Master's Thesis

Title

Bio-inspired autonomous device assignment for cooperative resource sharing in a wireless sensor and actor network

Supervisor

Professor Masayuki Murata

Author

Takuya IWAI

February 14th, 2012

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Bio-inspired autonomous device assignment for cooperative resource sharing in a wireless sensor and actor network

Takuya IWAI

Abstract

In the forthcoming information society, users will expect to be provided with various types of information services and environmental and machinery control which are suited for time, place, occasion, and people. As a fundamental technology to realize the society, many researchers focus their interests on a wireless sensor and actor network (WSAN) which can detect and conjecture environmental and personal condition and provide a user with various types of information services and environmental and machinery control. Regarding required type and installation location of nodes, they shapely depend on an application which they offer their sensing or actuating function. Therefore, an application constructor deploys dedicated nodes with appropriate sensors and actuators at optimal locations in advance. Then, these nodes are connected each other in multi-hop communication and an application is constructed. Although a single application cannot meet diverse requirements of different and dynamic situations, it is hard to deploy and configure a variety of sensors and actuators for each of envisioned applications. In addition, as another problem, duplicated nodes playing the same role among applications might exist when multiple applications are constructed. This is not a trivial issue for a WSAN which is required to strongly save resource consumption. To solve these problems, it is desirable that an application is dynamically constructed by combining not dedicated nodes but multi-functional and common nodes among applications. Furthermore, it is expected to share sensors and actuators among different applications for decreasing the number of activated nodes. In general, a complicated and deterministic rule-base mechanism, e.g. the usage of *if-then* type of rules, can realize above-mentioned cooperative resource sharing among multiple applications. However, it might become more difficult to write correct rules as the number

of assumed situations or concurrent applications increases. Even if we can write correct rules, wrong parameter setting also leads to the failure of resource sharing. In this thesis, we adopt a mathematical model of division of labors in a colony of social insects to accomplish autonomous assignment of devices to an application while taking into account cooperative device sharing among multiple applications. Through simulations, we confirmed that the proposal can accomplish device assignment where devices are shared among multiple applications in a fully-distributed and self-organizing manner which does not require deterministic and complicated rules of existing device assignment mechanisms. We also showed that the proposal was less sensitive to parameter setting than the existing mechanism.

Keywords

wireless sensor and actor network
response threshold model
parameter insensitivity
device assignment
resource sharing

Contents

List of Figures

List of Tables

1 Introduction

In recent years, many researchers have been actively working in the field of wireless sensor and actuator/actor networks (WSANs) [1, 2]. 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 [3]. 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 [4, 5], 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 [4, 6]. Once heterogeneous nodes can

be handled through the common interface, another challenge arises in selection of nodes and devices [7, 8]. 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. To share sensors among multiple applications, TinyONet is proposed in [9]. In TinyONet, a sink manages virtual sensors each of which corresponds to a physical node and groups them into a slice in accordance with application requirements. Since the authors' focus is on reusability of sensing data, TinyONet assumes that sensing data is periodically collected from all physical sensors to the sink. To keep cached data up-to-date, TinyONet consumes bandwidth and energy and it does not fit to a WSAN. In [10], the authors propose VSNs (Virtual Sensor Networks) which are constructed over independent WSNs. In their proposal, a VSN is constructed of, based on our naming, member nodes and relay nodes belonging to different WSNs, forming the tree topology. A VSN allows a service-oriented and inter-WSN overlay, but they do not specify the way to assign nodes and devices to VSN. 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 challenge, there are several proposals on dynamic device assignment [11], 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 this thesis, we presented an idea of autonomous device assignment mechanism where each node determines whether to offer its own devices to an application in a fully-distributed and self-organizing manner. Our mechanism cooperates with SPAN [12] to efficiently share nodes engaged in message relaying among applications. In our proposal, the minimum connectivity is maintained by SPAN, where a set of coordinator nodes construct 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 [13], 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 task. In our proposal, a request message advertised by a request node expresses the demand intensity to stimulate nodes to offer their devices.

The remainder of this thesis is organized as follows. First, in section 2, we describe related work. Next, in section 3, we describe application scenario that our proposal assumes and example of applications. Then, in section 4, we explain a response threshold model and analyze its characteristic. In section 5, we propose a response threshold model-based device assignment mechanism. In section 6, we show simulation results to evaluate our proposal and compare it with an existing mechanism. Finally, in section 7, we provide concluding remarks and future work.

2 Related work

2.1 Multi-purpose sensor and actor networks

Today, researchers' interests are shifting from a *special purpose WSAN* to a *multi-purpose WSAN* where multiple concurrent applications are running over a WSAN. In a multi-purpose WSAN, various types of applications use common infrastructures and we can expect the deployment and maintenance cost to be significantly reduced [3]. To realize a multi-purpose WSAN, there are many critical challenges which need to be solved. Here, we briefly explain the three basic challenges.

1. heterogeneous node
2. dynamic separation of nodes
3. resource confliction

Distributing multifunctional and common nodes in advance, organizing them as a single and monolithic WSAN, and accommodating all applications are one of the most realistic solutions to realize a multi-purpose WSAN. To realize this, a mechanism to control heterogenous nodes is needed. For this challenge, a network virtualization is very useful for seamless interoperability of different nodes and networks [5]. In addition, the concept of SOA (Service Oriented Architecture) [4, 6] is very useful for controlling heterogeneous nodes without concerning differences in node architectures, OS, and programming languages. As another challenge, if static applications are just organized, it is difficult for users to be provided with various types of information services and environmental and machinery control which are suited for time, place, occasion, and people. For this challenge, we need a mechanism to dynamically group nodes, their sensors, and actuators for each application while taking into account their status and sharing them is needed for flexible, efficient, and on-the-fly deployment of multiple applications [7]. However, nodes cannot always fulfill requests from multiple applications simultaneously [14]. For this challenge, a mechanism to solve resource contention among multiple applications is also needed [15, 16, 17]. The final couple of challenges might be able solved by rigorous rule-based mechanism such as Generic Role Assignment [11]. However, as the network be-

comes larger and the number of concurrent applications increases, it becomes impossible to write appropriate rules without contradictions.

2.2 Resource sharing in sensor and actor networks

To share sensors among multiple applications, TinyONet is proposed in [9]. In TinyONet, a sink manages virtual sensors each of which corresponds to a physical node and groups them into a slice in accordance with application requirements. From a viewpoint of an application, a dedicated sensor network, i.e. a slice, is tailored over heterogeneous sensors. Since the authors' focus is on reusability of sensing data, TinyONet assumes that sensing data is periodically collected from all physical sensors to the sink. On request from an request, a virtual sensor returns cached sensing data as if it was obtained from a physical sensor. To keep cached data up-to-date, TinyONet consumes bandwidth and energy and it does not fit to a WSAN. In [10], the authors propose VSNs (Virtual Sensor Networks) which are constructed over independent WSANs. In their proposal, a VSN is constructed of, based on our naming, member nodes and relay nodes belonging to different WSANs, forming the tree topology. A VSN allows a service-oriented and inter-WSAN overlay, but they do not specify the way that member nodes is selected. Furthermore, it is inefficient to make all messages pass through a root node though it benefits from data aggregation. Regarding on-demand selection of nodes which offer functions to an application, [11] proposes an algorithm for generic role assignment. An application developer provides a role specification using if-then type rules. It is translated to a role specification message and distributed to nodes by a gateway. Each node decides whether to become ON or OFF based on the received message and property information.

3 Application scenario

3.1 Scenario

We assume that many nodes with various types of device, i.e. sensors and actuators, are deployed in the area and organize a WSAN. 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. An application consists of a series of *processes*, such as turn on or off the light, which are realized by devices embedded in the area.

Each process of an application has a priority value, which is determined in advance but can dynamically change in response to, for example, emergency. Priority helps in solving an 'actuator contention' problem. Sharing a sensor device among multiple application processes is not harmful as far as the sensor can provide them with requested sensor data at the desired precision and frequency and it does not lead to energy or bandwidth depletion. On the contrary, sharing an actuator among multiple application processes sometimes causes a problem, which we call '*actuator contention*'. Assume that two processes of different applications need to turn on and off the light using the same switch. When one process wants to turn on the light while the other has an opposite request, i.e. turning off the light, the switch cannot simultaneously fulfill requests of the both. The occurrence of contradicting requests which need different and exclusive actions or operational modes of an actuator is called 'activator contention' in this thesis. The simplest way to solve activator contention is to assign a requested actuator to a process with the highest priority and let it operate in the desired mode.

3.2 Application examples

In this section, we give three examples of application, i.e. home theater, home security [18], and HVAC (heating, venting, and air conditioning) [19], operating in a room. Table 1 summarizes outlines, processes, and their priority of applications. A smaller priority value means that a corresponding process has the higher priority. Regarding those processes

that use a sensor, a priority value is not specified and shown as 'NA' in the table.

When there are some people in the room, an HVAC application tries to maintain the temperature and humidity of the room at the appropriate and comfort level. It needs several thermal sensors, humidity sensors, and CO/CO2 sensors to report sensing information to an HVAC server. Based on the obtained information, the server requests a controller of an air conditioner to adjust the room temperature and a switch of a ventilation fan to clean the air. Now, a person begins to watch a movie. A server of a home theater application wants to control a ventilation fan, curtains, room lights, a TV monitor, a video player, and speakers. Then, a WSAN in the room tries to assign actuators corresponding to those facilities to processes of the home theater application when requested. To avoid a noise a process of the home theater application intends stopping a ventilation fan, but it conflicts with a demand of a process of the HVAC application to clean the air. Since those competing processes have the same priority in Table 1, a switch of a ventilation fan is assigned to either process which strongly requests a ventilation fan. For purpose of illustration, we here assume that a switch of a ventilation fan is assigned to a process of HVAC application. As air gradually clears, the process of home theater application more strongly requests a ventilation fan in comparison. As a result, a switch of a ventilation fan is assigned to a process of home theater application and a ventilation fan will stop. At nightfall, a server of home security application requests a WSAN to close a curtain, turn on room lights and security lights, and lock a window and a door. Both home theater and home security applications want to controll room lights. As shown in Table 1, the priority relating to room lights of home theater application is higher than home security application. Therefore, switches of room lights is assigned to home theater application and room lights continue extinction. When the person quits watching a movie, only home security application requests switches of room lights and room lights are lighting up.

Table 1: Example of applications

Home theater	
Outline	When an user tries to watch a video, it makes the room dim and quiet and makes the monitor, player and speaker playable.
process (priority)	It keeps a ventilation fan off while the user uses the home theater (3). Next, it keeps the curtain closed (3) and keeps the light blackout (3). Then, it keeps on a player and the player is playing (2). After that, it keeps the input of monitor connected to the player (2), and the input of speaker connected to the player (2).
Home security	
Outline	It monitors troubles using various types of sensors. When troubles are detected, it sounds an alarm. At nightfall, it closes a curtain and turns on a room light and a security light for security.
process (priority)	It steadily collects sensing information from sensors (NA). At nightfall, it keeps windows and doors locked (4), keeps curtains closed (4), and keeps on room lights and security lights (4). When it detects a trouble, it keeps the input of speaker connected to the home security (1) and sounds an alarm (1).
HVAC	
Outline	In response to personal condition, room temperature, and air pollution, it ventilates a room and adjusts room temperature.
process (priority)	It steadily collects sensing information from sensors (NA). When room temperature is higher than target temperature, it keeps on a heater and cools the room (3). When room temperature is lower, it keeps on a cooler and heats the room (3). When concentration of CO/CO2 is high, it keeps on a ventilation fan and ventilates the room (3).

4 Response threshold model

In this section, we firstly explain the mathematical model of the division of labors in social insects. Then, we verify characteristics of the model through some simulations.

4.1 Mathematical model

A response threshold model [13] is a mathematical model which imitates a mechanism of adaptive division of labors in a colony of social insects. A colony is divided into two groups of workers and non-workers based on autonomous decision of individuals using a simple rule. The size of groups is well adjusted based on the task-associated intensity of stimuli. As to the basic behavior of the model, please look at section 4.2.2. In the following, we consider there is one task to be performed in the colony for the sake of simplicity.

Let s $(0 \leq s)$ be the task-associated stimulus intensity, which gradually increases over time and decreases as individuals work as formulated by the following equation.

$$s(t) = s(t-1) + \delta - \frac{\alpha N_{act}}{M},$$

where δ $(0 \leq \delta \leq 1)$ is the increasing rate of the stimulus intensity. α $(0 < \alpha)$ is the impact of a worker on the task. N_{act} is the number of workers among N individuals which are capable of performing the task. In the model, each individual stochastically decides on whether to perform the task or not. Individual i has a state value $X_i \in \{0, 1\}$. Individual i with $X_i = 0$ does not perform the task and the other performs the task.

The probability $P(X_i = 0 \to X_i = 1)$ that individual i begins performing the task at time t is given by the following equation.

$$P(X_i = 0 \to X_i = 1) = \frac{s(t)^2}{s(t)^2 + \theta_i(t)^2},$$

where $\theta_i(t)$ $(\theta_{min} \leq \theta_i(t) \leq \theta_{max})$ is a threshold that corresponds to hesitation of individual i in performing the task at time t. The probability $P(X_i = 1 \to X_i = 0)$ that individual i quits the task is given by a constant p $(0 \leq p \leq 1)$.

$$P(X_i = 1 \to X_i = 0) = p$$

This enables rotation of performing the task among individuals. The average duration that an individual performs the task is $1/p$. When the number of idle individuals occasionally increases for perturbation, the stimulus intensity increases accordingly and those

individuals with high threshold become workers based on the increased stimulus intensity. As a result, the ratio of workers is maintained at the same level.

Threshold $\theta_i(t)$ can be identical or different among individuals. Individuals with smaller threshold are more likely to become workers. In the response threshold model proposed in [13], there is a mechanism of reinforcement, which differentiates workers from non-workers and makes them specialists. Threshold θ_i is adapted by the following equations.

$$\theta_i(t) = \begin{cases} \theta_i(t-1) - \xi, & \text{if individual } i \text{ performs a task} \\ \theta_i(t-1) + \varphi, & \text{otherwise} \end{cases}$$

where ξ $(0 < \xi)$ and φ $(0 < \varphi)$ are related to the speed of differentiation. By reinforcement, an individual performing the task becomes a worker more often than one that does not work. Eventually the colony is divided into distinct groups of workers and non-workers.

4.2 Characteristic analysis

4.2.1 Metric

In characteristic analysis, we analyze the effect of parameter setting on the behavior of response threshold model. We define worker specialist ratio R_{work} $(0 \leq R_{work} \leq 1)$ which represents the propotion of worker specialists to individuals and non-worker specialist ratio R_{nwork} $(0 \leq R_{nwork} \leq 1)$ which represents the propotion of non-worker specialists to individuals. Here, a worker specialist corresponds to a node with $\theta_i(t) < \theta_{min} + \Delta_a$ and a non-worker specialist corresponds to a node with $\theta_i(t) > \theta_{max} - \Delta_i$, where Δ_a and Δ_i are a constant.

$$\begin{aligned} R_{work}(t) &= \frac{|\{i \in \mathbf{I} \mid \theta_i(t) < \theta_{min} + \Delta_a\}|}{M} \\ R_{nwork}(t) &= \frac{|\{i \in \mathbf{I} \mid \theta_i(t) > \theta_{max} - \Delta_i\}|}{M} \end{aligned}$$

Here, i is an identifier of an individual. $\mathbf{I}$ is a set of identifiers of individuals. M is the total number of individuals.

To analyze the stability of response threshold model, we calculate the coefficient of variation (c.v.) C_n of the number of workers. The average number $\overline{N}$ of workers from

time T $(0 < T)$ to time $T + k - 1$ $(1 \leq k)$ in discrete time step is calculated as below.

$$\overline{N} = \frac{1}{k} \sum_{t=T}^{T+k-1} N(t)$$

Using the average number of workers $\overline{N}$, the variance of the number of workers σ_w^2 $(0 \leq \sigma_w^2)$ from time T to time $T + k - 1$ in discrete time step is calculated as below.

$$\sigma_w^2 = \frac{1}{k} \sum_{t=T}^{T+k-1} \left(N(t) - \overline{N} \right)^2$$

Using the average number of workers $\overline{N}$ and the variance of the number of workers σ_w^2, the c.v. C_n of the number of workers is derived as below.

$$C_n = \frac{\sqrt{\sigma_w^2}}{\overline{N}}$$

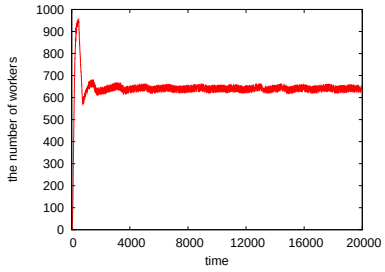
Larger C_n means that the number of workers is more fluctuating from time T to time $T + k - 1$. Similarly, we can calculate the average worker specialist ratio $\overline{R_{work}}$, the average non-worker specialist ratio $\overline{R_{nwork}}$, the average stimulus intensity $\overline{s}$, and the c.v. C_s of stimulus intensity.

4.2.2 Basic behavior

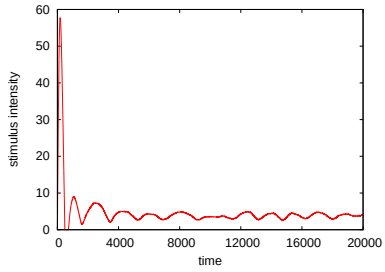
We briefly describe the behavior of response threshold model. Here, the total number M of individuals is 1,280. The impact α of a worker is 1. The increasing ratio δ of stimulus intensity is 0.5. The maximum θ_{max} of threshold is 1,000 and the minimum θ_{min} of threshold is 1. Δ_a and Δ_i are 50. The probability p that a worker becomes a non-worker is 0.002. The initial stimulus intensity $s(0)$ is 0 and the initial threshold $\theta(0)$ is 500.

Figures 1(a), 1(b), and 1(c) illustrate variation in the number of workers, the stimulus intensity, and the specialist ratio, respectively. Figures 1(d) and 1(e) are a histogram representing individual distribution in time ratio staying a worker specialist and a non-worker specialist during the whole simulation, respectively.

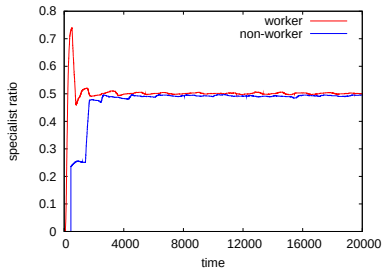
As illustrated in Fig. 1(a), the initial number of workers is 0 and a sufficient number of individuals are not workers. Therefore, as shown in Fig. 1(b), the stimulus intensity increases to about 60. As the stimulus intensity increases, the probability that a non-worker becomes a worker increases and the number of workers increases. Figure 1(c) shows



(a) the number of workers

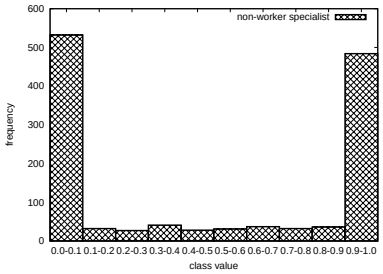


(b) the stimulus intensity



(c) the specialist ratio

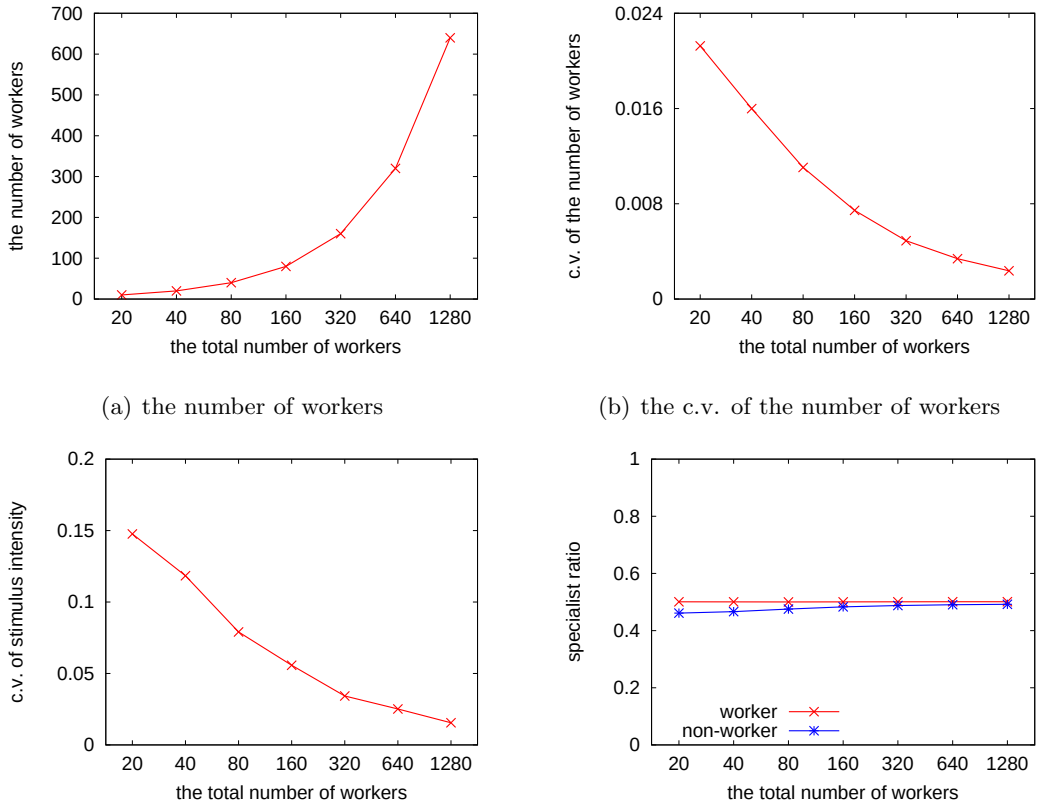
(d) the histogram of worker specialist



(e) the histogram of non-worker specialist

Figure 1: Basic behavior

that the number of worker specialists increases due to the reinforcement mechanism. In the parameter setting of this simulation, constant ξ which corresponds to the speed of becoming a worker specialist is larger than constant φ which corresponds to the speed of becoming a non-worker specialist. Therefore, worker specialist ratio increases in advance of non-worker specialist ratio. Once a sufficient number $\delta M/\alpha = 640$ of individuals become workers (Fig. 1(a)), the stimulus intensity starts to gradually decrease (Fig. 1(b)). While the stimulus intensity is 0, a new non-worker does not become a worker. In addition, a worker becomes a non-worker with probability p. As a result, the number of workers gradually decreases to less 640 (Fig. 1(a)) and the demand intensity starts to increase



(a) the number of workers

(b) the c.v. of the number of workers

(c) the c.v. of stimulus intensity

(d) the specialist ratio

Figure 2: Influence of colony size

again (Fig. 1(b)). While the demand intensity is more than 0, a non-worker stochastically becomes a worker and the number of workers will increase. The reinforcement mechanism divides individuals into worker specialists or non-worker specialists as shown in Figs. 1(d) and 1(e). Even if a worker specialist temporarily becomes an non-worker, it immediately becomes a worker again in response to the increase in the demand intensity. The increase in the number of workers decrease the demand intensity. Once individuals are divided into two groups, the fluctuation of the number of workers becomes small and the number of workers is kept at 640 in the parameter setting.

4.2.3 Influence of colony size

To analyze the influence of the number of individuals, we vary the number of individuals from 20 to 1,280. Other parameters are the same as a simulation in section 4.2.2. Results

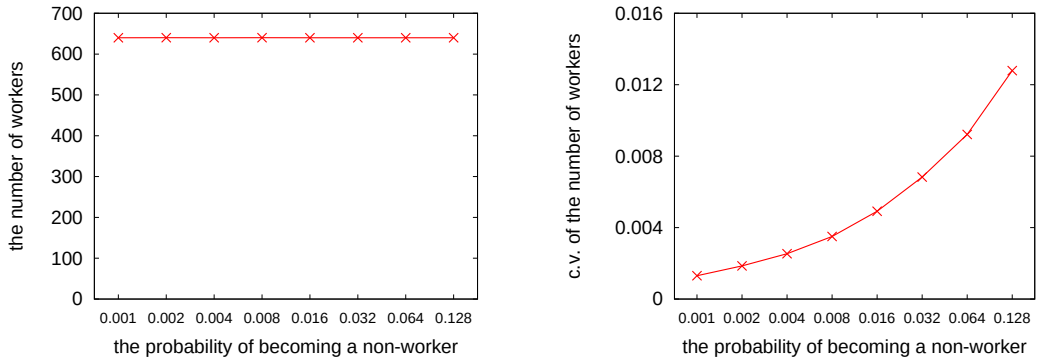
are the average of 1,000 simulation runs.

Figures 2(a), 2(b), 2(c), 2(d) illustrate variation in the average number $\overline{N}$ of workers, the c.v. C_n of the number of workers, the c.v. C_s of the stimulus intensity, and the specialist ratio $\overline{R_{work}}$ and $\overline{R_{nwork}}$, respectively. Here, T is 19,900 and k is 100. Figure 2(a) shows that the average number of workers is roughly equal to the target value, i.e. $\delta M/\alpha$. However, Fig. 2(b) shows that the c.v. of the number of workers decreases as the number of individuals increases. As the number of individuals is more, the impact per worker on the stimulus intensity becomes smaller. When the number of workers is less than the target value, the degree of increases in the stimulus intensity becomes larger. On the contrary, when the number of workers is more than the target value, the decreases in the stimulus intensity becomes larger. Therefore, as illustrated in Fig. 2(c), the stimulus intensity more fluctuates and the probability that a non-worker becomes a worker also fluctuates. As a result, the number of workers fluctuates and the c.v. of the number of workers becomes larger. In this parameter setting, constant ξ is larger then constant φ. Threshold θ is more likely to decrease and a non-worker is more likely to quit a non-worker specialist. As a result, as shown in Fig. 2(d), the specialist ratio of non-workers is lower than the specialist ratio of workers as the number of individuals decreases.

4.2.4 Influence of frequent task rotation

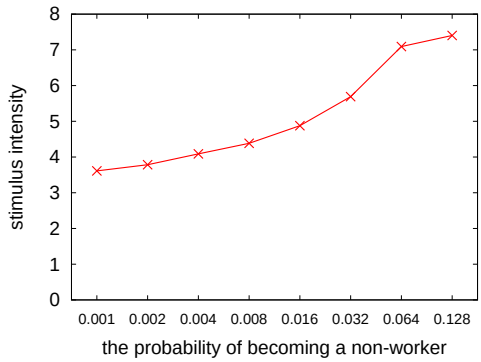
To analyze the influence of frequency of task rotation, we vary the probability that a worker becomes a non-worker from 0.002 to 0.128. Other parameters are the same as a simulation in section 4.2.2. Results are the average of 1,000 simulation runs.

Figures 3(a), 3(b), 3(c), 3(d) illustrate variation in the average number of workers $\overline{N}$, the c.v. of the number of workers C_n, the average stimulus intensity $\overline{s}$, and the specialist ratio $\overline{R_{work}}$ and $\overline{R_{nwork}}$, respectively. Here, T is 19,900 and k is 100. Figure 3(a) shows that the average number of workers is roughly equal to the target value, i.e. $\delta M/\alpha$. However, Fig. 3(b) shows that the c.v. of the number of workers increases as the probability p increases. It is apparently because a worker more frequently becomes a non-worker with higher probability p and the number of workers is more fluctuating. As the probability p increases, workers more frequently become non-workers and an insufficient number of workers exist. In order to cover the deficiency of workers, Fig. 3(c) shows that the stimulus

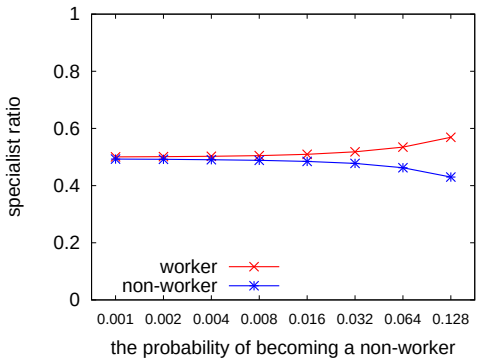


(a) the number of workers

(b) the c.v. of the number of workers



(c) the stimulus intensity



(d) the specialist ratio

Figure 3: Influence of frequent task rotation

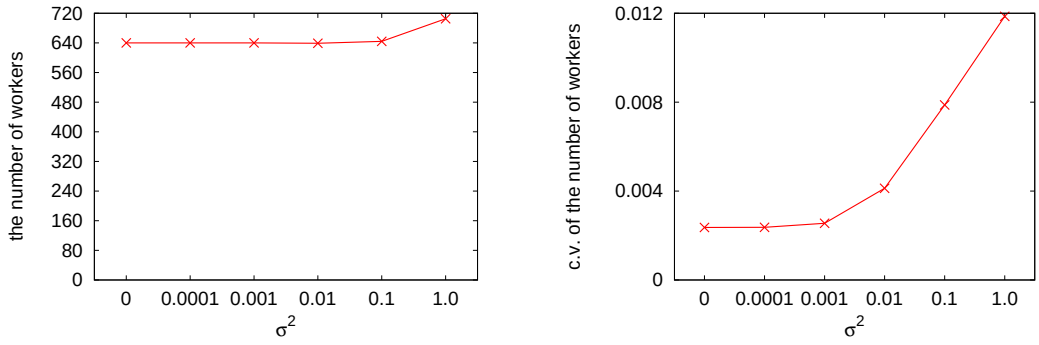
intensity increases and more individuals are inclined to become a worker. In addition, constant ξ is larger then constant φ. Therefore, threshold θ is more likely to decrease and a non-worker is more likely to quit a non-worker specialist. As a result, as shown in Fig. 3(d), the specialist ratio of non-workers is lower than the specialist ratio of workers as the probability increases.

4.2.5 Influence of demand perturbation

To analyze robustness of response threshold model to instantaneous noise, we add gauss-noise to stimulus intensity as follows.

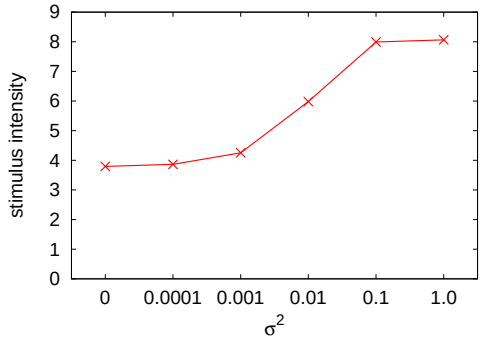
$$s(t+1) = s(t) + \delta - \frac{\alpha N(t)}{M} + \eta(t+1)$$

Here, $s(t)$ is stimulus intensity at time t. δ is the increasing ratio of stimulus intensity. α is the impact of a worker. $N(t)$ is the number of workers at time t. M is the number of

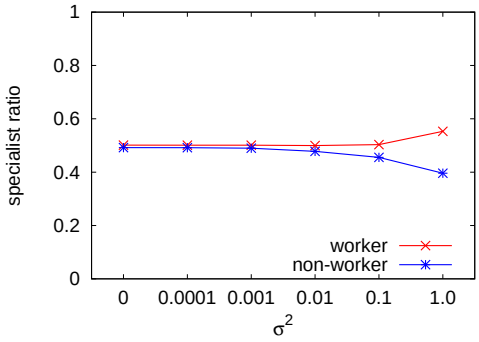


(a) the number of workers

(b) the c.v. of the number of workers



(c) the stimulus intensity

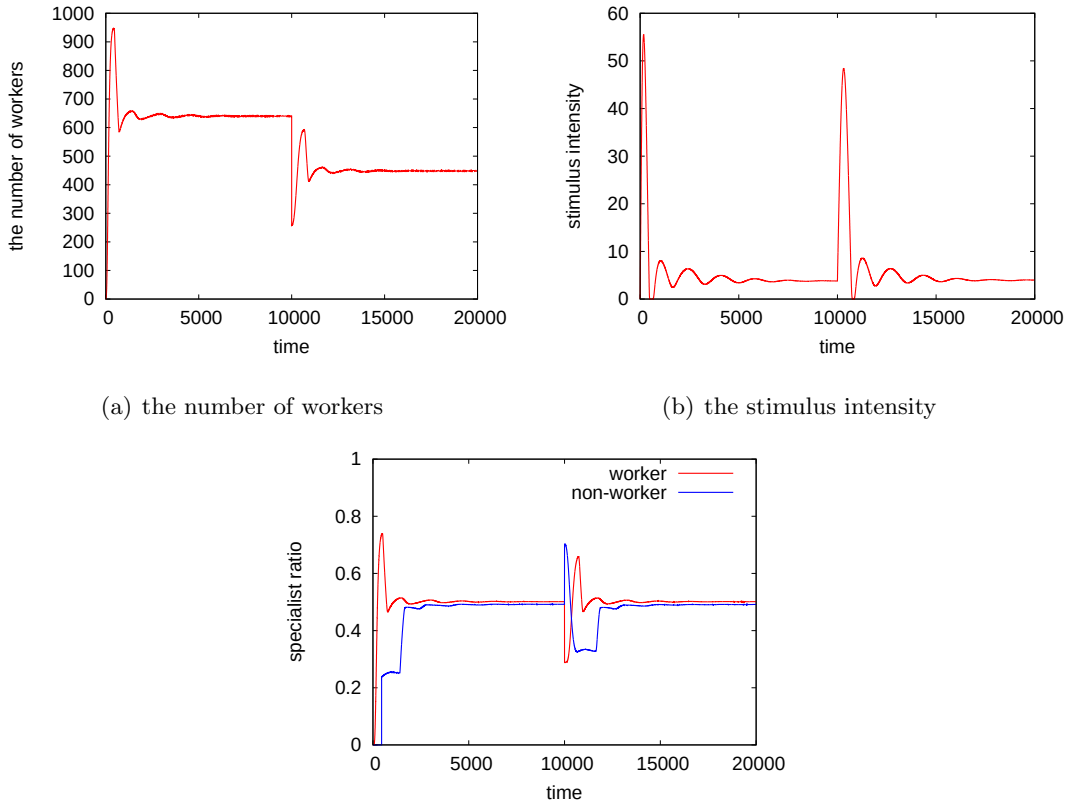


(d) the specialist ratio

Figure 4: Influence of demand perturbation

individuals. $\eta(t+1)$ is a gauss-noise at time $t+1$ where the average is 0 and the variance σ_s^2. In order to change the impact of noise, we change the variance σ_s^2 from 0.0 to 1.0. Other parameter settings are the same as a simulation in section 4.2.2. Results are the average of 1,000 simulation runs.

Figures 4(a), 4(b), 4(c), 4(d) illustrate variation in the average number $\overline{N}$ of workers, the c.v. C_n of the number of workers, the c.v. of the stimulus intensity C_s, and the average specialist ratio $\overline{R_{work}}$ and $\overline{R_{nwork}}$, respectively. Figure 4(a) shows that the average number of workers in $\sigma_s^2 = 0,\ 0.0001,\ 0.001,\ 0.01$ and 0.1 are roughly equal to the target value. However, when σ_s^2 is 1.0, the number of workers increases by about 80. However, the size of noise with $\sigma_s^2 = 1.0$ is too large. The increasing ratio δ of the demand intensity is ranging from 0 to 1. In case of $\sigma_s^2 = 1.0$, the noise with about more 30 is often added to stimulus intensity. Therefore, this situation is not realistic and we can ignore the result. Fig. 4(b)



(a) the number of workers

(b) the stimulus intensity

(c) the specialist ratio

Figure 5: Influence of individual extinction

shows that the c.v. of the number of workers increases as the variance σ_s^2 increases and we can find the stability of the number of works decreases as the size of noise becomes larger. On the contrary, the average number of works is not influenced by noise. Therefore, we conclude that response threshold model is highly robust to noise.

4.2.6 Influence of individual extinction

To analyze the influence of individual extinction, we remove a worker with probability 0.6 from colony at time 10,000. Other parameters are the same as a simulation in section 4.2.2. In the simulation, we dose not change parameter δ of the response threshold model around the time of death of works. Results are the average of 1,000 simulation runs.

Figure 5(a) shows that the number of workers suddenly decreases at time $10,000$ due to individual extinction. However, the number of works increases in the very short term.

Table 2: Characteristic of response threshold model

	scale	optimality	robustness	resilience
response threshold model	larger is better	average	high	high

It is because demand intensity increases due to the deficiency of works and the probability that a non-worker becomes a worker also increases. In the simulation, parameter δ does not adaptively change in response to the change of the number of individuals. Therefore, the number of workers decreases to about 450 until stabilization. In addition, as shown in the Fig. 5(c), the specialist ratio of workers and non-workers recovers and maintained at 0.5. This means that a response threshold model is highly resilient.

4.3 Conclusion

We organize characteristics of response threshold model as Table 2. From section 4.2.3, we can find that the stability which is represented by the c.v. of the number of workers advances as the number of individuals increases. However, the average number of works is not influenced by the number of individuals, the probability that a worker becomes a non-worker, and the demand perturbation based on the results in sections 4.2.3, 4.2.4, and 4.2.5. Therefore, we can conclude that the more is better as to the scale which means the number of individuals. Then, the optimality which means a sufficient number of individuals become a worker is averagely achieved. Therefore, we put 'average' in optimality of the table. Finally, as described in sections 4.2.5 and 4.2.6, this model is highly robust to noise and resilient to individuals extinction. Therefore, we put 'high' in both 'rubustness' and 'resilience'.

5 Response threshold model-based device assignment

The proposal adopts a response threshold model of division of labors in a colony of social insects [13] to accomplish autonomous and fully distributed decision making of nodes on whether to assign embedded devices to an application.

5.1 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 mode active device and some idle devices, but an idle member node has only idle devices. A decision 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,

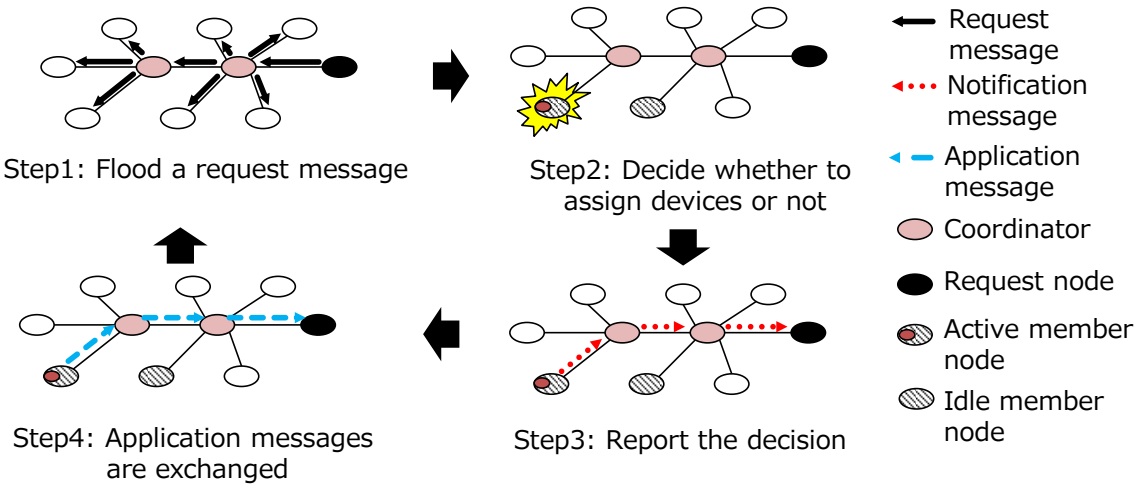


Figure 6: Overview of device assignment

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.

5.2 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 Fig. 6). The minimum connectivity of a WSAN is maintained by SPAN [12]. 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 whole area or nodes in the specified area of interest when location information is available, through the forwarding backbone [20, 21].

When a node receive 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

Table 3: Internal values of a node

Notations	Default	Description
$\mathbf{D}$	ϕ	set of devices
$\mathbf{O_j}$		set of possible operational modes of device j
$\mathbf{S}$	ϕ	set of requirements of applications
$\mathbf{X}$	ϕ	set of $X_{i,j}$
$\mathbf{Y}$	ϕ	set of Y_j
$\mathbf{\Theta}$	ϕ	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

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 (step4).

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 also is possible for a member node to issue a new request message for the application. At the end of 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.

5.3 Internal values of nodes

In our proposal, a node maintains a set of information summarized in Table 3. 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 $n = |O_j|$, i.e. the number of operational modes.

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

A device cannot operate in different operational modes simultaneously. 'mode_n' is a '*default mode*' of a device and an idle device is in 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.

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 $\theta_{i,j}$ ($i \in \mathbf{I}$ 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 operational mode 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 $i \in \mathbf{I}$, where 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.

5.4 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 an application performs a process for a 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 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 $(\text{motion sensor}, \text{sensing}, (x, y), I_{data})$. Content information can be extended by using an XML-based method [22].

When a node other than a request node of an application receives a request message, it behaves following a flow chart shown in Fig. 7. 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 $\mathbf{S}$, it generates a new 7-tuple element. If the corresponding tuple exists, it is updated. Then, a node checks the priority of the requesting application and 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 $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{\Theta}$, respectively, if not exist. If the priority of the application is the highest in all applications in set $\mathbf{S}$ or not assigned, values $X_{i,j}$, Y_j, and $\theta_{i,j}$ are updated by a decision making algorithm explained in section 5.5. A request message requesting assignment of an actuator, to help solving actuator contention, a priority is assigned. On the contrary, a request message requesting assignment of a sensor, a sensor can be shared among multiple applications and priority is not assigned. After decision making, if all applications with the highest priority in a set $\mathbf{S}$, device j is not assigned to any application and the state of device j will be set at 'default mode'. When $X_{i,j}$ is *true* and application i has the highest priority in all applications, the state is set at the mode requested by application i. Otherwise, the state is not updated. 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 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. 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 is calculated from the number of notification messages by the following equation, where the initial demand intensity $s_i(0)$ is set at 0.

$$s_i(t+1) = s_i(t) + \delta_i - N_i(t) \tag{1}$$

Here, δ_i $(0 \leq \delta_i)$ is an increasing rate of demand intensity of application i. $N_i(t)$ $(0 \leq N_i(t))$ is the number of active member nodes which is equal to the number of notification messages stating $X_{i,j} = true$ received in response to the t-th request message. The equation means that, when the number of active member nodes is less than δ_i, the demand intensity gradually increases and the request node requests more member nodes to become an active member node. On the other hand, when the number is greater than δ_i, the demand intensity gradually decreases and active member nodes become inactive. The updated demand intensity is notified to member nodes by a request message at the beginning of the next round. Until the next round, a request node exchanges messages with active member nodes.

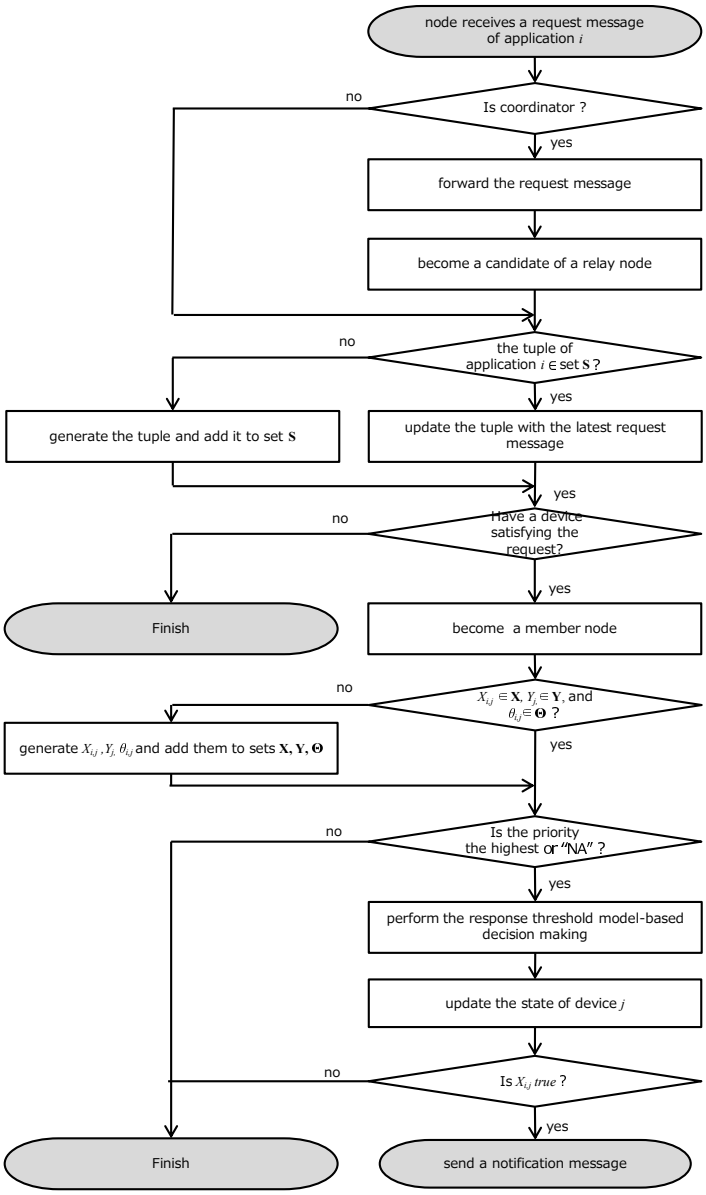


Figure 7: Behavior of a node on receiving a request message

5.5 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. The size of group is well adjusted based on the task-associated intensity of stimuli [23]. A response threshold model is a mathematical model of the division of labors of social insects [13]. 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 [13] or Appendix.

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

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

Here, $s_i(t)$ ($0 \leq s_i(t)$) is the demand intensity of application i at the t-th round. $\theta_{i,j}$ ($0 < \theta_{i,j} \leq \theta_{max}$) 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 ($1 \leq A_j$) is a variable related to the degree that device j is shared among application, and the residual energy of the node. Derivation of A_j will be explained in section 5.6.

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

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

Here, p_j ($0 \leq p_j \leq 1$) is a constant defined per device. This enables rotation of task among member nodes and prevents active member nodes from redundant device assignment. The average duration that an active member node assigns device j 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.

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

where ξ_j $(0 < \xi_j)$ and φ_j $(0 < \varphi_j)$ 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.

5.6 Variable A_j for device sharing and energy efficiency

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

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

Parameters are summarized in Table 4. Here, the first term of right side is used for device sharing among applications. Variable S_j $(1 \leq S_j)$ represents the number of applications where a node is a member regarding device j. S_j is derived as $S_j = |\{X_{i,j} \in X \mid i \in L\}|$ where L is a set of identifiers of application where a node is a member node. Variable F_j $(0 \leq F_j)$ represents the number of applications where a node is an active member node. F_j is derived as $F_j = |\{X_{i,j} \in X \mid i \in I, X_{i,j} = true\}|$ and $F_j \leq S_j - 1$. Exponent m $(1 \leq m)$ influences the sensitivity to the degree of sharing. The second term is used for balancing energy consumption. P_{full}/P_{res} is the ratio of the battery capacity P_{full} $(0 < P_{full})$ to the residual energy P_{res} $(0 < P \leq P_{full})$. Exponent n $(1 \leq n)$ influences the sensitivity to the residual energy. The third term is used for shifting minimum value of valuable A_j from 2 to 1. Variable A_j becomes smaller and probability $P(X_{i,j} = false \rightarrow X_{i,j} = true)$ becomes higher on a node which is engaged in more applications as an active member node and has more residual energy.

Table 4: 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_{res}	amount of residual energy of a node
P_{full}	total capacity of battery of a node

6 Performance evaluation

In this section, we evaluate our proposal through comparison with directed diffusion [24] and our former proposal [25]. 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. Our proposal is also designed with actuator contention. We are sure that our proposal can solve actuator contention when each process of applications is given an appropriate priority [17]. Therefore, we do not evaluate this issue in this paper.

6.1 Directed diffusion

Directed diffusion is a data-centric information gathering mechanism [24]. 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 member node in our proposal, and begins to send sensing data as data messages at the specified report interval. Data messages reach the sink by following

Table 5: 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

gradients. The first data message is called an *exploratory data* message.

A sink would receive multiple exploratory messages from different sources. 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.

6.2 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 6.4, while device assignment relies on an autonomous decision of each node. On the other hand, the number of sources 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, interest messages and reinforcement messages do not always reach the same set of sources. 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, we extend a data message to have ad-

ditional fields for an identifier id_{sink} of a sink, id_{src} of a source, the amount P_{res} ($0 < P_{res} \leq P_{full}$) of residual energy, the battery capacity P_{full} ($0 < P_{full}$), the number M_{dd} ($1 \leq M_{dd}$) of sinks from which it receives interest or reinforcement messages, and the number N_{dd} ($0 \leq N_{dd}$) 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 [24] for details of other fields than those newly introduced. Next, to identify a path between a specific sink and a specific source, we add a field of an identifier id_{sink} of a sink and id_{src} of a source 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]. As example of message exchange, an interest message is flooded and generates gradient with a field id_{src} empty. Then, a data message is forwarded to a sink following gradient and fills in a field id_{src} of gradient. Then, a reinforcement message from a specific sink to a specific source is forwarded based-on data-caches.

A sink in the extended directed diffusion first disseminates an interest message to all nodes. Next sources begin to send data messages. A data message sent by a source 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 5, where T_{energy} and T_{share} are thresholds. Numbers show priority values. For example, if a source 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 to reinforce. Finally, following an ascending order of priority value, a sink selects the required number N of sources and send reinforcement messages to them. In the following, we call a source which receives a reinforcement message an active source.

Table 6: Parameter setting of performance evaluation

Notations	Description	Value
p_j	probability of quitting task in Eq. (3)	0.01
ξ_j	threshold adaptation parameter in Eq. (4)	0.1
φ_j	threshold adaptation parameter in Eq. (4)	1
m	influence of degree of sharing devices in Eq. (5)	3
n	influence of residual energy in Eq. (5)	3
L	interval for collecting exploratory data messages	0.5
T_{share}	threshold of reinforcement rule	0.5
T_{energy}	threshold of reinforcement rule	0.5
I_{demand}	interval of request message	10

6.3 Simulation setting

We used OMNet++ [26] for simulation. 25 nodes are placed in the area of 25 m $\times$ 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 8 illustrates an example of node layout where the x and y axes are coordinates 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 MICAz [27], 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 μA 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 a 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. Pa-

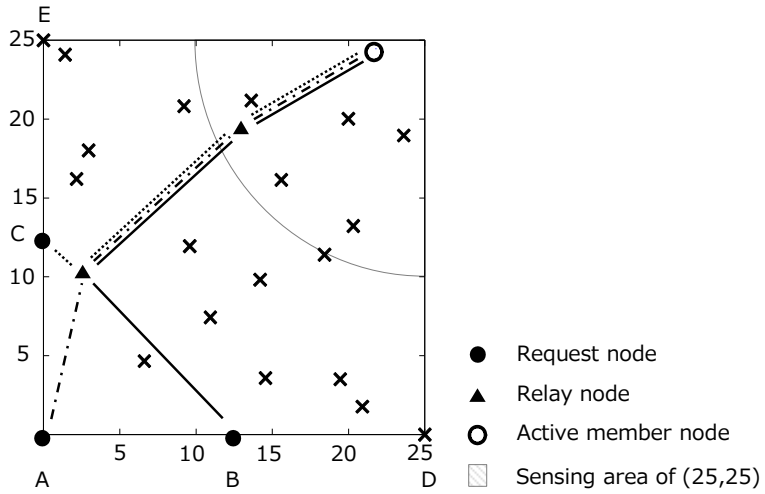


Figure 8: Snapshot of a simulation

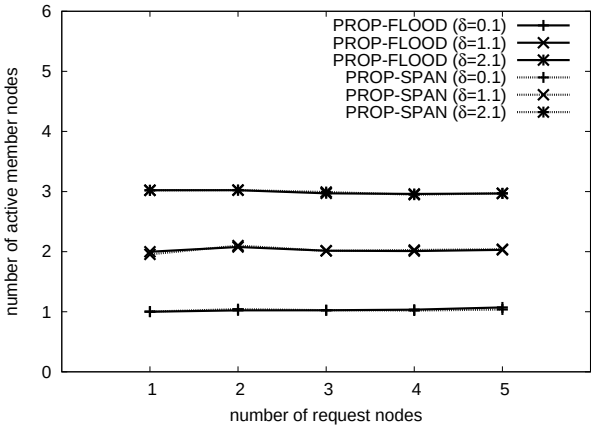
rameters used in the simulation experiments are summarized in Table 6, which are chosen based on preliminary experiments.

6.4 Evaluation of task assignment

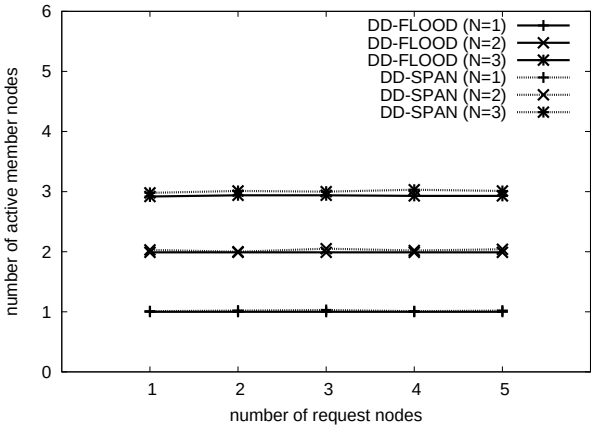
Since self-organization does not always achieve the optimal result due to its autonomous behavior, in this section, we first verify that our proposal can accomplish as good device assignment as directed diffusion which employs deterministic rules. Evaluation is conducted, from a viewpoint of the number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

We configure 5 edge nodes as request nodes of 5 independent applications. All request nodes require the information about the corner point 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 Fig. 8. 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 δ_i from 0.1 to 2.1, and the required number N of active sources from 1 to 3.

Figures 9 and 10 summarize results of the number of active member nodes or active



(a) our proposal

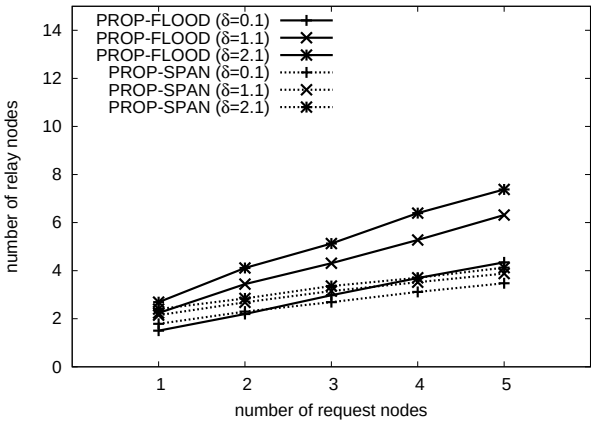


(b) directed diffusion

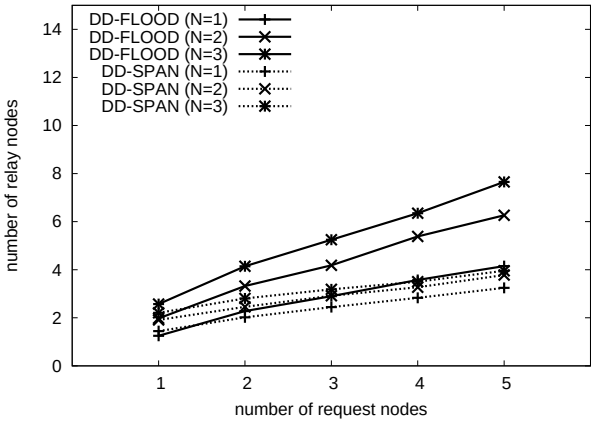
Figure 9: Number of active member nodes

sources 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 δ 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 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 9(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



(a) our proposal



(b) directed diffusion

Figure 10: Number of relay nodes

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 δ_i results in the same number of active member nodes independently of the number of applications, while different δ_i results in the different number of active member nodes. When parameter δ_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 $P(X_{i,j} = false \rightarrow X_{i,j} = true)$ in Eq. (2) becomes large at an idle member node. Once some idle member nodes become active, the demand intensity gradually decreases to 0. It prevents

other idle member nodes from becoming an active member node. At the beginning, the number of idle member nodes that are stimulated to become active 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. 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 δ_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 Fig. 9(b). A sink selects the predetermined number N of source nodes based on the algorithm explained in section 6.2. Then, it sends reinforcement messages to those nodes. By receiving a reinforcement message, R_{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, Fig. 10(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 δ 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 Fig. 10(b), directed diffusion also benefits from

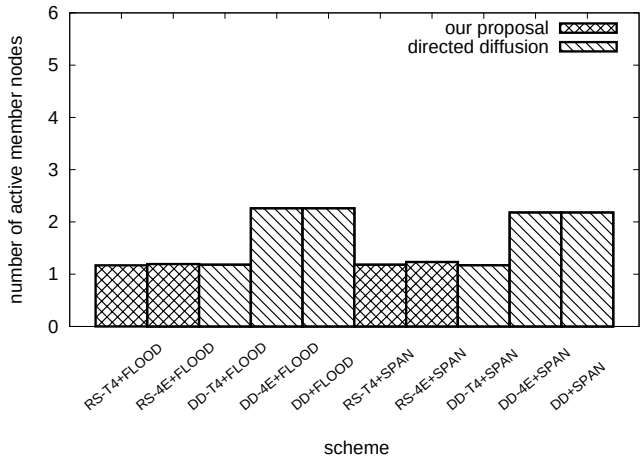
SPAN. Comparing the proposal and directed diffusion, the number of relay nodes is similar, since the number of active member nodes and the number of active source nodes are the same.

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.

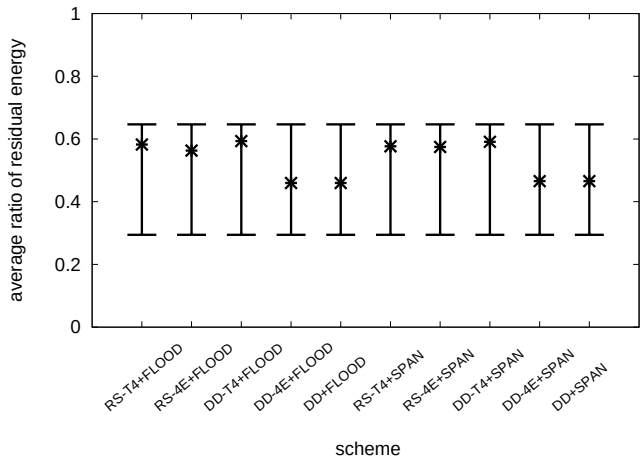
6.5 Evaluation of robustness against parameter setting

We discuss advantages of the self-organization based proposal over the deterministic and complicated rule-based directed diffusion in regard to the robustness against 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 Fig. 8. The requested number N of active source nodes per application is set at 1 and parameter δ_i of the proposal is set at 0.1. To make nodes heterogenous 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 [28]. To confirm this, we changed m and n in Eq. (5) from 3 (Table 6) to 6. With larger m and n, it becomes more difficult for a member node to become an active member node and the sufficient number of active member nodes might not be achieved. To examine the robustness of a decision making algorithm against parameter setting, we also change thresholds T_{share} and T_{energy} from 0.5 in Table 6 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 $(3 - 2)/3 \simeq 0.33$ because in reinforcement of each sink a node with most sharing its device is requested from three



(a) Number of active member nodes



(b) Residual energy

Figure 11: Rubustness of our proposal against parameter setting

applications ($M_{dd} = 3$) and has already provided two applications ($N_{dd} = 2$) with the device. As a result, all source nodes have the same priority of 4 in Table 5. That is, directed diffusion cannot take into account the heterogeneity of nodes in device assignment for inappropriate parameter setting for conditions of a WSAN.

Simulation results averaged over 100 runs are depicted in Fig. 11. We considered 10 different schemes in the experiments. Their names are in the form of 'scheme-parameter+ dissemination'. First, 'scheme' is either of RS or DD, where RS corresponds to our proposal and DD is directed diffusion. Second, 'parameter' is either of 'T4' or '4E'. 'T4' uses parameters summarized in Table 6, whereas '4E' uses parameters in section 6.5. Finally,

'dissemination' corresponds to a method used for message dissemination and it is either of 'FLOOD' or 'SPAN'. 'DD-FLOOD' is the original directed diffusion, to which no extension is applied. 'DD-SPAN' uses SPAN instead of flooding. In Fig. 11(a), the x-axis indicates schemes and the y-axis is the average number of active member nodes or active sources at 20000 s. In Fig. 11(b), the x-axis is the same as Fig. 11(a) and 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 of each bar indicates the maximum value of member nodes. The bottom of each bar indicates the minimum value of member nodes.

As shown in Fig. 11(a), from a view point of the number of active member nodes, the change of parameter setting does not strong impact on our proposal. On the contrary, it impacts on directed diffusion and the number of active sources increases. This is because, as mentioned above, thresholds in reinforcement rule is not appropriate to the current condition and all sources are categorized into one. As a result, similarly to DD-FLOOD/SPAN, a source nearer a sink is more likely to be selected as an active source. In this simulation, each sink is separately placed. Therefore, the number of active sources in DD-4E-SPAN/FLOOD are more than the number of active sources in DD-T4-SPAN/FLOOD. Finally, Fig. 11(b) shows RS-T4+FLOOD/SPAN can construct applications from nodes with higher residual energy similarly to DD-T4+FLOOD/SPAN.

From the results, in appropriate parameter setting, our proposal can achieve comparable performance with deterministic and complicated rule-based directed diffusion. Even if parameter setting is changed, its impact is less sensitive than directed diffusion. In general, in a rule-based mechanism like directed diffusion, it is difficult to draw up an appropriate rule as the number of metrics we have to take into account increases and decide on accurate parameter setting such as priorities and thresholds described in section 6.2. In addition, based on the above-mentioned simulation results, if parameters are not appropriate, it might lead to the decrease in the performance. On the contrary, our proposal can achieve efficient device assignment with very simple mechanism described in section 5.6 and this leads to the robustness against parameter setting.

7 Conclusion and future work

In this thesis, we proposed a mechanism for self-organizing device assignment mechanism. Results of simulation support the proposal, but there still remains room for further evaluation and improvement. First we need to consider how to solve actuator contention among processes with the same priority in our proposal. Evaluation from a viewpoint of robustness, scalability, and adaptability is one of future work as well.

Acknowledgments

I gratefully acknowledge the continuous support, tremendous advice, and encouragement from my supervisor, Professor Masayuki Murata of Osaka University. I sincerely appreciate Professor Naoki Wakamiya of Osaka University. All of my work would not be achieved without his support. He gave me invaluable help on my research experiments. And also I would express my sincere appreciation to Professors Koso Murakami, Makoto Imase, Teruo Higashino, and Hirotaka Nakano of Osaka University, for their invaluable advice. I am also grateful to Associate Professor Shin'ichi Arakawa, Assistant Professor Yuichi Oshita of Osaka University. They gave me helpful comments and feedback. Finally, I truly thank my friends and colleagues in the Department of Information Networking, Graduate School of Information Science and Technology of Osaka University for their support.

References

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

[2] F. Xia, Y. Tian, Y. Li, and Y. Sung, "Wireless sensor/actuator network design for mobile control applications," *Sensors*, vol. 7, pp. 2157–2173, June 2007.

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

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

[5] H. Lim, M. Iqbal, and T. Ng, "A virtualization framework for heterogeneous sensor network platforms," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, pp. 319–320, Nov. 2009.

[6] N. Mohamed and J. Al-Jaroodi, "A survey on service-oriented middleware for wireless sensor networks," *Service Oriented Computing and Applications*, vol. 5, pp. 71–85, June 2011.

[7] Y. Yu, L. J. Rittle, V. Bhandari, and J. B. LeBrun, "Supporting concurrent applications in wireless sensor networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems*, pp. 139–152, Oct. 2006.

[8] A. Majeed and T. Zia, "Multi-set architecture for multi-applications running on wireless sensor networks," in *Proceedings of the International Conference on Advanced Information Networking and Applications Workshops*, pp. 299–304, Apr. 2010.

[9] 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*, vol. 8, pp. 7930–7950, Dec. 2008.

[10] H. M. N. D. Bandara, A. P. Jayasumana, and T. H. Illangasekare, "Cluster tree based self-organization of virtual sensor networks," in *Proceedings of the International Workshops on Wireless Mesh and Sensor Networks*, pp. 1–6, Nov. 2008.

[11] C. Frank and K. Romer, "Algorithms for generic role assignment in wireless sensor networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems*, pp. 230–242, Oct. 2005.

[12] 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*, vol. 8, pp. 481–494, Sept. 2002.

[13] 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 *Proceedings of the International Conference on Biocomputing and Emergent Computation*, pp. 36–45, Jan. 1997.

[14] P. del Cid Garcia, D. Hughes, S. Michiels, and W. Joosen, "Middleware for resource sharing in multi-purpose wireless sensor networks," in *Proceedings of the International Conference on Networked Embedded Systems for Enterprise Applications*, pp. 20–28, Nov. 2010.

[15] C. Shin and W. Woo, "Service conflict management framework for multi-user inhabited smart home," *Journal of Universal Computer Science*, vol. 15, pp. 2330–2352, Dec. 2009.

[16] S. Baek, H. Lee, S. Lim, and J. Huh, "Managing mechanism for service compatibility and interaction issues in context-aware ubiquitous home," *IEEE Transactions on Consumer Electronics*, vol. 51, pp. 524–528, May 2005.

[17] M. Kolberg, E. Magill, and M. Wilson, "Compatibility issues between services supporting networked appliances," *IEEE Communications Magazine*, vol. 41, pp. 136–147, Nov. 2003.

[18] K. Gill, S. Yang, F. Yao, and X. Lu, "A zigbee-based home automation system," *IEEE Transactions on Consumer Electronics*, vol. 55, pp. 422–430, May 2009.

[19] C. Gomez and J. Paradells, "Wireless home automation networks: A survey of architectures and technologies," *IEEE Communications Magazine*, vol. 48, pp. 92–101, June 2010.

[20] J. Bruck, J. Gao, and A. Jiang, "Localization and routing in sensor networks by local angle information," *ACM Transactions on Sensor Networks*, vol. 5, pp. 7:1–7:31, Feb. 2009.

[21] Y. Ko and N. Vaidya, "Flooding-based geocasting protocols for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 7, pp. 471–480, Dec. 2002.

[22] N. Hoeller, C. Reinke, J. Neumann, S. Groppe, D. Boeckmann, and V. Linnemann, "Efficient XML usage within wireless sensor networks," in *Proceedings of International Conference on Wireless Internet*, pp. 1–10, Oct. 2008.

[23] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford University Press, 1999.

[24] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the International Conference on Mobile Computing and Networking*, pp. 56–67, Aug. 2000.

[25] T. Iwai, N. Wakamiya, and M. Murata, "Proposal for dynamic organization of service networks over a wireless sensor and actuator network," in *Proceedings of the International Conference on Ambient Systems, networks and Technologies*, vol. 5, pp. 240–247, Sept. 2011.

[26] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference*, pp. 319–324, June 2001.

[27] Crossbow Technology, "MICAz Datasheet." http://www.xbow.com.

[28] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Media streaming on P2P networks with bio-inspired cache replacement algorithm," in *Proceedings of the International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, vol. 3141, pp. 380–395, Jan. 2004.