社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

無線センサ・アクチュエータネットワークにおける複数サービス間での資源共有を実現する自己組織型デバイス割当機構の提案と評価

岩井　卓也†　　若宮　直紀†　　村田　正幸†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5
E-mail: †{t-iwai,wakamiya,murata}@ist.osaka-u.ac.jp

あらまし　多目的型の無線センサ・アクチュエータネットワークは IoT や M2M の基盤技術として注目を集めているが，遍在する様々なデバイスを活用し，多様なアプリケーションを実現するためには，利用可能なデバイスをアプリケーション間で効果的に共有するための機構が必要となる．本稿では，ノードの自律的な判断によって，ネットワーク全体として十分な数のデバイスが適切にアプリケーションに割り当てられる，自己組織型のデバイス割当機構を提案する．シミュレーション評価を通して，提案手法により，集中型の最適割当を行う手法と同等のデバイス共有が達成できるだけでなく，パラメータ設定の誤差に対して頑健性が高いことを示した．
キーワード　センサ・アクチュエータネットワーク，デバイス割当，資源共有，反応閾値モデル

A self-organization based device assignment mechanism for cooperative resource sharing in a wireless sensor and actuator network

Takuya IWAI†, Naoki WAKAMIYA†, and Masayuki MURATA†

† Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita 565-09871, Japan 565–0871 Japan
E-mail: †{t-iwai,wakamiya,murata}@ist.osaka-u.ac.jp

Abstract　A multi-purpose wireless sensor and actuator network is a promising technology to accommodate a variety of IoT and M2M applications without wasting and depleting network resources and node resources. For this purpose, we need a mechanism to effectively share available resources among concurrent applications while taking into account application requirements and resources. 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. 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.
Key words　sensor and actuator network, device assignment, resource sharing, response threshold model

1. ま え が き

環境に配置されたセンサによって取得された情報にもとづいてアクチュエータを制御することにより，環境制御や機器制御を行う無線センサ・アクチュエータネットワークは近年急速に注目が集まっている IoT [1] や M2M [2] の基盤技術として注目されている [3]．無線センサ・アクチュエータネットワークにおいて，配置されたノードの備えるデバイス，すなわち，センサやアクチュエータを組み合わせることにより，様々なアプリケーションを実現することができるが，これまでは，アプリケーションごとに，適切なデバイスを搭載するノードを，あらかじめ定められた適切な場所に配置し，ネットワークを構成していた．その結果，環境内に多数のノードが冗長配置され，それぞれがアプリケーション毎に独立して通信することとなり，通信帯域が逼迫するという問題が生じている．また，通信品質やアプリケーション要求の変化を予測して最適にノードを配置することは困難であることからも，環境内に存在する利用可能なデバイスを必要に応じて組み合わせ，アプリケーションを実現することが望ましい．

例えば，文献 [4] では，アーキテクチャや機能の異なる様々なデバイスやノードを統一的に取り扱うための API を提供する TinySOA を提案しており，これを用いることで，アプリケーション開発者はデバイスやノードの差異を意識することなくプログラムを作成することができる．アプリケーションへの動的な

デバイス割当については，文献 [5] において，if-then-else ルールによって記述されたデバイス割当要求のメッセージをネットワーク内に拡散し，これを受け取ったノードが自らの役割を決定することによって，アプリケーション要求に応じたノードの選出を行う Generic Role Assignment が提案されている．また，文献 [6] で提案されている directed diffusion では，シンクノードと呼ばれるノードがネットワーク内に要求メッセージを拡散すると，その要求を満たせるノードがシンクにセンサデータを返信し，シンクノードが受信結果に基づいてアプリケーションで利用するデータを選出する．これらの手法によって，アプリケーション要求に応じたデバイス割当を行うことができるが，いずれもあらかじめ定められたルールを用いる決定論的な手法であるため，デバイスの共有度合い，ノードの残余電力など，考慮すべき条件が多くなるに従い，適切な割当ルールの設定が困難になるという問題がある．

我々の研究グループでは，社会性昆虫の群れにおける自己組織的な分業の仕組みの数理モデルである反応閾値モデル [7] を用いることで，複雑なルールを必要しない適応的なデバイス割当機構を提案している [8]．その有効性はシミュレーション評価によって検証済みであるが，メッセージを中継するノードをアプリケーション間で共有できておらず，また，従来手法との比較評価も不十分であった．そこで本稿では，SPAN [9] を用いることにより，アプリケーション間でのメッセージ中継ノードの共有を図るとともに，デバイス割当アルゴリズムを簡単化する改良を行った．また，制御パラメータ設定に対する頑健性について新たに評価した．本稿の構成は以下の通りである．まず，2. 章において，我々の提案する反応閾値モデルを用いたデバイス割当機構について述べる．次に，3. 章において，directed diffusion との比較評価により提案手法の有効性を示す．最後に 4. 章で本稿のまとめと今後の課題を述べる．

2. 自己組織型デバイス割当機構

提案するデバイス割当機構では，反応閾値モデルに基づいてそれぞれのノードがアプリケーションにデバイスを割り当てるか否かを自律的に決定し，無線センサ・アクチュエータネットワーク全体としてアプリケーション要求を満足するデバイス割当が行われる．

以降では，ホームゲートウェイやアプリケーションサーバなど，無線センサ・アクチュエータネットワークに対してデバイス割当の要求を出すノードをリクエストノード，要求にあったデバイス（センサ，アクチュエータ）を有するノードをメンバノード，ノード間のメッセージ中継を行うノードをリレーノードと呼ぶ．また，あるアプリケーションに対してデバイスを提供しているメンバノードを，そのアプリケーションのアクティブメンバノードと呼び，デバイスを提供していないノードをアイドルメンバノードと呼ぶ．さらに，アプリケーションに割り当てられているデバイスをアクティブデバイス，割り当てられていないものをアイドルデバイスと呼ぶ．したがって，アクティブメンバノードは 1 つ以上のアクティブデバイスを有し，アイドルメンバノードのデバイスはすべてアイドルデバイスである．

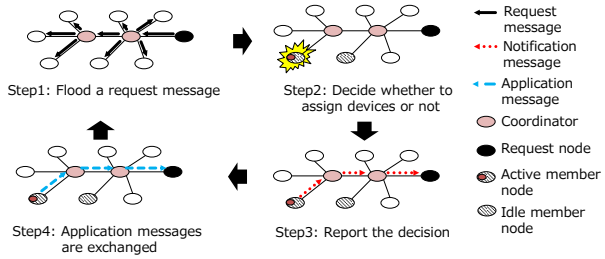


図 1　デバイス割当機構の動作

提案手法では，センサやアクチュエータなどのデバイスには，動作モードが定められているものとする．デバイス j の取り得る動作モードは集合 $\mathbf{O_j} = \{\mathrm{mode}_1, \cdots, \mathrm{mode}_{n-1}, \mathrm{mode}_n\}$ で表され，同時に複数の動作モードをとることは出来ない．あるアプリケーションの望む動作モードにあるデバイスは，そのアプリケーションに割り当てられているものとみなす．なお，どのアプリケーションにも割り当てられていないデバイスの動作モードは mode_n である．例えば，センサ a の動作モードの集合は $\mathbf{O_a} = \{\mathrm{sensing}, \mathrm{sleep}\}$ であり，ON/OFF スイッチ b の動作モードの集合は $\mathbf{O_b} = \{\mathrm{on}, \mathrm{off}\}$ である．

2.1 デバイス割当機構の動作

デバイス割当機構の動作を図 1 に示す．アプリケーション i のリクエストノードは，アプリケーション i の要求を記述したリクエストメッセージをすべてのノードに対して I_{demand} (> 0) 毎に送信する．リクエストメッセージのヘッダには，アプリケーションの識別子 i，リクエストメッセージのシーケンス番号 k，アプリケーションの要求強度 $s_i(t)$ (≥ 0) が設定される．要求強度については後述する．また，リクエストメッセージの本文には，アプリケーションが要求するデバイスの種類や動作モードなどが設定される．例えば地点 (x, y) の温度情報を I_{data} 毎に収集するアプリケーションの場合には，リクエストメッセージの本文は $(\mathrm{thermal\ sensor}, \mathrm{sensing}, (x, y), I_{data})$ となる．リクエストメッセージは，SPAN [9] のフォワーディングバックボーンを利用することでネットワーク内に拡散される．

ノードは，アプリケーションに関する情報 $(i, j, m, k, h, s_i(t))$ の集合 $\mathbf{S}$ を保持している．ここで，i はアプリケーションの識別子を，j はアプリケーションが求めるデバイスの識別子を，m はデバイスの動作モードを，k は受信したリクエストメッセージのシーケンス番号を，h は受信したリクエストメッセージの転送元である隣接ノードの識別子を，$s_i(t)$ は t 回目に受信したリクエストメッセージに設定されたアプリケーションの要求強度を表す．ノードは，アプリケーション i のリクエストメッセージを受信すると，アプリケーション i の情報が集合 $\mathbf{S}$ に含まれていない場合には，新しく $(i, j, m, k, h, s_i(t))$ を作成し，集合 $\mathbf{S}$ に加える．一方で，すでに情報が集合 $\mathbf{S}$ に含まれている場合には，受信したリクエストメッセージを用いて情報を更新する．ノードは，アプリケーションの要求を満たすことのできるデバイスを有していれば，そのアプリケーションのメンバノードになる．ノードは，メンバノードとなっているアプリ

ケーション i について，デバイス j を提供するか否かを表す論理値 $X_{i,j}$ の集合 $\mathbf{X}$ と，閾値 $\theta_{i,j}$ $(0 < \theta_{i,j} \leqq \theta_{max})$ の集合 $\mathbf{\Theta}$ を生成，保持している．

リクエストメッセージを受信したメンバノードは，後述するデバイス割当アルゴリズムによってアプリケーション i にデバイス j を割り当てるか否かを決定する．アプリケーション i にデバイス j を割り当てる場合には，アクティブメンバノードとなり，デバイス j の動作モードをアプリケーション i が求める動作モード m に変更し，隣接ノード h を介してリクエストノードに通知メッセージを送信する．通知メッセージを受信したノードはさらにその隣接ノード h に通知メッセージを転送する．なお，ノードが最後にアプリケーション i のリクエストメッセージを受信してから E_i $(> I_{demand})$ の間に一度もリクエストメッセージを受信しない場合には，ノードは集合 $\mathbf{S}$，$\mathbf{X}$，$\mathbf{\Theta}$ からアプリケーション i に関する情報を削除する．

リクエストノードは，t 回目のリクエストメッセージに含める要求強度 $s_i(t)$ を，$t-1$ 回目のリクエストメッセージ送信後から t 回目のリクエストメッセージ送信までにアクティブメンバノードから受信した通知メッセージの数 $N_i(t)$ を用いて次式により計算する．なお，要求強度の初期値 $s_i(0)$ は 0 とする．

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

ここで，係数 δ_i $(\geqq 0)$ はアプリケーション i の要求強度の増加係数を表す．アクティブメンバノードの数が係数 δ_i よりも小さい時には，要求強度は増加し，メンバノードがアクティブになることを促すこととなる．一方で，係数 δ_i よりも大きい時には，要求強度は減少し，メンバノードが新たにアクティブになることを抑える．このように，提案手法では，係数 δ_i を用いてアクティブメンバノードの数を調整することができる．

2.2 反応閾値モデルを用いたデバイス割当アルゴリズム

社会性昆虫の群れにおいて，それぞれの個体は仕事をするか否かを自律的に決定し，結果として仕事をしている個体，仕事をしていない個体からなる 2 つのグループを形成する．反応閾値モデル [7] はこの分業の仕組みを表す数理モデルであり，提案手法は反応閾値モデルにおける仕事をデバイス割当とみなすことで自己組織的なデバイス割当を実現する．

アプリケーション i にデバイス j を割り当てていないアイドルメンバノード $(X_{i,j} = false)$ が，t 回目のリクエストメッセージを受信した時にアクティブメンバノードになる確率は次式で与えられる．

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

ここで，$s_i(t)$ $(\geqq 0)$ は t 回目のリクエストメッセージに含まれるアプリケーション i の要求強度であり，$s_i(t)$ が大きいほどアイドルメンバノードはよりアクティブメンバノードになりやすい．また，$\theta_{i,j}$ $(0 < \theta_{i,j} \leqq \theta_{max})$ は，アプリケーション i に対するデバイス j に関する閾値であり，$\theta_{i,j}$ が小さいほどアクティブメンバノードになりやすい．変数 A_j $(\geqq 1)$ は，ノードの電力残存率およびデバイス j の共有度合いに応じたデバイス割当を実現するための変数である．変数 A_j の定義は 2.3 章において述べる．

一方，アプリケーション i にデバイス j を割り当てているアクティブメンバノード $(X_{i,j} = true)$ が，アイドルメンバノードになる確率は次式で与えられる．

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

ここで，p_j $(0 \leqq p_j \leqq 1)$ はデバイス毎に定められた定数である．アクティブメンバノードが確率 p_j でアイドルメンバノードになることで，冗長なメンバノードがアクティブになることを防ぐと同時に，メンバノード間でアクティブメンバノードを交代するようになり負荷分散が図られる．

デバイス割当アルゴリズムは強化学習の仕組みを有しており，リクエストメッセージの受信時に式 (2) または式 (3) を評価する毎に，ノードの閾値 $\theta_{i,j}$ を次式に従って調整する．

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

ここで，係数 ξ_j (> 0) と係数 φ_j (> 0) は，反応閾値モデルによる役割分担の分化の速さと可塑性に影響するパラメータである．閾値調整により，一旦アクティブメンバノードになったノードはよりアクティブメンバノードになりやすくなり，たまたまアイドルメンバノードになっても再度アクティブメンバノードに戻りやすくなる．また，一旦アイドルメンバノードになったノードは，アイドルメンバノードに留まりやすくなる．

2.3 デバイスの共有，消費電力の均一化のための変数 A_j

メンバノードのデバイスをより多くのアプリケーション間で共有するとともに，メンバノード間での電力消費を均一化するための変数 A_j $(\geqq 1)$ を，次式により定義する．

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

右辺第 1 項はアプリケーション間でのデバイスの共有を促進するための項である．変数 S_j $(\geqq 1)$ は，デバイス j に関してメンバノードであるアプリケーションの数であり，$S_j = |\mathbf{X}|$ で与えられる．変数 F_j $(\geqq 0)$ は，ノードがアクティブメンバノードであるアプリケーションの数を表し，$F_j = |\{X_{i,j} \in \mathbf{X} \mid X_{i,j} = true\}|$ として表される．右辺第 2 項は，ノードの電力残存率の均一化を図るための項である．なお，変数 P_{res} (> 0) はノードの残余電力量を表し，P_{full} $(> P_{res})$ はバッテリ容量を表す．右辺第 3 項は，変数 A_j を正規化するためのものである．より多くのアプリケーションのアクティブメンバノードであり，かつ，電力残存率のより大きいノードほど変数 A_j が小さくなる．

3. シミュレーション評価

本章では，directed diffusion [6] との比較評価を通して，提案するデバイス割当機構の有効性を検証する．

3.1 directed diffusion の拡張と reinforcement ルール

directed diffusion では，シンクノードは，アプリケーションに必要なセンサデータを取得するため，観測対象や条件，センサデータの報告頻度などを記述した interest メッセージをネットワークに拡散する．なお，報告頻度はアプリケーションの求め

表 1　directed diffusion におけるソースノードの優先度

		R_{energy}	
		$\geqq T_{energy}$	$< T_{energy}$
R_{share}	$\leqq T_{share}$	1	3
	$> T_{share}$	2	4

る頻度より低く設定される．interest メッセージを受信したノードは，転送元の隣接ノードを記憶し（この情報を gradient と呼ぶ），他の隣接ノードに転送する．interest メッセージを受信したノードのうち，要求に応えられるノードはソースノードと呼ばれ，指定された条件でシンクノードに対してセンサデータを送信する．これを exploratory data メッセージと呼ぶ．

exploratory data メッセージを受信したシンクノードは，reinforcement ルールと呼ばれるルールによって，例えばメッセージの受信順や，センサデータの確度などに基づいて，アプリケーションにとって望ましい exploratory data メッセージを選出し，その転送元である隣接ノードに，報告頻度をアプリケーションの望む頻度に設定した reinforcement メッセージと呼ばれる interest メッセージを送信する．reinforcement メッセージはシンクノードと同じ reinforcement ルールに基づいて選ばれた隣接ノードを経由してソースノードに到達し，アプリケーションの望む頻度で data メッセージが送信されるようになる．interest メッセージは定期的に送信され，reinforcement メッセージを受信したソースノードからはアプリケーションの望む頻度でセンサデータを受信するとともに，他のソースノードからも低い頻度でセンサデータを受信する．

上記のように，directed diffusion では望ましいセンサデータを選択的に受信することができるが，directed diffusion の interest メッセージや data メッセージ，gradient には，ソースノードやシンクノードの情報が含まれないため，reinforcement メッセージが常に同じソースノードに到達するとは限らない．そのため，デバイスの共有度や残余電力などの情報を data メッセージに付与したとしても，それらを考慮したソースノード選出を行うことができない．そこで，本稿では，data メッセージに，ソースノードとシンクノードの識別子，ソースノードの残余電力量 P_{res} とバッテリ容量 P_{full}，さらに受信した interest メッセージのうちそのノードが要求に応えられるものの種類数 M_{dd} と，受信した reinforcement メッセージの種類数 N_{dd} を記述する．M_{dd} は式 (5) における S_j に，N_{dd} は F_j にそれぞれ相当する．また，gradient と reinforcement メッセージにソースノードとシンクノードの識別子を設定する．これにより，特定のソースノードとシンクノード間での reinforcement が行えるようになる．

上記の拡張の下で，デバイスの共有度とノードの残余電力を考慮して N 台のソースを選出する reinforcement ルールを定める．シンクは，複数の exploratory data メッセージを受信すると，ソース毎に電力残存率 $R_{energy} = P_{res}/P_{full}$，および，デバイスの共有度合い $R_{share} = (M_{dd} - N_{dd})/M_{dd}$ を計算する．次に，表 1 に示すように，閾値 T_{energy} および閾値 T_{share} との大小関係にもとづいてソースノードに優先度を定め，値が小さく優先度の高いソースノードから順に N 台に対して reinforcement メッセージを送信する．

表 2　シミュレーションのパラメータ設定

Notation	Description	set A	set B
p_j	仕事を辞める確率（式 (3)）	0.01	0.01
ξ_j	分化の速度に関する係数（式 (4)）	0.1	0.1
φ_j	分化の可塑性に関する係数（式 (4)）	1	1
m	共有度に関する係数（式 (5)）	3	6
n	電力残存率に関する係数（式 (5)）	3	6
L	exploratory メッセージの送信間隔	0.5	0.5
T_{share}	デバイスの共有度の閾値	0.5	0.1
T_{energy}	電力残存率の閾値	0.5	0.9
I_{demand}	リクエストメッセージの送信間隔	10	10

3.2　シミュレーションの設定

シミュレーションには OMNet++ [10] を用いる．25 m × 25 m の領域に，5 台のノード A ～ E をそれぞれ座標 $(0,0)$，$(12.5,0)$，$(0,12.5)$，$(25,0)$，$(0,25)$ に，20 台のノードをランダムに配置する．すべてのノードは単 3 電池 2 本により稼働し，MICAz のデータシートに基づいて，リスニング時，受信時に 18.8 mA を，送信時に 17.4 mA を，スリープ時に 0.021 μA をそれぞれ消費するものとする．リクエストメッセージ，interest メッセージ，reinforcement メッセージのサイズは MAC 以下のヘッダを除いて 36 byte とし，通知メッセージ，data メッセージは 64 byte とする．表 2 にシミュレーションにおけるパラメータ設定を示す．set A は事前評価によって設定したパラメータであり，set B はパラメータ設定に対するロバスト性の評価に用いる．すべてのノードは半径 15 m の正円内をセンシングするセンサを 1 つ有しているものとし，デバイスの電力消費は無視できるほど小さいものとする．また，すべてのノードは IEEE 802.15.4 のノン・ビーコンモードによって 15 m 以内のノードと通信できるものとする．

3.3　アプリケーション間のデバイス共有

本節では，ノードの自律的な判断に基づいてデバイス割当を行う提案手法と，集中型で決定論的なルールに基づいてデバイス割当を行う directed diffusion とを比較し，提案手法によって効果的なデバイス割当が行えることを確認する．

シミュレーションでは，ノード A ～ E をそれぞれ異なるアプリケーションのリクエストノードまたはシンクノードとする．いずれのアプリケーションも，地点 $(25,25)$ のセンシングを要求するものとする．したがって，地点 $(25,25)$ から距離 15 m 以内に位置するノードが割り当て対象となる．リクエストノードまたはシンクノードを 1 台～ 5 台，1 つのアプリケーションが求めるデバイスの数を 1 台（$\delta = 0.1$，$N = 1$），2 台（$\delta = 1.1$，$N = 2$），3 台（$\delta = 2.1$，$N = 3$）の場合の計 15 通りのシミュレーションをそれぞれ 100 回行い，以降のグラフではそれらの平均を示す．図 2 に，20,000 s 時点のアクティブメンバノード，または，アクティブソースノードの数を，図 3 にリレーノードの数をそれぞれ示す．なお，directed diffusion においては，reinforcement メッセージを受信したソースノードをアクティ

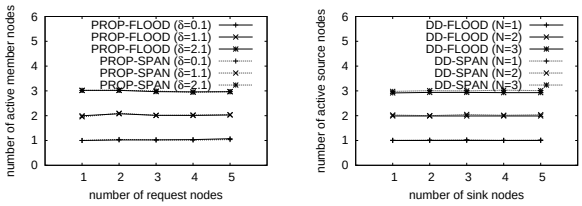
ブソースノード，シンクノードとアクティブソースノードとの間のメッセージ中継を行うノードをリレーノードと呼ぶ．図には，フラッディングによりリクエストメッセージや interest メッセージを拡散する場合（FLOOD) と，SPAN を組み合わせる場合（SPAN）をそれぞれ示している．

図 2(a) より，リクエストノードの数によらず，提案手法におけるアクティブメンバノードの数は一定であり，アプリケーション毎に割当を要求するデバイス数とほぼ等しいことがわかる．これは，提案手法により，アプリケーション間でアクティブメンバノードを効果的に共有できていることを表している．アクティブメンバノードが 0 台の時には，要求強度が係数 δ により次第に増加する．その結果，式 (2) における確率 $P(X_{i,j} = false \rightarrow X_{i,j} = true)$ が高くなるため，いくつかのアイドルメンバノードがアクティブになる．アクティブメンバノードが通知メッセージを送信することで，要求強度が次第に減少し，他のアイドルメンバノードがアクティブになりにくくなる．また，アクティブメンバノードは確率 p_j でアイドルになるため，十分な数のアクティブメンバノードがいなくなると，再び要求強度が増加するようになる．これを繰り返すことで，いずれアクティブメンバノードの数が一定に保たれるようになる．さらに，強化学習の仕組みにより，特定のノードがアクティブメンバノードになり，安定したデバイス割当が実現される．ただし，電力残余率が低下すると確率 $P(X_{i,j} = false \rightarrow X_{i,j} = true)$ が小さくなるため，アイドルメンバノードとの交代が促される．

アクティブメンバノード数は要求強度の増加係数 δ_i によって定まり，本シミュレーションにおいては，係数 δ_i を 0.1，1.1，2.1 に設定するとそれぞれ 1 台，2 台，3 台のノードがアクティブとなる．また，図 2(b) より，directed diffusion においても，シンクノード数によらずアクティブソースノード数がアプリケーション毎の割当数 N にほぼ等しいことが分かる．これは，あるシンクノードから reinforcement メッセージを受信したソースノードでは R_{share} が減少し，表 1 において優先度が高くなり，他のシンクノードにも選ばれやすくなるためである．

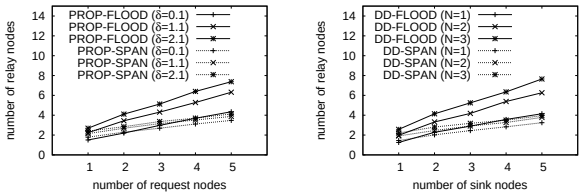
図 3(a) より，提案手法においては，リクエストノード数と割当要求デバイス数のいずれも 1 の場合を除き，SPAN を用いることによってリレーノードの数が少なくなるため，ネットワーク資源，ノード資源の節約が期待できる．SPAN により構成されるトポロジはすべてのアプリケーション間で共通である．このため，リクエストノードとメンバノード間の経路は必ずしも最短経路ではないものの，アプリケーション間でリレーノードや経路が共有されやすい．一方で，フラッディングを用いた場合には，経路は共有されにくいものの，最短になる傾向が高い．そのため，アプリケーション毎のデバイス数が 1 の場合には，フラッディングにより個別に独立した最短経路を構築した方がリレーノード数が少なくなっている．また，図 3(b) より，提案手法と同様に，directed diffusion においても SPAN を用いることでリレーノード数が削減されることがわかる．

以上より，同一条件下では，自己組織的なデバイス割当を行う提案手法によって，集中型で決定論的なデバイス割当を行う



(a) 提案手法　(b) directed diffusion

図 2　アクティブメンバ／ソースノードの数



(a) 提案手法　(b) directed diffusion

図 3　リレーノードの数

directed diffusion と同等に効果的なデバイス割当が行えることが明らかとなった．

3.4　パラメータ設定に対するロバスト性

一般的に生物モデルに基づく制御機構はパラメータ設定の影響を受けにくく，厳密なパラメータチューニングが不要であるという特徴がある．そこで，デバイス割当の決定に影響を持つパラメータである指数 m と n，および閾値 T_{share} と T_{energy} を表 2 の set B のように変更し，シミュレーションを行った．これらの設定により，アクティブメンバノード，または，アクティブソースノードになるための条件が set A よりも厳しくなる．シミュレーションでは，ノード A 〜 C を地点 $(25, 25)$ をセンシングするアプリケーションのリクエストノードまたはシンクノードとする．ノードの電力残存率は 25% から 80% の範囲でランダムに設定した．このシミュレーション設定では，R_{share} の最小値は $(3-2)/3 \simeq 0.33$ であり，シミュレーション開始時の R_{energy} は 0.25 から 0.80 である．従って，すべてのソースノードは等しく表 1 の優先度 4 に 分類されるため，directed diffusion では共有度合いや残余電力を考慮したデバイス割当が行えない．

図 4(a) に，各手法における 20,000 s 時点のアクティブメンバノード，または，アクティブソースノードの数を示す．なお，括弧内は使用したパラメータ設定を表し，DD-FLOOD や DD-SPAN は残余電力やデバイス共有度合いを考慮しない reinforcement を行った場合の結果である．また，図 4(b) に，メンバノードの電力残存率の平均を示している．それぞれの上辺，下辺のそれぞれはメンバノードのうちの最大および最小電力残存率を，点は平均電力残存率を示す．

図 4(a) に示されるように，提案手法はパラメータ設定の影響をほとんど受けず，3 つのアプリケーションに対しておよそ 1 台のノードのみがデバイスを割り当てていることが分かる．一方で，directed diffusion においては，パラメータ設定 set A よりも set B でのアクティブソースノード数が多く，残余電力やデバイス共有度合いを考慮しない場合と同等の結果となってい

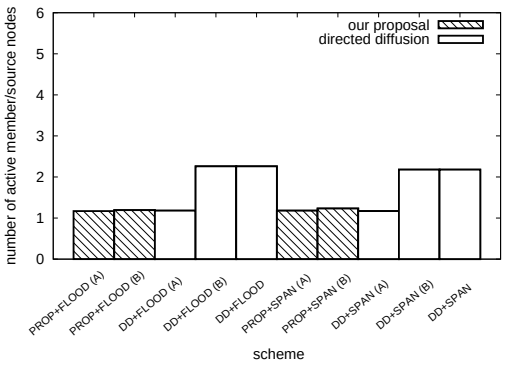
る．これは，不適切な閾値設定により，すべてのノードに同じ優先度が割り当てられたためである．また，図 4(b) が示すように，提案手法ではメンバノードの平均電力残存率が高いことから，残余電力の多いメンバノードがアクティブになっていると考えられる．一方，directed diffusion はパラメータ設定の影響を受け，set B では平均電力残存率が低くなっている．

以上のシミュレーション結果より，提案手法は directed diffusion と比べてパラメータ設定の誤差による影響を受けにくいことがわかった．表 1 における適切な閾値設定は，同時動作するアプリケーション数やノードの電力残存率の分布を考慮して決定する必要があり，その設定は困難であることから，多目的型無線センサ・アクチュエータネットワークにおけるデバイス割当には，パラメータ設定の影響を受けにくく，また，動的なパラメータ調整も不要な提案手法が効果的であると言える．

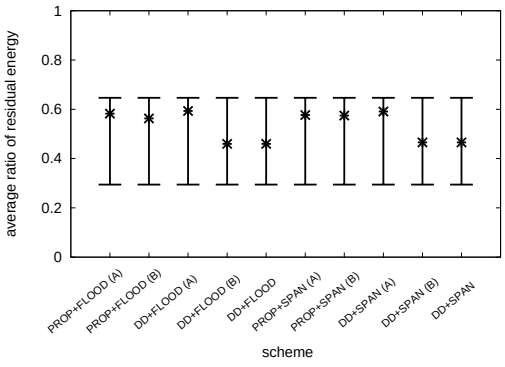
4. おわりに

本稿では，反応閾値モデルを応用した自己組織型デバイス割当機構の改良を行い，複数のアプリケーション間でのリレーノードの共有を実現した．併せて，提案手法が既存手法と比べてパラメータ設定の誤差による影響を受けにくいことを確認した．なお，提案手法は，リクエストノードが要求強度の通知によってデバイス割当の制御を行う点において，集中型と見なすこともできるが，リクエストノードはアクティブメンバノード数だけを把握しており，ノード毎の特性については関知していない．

多目的型無線センサ・アクチュエータネットワークでのデバイス共有においては，複数のアプリケーションがアクチュエータに異なる動作モードを要求するアクチュエータ競合が問題となる．提案手法を適用すると，アクチュエータが割り当てられたアプリケーションの需要強度が減少する一方で，アクチュエータが割り当てられていないアプリケーションの需要強度が増加するため，アプリケーション間で交代でアクチュエータが割り当てられることになる．また，例えば侵入検知などの緊急性の高いアプリケーションの場合には，需要強度を高く設定することによって，他のアプリケーションよりも優先的にアクチュエータが割り当てられるように制御することも可能である．一方で，アクティブメンバノード数が安定し，ノードが分化するまでには，リクエストノードとメンバノードとの間のインタラクションが必要となるため，ノード故障や移動などの変化に対する応答性の問題があると考えられる．今後は，これらの検証と，手法の改良に取り組む．



(a) アクティブメンバ／ソースノードの数



(b) 残余電力

図 4　パラメータ設定に対するロバスト性

謝　辞

本研究の一部は，独立行政法人情報通信機構国際共同研究助成制度によるものである．ここに記して謝意を表す．

文　献

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. Johnson, “M2M: From mobile to embedded internet,” *IEEE Communications Magazine*, vol. 49, pp. 36–43, Apr. 2011.

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

[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] 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.

[6] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed diffusion for wireless sensor networking,” *IEEE/ACM Transactions on Networking*, vol. 11, pp. 2 – 16, Feb. 2003.

[7] 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.

[8] 岩井 卓也, 若宮 直紀 , 村田 正幸, “無線センサ・アクチュエータネットワークにおける動的なサービスネットワーク構成のための端末選出機構の提案,” **電子情報通信学会技術研究報告** (NS2010-219), pp. 323–328, Mar. 2011.

[9] 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.

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