LL_010

センサネットワークにおける電源残量に基づいたクラスタヘッド選出方法
Cluster Head Selection Algorithm based on Battery Availability

永田 智大† 小熊 寿† 山崎 憲一†
Tomohiro Nagata Hisashi Oguma Kenichi Yamazaki

1 はじめに

実在のオブジェクト (以下「モノ」とする) 間の関連性に着目したコンテキスト取得研究は数多く存在する．最近では，モノのまとまり状況をコンテキストとして検知する研究が特に注目されている．我々は同じ状況にあるモノを1つのまとまり (グループ) として捉え，コンテキストを生成する研究を行っている．

コンテキストは，データマイニングのために蓄積する場合と，その場で即座に利用する場合が考えられる．後者のようにリアルタイムにコンテキストを扱えるようにするために，すばやく同じ状況にあるモノを発見する必要がある．我々はモノにセンサノードを取り付け，同じ状況にあるモノからグループを構築する方法を提案している．この方法では同じ状況にあることを認識するために，センサ測定時期と測定値それぞれの類似度に着目している (これ以降では同時類似性と呼ぶ)．我々はこのようなグループをクラスタと読んでいる．リアルタイムにコンテキストを取得するためには，素早くクラスタリングすることが求められる．

従来のセンサネットワーク研究ではクラスタリングを利用して，センサノードの消費電力を抑える方法が提案されている．これらの提案では，クラスタヘッドと呼ばれる代表ノードがクラスタに所属するセンサノードからの情報を集約し，送信する．クラスタヘッドはクラスタに所属する他のセンサノードに比べて通信量が多くなるため，消費電力も多い．センサネットワーク全体の動作時間をより長くするためには，電源残量の多いセンサノードをクラスタヘッドとして選ぶためのアルゴリズムが必要となる．

そこで，本研究では，センサノードがクラスタを形成する際のクラスタヘッドを短い時間でノードの電源残量に基づいて選出するためのアルゴリズムを提案する．その際，クラスタヘッドを選出する処理にかかる時間を短かくすることで，消費電力を抑えて効率的にクラスタリングすることを実現する．

本論文は以下のように構成される．第2章では，センサネットワークにおけるクラスタヘッドの選出方法について考察を行う．その考察を踏まえ，第3章で本論文の提案する電源残量に基づくクラスタヘッドの選出方法について詳細に述べる．第4章ではネットワークシミュレータを用いてクラスタヘッド選出までにかかる時間を測定し，その有効性について評価を行う．第5章にてまとめる．

2 クラスタヘッド選出方法への考察

クラスタヘッドは一度データを集約し，シンクノードに一括してデータを送信するため，他のセンサノードの消費電力を抑えられる．しかし，クラスタヘッドの負荷が高くなり，クラスタヘッド自身の消費電力も高くなってしまう．このようなトレードオフの関係となっているため，いくつかの方法が提案されている．そこで，本章ではクラスタヘッドの選出方法について考察を行う．

2.1 クラスタヘッドの選出方法

クラスタヘッドの選択方法として，一般的に以下の2つが考えられる．

- ネットワークトポロジを基に選ぶ方法
  センサノード間やデータを最終的に集約するためのシンクノードまでの経路を考慮して，クラスタを形成し，各要所にクラスタヘッドを配置する．
- 計算資源の平等性に基づいて選ぶ方法
  計算資源の豊かなノードをクラスタヘッドとして選択する．なお，センサネットワークの場合，最も重要な計算資源は電源である．

クラスタを形成して多くのデータを送受信する場合，ネットワークトポロジを基に，データの集約しやすい場所に配置されたノードをクラスタヘッドに選出することが多い．クラスタヘッドがデータの集約を行いやすくなるため，ネットワーク全体の消費電力を抑えられる．

コンテキスト取得のためにクラスタを形成することを考えた場合，モノに付随してセンサノードも移動し，クラスタを構成するメンバノードが変化する．1章で述べたように，我々はモノにセンサを装備することを想定しており，頻繁にネットワークトポロジも変化する．また，モノの置かれている状況が頻繁にかわると，クラスタを構成するメンバノードもそれに併わせてかわる．このような目的のためにネットワークトポロジを基にクラスタヘッドを選ぶと，トポロジ全体を把握するためのメッセージのやりとりが増大し，消費電力が増大してしまう．後者の場合，クラスタを構築するメンバノードの中からクラスタヘッドを選ぶ．そのため前者よりも通信量が少なく，消費電力を抑えられる．そこで，次節では電源残量を基準としてクラスタヘッドを選出する際の問題点について述べる．

2.2 電源残量を基準とした際の課題

電源残量の多いセンサノードをクラスタヘッドとして選出するために，お互いのノードが自身の電源残量を送受信すると，それによる消費電力が発生してしまう．そのため，ノード間のメッセージのやりとりをせずに選出する以下のような方法が提案されている．

- 確率的に均等となるように決められたノードがクラスタヘッドとなる
- 電源残量に反比例する待ち時間が最初に終了したノードがクラスタヘッドとなる

前者にはLEACH[1]が挙げらる．LEACHでは電源残量から求められる確率に基き，自身がクラスタヘッドになるべきか否かを判断する．センサノード同士の時間

†株式会社 NTT ドコモ ネットワーク研究所
Network Laboratories, NTT DoCoMo, Inc.

表 1: CHD メッセージの構成

NID	送信者のノード ID
TRtype	トリガタイプ
Timestamp	トリガが発生してから，そのメッセージが生成されるまでの時間

は同期しており，一定周期毎にクラスタヘッドの選出が行われる．センサノードの電源残量が多ければクラスタヘッドになる確率が高くなり，逆に電源残量が少なければ確率は低くなる．

後者には [2]，[3] などが挙げられ，電源残量から求められる待機時間待ったのち，先にタイムアウトしたセンサノードがクラスタヘッドの宣言を行う．各センサノードは待機時間を電源残量に反比例するような値として求める．そのため，電源残量の多いセンサノードほど待機時間が短かくなり，はやくタイムアウトを起こし，クラスタヘッドとなる．前者は十分に多くのセンサノードがある状態で十分に長い時間に渡って運用した場合には，消費電力が平均化される．しかし，1 章で述べたように，我々はセンサノードが移動することを想定し，その場合，短期間での消費電力の平均化が必要である．そのため，前者のようなクラスタヘッド選出方法は向いていない．後者は電源残量を直接的に制御に反映するため，少数ノードで短時間運用においても効果的である．しかし，これには次に述べるような問題がある．

後者で扱う待機時間 $WaitT$ を，各センサノードの装備する最大電源容量である $maxPower$，センサノード n の現在の電源残量である $Power_n$ を用いて現すと以下のようになる [3]．

$$WaitT = \alpha * (maxPower - Power_n) \qquad (1)$$

α は 0 以上の値をとる係数である．上の式からもわかるように，後者の方式では，ネットワーク上のセンサノードの電源残量が全体的に少なくなると，待機時間もそれにつれて長くなってしまう．待機時間が伸びることにより，クラスタを構築するまでの時間も長くなってしまい，結果としてセンサノードの消費電力が増えてしまう．

そこで，本論文では，[2] や [3] をふまえ，待機時間が一次増加関数のように増加せずに電源残量を基準としてクラスタヘッドを選出する方法を実現する．次章では，このような問題を解決する電源残量を基準としたクラスタヘッドの選出方法について述べていく．

3 電源残量を基準としたクラスタヘッド選出方法

3.1 クラスタリングアルゴリズム

具体的な電源残量を基準としたクラスタヘッド選出方法を述べる前に，本論文での提案がベースとする同時類似性に着目したクラスタリングアルゴリズム [3] について説明する．

各ノードは，トリガと呼ばれるセンサ値の変化パターンをあらかじめ指示する．トリガの例として，センサの値が 2 つの閾値で指定された範囲外の値になった Out-of-Range などが挙げられる．クラスタ形成時にクラスタヘッド候補から送信される CHD (Cluster Head Declaration) メッセージを表 1 に示す．

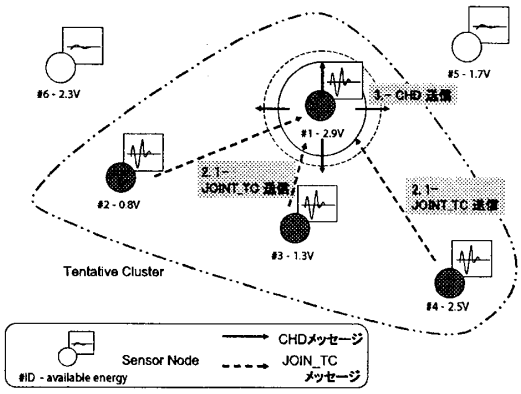

図 1: クラスタ生成時の様子

まず，センサノードが初期化された時に，センサノード上のアプリケーションがトリガタイプをパラメータとともに登録する．以下では，クラスタ生成アルゴリズムを図 1 を参照しつつ述べる．なお，文章の各処理に振られた番号は図中の番号に対応し，ノード #1〜4 は同じタイミングでトリガを検出し，ノード #5〜6 はトリガを検出していない．また，記述上，CHD メッセージの処理番号が JOIN_TC メッセージの処理番号より大きくなっているが，実際は CHD メッセージを受けて JOIN_TC メッセージが送信される．

クラスタ生成アルゴリズム:

1. 登録されたトリガ TR が検知されると，WaitT 待つための時間計測を開始する．
2. CHD メッセージ M が近隣のノード N から送信されてきた場合には，M.TRtype と TR のトリガタイプが同じで，かつ M.Timestamp と現在のタイムスタンプがほぼ同じかを調べる．(図中: ノード #2〜4)
   - **2.1.** 同じであるときは，WaitT の待ちが終了した後，JOIN_TC メッセージをノード N に送る．(図中: ノード #2〜4)
   - **2.2.** そうでない時は，WaitT の待ちを継続する．
3. WaitT の間，CHD が受信されない時は，CHD メッセージを作成し，ブロードキャストする．(図中: ノード #1)
   - **3.1.** TimeoutT 時間だけメッセージを待ち
   - **3.2.** JOIN_TC が受信されたら，その送信元ノードを TC に加える．(図中: ノード #1)

WaitT，TimeoutT は，時間に関するパラメータである．WaitT は，次節で詳しくは述べるが，センサノードの電源残量が十分多い場合，WaitT，すなわちステップ 3 実行までの時間は相対的に短くなる．その結果，最も電源残量のあるセンサノード (図中では 2.9V と電源残量の一番多いノード #1) がクラスタヘッドとなり，CHD を送信する．この方法ではクラスタヘッドの消費電力が大きくなるため，電源残量の最も多いセンサノードをクラスタヘッドとしている．CHD を発行するクラスタヘッドは TimeoutT をタイムアウト値として設定し，他のセンサノードからの JOIN_TC を待つ．

検出したトリガを同時類似性が高いと判断する基準として，CHD に含まれる Timestamp を利用する．この時間はサンプリングレートや状況などにより，センサノード間で誤差が生じる．時間の測定誤差許容値は対象となるサービスアプリケーションに依存する．センサノード

はアプリケーションが指定する許容値を用いて，検出したトリガの同時類似性を判断する．

3.2 クラスタヘッド選出方法

3.2.1 前クラスタヘッドの電源残量との比較

前章で述べたように，単純にWaitTの時間を電源残量に反比例するような値にした場合，ネットワーク上のセンサノードの電源残量が全体的に少なくなると，WaitTの値もそれにつれて大きくなり，クラスタを構築するまでの時間も長くなる．それを防ぐために，本提案方式では，クラスタを構築する際，クラスタヘッドの電源残量をメンバノードが保存し，次回クラスタヘッドを選出する際にその値を利用する．

メンバノードがクラスタヘッドの電源残量を知らせるため，クラスタヘッドから送信されるCHDメッセージにクラスタヘッドの電源残量を保持するPower値を追加する．クラスタを形成する際，クラスタヘッドから送られてきたCHDメッセージ中のPower値をメンバノードは保存する．

$Power_{ch}$，$Power_n$を以前のクラスタヘッドの電源残量，自身の電源残量，αを後述するWaitT係数とする．次にクラスタを形成する際，各ノードで計算される$WaitT$は以下のように定義する．

$$WaitT = \alpha * (Power_{ch} - Power_n) \qquad (2)$$

前節で述べたように，各ノードは上式で求められた$WaitT$時間後，それまでCHDメッセージを受信していなければ自身がクラスタヘッドとして動作し，CHDメッセージを送信する．また，$WaitT$時間待機している間にCHDメッセージを受信すれば，必要に応じてJOIN_TCメッセージを送信する．WaitTの係数であるαは，想定するクラスタのメンバノードの数によって変化すると考えられる．メンバノードが少なければ，複数のセンサノードからCHDメッセージの同時送信による衝突があまり発生しないと考えられ，αは小さくなるし，逆に多ければ衝突が頻繁に起こりうるので大きくなると考える．そこで，αに関しては，想定するシナリオに合わせセンサノード上で動作するアプリケーションによって設定することとする．

このようなクラスタヘッド選出方法に関しては，2つの問題がある．一つは，CHDメッセージの衝突である．例えば，電池が新品のノードは，ほぼ同じ時刻にCHDを送信する．これは，無線アクセスに関するさまざまな従来技術を利用して解決可能であると思われるが，現在の実装では，ノードごとにランダムな値を加えてWaitTを決定することで回避している．

もう1つの問題は，電源残量の多いセンサノードが新たにクラスタに所属しようとした時，そのセンサノードをクラスタヘッドとして選出できないことが挙げられる．この場合，クラスタヘッドの電源残量が自分の電源残量よりも小さいため，式(1)では$WaitT$の値がマイナスとなってしまい，クラスタヘッドとして選出できない．この問題に対して，次節で述べるハイプライオリティスロットを用意することで解決を行う．

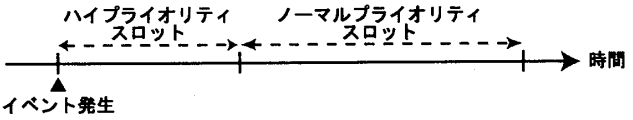

図2: ハイプライオリティスロットとノーマルプライオリティスロット

3.2.2 ハイプライオリティスロット

ハイプライオリティスロットとは，既存のクラスタヘッドより電源残量の多いセンサノードがクラスタに加わった場合に，そのノードが他のノードより先にCHDを送信するための時間として用意される．図2にハイプライオリティスロットと通常のCHD送信に使われるノーマルプライオリティスロットの関係を示す．ただし，上記のようなセンサノードが1回目に加わった時，前のクラスタヘッドの電源残量を正確に把握していないため，実際にクラスタヘッドになりうるのは2回目以降となる．

式(2)で$WaitT$の値がマイナスとなるようなセンサノードに対して，そのノードがハイプライオリティスロット内で$WaitT$がタイムアウトし，クラスタヘッドになるよう，別の計算式を用いて$WaitT$を求める．ハイプライオリティスロットの時間をγで表わすと，以下のような式になる．

$$WaitT = \beta * (Power_{ch} - Power_n) + \gamma \qquad (3)$$

これに伴い，式(2)でプラスの値になるセンサノードはハイプライオリティスロットより後にタイムアウトをしなければならない．そこで，以下に上記のようなセンサノードが式(1)のかわりに用いるハイプライオリティスロットを考慮した$WaitT$の式を示す．

$$WaitT = \alpha * (Power_{ch} - Power_n) + \gamma \qquad (4)$$

しかし，上記式(4)を用いると，センサノードの電源残量がほぼ一杯な場合でも$WaitT$の値はγ以上になる．これを回避するために以下のように各センサノードは式(1)，(3)，(4)を使い分けて$WaitT$を計算する．

1. まず式(1)を用い，$WaitT$の値を計算する．

1.1. $WaitT$の値がγ以下なら$WaitT$時間待機．

1.2. γ以上ならばステップ2へ．

2. $(Power_{ch} - Power_n)$の値を求める．

2.1. 値がプラスならば式(4)を用いて$WaitT$を求める．

2.2. 値がマイナスならば式(3)を用いて$WaitT$を求める．

これにより，$WaitT$の値を極力小さくしつつ，センサノードの電源残量が減った場合でも，$WaitT$の値が一次関数的に増えることがなくなる．

ここでα，β，γの各変数について簡単に考察する．γと$WaitT$係数であるα，βについては，クラスタに所属するセンサノードの数とセンサノードの電源残量の分散度合いに依存すると考える．各センサノードの電源残量が広い範囲で均一に分散している場合，αおよびβの値は小さくてよいと考えられる．逆に狭い範囲に集中している場合，αおよびβの値を大きくする必要がある．また，$(Power_{ch} - Power_n)$の値がマイナスとなるノードが多くなると，βおよびγの値を大きく設定する必要が

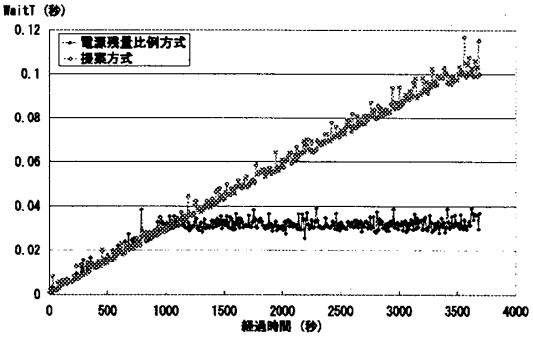

図 3: WaitT の変化の比較

あるだろう．この3つのパラメータに関しては詳しい考察が必要だと考えており，今後の課題として挙げられる．

4 評価

電源残量に基づいたクラスタヘッド選出方法を評価するため，ns-2 ネットワークシミュレータ上に実装し，式 (1) で現される従来の選出方法との比較を行った．シミュレーションは，ノード数 50，エネルギー初期値 2J，データ送受信の消費電力量は同じ，すべてのセンサノードが所属するクラスタ構築を 10 秒毎に 1 回行うようにした．また，提案方式のパラメータとして，α および β を 0.2，γ を 0.03 とした．この時の経過時間に応じた $WaitT$ の変化を示したものを図 3 に示す．

式 (1) で示される方法の場合，時間が経過するにつれ $WaitT$ の値が一次関数的に増加している．トリガが発生する度にクラスタを形成するための一定量のメッセージの送受信が行われ，センサノード自体の電源残量が一次関数的に減っていくためである．その結果，$WaitT$ の値自体も逆に一次関数のように増加する．

一方，提案方式では，1000 秒付近まで $WaitT$ の値は式 (1) の方法と同じように増えていくが，その後，ほぼ 0.03 秒付近で推移する．最初は式 (1) と同じ計算式のため，γ の値である 0.03 付近まで $WaitT$ は同じ増加をする．しかし，式 (1) で $WaitT$ の値が γ を越えた後，式 (4) を用いて $WaitT$ の計算を行うことで，ほぼ γ に近い値となる．この結果から式 (1) で表される従来の電源残量を基にしたクラスタ選出方法と比較し，提案方式の有効性を示せた．本方式により，クラスタを構築するまでの時間を短くできる．

センサノードのトリガ発生率と β の値を変えて $WaitT$ の変化の推移を評価した．トリガ発生率が低くなると，クラスタに所属しないセンサノードが増加し，クラスタの構成メンバの変化も多くなる．図 4，5 にトリガ発生率 50%で β 0.2 の場合，トリガ発生率 75%で β 0.4 の場合の $WaitT$ の推移を示す．それ以外の条件，パラメータは上記した評価と同じである．

上記 2 つの図から $WaitT$ の揺れ幅について考察を行う．式 (1) および (4) を用いて $WaitT$ を計算した時，図 4 では揺れ幅が大きい (標準偏差: 0.00222) のに対し，図 5 では小さい (標準偏差: 0.00208) ことがわかる．トリガ発生率が低下すると，クラスタの構成メンバが頻繁にかわり，構成メンバ数自体も少くなるため，$WaitT$ の揺れ幅が大きくなると考えられる．また，式 (3) を用いて $WaitT$ を計算する場合，β が異なるにもかかわらず，

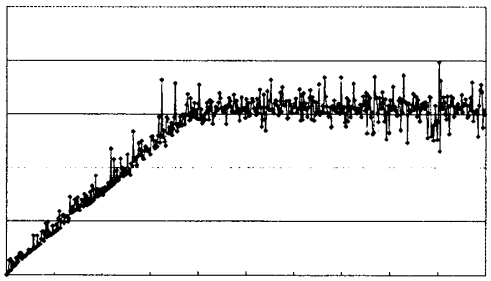

図 4: $WaitT$ の変化 (トリガ発生率 50%，β 0.2)

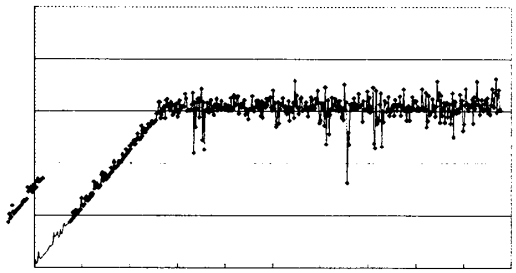

図 5: $WaitT$ の変化 (トリガ発生率 75%，β 0.4)

図 4 も図 5 も 同じような値を出している．これはトリガ発生率が低下しても，β の値が大きくなっても，同じように $WaitT$ の値を小さくするように影響すると考えられる．

5 まとめ

電源残量に比例した待ち時間でクラスタ形成を行うと，クラスタ形成時間が長くなり，消費エネルギーが増加するという問題を指摘し，これを解決するために，クラスタヘッドの電力残量をオフセットとして減算する方式を提案した．さらに，電力残量がＣＨより多いノードが存在する場合や，電力残量が十分にある場合などの例外的なケースに対処するための改良方式についても提案した．

今後の課題として，本論文では特定のパラメータのみの評価となっている．そこで，本提案方式で用いられている α や γ，センサノード数といった各パラメータの値を変えて評価を行い，どのようなネットワーク構成の場合，どのようなパラメータを設定すべきか検討することが挙げられる．

参考文献

[1] Heinzelman, W., Chandrakasan, A. and Balakrishnan, H.: Energy-efficient communication protocol for wireless sensor networks, *Hawaii International Conference System Sciences* (2000).

[2] 首藤幸司, ランバツェンゲェウテ, 西尾信彦: ユビキタスセンシングネットワークにおける自律的なデータ集約機構, 日本ソフトウェア科学会第 8 回プログラミングおよび応用のシステムに関するワークショップ (SPA 2005) (2005).

[3] Nagata, T., Oguma, H. and Yamazaki, K.: A Sensor Networking Middleware for Clustering Similar Things, *International Workshop on Smart Object Systems In Conjunction with International Conference on Ubiquitous Computing (Ubicomp 2005)*, pp. 53–60 (2005).