センサデータベースにおける協調質問処理を効率化する動的ルーティング手法

福島 海† 新美 礼彦† 小西 修†

†公立はこだて未来大学システム情報科学部 〒041-8655 北海道函館市亀田中野町 116- 2
E-mail: †{m1202142,niimi,okonishi}@fun.ac.jp

あらまし センサネットワークでは，ある集団における最大値や平均値取得のような，センサネットワーク中におけるセンサ間の協調的なセンシングデータ取得がよく行われる．このような処理を行う上で，センサデータベース環境では SQL ライクな質問処理言語を用いて，非常に様々な質問処理を協調的に行うことができる．このようなセンサデータベース環境の協調性を最大限に活かすためには，効率の良い協調質問処理の方式を提案することが必要となってくる．本研究では，質問処理実行時に頻繁に使用されるノード集合をネットワーク内で判断し，自動的に質問処理を効率化させるようなルーティングを行うアルゴリズムを提案する．

キーワード センサデータベース，協調質問処理，TinyDB，動的ルーティング

Dynamic Routing Method to Improve Cooperative Query Processing in Sensor Databases

Umi Fukushima† Ayahiko Niimi† Osamu Konishi†

† Future University Hakodate 116-2 Kamedanakano-tyo, Hakodate-shi, Hokkaido, 041-8655 Japan
E-mail: †{m1202142,niimi,okonishi}@fun.ac.jp

Abstract In wireless sensor networks, it is popular to fetch sensing data between in-network sensors cooperatively like max or average values in some sensor aggregations. In this process, sensor database environments present a solution to process various queries cooperatively by SQL-like language. In order to maximize the cooperativeness of sensor database environments, it is necessary to make a suggestion of the effective and cooperative query method. In this paper, we describe a dynamic routing algorithm that improves cooperative query processing in the way that in-network nodes find frequently used nodes while processing a query.

Keyword Sensor Database, Cooperative Query Processing, TinyDB, Dynamic Routing

1. はじめに

1.1. 背景

近年，ユビキタスコンピューティングの中でも注目される技術として，超小型コンピュータにセンサを搭載し，無線によりネットワークを形成してセンシング情報をやりとりする無線センサネットワークが存在する．無線センサネットワークでは，センサノードが電池等のバッテリで動作するので，電波が届きさえすれば特定のインフラを必要とせずに通信ができるなどといった利点がある．

無線センサネットワークの構築例として，動物の生態調査や作物の収穫時期の特定，販売・流通などのトレーサビリティ，病院・介護施設など様々な場面が検討されている．

センサネットワークを構築する無線センサ機器には様々なものがあるが，中でも MOTE は世界中で様々な研究に用いられている．センサ基盤と無線通信や CPU のある基盤は別々となっていて，利用者は好みのセンサ基盤を用い装着することが可能である．センサにより収集できる情報としては，光度，温度，音，加速度，磁気などがある．

センサネットワークを有効に利用するためには，個々のノードのセンシングデータのみを利用するだけでは不十分である．なぜなら，データを収集するだけならばノードに情報処理機能は必要ないからである．センサネットワークの最大の特徴は，センサノード自体が CPU を持つため，複数のセンサが自律分散的に協調してネットワーク内部でセンシング情報を解析し，データを閲覧するときには利用者が欲する情報のみに絞らせることができるという点にある．しかし，ある目的専用にネットワーク内部でセンシング情報を解析

させるためには，組み込み機器専用のプログラミング言語を習得してコーディングを行わなければならず，気軽に作成することが出来ない．

この問題を解決するシステムとしてセンサデータベースという，センサネットワークを 1 つのデータベースとして扱うミドルウェアの研究が行われている[1][3][4]．この中には MOTE 上の TinyOS[6]という，無線センサ機器用のコンパクトな OS 上で動く TinyDB [8]という質問処理システムが存在する．TinyDB では，SQL に似たクエリを発行することでセンサ機器をデータベースであるかのように運用できるという利点があり，集約関数などを用いて個々のセンサ間で計算させながら協調的に情報を取得できる．クエリ例を以下にあげる．

```
SELECT AVG (temp) FROM sensors
WHERE loc in (0, 0, 100, 100) and light > 1000 lux
SAMPLE PERIOD 10 s
```

上記のようなクエリをネットワークに送信するとする．そうすると，矩形 (0,0,100,100) で定義される領域中の明るいノード(1000 ルクスより高いもの)をセンサ間で協調して判断し，温度情報の平均値をセンサ間で解析・計算して，10 秒ごとに PC へ報告するという動作が行える[8]．こうすることにより，利用者はある範囲における明るい部分の温度情報が，まるで 1 つのセンサから情報を受けとるようにして閲覧できる．

図 1 に TinyDB によるセンサデータベースの概略図を示す．矢印がクエリの伝播を表し，その質問処理の結果が矢印の逆向きにルートへと報告される．これは 1 回分のセンシング情報の取得例であるが，実際にはある一定期間ごとに連続更新されるテーブルとなる．

1.2. 研究の目的

センサデータベースによる前述のような協調性を最大限に活かすためには，効率の良い協調質問処理方式の提案が必要となる．センサネットワークにおいては消費電力の削減という問題は最重要課題の 1 つであり，センサデータベースでは質問処理をできるだけ長時間持続させるため，複数のクエリ結果をルートノードまで配送する途中の中継ノードで集約し 1 つにまとめる手法が提案されている[5][9]．これにより，データの通信量を減らして消費電力の削減を図っている．しかし，このような協調的なデータ処理にも，集約を行えるノードが結果を報告するノードから離れすぎていると，無駄な中継ノードが増え，その利点を活かせなくなってしまうという問題点が存在する．

本論文では，集約を行えるノードが結果を報告するノードから離れている場合，無駄な中継ノードが増えて集約による利点を活かせなくなるという問題を解決する．そのために，ネットワークのルーティングの観点から見たデータの質問処理の向上を行う方法を提案する．センサネットワークにおけるルーティング分野の研究は非常に様々に行われているが，我々はネットワーク上におけるクエリ結果の集約[5][9]を効率化させるための動的ルーティングのアルゴリズム提案を行う．これにより，センサデータベース環境におけるさらなる通信量の削減を図ることが目的となる．このように，センサデータベースにおける集約のような協調質問処理に関する問題を解決することも，今後のセンサネットワークに重要な研究課題となるだろう．

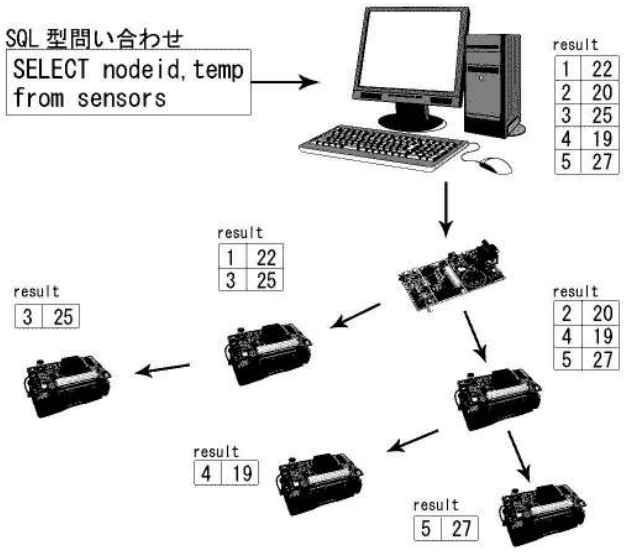


図 1．センサデータベース概要

2. 関連研究

センサデータベース環境を改善するための研究は従来から行われている．TinyDB 向けの論文[1]では ACQP (ACquisitional Query Processing)という処理方法を提案している．ACQP では，伝統的な分散クエリ処理がマルチホップネットワークにおいて省電力化・通信量の削減に効果を持つことを，センサネットワーク向けに進化させたクエリ処理として提案している．データ取得のコストと順序・いつ取得すべきかなどの点に着目して省電力化を実現するなど，非常に消費電力に気を使った提案を行っている．また，TinyDB への改良として SRT というネットワーク構築方法も合わせて提案している．SRT はネットワーク構築時に，自ノードより下位木のノードの何らかの定数属性(nodeid,場所など)を格納する．そしてクエリが来たとき，その定数属性を見て子ノードに適用されないなら，クエリを

破棄し電力消費の軽減を実現している．また，ネットワーク構築には clustered approach という，各親ノードが持つ定数属性の範囲が最小になるような構築法が提案されている．しかし，この論文[1]では一旦ルート木が構築されると，電力切れや移動などによりノードをロストしない限り，変更は行われない．我々は，クエリによる質問処理を行っている時に，良く使用されているノードによる動的なルーティングを行って，ネットワークの最適化を図る．

REED[7]では，TinyDB に基づいたロバストで効果的なフィルタリングとイベント検出システムを提案している．まず外部で用意したセンシングデータのフィルタリングに使う条件テーブルをネットワークに投げておき，SQL の表結合(join)の機能を実装しフィルタリングを行うという拡張を行っている．こうすることで，より柔軟な条件付けの表現を可能にし，その結果様々なイベント検出の条件付けも可能にしている．また，比較的高いデータレートのサンプリング(例えば，100Hz)で要求をモニタリングする必要があるときに，条件付けによって通信量を減らすことは，多くの産業の場面で特に重要であるとしている．このように，REED[7]では SQL の拡張を行うことによりセンサデータベースの協調的な質問処理を向上させているが，我々はネットワークの観点から，質問処理を効率化させる動的なルーティングの提案を行う．

論文[5]は，センサデータベースでの質問処理中に，グループ関数(COUNT, MIN, MAX, SUM など)による集約クエリの実行の仕方をどのように行うかという点に着目した論文である．具体的には，途中の中継ノードにおいてデータの集約処理を行い，集計結果のみを木グラフの上流ノードへ配送する提案をしている．その結果，データの配送を削減し消費電力の減少を図っている．しかしこの論文では動的なルーティングなどの記述はないため，集約を行えるノードが離れていた場合，ルートに近いノードで集約を行うほど無駄な中継ノードができてしまい，集約による有効性が失われてしまうことがある．我々はこの問題点を解決するために，現在のルート木を変更させてより集約の利点を活かせるようにするアルゴリズムを提案する．

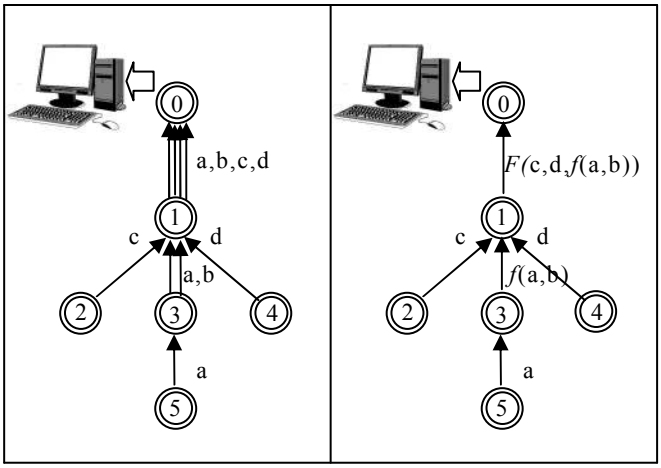
またセンサデータベースを利用した例として，実世界情報ストリームの取得を TinyDB にまかせ，それらを利用するための統合環境を構築した例がある[2]．この論文ではストリーム統合システムをベースに，センサネットワークに連続的にクエリを送信して情報を取得するためのセンサネットラッパーを開発し，それらをまとめた統合環境を構築している．しかし，この論文は TinyDB をセンシング情報の取得のために利用しているだけであり，センサデータベースの内部，特にセンサ群の協調による質問処理を改良するといったことは行われていない．我々は提案した拡張をセンサデータベースに適用させ，評価を行い質問処理の効率化を図る．

3. クエリ結果の集約

論文[5]では，データ配送途中の中継ノードで集約を行えばデータの配信数が減り電力削減に効果があることを示していると前章で述べたが，それを通常のサーバベースのデータ配信法と比べたものを図 2 に示す．

図 2 は簡略化したセンサネットワークの構成図である．まず前提として，ネットワークはシンクノードをルートノードとした木構造とし，ルート以外のノードは親ノードを 1 つだけ持つとする．無線通信はブロードキャストであるため，基本的に電波が届けばどのノードとも通信はできる．しかしクエリ結果の報告やクエリの配信では，通信の宛先に親ノードや子ノードのアドレスが設定されているため，そのノードのみが受信することで木構造に沿って通信が行われるとする．(a)は通常のサーバベースのデータ配信方法である．一番下の 5 番のノードは 3 番・1 番のノードをホップしてルートに a というデータを報告する．他のノードも同様にして報告し，計 9 つのデータ配信が行われる．これに対し(b)では各ノードで集約を行い，5 番ノードからの a というデータは，3 番ノードの b というデータと共に集約関数 f によって 1 つにまとめられ，f(a,b)という 1 つのデータが報告される．同様にして 1 番ノードでも集約が行われ，計 5 つのデータ配信が行われる．(a)と(b)を比べると，2 倍近くのデータ配送量の違いがあることが分かる．

このようにして，中継ノードで集約を行うことでデータ配送量を減らし，電力消費の削減を行っている．



(a) (b)

図 2. サーバベース配信法と集約を使った配信法

3.1. クエリ結果を集約する手法

また論文[5]では，集約クエリの計算は伝播フェーズと集約フェーズの，2 つのフェーズから成るとしている．伝播フェーズではセンサネットワーク上の全ノードに集約クエリ(SQL による質問)を投げかける段階である．全てのノードにクエリが渡ったら，そのクエリの質問にしたがって集約したデータを報告する集約フェーズが開始される．

クエリ結果を報告するには，まず葉のノードからデータを報告しなければ中継ノードの値も報告できない．よって自分が葉のノードであるかを判断させる必要がある．以下にその過程を示す．

1) 葉ノードであるかの判断は，まず図 3 の(a)のように，あるノード p (図中の黒色のノード)が子ノードに集約クエリ a を送信する．

2) その後図 3(b)のように，p の子ノードがさらに子ノード(p の孫ノード)向けに a を送信するのを p が受信できたら，p は葉ノードではないというように判断する．センサ機器の無線通信はブロードキャストであるため，p が子ノードに通信できれば，その子ノードが孫ノードに通信したときに p もその通信を受信することが出来る．

3) そして図 3(c)のように，p が a を送信したときに何も反応がなければ(p に子ノードがなければ)，自身は葉ノードであると判断する．

4) 最後に(d)のように親にクエリ結果を報告する．子ノードがいるならば，少しの時間待って子ノードの結果の報告を待ち，来たら自分自身の値と子ノードの値を適用した値を親ノードに送る．このような形でクエリ結果の集約は行われる．

3.2. クエリ結果の集約における問題点

クエリ結果の集約は無駄なデータ配送を無くすという点で理想的であるが，それはクエリ結果を報告するノードの比較的近くの親ノードで集約できた時に有効である．集約を行えるノードがクエリ結果を報告するノードから離れていた場合，ルートに近いノードで集約を行うほど無駄な中継ノードができてしまい，集約による有効性が失われてしまうことがある．

図 4 にその概要を示す．丸はセンサネットワークのノードを示し，黒色はクエリ結果を返すノード，灰色は集約を行うノードを表している．

1) 近くの親ノードで集約できる場合(図 4(a))

このような場合，集約できるノードが近くにいるため，クエリ結果の集約による効果を最大限発揮できるといえる．

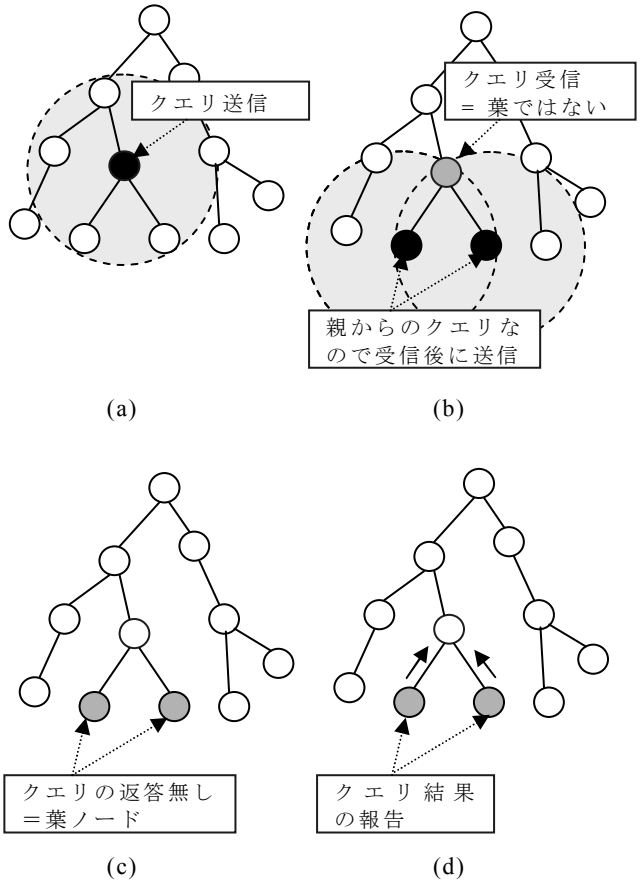


図 3．葉ノードであるかを判断する過程

2) 集約できるノードが離れている場合(図 4(b))

集約できるノードが一番上のルートノードのみであるため，無駄な中継ノードが出来てしまい集約による効果を発揮できない．配送されるデータ量も，(a)よりも(b)のほうがクエリ結果を返すノードが少ないにもかかわらず，(b)の方が多くなっている．ノードが増えてルート木がさらに深くなってくると，無駄な中継ノードができて集約による利点が抑えられてしまうような事態が少なくない頻度で発生する可能性がある．

3) 2)による問題点の改善(図 4(c))

無駄な中継ノードを減らし，クエリ結果の集約を最大限に活かすためには，現在のルーティングを変更させて，クエリ結果を報告するノードに，より近い親ノードで集約できるようにすればよい．図 4(c)では，右側の黒色ノードの親ノードを，左上のノードに変更させている．そうすると，2 つ中継ノードを経由しなければ集約できなかったところが，経由するノード無しで集約できるようになる．この例では 2 つ分のノードしか変わらないが，もっと階層が深くなるとより中継ノードが減らせるようになり，恩恵が大きくなるだろう．

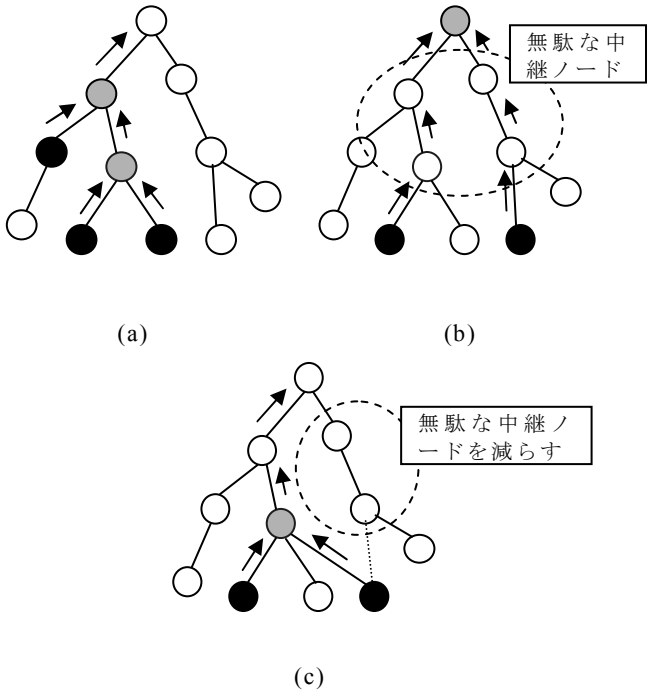


図 4. クエリ結果の集約における問題と解決法

4. 提案するアルゴリズム

前章で述べたクエリ結果の集約における問題の解決法である，構築済みのルート木を変更させ，クエリ結果を報告するノードに，より近い親ノードで集約する手法について述べる．

4.1. 基本的考え方

前提として，ネットワークはシンクノードをルートノードとした木構造とし，ルート以外のノードは親ノードを 1 つだけ持つものとする．そして，クエリ結果の報告やクエリの配信では，木構造に沿って通信が行われるとする．その上で，我々のアルゴリズムの基本的な考え方は，

1) クエリ結果の集約による通信量の最適化をさらに生かすため，無駄な中継ノードを減らす．
2) そのために，頻繁に結果を報告するノードと，結果を報告する経路上のノードは集約に適するため，それらのノードに優先度付けをする．
3) 優先度とは，集約に有利なノードを判別するための指標であり，1 が最も優先順位が高く，2，3 と減るにしたがって低くなっていくものとする．
4) 優先度付けされた後，同じように頻繁に結果を報告するノードがいた場合，近隣に，より優秀な優先度を持ったノードがいれば親を変更する．

このような優先度付けによる動的ルーティングを行うものである．優先度を設定する理由は，ただ近隣にあるノードで集約に適していれば何にでも親を変更するのではなく，近隣のノードの中で最も集約に適したノードに親を変更させるためである．また，この優先度の値はクエリ結果を報告するときにメタデータ(送信先，送信元アドレスなどが含まれる)として一緒に送信されるものとする．これにより，無駄なパケットを増やすことなくルーティングを行うことが出来る．

4.2. 手順

事前条件として，センサデータベース側でルート木は構築済みであり，ネットワーク内には 1 つの質問処理を行うクエリが投げかけられているとする．ここでいうクエリとは，全てのノードから値を取得するものではなく，グループ関数(AVG，MAX，MIN，SUM など)による値の集約と，where 句による値の取得条件が設定されているものとする．このようなクエリによる質問処理では，センサの値が相当な頻度で変化する環境を除けば，頻繁にクエリ結果を報告するノード群とそうでないノード群に分かれていく．

図 5 にアルゴリズムの簡略化したフローチャートを示す．このアルゴリズムは各ノードがクエリ処理を行っているときのものなので，開始点はクエリ処理の部分である．また，このアルゴリズムはルートノード（直接シリアルケーブル等で PC に接続されたノード）以外の，すべてのセンサノードが行うものである．大まかな流れとしては，最初はルート木に優先度が設定されていないので，まず優先度を設定する必要がある．その後設定された優先度を基準に親変更を行う．優先度付けは図 5 の(a)と(b)の処理により行われ，親変更は図 5 の(c)の処理によって行われる．

以下に構築済みのルート木を変更させ，クエリ結果を報告するノードに，より近い親ノードで集約する手法の概要について述べる．

1) 以下の場合，集約に適するため，そのノード(ノード n とする)の優先度を 1 とする．
   1-1) ノード n が優先度 1 ではない．
   1-2) クエリ処理時にノード n が頻繁にクエリ結果を報告している(例：10 回連続でクエリ結果を報告)．
   1-3) 近隣に優先度付けされたノードがいない．
2) 優先度を設定したノードは自分の優先度をメタデータに設定しクエリ結果を報告する．
3) 優先度付けされたクエリ結果を受信した時，自分の優先度が 1 ではないときは以下の処理を行う

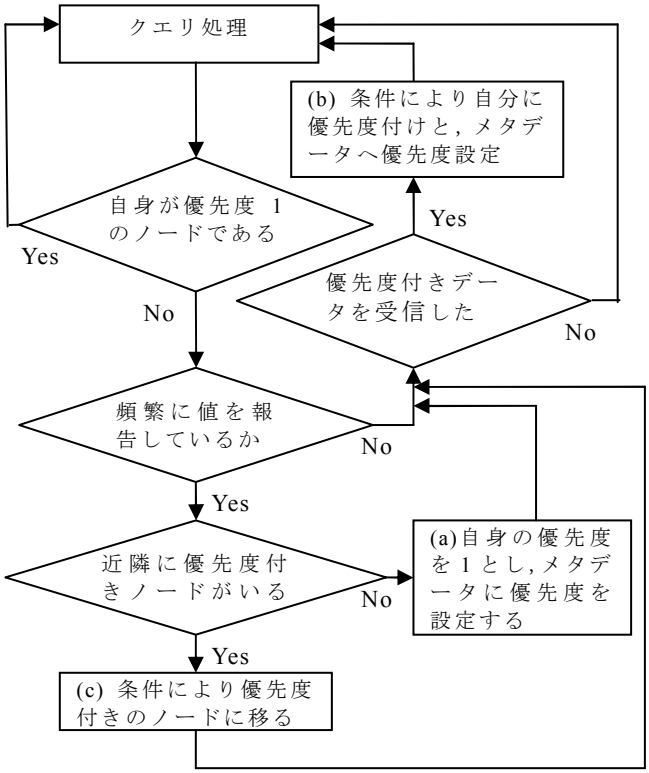


図 5. アルゴリズムの簡易フローチャート

3-1) そのクエリ結果を送信したノードの親ノードであった場合，優先度を 1 と設定する．

3-2) 自分が優先度 1 であるノードの子ノードであれば優先度 2 と設定し，同じように優先度 2 であるノードの子ノードならば優先度 3 と設定する(後に説明する図 7(a)のようになる)．

3-3) 受信した優先度が自分の優先度よりも高いとき，新しい優先度で上書きする．例えば，最初優先度が 3 であったノードに優先度 1 をつけるよう要求があった場合，1 を優先度として設定する．こうすることにより，新しく優先度付けしたノードによるデータの集約が行いやすくなる．

4) 優先度付きのデータを受信したノードは，それを近隣ノードの情報として保持する．その情報から近隣の優先度付きノードを調べる．

5) あるノード m が以下の条件を満たす場合，親ノード変更の処理に移る．

5-1) ノード m は優先度 1 ではない．

5-2) クエリ処理時にノード m が頻繁にクエリ結果を報告している．

5-3) 近隣に優先度付けされたノードがいる．

6) 5)を満たす場合，ノード m の親ノードが優先度 p で，近隣ノードの優先度の最大値を q としたとき，以下の場合により親を変更するかどうか決定する．

6-1) ノード m の親ノードが優先度なし，また，p＞q の場合，一番高い優先度を持つノードに親ノードを移す．近隣に同値の優先度を持つノードがいる場合，一番初めに優先度付きのデータを受信したノードに移るものとする．

6-2) p＝q，また，p＜q の場合は近隣ノードに親を移さず現在の状態を保つ．

7) パケットのロストや電源断などによって親ノードを見失ったとき，見失った親ノードが優先度を持つ場合は以下の処理を行う．

7-1) 見失ったノードが優先度 2 以下の場合

ノード n のクエリ結果を報告する経路上のノードではないため，自身とその下位ノードの優先度の値だけをリセットする(図 6 の(a))．

7-2) 見失ったノードが優先度 1 の場合

ノード n のクエリ結果を報告する経路が使えなくなったということであり，ノード n による優先度は意味を成さなくなる．よって設定されている，ノード n による優先度を一旦リセットし，新たに違う経路の優先度付けを行わせるようにする(図 6 の(b))．

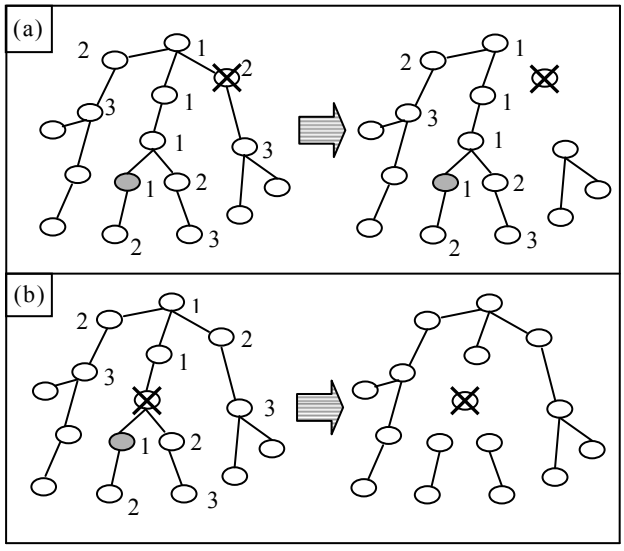


図 6. ノードを見失ったときの振る舞い

4.3. 具体的な適用例

このようなクエリ結果の集約における動的ルーティングの詳しい例を以下の図 7 に示す．白の丸はセンサネットワークのノードを示し，ノードとノードを結ぶリンクは親ノードと子ノードの関係を表している．

また，ノードの横に示した数字は各ノードに設定されている優先度の値を表す．

1) 図 7(a)では，灰色のノードが連続してクエリ結果を報告したノード(前述のノード n)を表し，自身の優先度を 1 と設定している．その後クエリ結果を報告すると同時に優先度の値を送信し，自身からルートまでの親ノードを経由して報告する経路を優先度 1 と設定する．次に優先度 1 の報告データを受信した子ノードは優先度 2，同じように優先度 3 と設定している．
2) 図 7(b)では，黒色のノードが連続して呼ばれたノードを表していて，灰色の大きな円は無線の届く範囲を示している．無線の届く範囲で優先度が設定されているノードは，優先度 3 の親ノードと優先度 1 のノードである．p＞q であるため 6-1) の条件が適用され，親ノードは優先度 1 のノードに変更される．これにより，優先度 3，2 のノードによるデータ中継がなくなり，効率良いデータ配送が実現できる．
3) 図 7(c)では，右側の黒色ノードが連続で呼ばれ，灰色の無線通信の範囲内には親ノードが優先度 2，近隣にも優先度 2 のノードが存在する．p＝q であるため，6-2) の条件が適用され，親ノードは変更しない．このような形で動的ルーティングが行われ，最終的には図 7(d)のようなルート木に更新される．

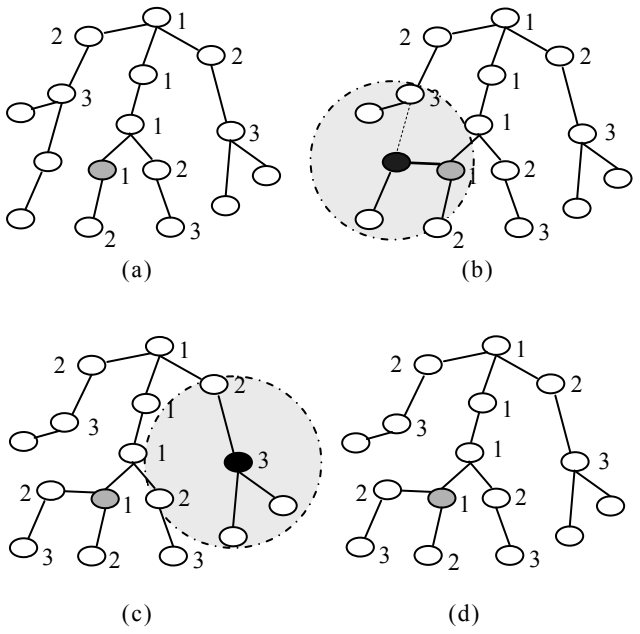


図 7．動的ルーティングの実行例

5. 実験と評価

我々のアルゴリズムは質問処理中における不要な中間ノードを取り除き，データ通信量の削減を目的としたものである．不要な中間ノードを取り除けられれば，全体的なホップ数が減少するはずであり，その結果全体的な通信量の削減を図ることができる．よって，ここではクエリの結果を報告したときの，集約されるノードまでの距離に着目し，評価を行う．

5.1. 実験環境

実装するセンサデータベース環境には，TinyDB[8]を用いた．これは，TinyDB がクエリ結果の集約を行えるセンサデータベースの環境であるからである．よって，これにアルゴリズムの実装を行った．また，TinyDB は TinyOS で動くシステムであるため，TinyOS 用のシミュレータ TOSSIM[10]を使用した．TOSSIM を用いることで，仮想的なノードによるセンサネットワークを構築でき，パケットの解析やデバッグなど，その環境の中で TinyDB の様々なシミュレーションを行うことができる．パケットの損失率などは実際のセンサノード(MOTE)の実験データを用いているため，ほぼ正確なシミュレートが期待できる．

実験ではパケットの損失はなしとし，25 個の仮想的なノードを使用して 5×5 の正方形の形に配置し，隣り合うノードとのみ通信が行えるような環境とした．また実験を行う際，事前に頻繁にクエリ結果を報告するノードとそれ以外のノードに分け，頻繁に報告するノードはセンサの値として light = 1000 を報告し，それ以外のノードは light = 500 を報告する．そして，クエリにはセンサ値の取得条件として light > 800 を設定することで，light = 1000 と設定したノードを頻繁に報告させることとした．しかし，同じタイミングで親変更を判断させると，それら全てが優先度 1 となり親変更が行えないため，アルゴリズムが有効に働かなくなってしまう．そのため，各ノードに 5～15 のランダムの数値を設定させ，その設定した数値分，クエリ結果を報告すると優先度付けや親変更を行うようにした．

そして，TinyDB 側からネットワーク内に以下のような，およそ 4 秒ごとの周期で平均値を PC 側で取得するクエリを投げかけた．

```
SELECT AVG (light) FROM sensors
WHERE light > 800 SAMPLE PERIOD 4096
```

さらに，TinyDB では起動するごとにルーティングが行われ，毎回同じ木構造にはならないので，20 回 TOSSIM によるセンサネットワーク構築を行ってクエリを投げかけ，ルーティングの変化を記録した．また，クエリ結果を頻繁に報告するノードの数によっても結

果が変化すると思われる．よって，頻繁に報告するノードを 5 個, 10 個, 15 個, 20 個の 4 パターンに設定した．

5.2. 実験結果

実験結果を図 8 に示す．実験では提案するルーティングを行う前の 1 周期分の全ホップ数と，行った後にルーティングが安定した 1 周期分の全ホップ数を測定した．あらかじめ親変更を行うノードが決まっているため，ルーティングは時間が経つと安定し，親変更を行うノードが 3 周期程度変化しなければ安定したと見なした．実験ではおよそ 30 周期で安定した．そして，それぞれの試行数 20 に対し通信量が 1 ホップ以上減少した割合と，減少したホップ数の平均値と分散をグラフ化した．横軸がクエリ結果を頻繁に報告するノード数であり，左側の縦軸が棒グラフに対応した平均値と分散，右側の縦軸が折れ線グラフに対応した割合を示している．全体的な平均としては，減少した試行の割合が 50%，減少ホップ数が 0.76 となった．

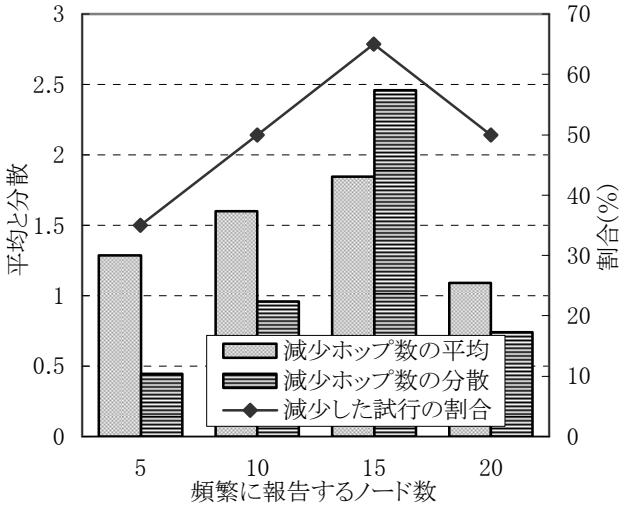


図 8. 実験結果

5.3. 実験の評価

この実験により，ネットワークの通信量を減らすことに成功した．図 8 では，頻繁にセンサ値を報告するノードが 15 の時に一番良い結果となり, 全試行に対し 65%の割合でホップ数を減少させることができた．また，最大で 7 ホップ減少させた試行もあり，このアルゴリズムでは頻繁にセンサ値を報告するノードが多すぎず少なすぎないときに効果を発揮する事が分かった．

他の場合を見ると，図 8 のノード数が 20 個のときは，頻繁に報告するノードが密集しているため，どの親に変更してもホップ数が変化しないことが多かった．この場合に関しては，改良を行っても減少できるホップ数に限界があるため，アルゴリズムの限界であると考えられる．図 8 のノード数が 5 個のときは，優先度がルート木全体に設定されないことが多く，そのため親変更を行えるノードが少なくなり，ホップ数が変化しない場合が多かった．この場合は優先度付けの改良などを行ってルート木全体に優先度を設定させれば，ホップ数を減少できると考えられる．

6. おわりに

本研究では，センサデータベース環境におけるクエリ結果の集約による通信量の最適化を生かすため，頻繁に結果を報告するノードと集約に有利なノードに優先度付けをし，優先度を持ったノードへ移るような動的なルーティングを行うアルゴリズムを提案し，実験により有効性を検証した．

今後は課題として，ノード数を増やした場合やパケットの損失を考慮に入れた実験を行うこと，また，ネットワーク全体のオーバーヘッドを考慮にいれたアルゴリズムの提案が考えられる．

参　考　文　献

[1] S. R. Madden, et al., TinyDB: An Acquisitional Query Processing System for Sensor Networks, ACM Transactions on Database Systems, Vol.30, No.1, pp.122-173, March 2005

[2] 山田真一，渡辺陽介，北川博之, 実世界情報ストリームの高度利用のための統合環境，DBSJ Letters, Vol.4, No.1, pp.105-108, June 2005

[3] Intanagonwiwat C., el al, Directed Diffusion for Wireless Sensor Networking, IEEE/ACM TRANSACTIONS on Networking, Vol.11, pp.2-16, February 2002

[4] Yao Y. and Gehrke J., The Cougar Approach to In-Network Query Processing in Sensor Networks, ACM SIGMOD Record, Vol.31, No.3, pp.9-18, September 2002

[5] Samuel R. Madden, et al., Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks, Workshop on Mobile Computing and Systems Applications, pp.49-58, 2002

[6] Hill J., Szewczyk R., et al., System Architecture Directions for Networked Sensors, Architectural Support for Programming Languages and Operating Systems, pp.93-104, 2000

[7] Daniel J. Abadi, Samuel Madden and Wolfgang Lindner, REED:Robust, Efficient Filtering and Event Detection in Sensor Networks, Proceedings of the 31st VLDB Conference, pp.769-780, 2005

[8] S. Madden, et al, The disign of an acquisitional query processor for sensor network, ACM SIGMOD international conference on Management of data, pp.491-502, 2003

[9] S. Madden, et al, TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks, In Proceeding of OSDI, pp.131-146, December 2002

[10] P.Levis, et al, TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications, Proc. of the First ACM Conference on Embedded Networked Sensor Systems (SenSys), pp.126-137, 2003