Vol.2011-MPS-82 No.7
2011/3/7

# 無線センサネットワークでのノードの停止・追加を考慮した省電力データ収集手法

坂口 隼[†1] 勝間 亮[†1] 山内 由紀子[†1]
安本 慶一[†1] 伊藤 実[†1]

無線センサネットワーク（WSN）は電池駆動のノードで構成される．ノードはいずれ電池を使い果たし停止するが，データ収集を継続させるために，新たなノードを追加することが多い．そのため，ノードの停止や追加に応じて自律的に転送経路を再構成できることが望ましい．本稿では，データ転送の負荷分散と自律的な転送経路の再構成を実現する自己安定アルゴリズムを提案する．提案手法では，DAG を構成することで各ノードからシンクノードへの経路を複数構成する．さらに，ノードの停止によってセンシングデータがシンクノードに到達しないことを回避するために，各センシングデータのコピーを複数のノードに保持させ，それぞれ独立に転送する．

# Energy-Efficient Data Gathering Method for Wireless Sensor Networks Resilient to Crash and Addition of Nodes

Shun Sakaguchi, Ryo Katsuma, Yukiko Yamauchi,
Keiichi Yasumoto and Minoru Ito

A wireless sensor network (WSN) consists of battery-driven sensor nodes. Because nodes eventually exhaust their batteries and crash, new nodes may be added to continue sensing and gathering sensing data. Hence, autonomous adaptability is expected to data gathering protocols for WSNs. In this paper, we propose a self-stabilizing data gathering algorithm that considers load balancing and provides autonomous reconstruction of data gathering paths. The proposed algorithm constructs a DAG to have multiple data gathering paths. Additionally, to tolerate crash of relay nodes, the proposed method makes multiple nodes have copies of each sensing data and relays the copies through different paths.

## 1. まえがき

無線センサネットワーク（WSN）は多数のセンサノードが周辺環境から計測した情報を取得・収集するためのシステムである．各ノードは電池で駆動し，センシング機能と無線通信機能を備えている．本稿では，各ノードが計測した情報をシンクノードに収集するデータ収集型 WSN を対象とする．WSN でのデータ収集は，各ノードからシンクノードへのマルチホップ通信によって行う．ノードは頻繁に通信を行うと電池を使い果たし停止してしまう．ノードが停止するとデータの中継ができなくなり，より遠方のノードからのデータを収集できなくなる可能性がある．そのため，ノードへのデータ転送の負荷を分散することは，WSN の稼働時間を延長するために必須である．停止したノードが保持していた転送途中のデータが損失することも避けるべきである．また，WSN の稼働期間中にその地理領域の情報を漏れなく収集できることが望ましい．ノードが停止すれば，計測できない領域が発生する可能性がある．一方，WSN は多数のノードによって構成されるため，異なるノード同士で計測できる領域が重複することがある．計測に不要なノードを休止させ，電池の消耗を抑えることも WSN の稼働時間を延長させるために有用である．

WSN は広大な地理領域に適用する際，非常に多くのノードによって構成される．そのため，WSN の制御手法はスケーラビリティのある設計が望ましく，分散処理による制御が適している．また，停止したノードを補うために新たなノードが追加されることも多い．そのため，ノードの停止や追加に応じて自動で転送経路を再構成することが必要となる．自律適応性を持つ分散アルゴリズムとして，自己安定アルゴリズム[3] がある．自己安定アルゴリズムは，初期状況にかかわらず，システムがやがて目的のふるまいを行うことを保証する．つまり，ノードの故障やトポロジ変化の後の状況を初期状況と見なし，そこから有限時間以内にシステムが目的の振る舞いを行うことを保証する．

本稿では，WSN の稼働時間延長とノードの停止や追加への自律適応性の実現を目的とし，各ノードからシンクノードへの経路を構成するアルゴリズムとデータ収集アルゴリズムを提案する．提案手法では，WSN の稼働時間を延長するために，データ収集に不要なノードを休止させる．そして，起動しているノード上に転送経路となる DAG （Directed Acyclic Graph：非循環有向グラフ）を自己安定的に構成する．DAG 上でのデータ収集は，ノードの電池残量に応じた経路を選択し，データ転送を行う．このデータ収集アルゴリズムは，ノードの停止によるセンシングデータの損失を避けるために，各センシングデータを複数のノードに保持させつつデータ収集を行う．

これまで，自律適応性を持つ手法は存在したが，ノードの停止による収集途中のデータの損失を防ぐ手法は無かった．本稿では，自律適応性を備え，かつ，ノードの停止による収集中のデータの損失を防ぐ WSN を実現できることを示す．

†1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

Vol.2011-MPS-82 No.7
2011/3/7

## 2. 関連研究

WSN でのデータ収集には，シンクノードと，一部のノード，通信リンクのみから成るネットワーク（**データ収集トポロジ**と呼ぶ）を用いることが多い．データ収集トポロジに含まれないノードを休止させることで消費エネルギーを抑制する．このように WSN の稼働時間を延長する手法は，よく用いられている[2),4),5)]．

データ収集トポロジとしては，生成木がよく利用されている[1),2),4)]．この方法では，WSN 上にシンクノードを根ノードとする生成木を構成し，各ノードは生成木に沿って，データをシンクノードへ転送する．Burri らが提案した Dozer[1)] は，1 つの親に接続できる子の数を制限し，さらにシンクノードからの距離が小さい親を選択することで，データ転送の負荷を分散する生成木を構成する．しかし，子は自身の親だけにデータを転送するため，親に負荷が集中する可能性がある．

木庭[4)] は複数の生成木を構成し，使用する生成木を動的に切り替えることで負荷分散を行い，WSN の稼働時間を延長する自己安定アルゴリズムを提案した．しかし，この手法はセンシング対象領域すべてをセンシングできることを保証していない．

勝間ら[5)] は，センシング対象領域を 1 重被覆するノードの集合（レイヤ）を複数作成し，レイヤを 1 つずつ選択し稼働させ，他のレイヤに属するノードは休止させることにより，WSN の稼働時間延長を実現している．しかし，文献 5) で提案されている手法は集中型のアルゴリズムである．

提案手法では，上記の既存手法と異なり，データの転送先の選択肢を増やすために，DAG をデータ収集トポロジとして用いる．また，センシング対象領域すべての情報の収集に必要なノードは残し，不要なノードを休止させる．このとき，データ転送の負荷が集中するシンクノード付近では中継用のノードを多く起動させる．ノードの休止と DAG の構成を行う自己安定アルゴリズムを提案することで，分散処理による制御を実現した．さらに提案手法では，ノードの停止により転送途中のデータが損失することを防ぐために，各センシングデータを常に複数のノードに保持させながらデータ収集を行う．

## 3. システムモデル

### 3.1 分散システム

分散システムをグラフ $G = (V, E)$ と表す．$V = \{v_0, v_1, \ldots, v_n\}$ はノードの集合で，$E$ はノード間の双方向通信リンクの集合である．ノード $v_i, v_j$ 間に通信リンク $(v_i, v_j) \in E$ が存在するとき，$v_i$ と $v_j$ は**隣接する**という．$v_i$ に隣接するノードの集合を $N_i$ と表す．2 つのノード間の**距離**を，この 2 ノード間の最短経路に含まれる通信リンクの数と定義する．

各ノードは**局所変数**の集合を管理し，ノードの**状態**はすべての局所変数の値によって定義する．ノード間の通信モデルとして**局所的な共有メモリモデル**を用いる．各ノード $v_i$ はノード $v_j \in N_i \cup \{v_i\}$ の局所変数を遅延なく読むことが可能である．

各ノードのアルゴリズムはガード付きアクションの集合から成る．ガード付きアクションは，$\langle Guard \rangle \rightarrow \langle Action \rangle$ の形式で表される．ノード $v_i$ の $Guard$ は，$v_i$ 及び，$v_i$ の隣接ノード $v_j \in N_i$ の局所変数に対する論理式であり，$Action$ は $v_i$ の局所変数の値を書き換えるステートメントである．ノードは自身の $Guard$ のいずれか 1 つが真となるとき，**動作可能**であるといい，対応する $Action$ のうち 1 つを実行する．本稿では**スケジューラ**として $d$-**デーモン**を想定する．各**計算ステップ**において，$d$-デーモンは，動作可能であるノードの任意の空でない部分集合を選択し，選択されたノードは対応する $Action$ を実行する．すべてのノードが 1 回以上の動作を実行する期間を 1 **ラウンド**という．

システムの**状況**をノードすべての状態の組として定義する．アルゴリズムの**実行**は状況の無限系列 $e = c_0, c_1, c_2, \ldots$ で表記する．ここで $c_{i+1}$ は $c_i$ に 1 つの計算ステップを適用して得られる状況，もしくは動作可能なノードが存在しない終端状況である．

**問題**は，アルゴリズムの実行が満たすべき条件を定めている．問題 $T$ が要求する条件を満たす実行を $T$ の**正当な実行**と呼ぶ．

アルゴリズム $A(T)$ の任意の実行 $e = c_0, c_1, c_2, \ldots$ において，部分実行行 $e' = c_i, c_{i+1}, c_{i+2}, \ldots$ が問題 $T$ の正当な実行となるような $c_i\,(i \leq 0)$ が存在するとき，アルゴリズム $A(T)$ を問題 $T$ を解く自己安定アルゴリズムと呼ぶ．この条件を満たす状況 $c_i$ を自己安定アルゴリズム $A(T)$ の**正当な状況**と呼ぶ．

**一時故障**は複数のノードの局所変数の値を任意の値に書き換える．自己安定アルゴリズムは有限個の一時故障やトポロジ変化に対して自律適応性を保証する．つまり，自己安定アルゴリズムは一時故障やトポロジ変化が発生した後の状況を任意の初期状況と見なし，システムがやがて問題が要求する条件を満たすことを保証する．

本稿では，ノードが電池切れによって停止することを分散システムにおける**停止故障**として扱う．停止故障を起こしたノードは停止し，その後起動することはない．本稿では，正当な状況において生じる 1 ノードの停止故障のみを考える．停止故障が生じる間隔は，任意のセンサノードからシンクノードにデータが到達までの時間より長いと仮定する．正当な実行において，1 つのノードの停止故障が生じて得られる状況を 1-**停止故障状況**と呼ぶ．

### 3.2 無線センサネットワーク

WSN は，シンクノード $v_0$ 及びノード $v_i\,(i = 1, 2, \ldots n)$ の集合 $V = \{v_0, v_1, \ldots v_n\}$ から成る分散システム $G = (V, E)$ である．各ノードの通信範囲はそのノードを中心とした半径 $r_c$ であり，ノード $v_i$ と $v_j\,(v_i \neq v_j, i, j \geq 0)$ が互いの通信範囲にある場合，ノード $v_i, v_j$ 間に通信リンク $(v_i, v_j) \in E$ がある．シンクノードからノード $v_i$ までの最小ホップ数が $\ell$ であるとき，ノード $v_i$ は第 $\ell$ 層に属するという．各ノードのセンシング範囲はそのノードを中心とした半径 $r_s$ の円形領域である．ノードが自身のセンシング範囲の情報を計測することを**センシング**という．アプリケーションがセンシングを要求する地点を**センシングポイント**といい，センシングポイントすべてが起動中のノードのセンシング範囲に含まれるとき，センシング対象領域の**被覆**ができているという．センシング対象領域にあるセンシングポイントの集合を $SP = \{sp_1, sp_2, \ldots\}$ とする．各ノードの通信範囲の半径 $r_c$ はセンシング範囲の半径 $r_s$ の 2 倍以上とする．つまり，あるセンシングポイントをセンシングできるすべ

てのノードは互いに隣接している．シンクノード $v_0$ は $store(d_i)$ という関数を実行することで，受信したデータ $d_i$ をセンシングデータを保持するための外部記憶装置に保存する．

各ノード $v_i$ は，$v_i$ のセンシング範囲にあるすべてのセンシングポイントの集合 $SP_i$，$v_i$ の電池残量 $v_i.btr$ を持つ．アルゴリズムが収束するまでの間は，ノードの停止や追加，休止や起動は起こらないと仮定する．電池残量は，アルゴリズムの実行が開始する時点の量から，アルゴリズムが収束するまでに行う通信やセンシングに必要な最大の量を差し引いた値とする[*1]．あるノードが停止故障等によって通信できなくなると，周囲のノードはそのノードが停止したことを一定時間 $t_{detect}$ 以内に検知することができる[*2]．

ノードは**起動**，**休止**，**仮起動**のいずれかの**参加状態**をとる．起動状態のノードは他のノードとの通信とセンシングを行うことができるが，これらを行うために電力を消費する．休止状態のノードは消費エネルギーが小さいが，通信とセンシングを行うことができない．休止状態のノードは一定間隔（**休止間隔**と呼ぶ）で仮起動状態になり，隣接ノードの状態に応じて起動状態か休止状態になる．仮起動状態のノードはセンシングとデータ転送は行わない．ノードは起動状態及び仮起動状態ではアルゴリズムを実行するが，休止状態では実行しない．

各ノードは，一定間隔での実行や経過時間の管理を行うために，タイマーを保持している．各ノードの一定間隔での仮起動とセンシングはタイマーによって制御し，$d$-デーモンに依存しない割り込み処理として扱う．

### 3.3 データ収集問題

本節では，本稿で扱うデータ収集問題を定義する．

**定義 1 (データ収集問題)** 任意の実行 $e = c_0, c_1, \ldots$ において，状況 $c_j$ でノード $v_i$ がセンシングしたデータを $d_{i,j}$ とする．$d_{i,j}$ が $v_i$ から他のノードに転送されたならば，状況 $c'_j$ $(j < j')$ においてシンクノード $v_0$ で $store(d_{i,j})$ が実行される．

提案手法では，1-停止故障状況から始まる任意の実行 $e = c_0, c_1, \ldots$ においても，上記を保証する．さらに，$c_0$ において転送途中のセンシングデータについても上記を保証する．

## 4. 提 案 手 法

本章では，1-停止故障状況においてもデータ収集を可能にしつつ，ノードへのデータ転送の負荷を分散するために，センシング対象領域を 2 重被覆するデータ収集 DAG を構成する自己安定アルゴリズムを提案する．シンクノードの出次数が 0，かつ，第 1 層のノードの出次数が 1，かつ，第 2 層以上のノードの出次数が 2 以上，かつ，連結である DAG を**データ収集** $DAG$ と呼ぶ．

提案手法では，1 ノードの停止故障によるデータの損失を防ぐために，各センシングデータをセンシングしたノードと転送中のノードの 2 ノードに保持させる．そのために，各転送において，最新のセンシングデータと前回のセンシングデータをパッキングして転送する．本稿では，ノードがセンシングしたデータを，自身の転送先に 2 度転送した時点で定義 1 における転送が完了したとする．

### 4.1 データ収集 DAG 構成アルゴリズム（$\mathcal{DGDAG}$）

本節では，データ収集 DAG を構成するアルゴリズム（$\mathcal{DGDAG}$）を示す．$\mathcal{DGDAG}$ は，各ノードが属する層を決定し，シンクノードへの連結に必要なノード（**連結用ノード**と呼ぶ）を求めるアルゴリズム（$\mathcal{CNS}$）と，各ノードが参加状態を決定するアルゴリズム（$\mathcal{ANP}$，$\mathcal{TNP}$）と，各ノードが転送先の候補を決定するアルゴリズム（$\mathcal{DAG}$）から成る．

各ノードは，自身が連結用ノードに選択されているかと 2 重被覆に必要かの判定を行うことで参加状態を決定する．すべてのノードが連結ノードを確保できるように，各ノードは連結用ノードの選択後，すべてのノードが連結用ノードを選択し終えるのに十分な時間だけ待機し，参加状態の決定を行う．提案手法では，この実行順序を実現するために各ノードが持つタイマーを用いる．そして，起動状態のノードが転送先の候補となる隣接ノードを選択することで DAG を構成する．

#### 4.1.1 連結用ノード選択アルゴリズム（$\mathcal{CNS}$）

$\mathcal{CNS}$ では，各ノードがシンクノードから自身までの距離を求めることで，自身が属する層を特定し，1 つ下の層に属する自身の隣接ノードのうち 2 ノードを連結用ノードとする．$\mathcal{CNS}$ をアルゴリズム 4.1 に示す．

シンクノード $v_0$ は，シンクノードからの距離 $v_0.dist$ を 0，2 つの連結用ノード $v_0.c1$，$v_0.c2$ を $NULL$ に保つために $S_{1.1}$ を実行する．シンクノードに隣接する各ノード $v_i$ は，シンクノードからの距離 $v_i.dist$ を 0，第 1 の連結用ノード $v_i.c1$ を $v_0$，第 2 の連結用ノード $v_i.c2$ を $NULL$ に保つために $S_{1.2}$ を実行する．シンクノードに隣接しない各ノード $v_j$ は，シンクノードからの距離と連結用ノードを正しい値に保つために $S_{1.4}$ を実行する．上記の局所変数の値が正しくないか，新たに起動したノードがあるか，連結用ノードの停止から休止間隔経過後に実行する．$v_j$ は，連結用ノードが停止故障した際に $S_{1.3}$ を実行し，より下の層のノードが起動するのを待ち，連結用ノードとする．

#### 4.1.2 起動状態のノードの参加状態決定アルゴリズム（$\mathcal{ANP}$）

$\mathcal{ANP}$ は起動状態のノードの参加状態を決定するアルゴリズムである．$\mathcal{ANP}$ をアルゴリズム 4.2 に示す．$\mathcal{ANP}$ は $\mathcal{CNS}$ の出力である連結用ノードと $\mathcal{CNS}$ の実行からの経過時間を入力とする．各ノードは，$\mathcal{CNS}$ を実行後，$\mathcal{CNS}$ の収束時間 $(d+2)$ ラウンド（$d$:ネットワークの直径）が経過するまで待機し，$\mathcal{ANP}$ を実行することで，連結用ノードとして選択されたノードを起動状態にする．起動状態の各ノードは，自身が被覆するセンシングポイントすべてが自身より電池残量が多い 2 つ以上の隣接ノードによって被覆されており，自身の隣接ノードすべてが自身以外を連結用ノードに選択しているときに休止状態となる．このとき，$1/(2 * v_i.dist)$ の確率で起動状態を保つことで，シンクノードに近いほど中継ノードを増やし，ノードへのデータ転送の負荷を分散する．

#### 4.1.3 仮起動状態のノードの参加状態決定アルゴリズム（$\mathcal{TNP}$）

新たに追加されたノードの参加状態は仮起動状態となる．また，休止状態になったノード

---

*1 アルゴリズムの実行中に電池残量が変化し，各ノードの実行に影響が出るのを防ぐため．
*2 ハローメッセージの交換ができなくなったノードが停止したと判断するといった実装方法がある．

**アルゴリズム 4.1** $\mathcal{CNS}$

**ノード $v_i$ の入力変数**
$t_{period}$: ノードが休止状態となってから仮起動状態になるまでの間隔
$IsSink_i$: $v_i$ がシンクノードならば真，さもなければ偽をとる Boolean 型変数
**ノード $v_i$ の内部変数**
$N_i^{select}$: 連結用ノードの選択を行ったときの $v_i$ の隣接ノードの集合
$t_i^{stop}$: $v_i$ の連結用ノードが停止故障してから経過した時間
**ノード $v_i$ の出力変数**
$v_i.c1$, $v_i.c2$: $v_i$ の第 1，第 2 の連結用ノードを指すポインタ
$v_i.dist$: シンクノードから $v_i$ までの距離
$t_i^{select}$: $v_i$ が連結用ノードの選択を行ってから経過した時間
**関数**
$start\_timer(t)$: 引数 $t$ のカウントアップを 0 から開始する関数
$stop\_timer(t)$: $t$ のカウントを終了し，$t$ に 0 を代入する関数
**ノード $v_i$ のアクション**
// シンクノード $v_0$ は距離を 0，連結用ノードを $NULL$ に保つ.
$S_{1.1} : IsSink_i \wedge (v_i.dist \neq 0 \vee v_i.c1 \neq NULL \vee v_i.c2 \neq NULL)$
$\longrightarrow v_i.dist := 0;\ v_i.c1 := NULL;\ v_i.c2 := NULL;$
// シンクノードに隣接するノード $v_i$ は距離を 1，連結用ノードを $v_0$ のみに保つ.
$S_{1.2} : \neg IsSink_i \wedge v_0 \in N_i \wedge (v_i.dist \neq 1 \vee v_i.c1 \neq v_0 \vee v_i.c2 \neq NULL)$
$\longrightarrow v_i.dist := 1;\ v_i.c1 := v_0;\ v_i.c2 := NULL;$
// シンクノードに隣接しないノード $v_j$ は連結用ノードが停止してからの時間待機.
$S_{1.3} : \neg IsSink_i \wedge v_0 \notin N_i \wedge (v_i.c1 \notin N_i \vee v_i.c2 \notin N_i)$
$\longrightarrow start\_timer(t_i^{stop});\ v_i.c1 := NULL;\ v_i.c2 := NULL;$
// 正しい層の番号を保持し，上層のノードから連結用ノードを選択.
$S_{1.4} : \neg IsSink_i \wedge v_0 \notin N_i \wedge ((v_i.dist \neq min\{v_j.dist + 1 | v_j \in N_i\}) \vee$
$(v_i.c1 \notin \{v_j | v_j \in N_i \wedge v_j.dist = min\{v_k.dist | v_k \in N_i\}\} \wedge v_i.c1 \neq NULL) \vee$
$(v_i.c2 \notin \{v_j | v_j \in N_i - \{vi.c1\} \wedge v_j.dist = min\{v_k.dist | v_k \in N_i - \{vi.c1\}\}\} \wedge$
$v_i.c2 \neq NULL) \vee (N_i - \{v_j | v_j \in N_i \wedge v_j.dist \neq NULL\} - N_i^{select} \neq \emptyset) \vee t_i^{stop} > t_{period})$
$\longrightarrow v_i.dist := min\{v_j.dist + 1 | v_j \in N_i\};$
$v_i.c1 := v_j : v_j \in N_i \wedge v_j.btr = max\{v_k.btr | v_k \in N_i \wedge v_k.dist = min\{v_l.dist | v_l \in N_i\}\};$
$v_i.c2 := v_j : v_j \in N_i - \{vi.c1\} \wedge v_j.btr = max\{v_k.btr | v_k \in N_i - \{vi.c1\} \wedge$
$v_k.dist = min\{v_l.dist | v_l \in N_i - \{vi.c1\}\}\};$
$N_i^{select} := N_i - \{v_j | v_j \in N_i \wedge v_j.dist \neq NULL\};$
$stop\_timer(t_i^{stop});\ start\_timer(t_i^{select});$

は休止間隔ごとに仮起動状態となる．仮起動状態の各ノードは $\mathcal{TNP}$ を実行することで自身の参加状態を決定する．$\mathcal{TNP}$ をアルゴリズム 4.3 に示す．仮起動状態のノードは，自身が 2 重被覆か他ノードの連結用ノードのいずれか一方にでも必要な場合，起動状態となり，いずれにも不要であるときに休止状態となる．自身より電池残量が多い仮起動状態の隣接ノードが起動することで，2 重被覆が保証されるならば自身は起動せずに休止状態となる．このときも，$\mathcal{ANP}$ と同様にシンクノードからの距離に応じて起動状態となることで，シンクノードに近いほど中継ノードを増やす．

**アルゴリズム 4.2** $\mathcal{ANP}$

**ノード $v_i$ の入力変数**
$v_i.c1$, $v_i.c2$: $v_i$ の第 1，第 2 の連結用ノードを指すポインタ（$\mathcal{CNS}$ の出力）
$v_i.data$: $v_i$ がセンシングしたデータ（$\mathcal{DG}$ の出力）
$v_i.rly$: $v_i$ が転送するデータ（$\mathcal{DG}$ の出力）
$t_i^{select}$: $v_i$ が $S_{1.4}$ を実行してから経過した時間
**ノード $v_i$ の内部変数**
$t_{converge}$: $\mathcal{CSN}$ の収束時間（$d$ ラウンド（$d$:ネットワークの直径））
$v_i.mode$: $v_i$ の参加状態を表す．$active$:起動状態，$sleep$:休止状態，$temp$:仮起動状態
**ノード $v_i$ の述語**
//$v_i$ が被覆するセンシングポイントすべてが $v_i$ よりも電池残量が多い 2 ノード以上の
//起動状態のノードに被覆されていれば真となる.
$CoverAct_i \equiv \forall sp_j \in SP_i : |\{v_k | v_k \in N_i \wedge sp_j \in SP_k \wedge v_k.mode = active \wedge$
$vk.btr > vi.btr\}| \geq 2$
//$v_i$ に隣接する起動状態のノードが $v_i$ 以外を連結用ノードに選択していれば真となる.
$Connect_i \equiv \forall v_j \in N_i : (v_j.c1 \neq NULL \wedge v_j.c2 \neq NULL \wedge v_j.c1 \neq v_i \wedge$
$v_j.c2 \neq v_i) \vee (v_j.dist = 1 \wedge v_j.c1 \neq NULL \wedge v_j.c1 \neq v_i) \vee (v_j.dist = 0) \vee$
$(v_j.mode = temp)$
**関数**
$add(v_i.dist)$: $1/(2 * v_i.dist)$ の確率で真を返す関数
**ノード $v_i$ のアクション**
$S_{2.1} : t_i^{select} > t_{converge} \wedge CoverAct_i \wedge Connect_i \wedge v_i.data = NULL \wedge$
$v_i.rly = NULL \wedge \neg add(v_i.dist)$
$\longrightarrow v_i.mode := sleep;$

#### 4.1.4 DAG 構成アルゴリズム（$\mathcal{DAG}$）

$\mathcal{DAG}$ は各ノードに転送先の候補ノードの集合と迂回先の候補ノードの集合を保持させるアルゴリズムであり，$\mathcal{CNS}$ の出力であるシンクノードからの距離を入力とする．$\mathcal{DAG}$ をアルゴリズム 4.4 に示す.

図 1 のように，データ収集 DAG の辺は，起動状態の各ノード $v_i$ を始点とし，(1)$v_i$ の下の層に属する $v_i$ の隣接ノード $v_j$ を終点とする有向辺 $(v_i, v_j)$（この有向辺 $(v_i, v_j)$ を**転送辺**，$v_j$ を $v_i$ の**転送先の候補ノード**と呼ぶ）(2)$v_i$ と同じ層に属し，$v_i$ より多くの転送辺の始点となっている $v_i$ の隣接ノード $v_k$ を終点とする有向辺 $(v_i, v_k)$（この有向辺 $(v_i, v_k)$ を**迂回辺**，$v_k$ を $v_i$ の**迂回先の候補ノード**と呼ぶ）から成る．各ノード $v_i$ が $S_{4.1}$ により転送先の候補ノードの条件に合う隣接ノードの集合 $FWD_i$ を，$S_{4.2}$ により迂回先の候補ノードの条件に合う隣接ノード集合 $DTR_i$ を保持することでデータ収集 DAG を構成する.

### 4.2 データ収集アルゴリズム（$\mathcal{DG}$）

本節では，データ収集 DAG 上でデータ収集を行うアルゴリズム $\mathcal{DG}$ を示す．$\mathcal{DG}$ は $\mathcal{DAG}$ の出力である $FWD_i$ と $DTR_i$ を入力とする．ノードはタイマーによる制御で一定間隔ごとにセンシングを行う．各ノードは自身が被覆するセンシングポイント $sp_i$ について，自身が $sp_i$ を被覆する電池残量が最多のノードであるならば，自身が被覆するセンシングポイン

**アルゴリズム 4.3** $\mathcal{TNP}$

**ノード $v_i$ の内部変数**
$v_i.mode = temp$: $v_i$ の参加状態
$threshold$: 電池残量の差の閾値
**ノード $v_i$ の出力変数**
$v_i.mode$: $v_i$ の参加状態を表し，$active$:起動状態，$sleep$:休止状態，$temp$:仮起動状態
**ノード $v_i$ の述語**
$//v_i$ が被覆するセンシングポイントすべてが，2 ノード以上の $v_i$ よりも電池残量が
$//threshold$ を超えて少なくない起動状態のノード，または，$v_i$ よりも電池残量が
//多い仮起動状態の隣接ノードに被覆されていれば真となる
$CoverTemp_i \equiv \forall sp_j \in SP_i : |\{v_k | v_k \in N_i \wedge sp_j \in SP_k \wedge ((v_k.mode = active \wedge vk.btr + threshold > vi.btr) \vee (v_k.mode = temp \wedge vk.btr > vi.btr))\}| \geq 2$
$//\mathcal{ANP}$ と同様.
$Connect_i \equiv \forall v_j \in N_i : (v_j.c1 \neq NULL \wedge v_j.c2 \neq NULL \wedge v_j.c1 \neq v_i \wedge v_j.c2 \neq v_i) \vee (v_j.dist = 1 \wedge v_j.c1 \neq NULL \wedge v_j.c1 \neq v_i) \vee (v_j.dist = 0) \vee (v_j.mode = temp)$
**関数**
$add(v_i.dist)$: $1/(2 * v_i.dist)$ の確率で真を返す関数
**ノード $v_i$ のアクション**
$S_{3.1}$ : **if** $(CoverTemp_i \wedge Connect_i \wedge \neg add(v_i.dist))$ **then** $v_i.mode := sleep$;
**else** $v_i.mode := active$;

**アルゴリズム 4.4** $\mathcal{DAG}$

**ノード $v_i$ の入力変数**
$v_i.dist$: シンクノードから $v_i$ までの距離（$\mathcal{CNS}$ の出力）
**ノード $v_i$ の出力変数**
$FWD_i$: $v_i$ の転送先の候補ノードの集合
$DTR_i$: $v_i$ の迂回先の候補ノードの集合
**ノード $v_i$ のアクション**
$S_{4.1} : FWD_i \neq \{v_j | v_j \in N_i \wedge v_j.dist = v_i.dist - 1\}$
$\longrightarrow FWD_i := \{v_j | v_j \in N_i \wedge v_j.dist = v_i.dist - 1\}$;
$S_{4.2} : DTR_i \neq \{v_j | v_j \in N_i \wedge v_j.dist = v_j.dist \wedge |FWD_j| > |FWD_i|\}$
$\longrightarrow DTR_i := \{v_j | v_j \in N_i \wedge v_j.dist = v_j.dist \wedge |FWD_j| > |FWD_i|\}$;

トすべてのセンシングを行う．

提案手法では，転送中のセンシングデータを常に 2 つのノードに保持させるために，各ノードは，自身がセンシングした最新のデータと前回のデータの 2 つを 1 つの転送データにパッキングする．ノード $v_i$ がデータを $d_1, d_2, d_3$ の順にセンシングしたとする．このとき $v_i$ は，$d_2$ のセンシング後に $(d_1, d_2)$ を転送し，$d_3$ のセンシング後に $(d_2, d_3)$ を転送する．各ノードは転送データを 1 つずつ中継するため，$(d_1, d_2)$ と $(d_2, d_3)$ を 1 つのノードが同時に保持することはなく，$d_2$ は常に 2 つのノードが保持することになる．この方法で $d_2$ を転送することで，$(d_1, d_2)$ が転送中に 1 つの中継ノードの停止故障によって損失したとしても，

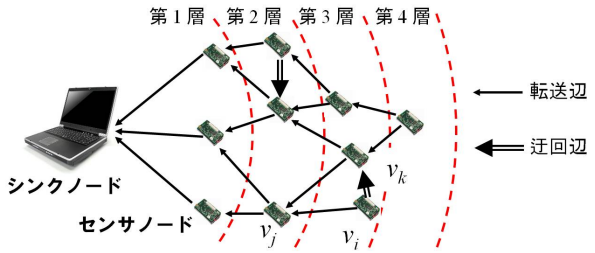


**図 1** データ収集 DAG を構成する有向辺

$(d_2, d_3)$ の転送中には停止故障が起きないため，$d_2$ は確実にシンクノードに到達できる．

センシングデータのシンクノードへの収集は，各ノードが自身を転送先とするノードから転送データを取得することでデータの中継を行い，これを繰り返すことでシンクノードに転送データを到達させる．各ノード $v_i$ は，$v_i$ に転送データが届いたことを $v_i$ を転送先とするノード $v_j$ が認識しており，$v_i$ の転送先 $v_k$ に転送データが届いたときに次の転送データを取得する．次の転送データの取得は，自身のセンシングデータを優先し，次に自身を転送先とするノードの集合からラウンドロビン方式でノードを選択し，そのノードから取得する．次の転送先は，$FWD_i$ と $DTR_i$ から電池残量とシンクノードからの距離に応じて選択する．ノード $v_j$ を選択する確率は，

$$\frac{v_j.btr/(v_j.dist - v_i.dist + 2)}{\sum_{v_k \in FWD_i \cup DTR_i} v_k.btr/(v_k.dist - v_i.dist + 2)}$$

である．次の転送先には，電池残量が多く，シンクノードに近いノードを選択しやすくすることで，各ノードの電池残量の消耗の平均化を行う．そして，転送先から自身の転送データが取得されるのを待つ．

シンクノード $v_0$ は，ラウンドロビン方式で自身の子から順に転送データ $d_i$ を取得する．そして，$store(d_i)$ を実行することで $d_i$ を外部記憶装置に保存する．

## 5. 正しさの証明

本章では，提案手法によってデータ収集問題を解くことができることを示す．補題 1 では，センシング対象領域の 2 重被覆が保証できることを示す．補題 2〜4 では，データ収集 DAG を構成できることを示す．以上より，$\mathcal{DGDAG}$ が，センシング対象領域が 2 重被覆するデータ収集 DAG を構成する自己安定アルゴリズムであること（定理 1）を示す．最後に $\mathcal{DG}$ がデータ収集問題を解くアルゴリズムであること（定理 2）を示す．ここでは，層の数を $\ell$ と仮定する．

**補題 1** $\mathcal{ANP}$ 及び $\mathcal{TNP}$ によって，任意の初期状況からセンシング対象領域が 2 重被覆されることが保証される．

$\mathcal{TNP}$ より，あるセンシングポイントが 2 重被覆されていない場合は，電池残量が多い仮起動状態のノードが起動状態になる．各ノードは休止間隔ごとに $\mathcal{ANP}$ を実行するため，任意の初期状況から休止間隔が経過するとセンシング対象領域の 2 重被覆が保証される．$\mathcal{ANP}$ より，ノード $v_i$ は自身がセンシング対象領域の 2 重被覆に不要なときにのみ休止状態となるため，センシング対象領域の 2 重被覆が保証された状況から 2 重被覆されていないセンシングポイントが現れることはない．したがって，$\mathcal{ANP}$ 及び $\mathcal{TNP}$ によって，任意の初期状況からセンシング対象領域が 2 重被覆されることが保証される．

**補題 2** $\mathcal{CNS}$ によって，任意の初期状況から最大 $(\ell+2)$ ラウンドで各ノードが属する層と連結用ノードが決定される．

$v_i$ のシンクノードからの距離 $v_i.dist$ の値は，シンクノードからの距離が最小である $v_i$ の隣接ノード $v_j$ を用いて $v_j.dist+1$ とするため，$v_i.dist$ が属する層の値より小さい場合は，ラウンドの経過とともに増加する．$\mathcal{CNS}$ より，任意の初期状況から 1 ラウンドが経過すると，定数が与えられる第 1 層以下の各ノードの局所変数は正しい値をとる．第 2 層以降のノード $v_i$ について，$v_i.dist$ が小さい値であってもいずれ増加し，下の層の隣接ノード $v_j$ を連結用ノードとし，$v_i.dist$ の値が $v_j.dist+1$ となる．以上を繰り返し，下の層のノードから各ラウンドごとに順繰りに正しい値を計算していく．第 $\ell$ 層では，連結性に必要な休止状態のノードが起動後，最大 $(\ell+2)$ ラウンドで各ノードのシンクノードからの距離と連結用ノードを決定することができる．

$\mathcal{ANP}$ は，$\mathcal{CNS}$ の出力を受け取る．各ノードは連結用ノードの選択後，$d+2$ ラウンドが経過してから，$\mathcal{ANP}$ を実行する．補題 2 より，このとき既に $\mathcal{CNS}$ は収束しているため，$\mathcal{ANP}$ は正しい入力を受け取ることができる．

**補題 3** $\mathcal{DAG}$ はデータ収集 DAG を構成する．

$\mathcal{TNP}$ より，連結用ノードを保持していないノードが存在する場合，その隣接ノードが起動し連結用ノードとなるため，データ収集トポロジは連結となる．また，$\mathcal{ANP}$ より，連結用ノードに選択されているノードは休止状態とならないので，データ収集トポロジは連結であり続ける．また，転送辺はより下の層のノードのみを指し，迂回辺は同じ層で転送先の候補ノードがより多いノードを指すため，閉路は構成されない．したがって，$\mathcal{DAG}$ が構成するデータ収集トポロジは DAG である．シンクノード $v_0$ について，$v_i.dist \leq v_0.dist$ となるノード $v_i$ は存在しないため，出次数は 0 となる．第 1 層のノード $v_i$ について，$v_j.dist < v_i.dist$ となるノード $v_j$ はシンクノードのみであり，同じ層の転送先の候補ノードはシンクノードのみであるため，出次数は 1 となる．第 2 層以降のノード $v_i$ について，$v_i$ の連結用ノード $v_i.c1$，$v_i.c2$ は起動し続けるため，出次数は 2 以上となる．以上より，$\mathcal{DAG}$ が構成するデータ収集トポロジはデータ収集 DAG である．

**補題 4** $\mathcal{DGDAG}$ によって，任意の初期状況から休止間隔の経過と最大 $(\ell+4)$ ラウンドでデータ収集 DAG が構成される．

補題 2 より，層の数が $\ell$ であるとき，任意の初期状況から，休止間隔の経過と最大 $(\ell+2)$ ラウンドで各ノードが属する層を特定することができる．$FWD_j$ は各ノードが属する層によって決定できるので，次のラウンドで得られる．$DTR_j$ は各ノードが属する層と $FWD_i$ によって決定できるので，さらに次のラウンドで得られる．したがって，$\mathcal{DGDAG}$ によって，任意の初期状況から休止間隔の経過と最大 $(\ell+4)$ ラウンドでデータ収集 DAG が構成される．

補題 1,2,3,4 より，定理 1 が成り立つ．

**定理 1** $\mathcal{DGDAG}$ はセンシング対象領域を 2 重被覆するデータ収集 DAG を構成する自己安定アルゴリズムである．

最後に $\mathcal{DG}$ がデータ収集問題を解くアルゴリズムであることを示す．

**定理 2** $\mathcal{DG}$ は 2 重被覆するデータ収集 DAG 上でデータ収集問題を解くアルゴリズムである．

$\mathcal{DG}$ より，各センシングデータは 2 ノードに保持させつつ 2 度転送する．そのため，停止故障によってセンシングデータが損失することが無く，どちらか一方の転送を行う際には停止故障は生じることはなくシンクノードに到達する．したがって，$\mathcal{DG}$ は 2 重被覆するデータ収集 DAG 上でデータ収集問題を解くアルゴリズムである．

## 6. ま と め

本稿では，1 ノードの停止故障発生時でもすべてのセンシングデータがシンクノードに到達する自己安定アルゴリズムを提案した．さらに，提案アルゴリズムでは，1 ノードの停止故障によるセンシングデータ損失を防ぐため，センシング対象領域を 2 重被覆し，各データを常に 2 ノードに持たせた状態を維持しながらデータ収集を行う．今後の発展として，シミュレーションや実機を用いた実験による稼働時間の評価などが挙げられる．

## 参 考 文 献

1) Burri, N., von Rickenbach, P. and Wattenhofer, R.: Dozer: ultra-low power data gathering in sensor networks, *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, ACM, pp.450–459 (2007).
2) Deng, J., Han, Y. S., Heinzelman, W. B. and Varshney, P. K.: Balanced-energy sleep scheduling scheme for high-density cluster-based sensor networks, *Comput. Commun.*, Vol.28, pp.1631–1642 (2005).
3) Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control, *Commun. ACM*, Vol.17, pp.643–644 (1974).
4) Kiniwa, J.: Construction of Self-Stabilizing Disjoint Sense-Sleep Trees with Application to Sensor Networks, *IEICE Transactions*, Vol.92-A, No.4, pp.1174–1181 (2009).
5) 勝間亮，村田佳洋，柴田直樹，安本慶一，伊藤実：無線センサネットワーク長寿命化のためのノード集合の分割に基づくスリープスケジューリング手法，情報処理学会論文誌 数理モデル化と応用，Vol. 3，No. 3，pp. 140-153（2010）.