社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－DPS－134 (32)
2008－CSEC－ 40 (32)
2008／3／7

センサネットワークにおける無線通信を利用した効率的ソフトウェア配布方式

宮丸卓也[†] 峰野博史[††] 寺島美昭[‡]
徳永雄一[‡] 水野忠則[††]

センサネットワークを構成するセンサノードには制御用の組込みソフトウェアが動作しており，それらのソフトウェアに変更が生じた場合はソフトウェア更新が必要となる．現在，大量のノードへ効率的にソフトウェアを配布する方式として無線通信を利用したものが研究されているが，それらの多くはネットワーク全体における電力消費量や配送速度の改善を目指し，個々のノードの性能や環境を考慮した配送は行われていない．本研究はパイプライニングという並列配送技術を想定し，グループごとに配送形態を選択可能な方式を提案することで，個々のノードのために局所的な性能改善を目指す．

Efficient Software Distribution with Wireless Communication for Sensor Networks

Takuya Miyamaru,[†] Hiroshi Mineno,[††] Yoshiaki Terashima,[‡]
Yuichi Tokunaga[‡] and Tadanori Mizuno[††]

Sensor nodes have some embedded softwares for control in sensor networks. If these softwares are changed, it needs software update. Currently, many software distribution techniques with wireless communication have been researched to distribute software to many nodes. However, these methods set the improvement of whole network performance as a goal. So they cannot consider each node environments. In this paper, we assume the parallel distribution technique called Pipelining and propose the method that can set the distribution configuration, and try to improve the performances locally for each nodes.

1. はじめに

センサネットワークは比較的新しい技術であることから，アドホックネットワークや位置推定，データ処理手法など発展途上の技術を多く含んでいる．それら技術の一部はセンサネットワークを構成する無数のセンサノードに組込みソフトウェアとして実装されている．発展途上の技術であるため，場合によってはソフトウェアの変更が必要となる場合が生じる．そのような場合，自動的にセンサノードへソフトウェアを配布し，ソフトウェア更新を効率的に行う必要がある．現在，無線マルチホップ通信を利用し，一度に大量のセンサノードへソフトウェアを配布する方式が研究されている．その多くはネットワーク全体の配送時間や電力消費量，信頼性の改善を目指し，個々のノードの性能差や配置環境に応じた配送というものを考慮していない．しかし実際のセンサネットワークでは，各ノードの配置環境は全て同じというわけではなく，電力残量が異なったり，障害物などによって通信環境の悪い場所も存在する．また現在の汎用コンピュータ環境がそうであるように，今後センサネットワークにおいても，様々な性能のセンサノードが混在する環境が想定される．そのような状況下でソフトウェア配布を行う場合，性能の低いノードや配置環境の悪いノードに対しては，配送性能を落としてでも環境にあった配送形態を割り当てるなど，個々のノードに対して状況に応じた配送形態を選択可能な手法が必要である．

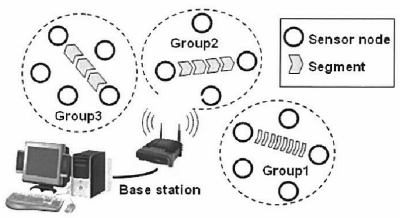

図1 ローカルパイプライニング

本稿ではパイプライニングと呼ばれるデータの並列配送技術を想定し，その配送性能を左右するセグメント分割数を，図1のように複数のセンサノードで構成されるグループに割り当てることで，各グループごとに状況に応じた配送性能を実現するローカルパイプライニングを提案する．またシミュレーションによる評価を行い，電力消費を左右するメッセージ送信回数などを局所的に増やしたり，減らしたりすることができることを確認した．

2. 関連研究

既存の無線マルチホップ通信を使ったソフトウェア配送

† 静岡大学大学院情報学研究科
Graduate School of Information, Shizuoka University
†† 静岡大学情報学部
Faculty of Information, Shizuoka University
‡ 三菱電機株式会社
Mitsubishi Electoric Corporation

方式は，主に電力消費や信頼性の改善を目指している[1]．最も原始的なソフトウェア配布方式はフラッディングであり，受信したデータをそのまま近隣ノードへ再ブロードキャストする．この方法は冗長な送信が多く，消費電力が大きく，信頼性も低いため使われていない．Deluge[2]は3ウェイハンドシェイクを利用し，不必要なデータ送信を除外することで電力効率を改善している．しかし3ウェイハンドシェイクを使うことでコントロールメッセージが増加し，近隣ノード数の多いネットワーク中央部で衝突が多発して信頼性が低下していた．MNP[3]ではコントロールメッセージを送信する代表者を選択するアルゴリズムを採用し，ネットワーク中央部での送信を抑制して信頼性を高めた．またMACプロトコルにTDMAを採用することでメッセージ衝突を回避し，信頼性を高めるSprinkler[4]などの研究がなされている．

これらの多くのプロトコルはネットワーク上の全てのノードへソフトウェアを配布する全体更新を想定しており，ネットワーク全体で電力消費を最小化，信頼性や配送速度を最大化しようとしている．その結果，電力残量の少ないノード，性能の低いノードなどは配送性能に考慮されず，ノード1台に対してかかるメッセージ送信回数などが大きな負担となっているのが現状である．

3. ローカルパイプライニングの提案

3.1 概　　要

データを並列配送することでソフトウェア配布を高速化するパイプライニングにおいても，ネットワーク全体の配送性能を最大化しようとする．配送性能はセグメント分割数というパラメータによって左右されるが，ネットワーク全体に対して1つの値が割り当てられる．筆者らは，ネットワークを複数のセンサノードで構成されるグループに分け，それぞれのグループにセグメント分割数を割り当てることで，ネットワークの局所でグループの環境に応じた配送性能を設定可能なローカルパイプライニングを提案している．

本方式では，性能の低いノードや電力残量や電力供給源の貧弱なノードの多いグループに対してはメッセージ送信回数や必要記憶領域などの負担を減らし，余裕のあるノードの多いグループには負担によらず，更なる配送高速化を実現する設定が可能となる方式を目指す．

3.2 パイプライニング

本方式は複数のパイプライニングで構成されるため，はじめにパイプライニングの仕組みと特性について述べる．パイプライニングは並列配送によってネットワーク全体へ高速にデータを拡散させる技術である．その特徴はデータをそのまま配送するのではなく，図2のようにセグメントと呼ばれる単位に分割して管理することである．更にセグメントは複数のパケットで構成され，複数のパケットを連続して配送することで1セグメントを配送する．　図3に通信半径が1ホップのセンサノード5台を直列に配置した場合のパイプライニングの配送過程を示す．パイプライニングでは複数のセグメントをオーバーラップさせて配送することができるため，全体の配送時間を短縮できる．例えばノード4が5に対して1番目のセグメントを配送していると同時に，ノード1は2に対して2番目のセグメントを配送することが可能となる．隠れ端末問題を避けるためセグメントは間隔を空けて配送しなければならないが，ネットワークサイズが十分に大きい場合，十分に配送時間を短縮可能である．n台の直列ネットワークにおいてサイズpのデータをm分割した場合，配送時間T_{np}は次の式で表現できる．

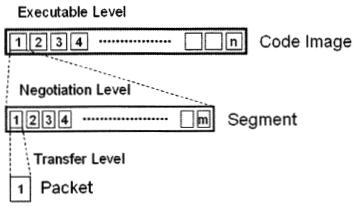

図2　データ構成

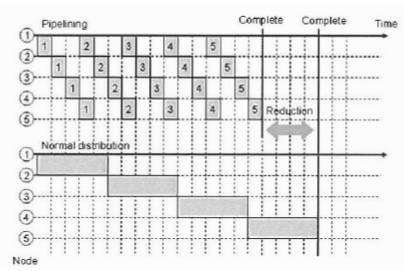

図3　パイプライニングによる配送

$$T_{np} = (n + 3(m-1)) \cdot p/m \quad (1)$$

3.3 セグメントの交渉

データは複数のセグメントで構成され，不必要なセグメントが送受信されないようにする必要がある．Delugeでは3ウェイハンドシェイクを使って冗長なデータ送信を抑制したが，セグメントの配送にも3ウェイハンドシェイクが適用されている．3ウェイハンドシェイクとは，ADV，REQ，DATAという3種類のメッセージで構成されるデータ送受信のための交渉手段である．はじめにノードはセグメントに関する情報が記されたメタデータをADVメッセージとして近隣ノードへ通知する．ADVメッセージを受信したノードは，通知されたセグメントが必要かどうかを判断し，必要ならばREQメッセージを返信する．最後にREQメッセージの受信後，セグメントを構成するパケットを連続して送りつけることで1つのセグメントの配送が完了する．REQメッセージを送らなければ，大きなセグメントのデータ本体が送信されることがないため，送信回数による消費電力を削減できる．

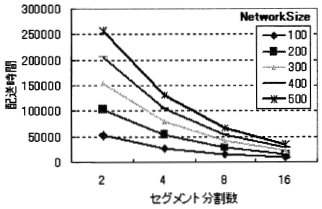

図 4 セグメント分割数と配送時間

また 3 ウェイハンドシェイクは，セグメントの再送にも適した手法である．ノードは 1 セグメント分のパケット管理ベクトルを保持しており，パケットを受信するとビットを立てる．もしセグメントの配送中にいくつかのパケットが欠落した場合は，次の ADV メッセージの受信時に，ベクトルを REQ メッセージに含めて返信することで，欠落パケットのみを再送できる．

3.4 セグメント分割数の特性

パイプライニングの配送性能を左右するのが，配送データをいくつのセグメントに分割するかというセグメント分割数 m である．配送性能は (a) 全体の配送時間，(b) メッセージ送信回数，(c) パケット管理領域量，(d) ノード 1 台の配送時間という 4 つに分けることができ，セグメント分割数を増減させることで (a)〜(d) の性能が変わる．1 つ目の (a) 全体の配送時間は，全てのノードへデータを配送するまでに必要な時間であり (1) 式によって算出可能である．(1) 式からセグメント分割数を変化させたときの全体の配送時間を示す図 4 が得られる．この図からデータをより細かく分割することによって並列度が増し，配送時間をより短縮できることがわかる．

(b) メッセージ送信回数は消費電力に大きく関わる．これはメッセージ送信はセンサノードの動作の中でも最も電力を消費するアクションの一つということに起因する．セグメント 1 つに対して ADV，REQ という 2 種類のコントロールメッセージが付加するため，最低でも $2m$ 回コントロールメッセージを送信する必要がある．つまりセグメント分割数を増やすほどコントロールメッセージの送信回数が増える．また再送やコントロールメッセージの欠落などを考慮すると更に多くの送信が必要となる．

(c) パケット管理領域量とは 1 セグメントを構成するパケットの欠落を監視するベクトルの大きさである．パケットが到着するとベクトル中の対応ビットを立て，全ビットが揃うまで再送要求を出す．必要なベクトルの大きさは 1 セグメント分の大きさであるので，全データが p パケット分の大きさだとすると p/m だけ必要となる．つまりセグメント分割数が大きいほど，1 セグメントの大きさは小さくなるのでベクトルの大きさも小さくなる．

(d) ノード 1 台の配送時間とは，1 台のノードが全てのセグメントを配送するのに必要な時間である．セグメント分割数を増やすほど，セグメントを送信する間隔の回数が増えるため，ノード 1 台が全てのセグメントを配送するのにより時間が必要となる．

3.5 ローカルパイプライニング

前章で述べたようにセグメント分割数は配送性能に大きな影響を与える．しかし既存のソフトウェア配送方式は，ネットワーク全体に 1 つのセグメント分割数を割り当てている．この仕様は各ノードにメッセージ送信回数など均等な配送負担を課し，ネットワーク全体の配送性能を最大化することができるが，性能の低いノードや配置環境の悪いノードに対しては大きな負担がかかるという課題がある．本稿では，このセグメント分割数を複数のセンサノードで構成されるグループに割り当てることで，より細かな配送性能の設定を目指す．

3.6 グループ境界での配送

ローカルパイプライニングでは複数のグループにセグメント分割数を割り当てるため，図 3 のような配送をそのまま適用したのでは問題が発生する．特に異なるセグメント分割数を持つグループの境界において配送がうまくいかなくなる．

第一に 1 セグメントを配送するために必要なデータが不足するという問題である．例えば 6 分割と 3 分割のセグメント分割数が割り当てられた 2 つのグループ A,B において，180 パケット分のデータを配送する場合を想定する．この場合，グループ B においてセグメントを送信するには 60 パケット分のデータが必要となるが，グループ A から送られてくるセグメントは 30 パケットしかない．これでは送信するためのデータが不足し，グループ B において新しいセグメント分割数で配送を行うことができない．

第二にメッセージ衝突の増加である．これはセグメント分割数が異なることによってセグメントの配送間隔が変わり，グループの境界において他グループのセグメント配送と自グループのセグメント配送が衝突するという問題である．

そこで本方式では図 5 のような配送アルゴリズムを採用する．このアルゴリズムはセグメントを受信したときの動作を規定している．まずローカルパイプライニングではパケットに自分が属しているグループに関する情報を含めており，その情報を参照し，配送方法を決定している．セグメントのデータを受信したとき，そこに含まれているグループが自分の属しているグループと同じならば，グループ内での配送を続行するために ADV メッセージを送信する状態へ移行する．しかし自分のグループとは異なるグループが受信メッセージに含まれていた場合は，セグメントを全て受信し終えるまで，他グループの配送を妨害しないように受信のみの待機状態へ移行する．もし全てのセグメントを受信し終えたら，新しいセグメント分割数を再設定し，自グループの配送を開始する．この配送アルゴリズムを採用すると，配送過程は図 6 のようになる．グループの境界にあるノード (境界

```
[ When a segment is received. ]

IF ownGroup == rcvMsg.group
        /* send segment as the same number of segment */
        GOTO Advertise phase
ELSE
        IF downloadComplete == TRUE
                /* send segment as the new number of segment */
                setOwnSegmentSize()
                GOTO Advertise phase
        ELSE
                /* wait until all segment data is complete */
                GOTO Receive phase
        ENDIF
ENDIF
```

図 5 配送アルゴリズム

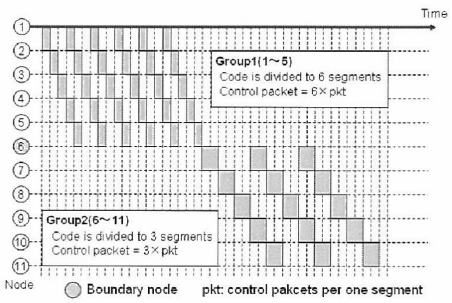

図 6 ローカルパイプライニングの配送

ノード) は，全てのセグメントを受信するまで待機し，全て受信したら自分のセグメント分割数で配送を開始する．また必要送信回数もグループごとにセグメント分割数を割り当てているので，グループごとに異なり，消費電力を局所的に減らすことができると考えられる．図 6 においては，1 セグメントを送るのに必要なメッセージ数を *pkt* とすると，グループ 1 で 6*pkt* 回，グループ 2 で 3*pkt* 回となり，グループ 2 だけ送信回数が減り，消費電力を減らすことが可能である．

3.7 グループの設定

セグメント分割数を割り当てるグループは複数のセンサノードで構成される．現在，その設定はユーザによって手動で設定することにしている．本稿ではノードを一列に配置した直列配置と格子状に配置した格子状配置という 2 種類のネットワークを想定し，それぞれに対してグループの設定方法を検討する．まず直列配置におけるグループ設定は，グループサイズ (ネットワーク内のグループ数) に依存する．本稿ではグループ数 4，3，2 の場合を想定し，それぞれにセグメント分割数を割り当てるようにしている．4 グループ以上の設定も可能だが，グループサイズが小さいとセグメントのオーバラップ回数が減り，パイプライニングが働かないため，本稿では 4 グループまでに限定している．続いて格子状配置において，データの伝播の仕方を考慮して，図 7 のような 4 種類のグループ設定を想定した．

(a) 平行タイプ 横の伝播速度が中央部分よりも速くデータが拡散した場合を想定している．グループ形状は三角形や台形などのシンプルな図形で，グループ 1 と 4 の大きさを小さく設定してある．

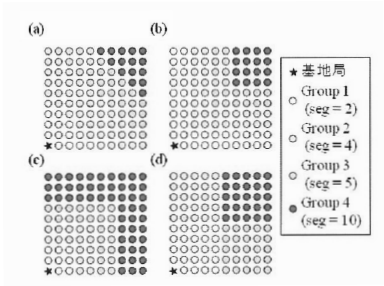

図 7 グループの設定

(b) 直角タイプ 中央よりも横の伝播速度が極端に速い場合を想定している．(a) や (d) と比較して複雑な形状をしており，グループサイズは基地局から離れるほど大きくなるようにしてある．

(c) 逆直角タイプ 中央の伝播速度が横の伝播速度よりも速い場合を想定している．(b) とは逆の形であり，グループサイズは基地局から離れるほど小さくなる．

(d) 十字タイプ 中央と横の伝播速度が等しい場合の拡散を想定している．他のグループ設定とは異なり，全てのグループが 2 つのグループに隣接している．更にグループの境界ノードと含まれるノード数は全て等しく，グループ形状も同じ形をしている．

4. シミュレーション評価

4.1 シミュレーション環境

本章ではシミュレーションによるローカルパイプライニングの性能評価を行う．シミュレータは多くのソフトウェア配布方式の評価に使われる TinyOS ネットワークシミュレータ TOSSIM[5] を利用する．ネットワークは 1×100 の直列配置と 10×10 の格子状配置の 2 種類を想定し，どちらも 1 ホップの通信半径を持つセンサノード 100 台で構成される．パケットロスは隠れ端末問題など MAC レベルで発生するもののみを想定している．評価する項目は消費電力を左右するメッセージ送信回数であり，配送するデータサイズが同じであることから，特にコントロールメッセージについて評価する．また全てのセグメントを受信するまでに要した時間 (配送時間) を各ノードについて評価した．

4.2 直列配置

直列配置のシミュレーションではネットワークを均等なネットワークサイズの複数グループに分割し，セグメント分割数をそれぞれに割り当てた．4 グループの場合は 1 グループの大きさは 25 台であり，基地局に近い側から，2，4，5，10 というセグメント分割数を割り当て，3 グループの場合は 1 グループは 33 台の大きさであり，2，4，10 というセグメント分割数に，2 グループの場合は 1 グループは 50 台の大きさで，4，10 というセグメン

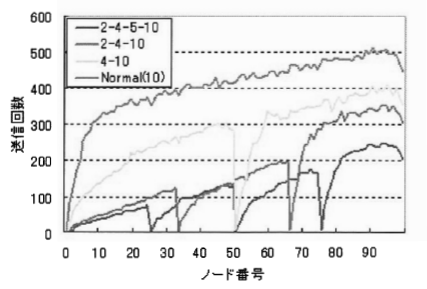

図 8 直列配置におけるコントロールメッセージ数

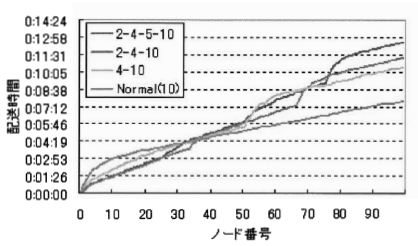

図 9 直列配置における配送時間

ト分割数を割り当てた．

このような環境下において，各ノードについて全てのセグメントを受信するまでに，送信したコントロールメッセージの送信回数を測定した結果が図 8 である．このシミュレーション結果から，まずわかることはネットワークの局所でコントロールメッセージ送信回数を増やしたり，減らしたりすることが可能となったということである．特に 4 グループの場合を見ると 2 分割にしたグループ 1 と 10 分割にしたグループ 4 とでは，明らかな差を見ることができる．これは第 3.4 章で述べたように，1 つのセグメントに最低でも 2 つのコントロールメッセージが付加しており，セグメント分割数が増えるほど，分割数に比例してコントロールメッセージが増えるためだと考えられる．またグループの大きさという観点から，このシミュレーション結果を見るとグループ数が多いほどメッセージ送信回数が少なくなることがわかる．4 グループの場合，1 グループの大きさは 25 台となり，3 グループの場合は 33 台となる．この 2 つのデータを同じ 10 という分割数を割り当てた最後のグループに注目して比較してみると，3 グループの場合の送信回数が最大で約 350 であるのに対して，4 グループの場合の送信回数は最大で約 250 というようにグループの大きさが小さいほどコントロールメッセージ送信回数を少なくできることがわかった．これは最初のセグメントを受信してから最後のセグメントを受信するまでの間に何回も ADV メッセージを送信することが原因であると考えられる．この最初のセグメントから最後のセグメントを受信するまでの時間は，送信開始地点から離れるほど長くなり，その分 ADV メッセージを送信する期間が長くなるため，グループの大きさが大きいほど送信回数が増えるのである．

次に各ノードが全てのセグメントを受信した時間 (配送時間) を測定した結果が図 9 である．このグラフを見ると各グループの境界で遅延が発生していることがわかる．この遅延は割り当てるセグメント分割数に依存していると考えられる．例えばセグメント分割数が 4 の場合と 10 の場合では，曲線の立ち上がり部分において大きな差がある．これはグループ境界では全セグメントを取得するまで待機するという配送アルゴリズムを採用しているこ

とが原因と考えられる．

しかしネットワークサイズが最も大きいグループなしで，分割数を 10 に設定した通常の配送 (Normal(10)) のグラフを見ると，立ち上がり以降での配送時間の増加率が，2 分割や 4 分割にした場合の増加率と比較すると少ないことがわかり，このことからネットワークサイズが大きいほど増加率は少なくなると予想される．

4.3 格子状配置

格子状配置のシミュレーションでは第 3.7 章で述べた平行，直角，逆直角，十字タイプという 4 種類のグループ設定を行い，それぞれに異なるセグメント分割数を割り当て，メッセージ送信回数と配送時間を調査した．グループはデータの伝播を考慮し，図 7 にあるような形を想定し，基地局に近い側からグループ 1，2，3，4 とし，それぞれに 2，4，5，10 というセグメント分割数を割り当てている．図 10 は全てのセグメントを受信するまでに必要なコントロールメッセージの送信回数を示しており，図 11 は各ノードについて全セグメントを取得した時間 (配送時間) をあらわしている．以下では想定した 4 種類のグループ設定に対して，共通した結果とグループ設定ごとの特徴的な結果について述べる．

全てのグループ設定に共通してわかることは，セグメント分割数を多く割り当てた場所ほどメッセージ送信回数が増加するということである．これは直列配置の場合の結果と同様に，1 つのセグメントにいくつかのコントロールメッセージが付加するため，分割数を増やすほどセグメントが増加し，それに比例してコントロールメッセージ送信回数が増加するためであると考えられる．またグループの境界ではメッセージ送信回数が極端に減少するという現象を観測することができる．これは境界ノードにおいて全てのセグメントを受信するまで待機し，セグメントの送信を開始しないという配送アルゴリズムを採用しているためである．境界ノード以外のノードは 1 つのセグメントを受信すると，すぐに近隣ノードへの配送を開始するために ADV メッセージを送信し始める．しかし境界ノードでは全セグメントを取得するまで ADV メッセージを送信し始めることはないため，このような現象が観測されると考えられる．次に配送時間に関しては，直列配置の場合と共通した傾向が見られる．セグメ

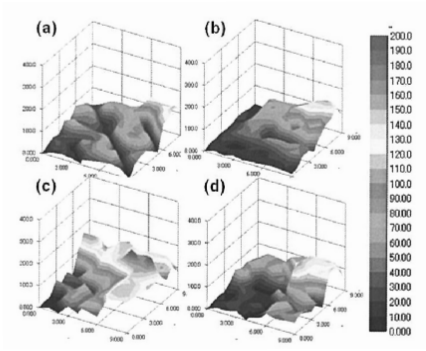

図 10 各グループ設定における送信回数

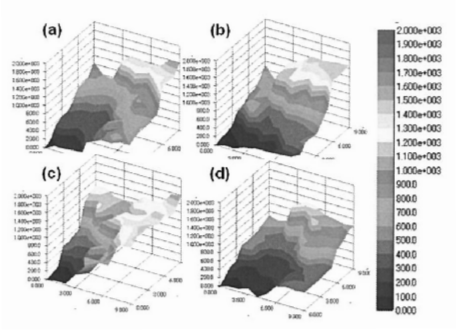

図 11 各グループ設定における配送時間

ント分割数を多く割り当てたグループの境界では，曲面の立ち上がり部分が大きく，遅延が発生していることがわかる．これはセグメント分割数を増やすほど，セグメントを送信する間隔が増え，その分 1 台のノードが全セグメントを取得する時間が増えるためである．

次にグループ設定ごとの特徴的な結果について述べる．まずメッセージ送信回数についてだが，境界ノードの数とセグメント分割数について注目する．(a) 平行タイプでは斜めにグループを分割しており，グループ 1 とグループ 4 の境界ノードの数は他のものと比較して小さい．境界ノードが少ないということは，セグメントを送信開始するときに他の境界ノードによって通信が妨害されず，再送によるメッセージ送信回数が少なくなると考えられる．(c) 逆直角タイプは最も境界ノード数が多く，その分だけメッセージ衝突が多くなり，図 10 のように送信回数が最も多くなっている．それに対して (c) 直角タイプのグループ 2 では同じように境界ノード数が多いが，セグメント分割数を少なく割り当ててあるので，送信回数は多くはない．(d) 十字タイプでは境界ノードの数は全てのグループで等しく，メッセージ送信回数の影響はセグメント分割数に限定される．

次に配送時間に関しては，シンプルな形状をしている平行タイプと十字タイプが少なく，複雑な形をしている直角と逆直角タイプのもののほうが時間がかかっている．またグループ 1 の大きさが大きいほど配送時間が短いこともわかる．これは配送の初期のほうが，データを送信するノードが少なく衝突が少ないためであると考えられる．

4.4 セグメント分割数の割り当てに関する考察

シミュレーション結果よりセグメント分割数の割り当て方法に関して，いくつかのことが判明した．まずセグメント分割数を多く設定すると，メッセージ送信回数が増加し，グループ境界で遅延が発生するが，その後の配送時間の増加率は少なくなる．グループの大きさを比較的大きなものにすれば，遅延以上に増加率が少ないことによる配送時間の短縮ができると予想される．またネットワークの大きさが大きいとメッセージ送信回数が多くなるため，送信回数を抑えたい場合は，できるだけグループの大きさを小さくしたほうがよい．現在，本方式はグループ設定を手動で行っているが，これらの特性を考慮して，グループを設定すれば，ユーザの希望通りの配送が可能でないかと考えられる．

5. おわりに

本稿では多様なセンサノード環境が混在するセンサネットワークを想定し，パイプライニングという並列配送技術を拡張したローカルパイプライニングを提案した．本方式はグループごとにグループの環境に応じてセグメント分割数を割り当てるものである．シミュレーションによる性能評価を行った結果，グループごとにメッセージの送信回数や配送時間を制御することが可能となった．また結果からセグメント分割数やグループ設定の割り当てに関する特性を見い出すことができた．しかし本方式ではグループの境界で遅延が発生するという課題があるため，配送アルゴリズムの改良により対応が必要である．また現在はグループ設定を手動でやっているが，自動的にグループを構築するということを考えたい．

参考文献

1) Qiang Wang, Yaoyao Zhu, Liang Cheng: Reprogramming wireless sensor networks: challenges and approaches. IEEE Network 20(3): 48-55 (2006)
2) Hui, J.W., Culler, D.: The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In: Proc. ACM SenSys 2004, pp. 81-94. ACM, New York (2004)
3) Kulkarni, S.S., Wang, L.: MNP: Multihop Network Reprogramming Service for Sensor Networks. In: Proc. IEEE ICDCS, pp. 7-16 (2005)
4) V.Naik et al.,"Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Wireless Embedded Devices," 26th IEEE Real-Time Sys. Symp.,Dec.(2005)
5) TinyOS: http://www.tinyos.net/