (1)$$

Here, $\delta_i$ ($\delta_i \geq 0$) is an increasing rate of demand intensity of application $i$. $N_i(t)$ ($N_i(t) \geq 0$) is the number of active member nodes which is equal to the number of notification messages stating $X_{i,j} = \textit{true}$ received in response to the $t$th request message. The equation means that, when the number of active member nodes is less than $\delta_i$, the demand intensity gradually increases and the request node requires more member nodes to become an active member node. On the other hand, when the number is greater than $\delta_i$, the demand intensity gradually decreases and active member nodes become inactive. As such, the parameter $\delta_i$ determines the number of active member nodes on convergence and it can be used to adjust the degree that nodes are involved in the process. The updated demand intensity is notified to member nodes by a request message disseminated at the beginning of the next round. Until the next round, a request node exchanges messages with active member nodes.

### 4.5 Response threshold model-based decision making

It is known that a colony of social insects is divided into two groups of workers and non-workers based on autonomous decision of individuals using a simple rule. A response threshold model is a mathematical model of the division of labors of social insects [4]. We adopt the model as

an algorithm for a member node to decide whether it assigns a device to an application or not. For details of the response threshold model, refer to [4]. The size of group is well adjusted based on the task-associated intensity of stimuli [5].

The probability $P(X_{i,j} = \textit{false} \rightarrow X_{i,j} = \textit{true})$ that an idle node ($X_{i,j} = \textit{false}$) assigns device $j$ to application $i$ is derived by the following equation:

$$P(X_{i,j} = \textit{false} \rightarrow X_{i,j} = \textit{true}) = \frac{s_i^2(t)}{s_i^2(t) + A_j \theta_{i,j}^2(t)}. \quad (2)$$

Here, $s_i(t)$ ($s_i(t) \geq 0$) is the demand intensity of application $i$ at the $t$th round. $\theta_{i,j}$ ($\theta_{\max} \geq \theta_{i,j} > 0$) is a threshold which corresponds to hesitation of the node in assigning device $j$ to application $i$. The equation is extended from the basic model by introducing variable $A_j$. $A_j$ ($A_j \geq 1$) is a variable related to the degree that device $j$ is shared among applications, and the residual energy of the node. Derivation of $A_j$ will be explained in Section 4.6.

The probability $P(X_{i,j} = \textit{true} \rightarrow X_{i,j} = \textit{false})$ that an active member node ($X_{i,j} = \textit{true}$) quits assigning device $j$ to application $i$ is derived by the following equation:

$$P(X_{i,j} = \textit{true} \rightarrow X_{i,j} = \textit{false}) = p_j. \quad (3)$$

Here, $p_j$ ($1 \geq p_j > 0$) is a constant defined per device. This prevents an active member node from devoting its devices too long and enables rotation of task among member nodes. It further leads to avoidance of redundant device assignment. The average duration that an active member node assigns device $j$ to an application is $1/p_j$ rounds.

Similarly to the basic response threshold model, our proposal also has a mechanism of reinforcement which makes specialists. Threshold $\theta_{i,j}$ is adjusted as follows:

$$\theta_{i,j} = \begin{cases} \theta_{i,j} - \xi_j, & \text{if } X_{i,j} \text{ is } \textit{true}, \\ \theta_{i,j} + \varphi_j, & \text{if } X_{i,j} \text{is } \textit{false}, \end{cases} \quad (4)$$

where $\xi_j$ ($\xi_j > 0$) and $\varphi_j$ ($\varphi_j > 0$) are parameters of the speed of differentiation. With the threshold adjustment, an active member node is more likely to become active again than an inactive member node.

### 4.6 Variable $A_j$ for device sharing and energy efficiency

In the proposal, for efficient device sharing and balancing energy consumption for a longer lifetime variable $A_j$ ($1 \leq A_j$) is derived by the following equation from the degree that device $j$ is shared among applications and the residual energy.

$$A_j = \left(S_j - F_j\right)^m + \left(\frac{P_{\textit{full}}}{P_{\textit{res}}}\right)^n - 1. \quad (5)$$

Parameters are summarized in Table 2. Here, the first term of the right side is used for device sharing among applications.

**Table 2:** Parameters of variable $A_j$.

| Notations | Description |
|---|---|
| $m$ | exponent regulating the sensitivity to the degree of sharing |
| $n$ | exponent regulating the sensitivity to the residual energy |
| $S_j$ | number of applications where a node is a member for device $j$ |
| $F_j$ | number of applications where a node is an active member node for device $j$ |
| $P_{\textit{res}}$ | amount of residual energy of a node |
| $P_{\textit{full}}$ | total capacity of battery of a node |

Variable $S_j$ ($S_j \geq 1$) represents the number of applications where a node is a member regarding device $j$. $S_j$ is derived as $S_j = |\{X_{i,j} \in \mathbf{X} \mid i \in \mathbf{L}\}|$ where $\mathbf{L}$ is a set of identifiers of application where a node is a member node. Variable $F_j$ ($F_j \geq 0$) represents the number of applications where a node is an active member node. $F_j$ is derived as $F_j = |\{X_{i,j} \in \mathbf{X} \mid i \in \mathbf{I}, X_{i,j} = \textit{true}\}|$ and $F_j \leq S_j - 1$. Exponent $m$ ($m \geq 1$) influences the sensitivity of the algorithm to the degree that the device is sharing. The second term is used for balancing energy consumption. $P_{\textit{full}}/P_{\textit{res}}$ is the ratio of the battery capacity $P_{\textit{full}}$ ($P_{\textit{full}} > 0$) to the residual energy $P_{\textit{res}}$ ($P_{\textit{full}} \geq P_{\textit{res}} > 0$). Exponent $n$ ($n \geq 1$) influences the influence of the amount of residual energy on decision making. The third term is used for shifting minimum value of valuable $A_j$ from 2 to 1. Variable $A_j$ becomes smaller and probability $P(X_{i,j} = \textit{false} \rightarrow X_{i,j} = \textit{true})$ becomes higher on a node which is engaged in more applications as an active member node and has more residual energy.

## 5 Performance evaluation

In this section, we evaluate our proposal through comparison with directed diffusion [11] and our former proposal [12]. We first briefly explain directed diffusion and its extension made for comparison purposes. Then, we will show results of evaluation from viewpoints of efficiency of device assignment and robustness against parameter setting errors.

### 5.1 Directed diffusion

Directed diffusion is a data-centric information gathering mechanism [11]. A *sink* which corresponds to a request node in our proposal first disseminates an *interest* message. An interest message specifies a required sensing task and a reporting interval. Initially, a reporting interval is set longer than one that an application requires.

When a node receives an interest message, it sets an entry called *gradient*, which consists of the information about a task, an identifier of a link with a neighbor node from which it received the interest message as a precursor, and a report interval specified in the message. If a node can perform the requested sensing task, it becomes a source node, which we call a member node in our proposal, and

begins to send data messages at the specified report interval. Data messages reach the sink by following gradients. The first data message is called an *exploratory data* message.

A sink would receive multiple exploratory messages from different source nodes. Among them, it selects one based on a reinforcement rule, for example, to select an exploratory data message received first. Then, the sink sends an interest message, called a *reinforcement* message to a neighbor node from which the selected exploratory data message comes. A reinforcement message is in the same format as an interest message, but it specifies an application-specific reporting interval which is shorter than a reporting interval written in an interest message. A reinforcement message is sent to a source node following gradients in the reverse direction while updating gradients on the route with the new reporting interval. The gradient does not hold information about a source node. Therefore, when there are two or more source nodes in the downstream of the selected neighbor node, a reinforcement message does not necessarily reach a source node which sent the corresponding exploratory data message. A sink keeps sending both of interest messages and reinforcement messages at regular intervals to maintain and update gradients.

### 5.2 Extension of directed diffusion

In our proposal, a request node can control the number of devices which contribute to an application by the demand intensity as will be verified in Section 5.4, while device assignment relies on an autonomous decision of each node. On the other hand, the number of source nodes is uncontrollable and it would dynamically change in directed diffusion. Since gradient on a node does not have information about either of a sink or a source node, interest messages and reinforcement messages do not always reach the same set of source nodes. Therefore, for comparison, we extended directed diffusion for controlling the number of nodes or devices which contributes to an application as follows.

First, for a sink to obtain information of a source node, we extend a data message to have additional fields for an identifier $id_{sink}$ of a sink, $id_{src}$ of a source node, the amount $P_{res}$ ($P_{res} > 0$) of residual energy, the battery capacity $P_{full}$ ($P_{full} > P_{res}$), the number $M_{dd}$ ($M_{dd} \geq 1$) of sinks from which it receives interest or reinforcement messages, and the number $N_{dd}$ ($N_{dd} \geq 0$) of sinks from which it receives reinforcement messages. Consequently, the extended data message takes the form of [type, data, $id_{sink}$, $id_{src}$, $P_{res}$, $P_{full}$, $M_{dd}$, $N_{dd}$, time-stamp]. Please refer to [11] for details of other fields than those newly introduced. Next, to identify a path between a specific sink and a specific source node, we add a field of an identifier $id_{sink}$ of a sink and $id_{src}$ of a source node to the gradient. First time when a node receives an interest message, it makes a gradient while leaving $id_{src}$ empty. It fills in the field when a data message is received. Consequently, the extended gradient has the form of [type, region, data rate, time stamp, expired-AT, $id_{sink}$, $id_{src}$]. While leaving the form of an interest message as it is, i.e. [type, region, interval, time-stamp, expired-AT], we extended the form of a reinforcement message to have a new field for an identifier $id_{sink}$ of a sink node and $id_{src}$ of a source node. As a result, the reinforcement message has the form of [type, region, interval, $id_{sink}$, $id_{src}$, time-stamp, expired-AT].

**Table 3:** Prioritization rule for reinforcement in directed diffusion.

| | | $R_{energy}$ | |
|---|---|---|---|
| | | $\geq T_{energy}$ | $< T_{energy}$ |
| $R_{share}$ | $\leq T_{share}$ | 1 | 3 |
| | $> T_{share}$ | 2 | 4 |

Now based on the extension, we consider a reinforcement rule which takes into account the amount of residual energy and the degree of device sharing. A sink in the extended directed diffusion first disseminates an interest message to all nodes. Next source nodes begin to send data messages. A data message sent by a source node contains information about its energy, $P_{res}$ and $P_{full}$ and its task $M_{dd}$ and $N_{dd}$. After receiving the sufficient number of data messages, a sink evaluates $R_{energy}$ which is derived as $P_{res}/P_{full}$ and $R_{share}$ which is derived as $(M_{dd} - N_{dd})/M_{dd}$ for each source node. Then, it determines priority of the node in reinforcement. For the sake of simplicity, we use threshold-based prioritization summarized in Table 3, where $T_{energy}$ and $T_{share}$ are thresholds. A smaller number has higher priority. For example, if a source node has plenty of energy, i.e. large $R_{energy}$, and is contributing to many sinks, i.e. small $R_{share}$, it is the best source node to reinforce. Finally, following an ascending order of priority value, a sink selects the required number $N$ of source nodes and send reinforcement messages to them. In the following, we call a source node which receives a reinforcement message an active source node.

### 5.3 Simulation setting

We used OMNet++ [19] for simulation. 25 nodes are placed in the area of 25 m × 25 m. 5 nodes, A, B, C, D, and E, among them are located at the edge of the area, while remaining 20 nodes are randomly distributed. Figure 3 illustrates an example of node layout where the x and y axes are coordinates and filled circles, open circles, crosses, and triangles indicate nodes. Each line corresponds to a path between an active member node and a request node to exchange application messages.

Nodes are identical in battery capacity, embedded device, and communication capability. They operate on two AA batteries of 3.3 V. Based on the data sheet of

**Table 4:** Parameter setting of performance evaluation.

| Notation | Description | set A | set B |
|---|---|---|---|
| $p_j$ | probability of quitting task in (3) | 0.01 | 0.01 |
| $\xi_j$ | threshold adaptation parameter in (4) | 0.1 | 0.1 |
| $\varphi_j$ | threshold adaptation parameter in (4) | 1 | 1 |
| $m$ | influence of degree of sharing in (5) | 3 | 6 |
| $n$ | influence of residual energy in (5) | 3 | 6 |
| $L$ | interval for exploratory data messages | 0.5 | 0.5 |
| $T_{share}$ | threshold of reinforcement rule | 0.5 | 0.1 |
| $T_{energy}$ | threshold of reinforcement rule | 0.5 | 0.9 |
| $I_{demand}$ | interval of request message | 10 | 10 |

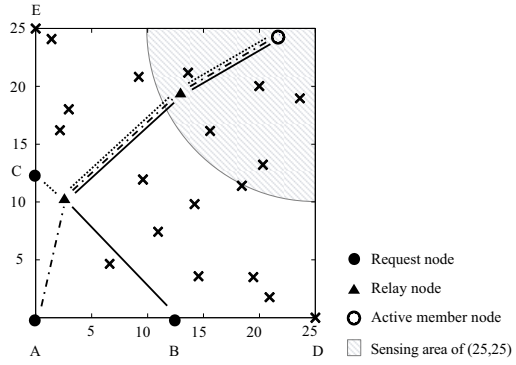



**Figure 3:** Snapshot of a simulation.

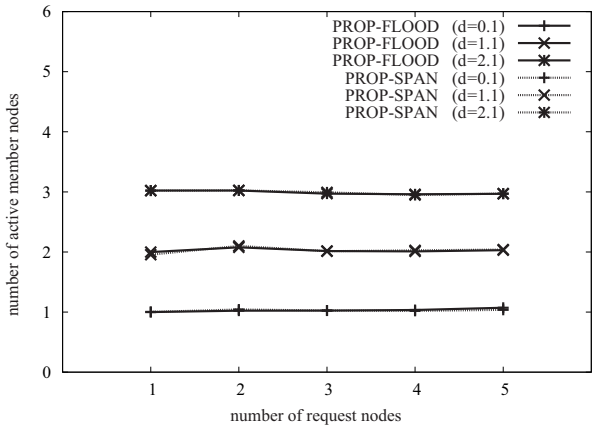
MICAz [7], a transceiver module consumes 18.8 mA in listening a channel and receiving a message, 17.4 mA in transmitting a message, and 0.021 $\mu$A in a sleep mode. A node is equipped with a sensing device with identifier $j$. A sensing device can obtain information about a certain point in the diameter of 15 m. We assume that energy consumption of the device in sensing is negligible in evaluation.

The communication range is 15 m on the IEEE 802.15.4 non-beacon mode MAC/PHY protocol. The length of a request message, an interest message, and a reinforce message is set at 36 byte without 6 byte header. Regarding a notification message, an exploratory message, and a data message, the length is set at 64 byte without a 6 byte header. Parameters used in the simulation experiments are summarized as set A in Table 4, which are chosen based on preliminary experiments.

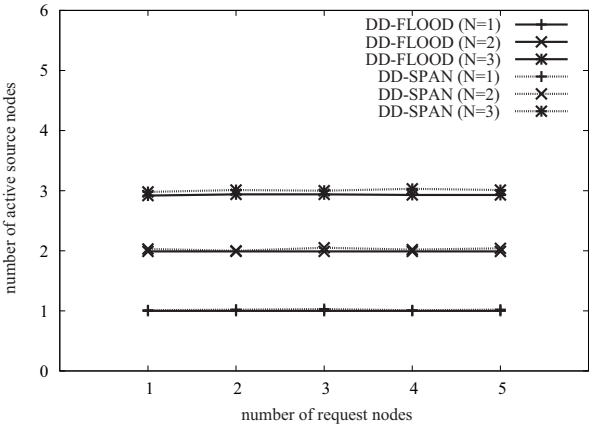
### 5.4 Evaluation of task assignment

Since self-organization does not always achieve the optimal result due to its autonomous behavior, in this section, we first verify that our proposal can accomplish as good device assignment as directed diffusion which employs deterministic rules. Evaluation is conducted from a viewpoint of the number of active member nodes and relay nodes.

We configure 5 edge nodes as request nodes of 5 independent applications of the same priority, respectively. All request nodes require the information about the corner point




(a) Our proposal.

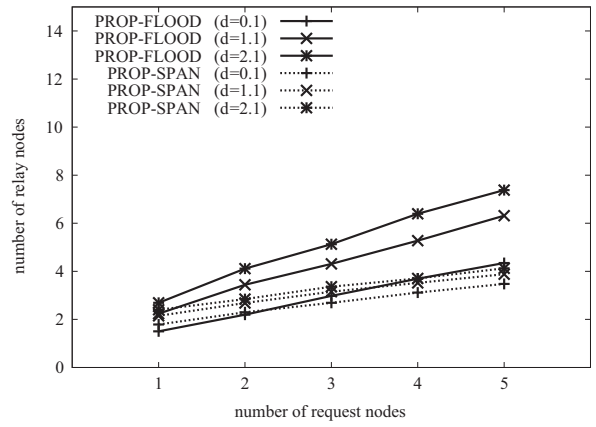



(b) Directed diffusion.

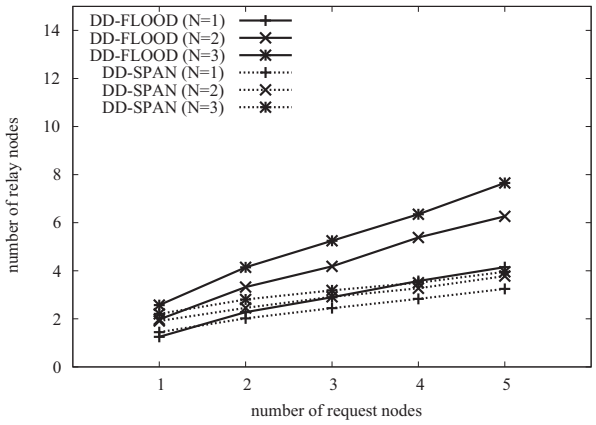
**Figure 4:** Number of active member or source nodes.

at $(25, 25)$. In the other words, they require assignment of a sensor device within a circular area centered at $(25, 25)$ with radius 15 m, which is illustrated as a shaded quadrant in Figure 3. Each request node sends a request message at a regular interval of $I_{demand}$, which is 10 s in the experiments. Timings of emission of the first request message from request nodes are randomly distributed in 1 sec to avoid collision. We conducted 100 simulation runs for each of 30 combinations of simulation parameters by changing the number of request nodes which send request messages from 1 to 5, the increase rate $\delta_i$ from 0.1 to 2.1, and the required number $N$ of active source nodes from 1 to 3.

Figures 4 and 5 summarize results of the number of active member nodes or active source nodes and the number of relay nodes in the network at the end of a simulation run of 20000 s, respectively. The number of active member or source nodes can be controlled by adjusting $\delta$ in our proposal and $N$ in directed diffusion. Each point is an average of 100 simulation runs. In the figures, PROP-SPAN means that our proposal is adopted, while PROP-FLOOD, that is our former proposal, employs our proposed scheme but




(a) Our proposal.




(b) Directed diffusion.

**Figure 5:** Number of relay nodes.

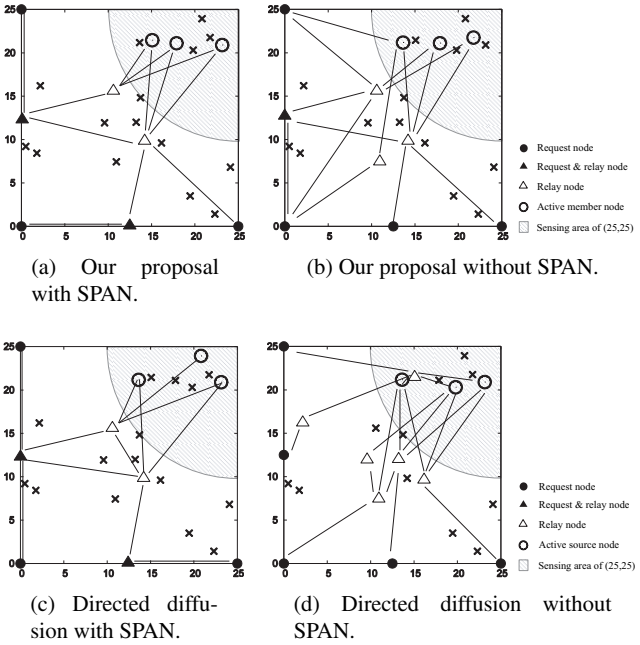
without SPAN. Instead, a request node use simple flooding to disseminate a request message in PROP-FLOOD. We also consider combination of directed diffusion with flooding and SPAN as DD-FLOOD and DD-SPAN, respectively.

Figure 4(a) shows that both of variations of the proposal, i.e. PROP-SPAN and PROP-FLOOD, keep the number of active member nodes constant even if the number of request nodes increases. Although not shown in the figure, the average number of active member nodes per application is one, two, and three with $\delta = 0.1$, 1.1, and 2.1.

This implies that our proposal can share active member nodes among applications without involving redundant devices. In addition, we also observe that the same $\delta_i$ results in the same number of active member nodes independently of the number of applications, while different $\delta_i$ results in the different number of active member nodes. When parameter $\delta_i$ is 0.1, the number of active member nodes stays 1. If there is no active member node, the demand intensity $s_i$ gradually increases. Consequently, the probability $P(X_{i,j} = \mathit{false} \rightarrow X_{i,j} = \mathit{true})$ in (2) becomes large at an idle member node. Then some idle member nodes become active and the demand intensity gradually decreases. At the beginning, the number of active member nodes is more than one. However, an active member node eventually becomes idle with probability $p_j$. If all active member nodes change to idle occasionally, the demand intensity increases again. Through the course, threshold $\theta_{i,j}$ is adjusted on each node. Consequently there appears a node which has the smallest threshold among all. As a result, the number of active member nodes converges to 1. Similarly, when parameter $\delta_i$ are 1.1 and 2.1, the number of active member nodes per application converges to 2 and 3, respectively.

In both of variations of directed diffusion, i.e. DD-SPAN and DD-FLOOD, the number of active source nodes is kept constant as shown in Figure 4(b). A sink selects the pre-determined number $N$ of source nodes based on the algorithm explained in Section 5.2. Then, it sends reinforcement messages to those nodes. By receiving a reinforcement message, $R_{\mathit{share}}$ increases and the priority of the source node becomes higher. Consequently, a source node selected by a sink node is likely to be selected by other sink. As a result, the desired number of source nodes, which are engaged in data reporting at the application-specific rate, are well shared among applications.

Regarding relay nodes, Figure 5(a) shows that incorporation with SPAN results in the smaller number of relay nodes than with flooding except for the case of $\delta = 0.1$ and the number of applications is 1. Being incorporated with SPAN, messages traverses the forwarding backbone between a request node and an active member node. Since there is only one forwarding backbone in the network and it is shared by nodes, a path between them is not necessary the shortest. On the contrary, a message disseminated by flooding follows the shortest path from a request node to a member node. As a result, the average number of relay nodes becomes larger with SPAN than with flooding, whereas the difference is only 0.2. When there are two or more applications or $\delta$ is set at a larger value to have two or more active member nodes, the proposal results in the smaller number of relay nodes. With flooding, in the worst case scenario, there exist the same number of independent and disjoint paths between all pairs of a request node and an active member node. On the other hand, the forwarding backbone is always shared among paths with SPAN. This apparently contributes to reduction in the number of relay nodes and the lifetime of a network can be prolonged. As shown in Figure 5(b), directed diffusion also benefits from SPAN. Comparing the proposal and directed diffusion, the number of relay nodes is similar, since the number of active member nodes and the number of active source nodes are the same. For example, Figure 6 shows snapshots of networks at the end of a simulation run. In the figure, the number of request nodes is 5, $\delta = 2.1$ for our proposal and $N = 3$ for directed diffusion. As shown in the figure, both of




(a) Our proposal with SPAN.

(b) Our proposal without SPAN.

(c) Directed diffusion with SPAN.

(d) Directed diffusion without SPAN.

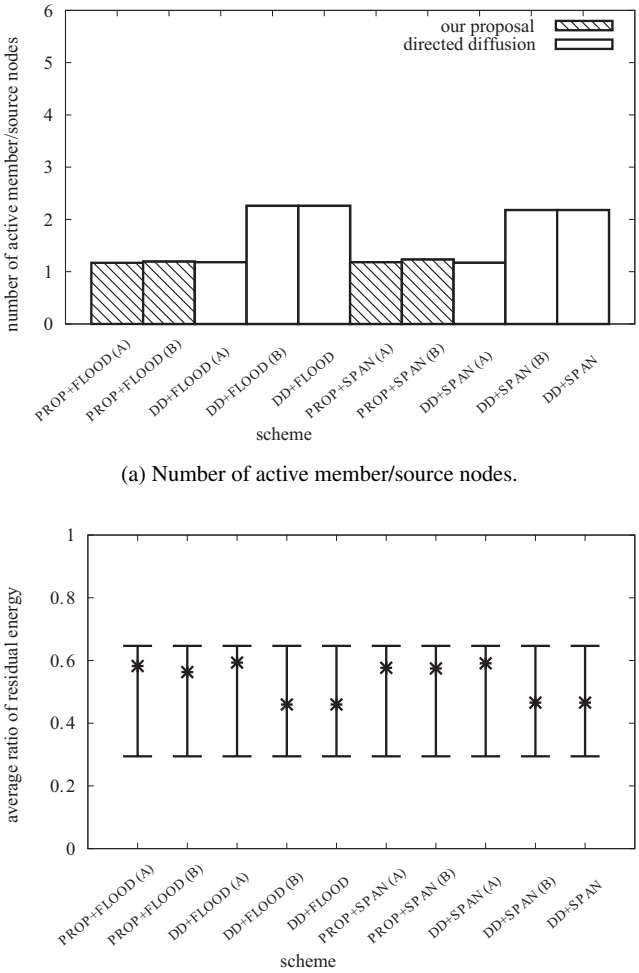
**Figure 6:** Snapshots of networks at the end of a simulation run.

our proposal with SPAN and directed diffusion with SPAN involve the minimum number of active source nodes and relay nodes to satisfy five applications. On the other hand, they involve more relay nodes and constructed networks become more complex if they do not adopt SPAN.

From the above results, we can conclude that the proposal can effectively share active member nodes and relay nodes among applications and keep the number of active member nodes constant independently of the number of applications in the current simulation setting. Since directed diffusion is a centralized protocol, where a sink decides source nodes to reinforce with rule-based decision making, it is not surprising that the number of nodes is kept as intended. On the other hand, each member node has the right to make a decision of device assignment in our proposal. Nevertheless, a response threshold model-based decision making algorithm brings results similar to directed diffusion's. That is, our proposal accomplishes self-organizing device assignment which is as optimal as a centralized and deterministic scheme.

### 5.5 Evaluation of robustness against parameter setting

We discuss advantages of the self-organization based proposal over the deterministic and complicated rule-based directed diffusion in regard to the robustness against errors in parameter setting. In this Section, we assume that three applications are operating in the area. Three nodes at coordinates $(0,0)$, $(12.5,0)$, and $(0,12.5)$ are their request nodes as illustrated in Figure 3. The requested number $N$ of active source nodes per application is set at 1 and parameter




(a) Number of active member/source nodes.

(b) Residual energy.

**Figure 7:** Rubustness of our proposal against parameter setting.

$\delta_i$ of the proposal is set at 0.1. To make nodes heterogeneous in energy condition, the initial residual energy of each node is set at random value ranging from 25% to 80%.

In general, a response threshold model is less sensitive to parameter setting similarly to other bio-inspired mechanisms [17]. To confirm this, we use the different parameter setting, i.e. set B in Table 4, we changed $m$ and $n$ in (5) from 3 to 6. With larger $m$ and $n$, it prevents a member node from becoming an active member node and getting the sufficient number of active member nodes become difficult. To examine the robustness of a decision making algorithm against parameter setting, we also change thresholds $T_{share}$ and $T_{energy}$ from 0.5 to 0.1 and 0.9, respectively. Since $R_{energy}$ ranges from 0.75 to 0.80 from the beginning of a simulation run and $R_{share}$ is always equal to or larger than $(3-2)/3 \simeq 0.33$. All source nodes have the same priority of 4 in Table 3. That is, directed diffusion cannot take into account the heterogeneity of nodes in device assignment.

Simulation results averaged over 100 runs are depicted in Figure 7. We considered 10 different schemes in the experiments. "DD+FLOOD" and "DD+SPAN" without an identifier of parameter setting corresponds to the original directed diffusion without and with SPAN. In Figure 7(a), the x-axis indicates schemes and the y-axis is the average number of active member nodes or active source nodes at 20000 s. In Figure 7(b), the y-axis indicates the averaged ratio of residual energy of member nodes. Each cross show an average ratio of residual energy of active member nodes. The top and bottom of each bar indicates the maximum and minimum ratios, respectively.

As shown in Figure 7(a), from a view point of the number of active member nodes, the change of parameter setting does not have strong impact on our proposal. On the contrary, directed diffusion suffers from an error and the number of active source nodes increases. This is because, all source nodes are categorized into one priority class. As a result, DD+FLOOD(B) and DD+SPAN(B) become identical to DD+FLOOD and DD+SPAN, respectively, where devices are not effectively shared among applications. In appropriate threshold setting further results in the unbalanced energy consumption as shown in Figure 7(b). We also observe that our proposal, independently of parameter setting errors, can achieve the same level of energy saving as the extended directed diffusion with appropriate thresholds.

From the above results, we can conclude that our self-organizing device assignment is less sensitive to errors in parameter setting and, as such, to operational conditions, than directed diffusion, while achieving as efficient device assignment as directed diffusion with appropriate parameters does.

## 6 Conclusion and future work

In this paper, we proposed a self-organizing device assignment mechanism for a multi-purpose WSAN. Results of simulation support the proposal, but there still remains room for further evaluation and improvement. When there is actuator contention among two applications with the same priority, our proposal first assigns an actuator to one application and then to another application by being stimulated by the increased demand intensity of the latter. The frequency that a device is assigned depends on $\delta_i$, i.e. an increasing rate of demand intensity. In other words, $\delta_i$ is another parameter with which an application can control device assignment. We need to confirm this by conducting additional experiments. We also need to evaluate the scalability and adaptability of the proposal, which are inherent characteristics of self-organizing systems.

**Acknowledgments** This research was supported in part by International Collaborative Research Grant of the National Institute of Information and Communications Technology, Japan and Grant-in-Aid for Scientific Research (B) 22300023 of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

## References

[1] I. F. Akyildiz and I. H. Kasimoglu, *Wireless sensor and actor networks: research challenges*, Ad Hoc Networks, 2 (2004), 351–367.

[2] E. Avilés-López and J. García-Macías, *TinySOA: a service-oriented architecture for wireless sensor networks*, Service Oriented Computing and Applications, 3 (2009), 99–108.

[3] H. M. N. Bandara, A. P. Jayasumana, and T. H. Illangasekare, *Cluster tree based self-organization of virtual sensor networks*, in Proc. of the International Workshops on Wireless Mesh and Sensor Networks, New Orleans, LO, 2008, 1–6.

[4] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J.-L. Deneubourg, *Adaptive task allocation inspired by a model of division of labor in social insects*, in Proc. of the International Conference on Biocomputing and Emergent Computation, Skövde, Sweden, 1997, 36–45.

[5] E. Bonabeau, G. Theraulaz, and M. Dorigo, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, 1st ed., 1999.

[6] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, *Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks*, Wireless Networks, 8 (2002), 481–494.

[7] Crossbow Technology, *MICAz Datasheet*. www.xbow.com.

[8] C. Frank and K. Römer, *Algorithms for generic role assignment in wireless sensor networks*, in Proc. of the 3rd International Conference on Embedded Networked Sensor Systems, San Diego, CA, October 2005, 230–242.

[9] C. Frank and K. Römer, *Solving generic role assignment exactly*, in Proc. of the 20th International Parallel and Distributed Processing Symposium, Rhodes Island, Greece, April 2006.

[10] N. Hoeller, C. Reinke, J. Neumann, S. Groppe, D. Boeckmann, and V. Linnemann, *Efficient XML usage within wireless sensor networks*, in Proc. of the 4th Annual International Conference on Wireless Internet, Maui, HI, November 2008, 1–10.

[11] C. Intanagonwiwat, R. Govindan, and D. Estrin, *Directed diffusion: a scalable and robust communication paradigm for sensor networks*, in Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00), Boston, MA, August 2000, 56–67.

[12] T. Iwai, N. Wakamiya, and M. Murata, *Proposal for dynamic organization of service networks over a wireless sensor and actuator network*, Procedia Computer Science, 5 (2011), 240–247.

[13] E. H. Jung and Y. J. Park, *TinyONet: a cache-based sensor network bridge enabling sensing data reusability and customized wireless sensor network services*, Sensors, 8 (2008), 7930–7950.

[14] H. B. Lim, M. Iqbal, and T. J. Ng, *A virtualization framework for heterogeneous sensor network platforms*, in Proc. of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09), Berkeley, CA, November 2009, 319–320.

[15] A. Majeed and T. A. Zia, *Multi-set architecture for multi-applications running on wireless sensor networks*, in Proc. of the 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA '10), 2010, 299–304.

[16] N. Mohamed and J. Al-Jaroodi, *A survey on service-oriented middleware for wireless sensor networks*, Service Oriented Computing and Applications, 5 (2011), 71–85.

[17] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, *Media streaming on P2P networks with bio-inspired cache replacement algorithm*, in Proc. of the 1st International Workshop on Biologically Inspired Approaches to Advanced Information Technology, A. J. Ijspeert, M. Murata, and N. Wakamiya, eds., vol. 3141 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2004, 380–395.

[18] J. Steffan, L. Fiege, M. Cilia, and A. Buchmann, *Towards multi-purpose wireless sensor networks*, in Proc. of the International Conference on Systems Communications, August 2005, 336–341.

[19] A. Varga, *The OMNeT++ discrete event simulation system*, in Proc. of the European Simulation Multiconference, Prague, Czech Republic, June 2001, 319–324.

[20] Y. Yu, L. J. Rittle, V. Bhandari, and J. B. LeBrun, *Supporting concurrent applications in wireless sensor networks*, in Proc. of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06), Boulder, CO, November 2006, 139–152.