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

2007－DBS－141 (24)
2007－GN－ 62 (24)
2007／1／26

WSNにおける無線通信を利用したソフトウェア更新効率化の検討

宮丸 卓也 † 峰野 博史 ‡
寺島 美昭* 徳永 雄一* 水野 忠則 ‡
† 静岡大学大学院情報学研究科 ‡ 静岡大学情報学部
*三菱電機株式会社

概要 ワイヤレスセンサネットワーク (WSN) は次世代のインフラとして注目されている．WSN は無数のノードで構成され，各ノードでは制御用ソフトウェアが動作し，システム変更時にソフトウェア更新が必要である．WSN 環境では，一度に無数のノードを高速に更新することが望まれるため，無線通信を使った更新手法が利用される．この更新手法では更新の高速化のため，ソフトウェアをセグメントに分割し並列配送するパイプライン方式が使われる．この方式ではセグメント分割数を増やして高速化するが，コントロールメッセージ増加によって電力消費が増える課題がある．本稿ではセグメント分割の扱いを変更することで，各ノードの状況に応じて課題に対処する柔軟なパイプライン方式を検討する．

A Study of Efficient Software Update for WSN with using Wireless Communication

Takuya Miyamaru † Hiroshi Mineno ‡
Yoshiaki Terashima* Yuichi Tokunaga* Tadanori Mizuno ‡
†Graduate School of Informatics, Shizuoka University
‡Faculty of Informatics, Shizuoka University
*Mitsubishi Electric Corporation

Abstract Wireless Sensor Network is focused as Next Generation Infrastructure. Wireless Sensor Network is composed of innumerable sensor nodes, and the software for the control process is running in each sensor nodes. If we have a change like the extension for new functions and bug fix release, it is necessary to update new software. In WSN environment, it is hoped to update an innumerable node at high speed at a time. Therefore, Over the air Update is used. In this method, we use pipelining that divide software into segments for the speed-up of the update. But there is a power consumption increasing problem because of a control message increase though the segment number of partitions is expanded for speed-up. In this report, we suggest a flexible pipelining that deals with this problem according to the situation of each node.

1 はじめに

MEMS 技術の発展，ZigBee といった低消費電力無線技術の出現によってワイヤレスセンサネットワーク (WSN) の実現が現実味を帯びてきている．WSN は通信機能を付加したセンサノードを協調動作させ，ありとあらゆる場所の環境情報を収集する．WSN は 1 つのセンサノードから集めた情報ではなく，複数のセンサノードからの情報を総合的に扱うことで，今まで観測することができなかった現象や事実を発見する可能性を秘めており，学問の分野から，セキュリティ，農業などの事業，ユビキタスコンピューティングを実現する次世代のインフラストラクチャーとして様々な分野で注目されている [1].

WSN は無数のセンサノードで構成されている．センサノードは温度センサ，加速度センサなどのセンサだけではなく，収集したデータを処理するためのプロセッサやメモリ，近隣ノードや基地局にデータを送るための通信機能を搭載している．例えば Crossbow 社の Mica シリーズ [2] が挙げられ，7MHz の 8 ビットマイクロコントローラ，128kB の

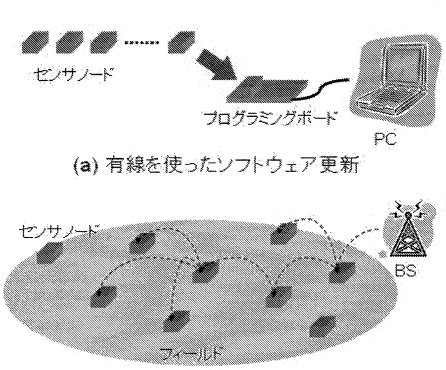

図 1: ソフトウェア更新の種類

プログラム用メモリを搭載している．このようなセンサノードでは，収集したデータの処理や各種機能（LED の点滅やメッセージ送信）を制御する専用のソフトウェアが動作している．このようなソフトウェアの更新は，WSN を利用したシステム管理者の重要な仕事の 1 つである．

現在，無線通信を利用してソフトウェア更新を簡単に行う Over the Air Programming(OTAP) という更新手法 [5][6] が研究されている．これらの方法ではソフトウェアをセグメントと呼ばれる単位に分割し，複数のセグメントを並列に配送することで高速化を実現するパイプライン方式が研究されている．パイプライン方式はセグメントの分割数を増やすほど高速化ができる一方で，セグメント 1 つに付随するコントロールメッセージによって，ネットワーク全体のコントロールオーバヘッドが増大してしまうという課題 (課題 1) を抱えている．またセグメント分割数を全体として固定しているため，バッテリ残量や通信状況など個々のセンサノードの状況がソフトウェアの配送に反映されないという課題 (課題 2) もある．そこで本稿では課題 2 に対して，各ノードでセグメント分割数を可変にする動的セグメント分割方式を提案し，課題 1 に対して，セグメントを一括要求することでコントロールメッセージを削減するセグメントバースト方式を提案する．更に現在のパイプライン方式を高速化するパケットレベルパイプライン方式について検討する．

以下，第 2 章では WSN におけるソフトウェア更新の特徴と既存研究の課題について述べ，第 3 章では課題を改善する方式である動的セグメント分割方式，セグメントバースト方式，パケットレベルパイプライン方式について検討する．第 4 章では本稿の内容をまとめ，今後の検討を行う．

2 WSN のソフトウェア更新

2.1 ソフトウェア更新

本稿におけるソフトウェアとはセンサノード上で動作する制御ソフトウェアのことを指す．このソフトウェアは収集したデータの処理やセンサノードに搭載された LED や通信機能を制御するために使われる．システムの拡張や機能変更によって，このソフトウェアに変更が生じた場合，ソフトウェア更新を行う必要がある．更新の方法は，図 1 の (a)(b) のように 2 種類の方法がある．(a) は古典的な手法であり，センサノードを PC に接続された専用プログラミングボードに接続し，一台ずつ有線を通じてソフトウェアを書き込むものである．(b) は無線通信を使い，マルチホップでソフトウェアを書き込む方式である．WSN 環境が無数のノードで構成されていることから，一度に大量のノードを更新することが望まれている．したがって WSN におけるソフトウェア更新は多くの場合，(b) の方式が採用されている．(b) の方式は一度に大量の更新が可能であるが，次の設計課題を考慮して更新方式の設計を行う必要がある．

2.2 冗長なメッセージ送信の抑制

1 つ目の設計課題は冗長なメッセージ送信の抑制である．冗長なメッセージ送信による弊害は，メッセージ衝突による信頼性の低下と電力消費の増加である．メッセージ衝突は複数のノードが大量のメッセージを送信することによって発生しやすくなる．したがって冗長なメッセージは出来る限り削減しなければならない．次に電力消費の増加である．図 1(b) の方法は，プログラミングボードから更新用の電力を取得できないため，センサノードのバッテリの電力を利用する．バッテリの電力は WSN のライフタイムを決定するため，出来る限り無駄な使用は避けなければならない．またメッセージ送信はセンサノードの動作の中で最も電力を消費するものの一つである．冗長なメッセージ送信は電力を無駄に消費する．電力消費の観点からも冗長なメッセージ送信を抑制する必要がある．

冗長なメッセージ送信を抑制する方法として SPIN[4] がある．SPIN ではハンドシェイクによって必要なデータのやり取りだけを行うことで，冗長なメッセージ送信を抑制する．ハンドシェイクでは最初に，データを持っているノードが ADV メッセージを送信する．ADV メッセージには自身がどのようなデータを持っているかを記述したメタデータが含まれる．ADV メッセージを受信したノードはメタデータを参照し，データの更新が必要かを判断する．データの更新が必要ならば REQ メッセー

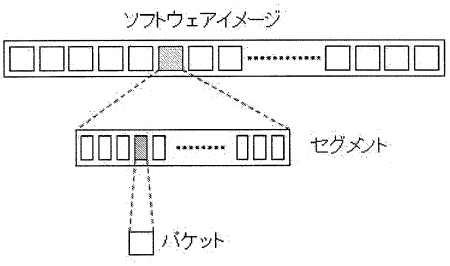

図 2: セグメント分割

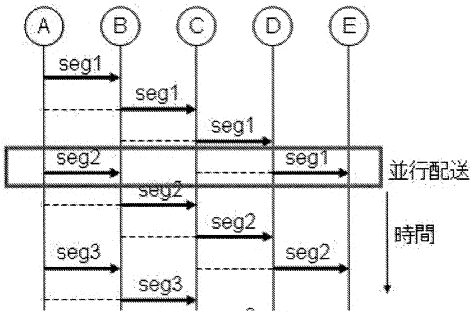

図 3: パイプライン化

ジを送り返す．REQ メッセージを受信したノードは，実際のデータを含めた DATA メッセージの送信を開始する．以上のような冗長なメッセージ送信の抑制機構を取り入れたソフトウェア更新方式として，Deluge[5]，MNP[6] がある．このように冗長なメッセージ送信の抑制機構はソフトウェア更新手法を設計する場合，取り入れてゆく必要がある．

2.3 更新完了時間の短縮

2 つ目の設計課題は更新完了時間の短縮である．WSN の本来の仕事は，ソフトウェア更新ではなく，周囲の情報を収集することである．ゆえに全ノードのソフトウェア更新を出来る限り早く終了させ，WSN が提供するサービスを再開させる必要がある．以上のように更新完了時間の短縮は，更新手法を評価する上でも重要な設計課題である．

更新完了時間を短縮させる方式としてパイプライン方式があり，Deluge[5] と MNP[6] で採用されている．パイプライン方式の特徴は，ソフトウェアイメージ全体をそのまま配送するのではなく，セグメントと呼ばれる単位に分割して配送することである．セグメントは図 2 に示すように，複数のパケットと呼ばれる送信単位の集まりである．前章で述べた ADV-REQ メッセージのやり取りはセグメント単位で行われ，DATA メッセージがパケットに相当する．図 3 にパイプライン方式の概要を示す．図 3 は 5 台のノードを直列に配置してパイプライン方式を適用した状況を示している．ノード D がセグメント 1 を送る間に，ノード A がセグメント 2 を配送するというように，セグメントを並行配送しているため，更新完了時間を短縮できる．

セグメント 1 つを隣接ノードに配送する時間を T_{seg}，セグメント分割数を m とし，n ホップの直列に配置されたネットワークにソフトウェアを配送する場合，パイプライン方式を適用すると更新完了時間 $T_{pipeline}$ は次の式で表すことができる [5]．

$$T_{pipeline} = (n + 3(m-1)) \cdot T_{seg} \qquad (1)$$

図 3 のように，メッセージ衝突を避けるため 1 つのセグメントを送信し終えてから，一定間隔を空けて次のセグメントを送る必要がある．この間隔は最低 3 ホップ分のノードが通信していない状況になるまでの時間であり，(1) 式の 3 という数字に反映されている．一方でパイプラインなしの場合の更新完了時間 T_{normal} は次のように表現される．

$$T_{normal} = n \cdot m \cdot T_{seg} \qquad (2)$$

ただしパイプラインなしのときは $m = 1$ であり，T_{seg} はソフトウェアイメージ全体を近隣ノードに配送する時間になる．

また図 4 のグラフは，(1) 式のセグメント分割数を変化させた場合の更新完了時間をあらわしている．前提となるパラメータはホップ数 n を 100 とし，ソフトウェアイメージ全体を近隣ノードに配送する時間を 1024 秒とする．パイプラインを利用しなかった場合の更新完了時間は，(2) 式より $100 \times 1 \times 1024 = 102400$ 秒かかる．一方でセグメントを 8 分割にしてパイプライン方式を利用した場合の更新完了時間は，(1) 式より $(100 + 3(8-1)) \times 1024/8 = 15488$ である．パイプライン方式を適用すると理論上，約 15 % (15488/102400) の更新完了時間に短縮できることになる．またグラフからは，パイプライン方式による高速化の効果はセグメント分割数を増やすほど現れることが読み取れる．以上のようにパイプライン方式による高速化は，更新完了時間の短縮につながるためソフトウェア更新手法に取り入れる必要がある．

2.4 既存方式の課題

前章までに WSN におけるソフトウェア更新の 2 つの設計課題について取り上げ，その対処方法につ

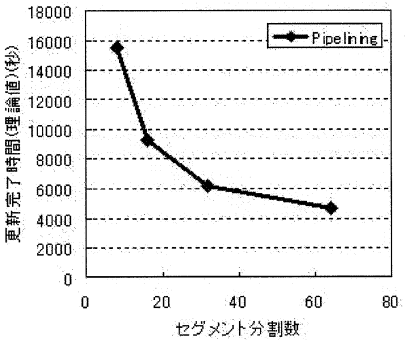

図 4: パイプラインの効果

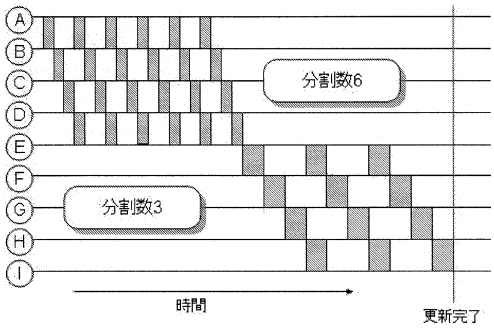

図 5: セグメント分割数を変更した場合

いて述べた．しかしパイプライン方式に関連して，2 つの課題が残されている．1 つ目はコントロールオーバヘッドの増加 [7] であり，2 つ目は各ノードの状況を考慮しない配送が行われていることである．

前章ではセグメント分割数を増やすほど更新完了時間を短縮できることがわかった．しかしセグメント分割数を増やすことで問題が発生する．セグメント一つのやり取りに ADV メッセージ，REQ メッセージというコントロールメッセージのやり取りが付随する．単純に高速化を求めてセグメント分割数を増やせば，セグメントの数が増えコントロールメッセージが急増する．コントロールメッセージの送信回数が急増することで，電力を大きく消費することにつながる．これがコントロールオーバヘッドの増加という課題である．

2 つ目の課題は，各ノードの状況を考慮しない配送である．各ノードの状況は，それぞれ異なる．バッテリー残量が多いノードがあれば，少ないノードもある．通信状況の違いもあればハードウェアの性能も異なる．しかし現在のパイプライン方式は，セグメント分割数をネットワーク全体で固定している．そのため各ノードの状況がパイプライン方式を利用した配送に反映されない問題が発生する．例えばバッテリー残量が少ないノードが存在したとしても，セグメント分割数の多い状態でソフトウェアを配送し，電力を消費するコントロールメッセージ送信を多く行わなければならない．

3 提案方式

3.1 動的セグメント分割

従来のパイプライン方式はネットワーク全体でセグメント分割数を決めて配送を行っていた．この方法はネットワーク全体で効率のよい配送を行うには都合がよいが，個々のノードの状況について考慮していない．例えばバッテリ残量の少ないノードは，メッセージ送信を頻繁に行うことを控えたい．しかしネットワーク全体では高速な配送を行うために，セグメント分割数を増やそうとする．セグメント分割数の増加は，コントロールメッセージの増加を招く．その結果，バッテリ残量の少ないノードでも多くのコントロールメッセージを送信しなければならない．このようにネットワーク全体でセグメント分割数を決めることで，個々のノードの状況 (バッテリ残量，通信状況，ハードウェア性能など) が全く無視されてしまう．

本章では，これらの課題に対処するため動的セグメント分割という方式を提案する．動的セグメント分割方式は，セグメント分割数を個々のノードで決定する方式である．もし，あるノードのバッテリー残量が少なかったり，通信環境の悪化，ハードウェア性能の問題などによってメッセージ送信回数を少なくしたい場合，それまで受信していたセグメント分割数を変更して，自身が定義した新しいセグメント分割数でセグメントを送信する．その結果，送信メッセージ回数を各ノードの状況に応じて調節することが可能となる．

図 5 はセグメント分割数を変更した場合の配送の様子をあらわしている．最初は各ノードの状況が比較的良好であるため，ソフトウェア更新時間を短縮するため高速な配送を目標とする．そして高速な配送を行うためにセグメント分割数を増やす．しかしノード E はセグメントを受信した段階で，バッテリー残量が少ないことや通信環境の悪化などによって，従来のセグメント分割数では不満な状況に陥っている．したがってノード E は，高速な配送よりも不満を解消するために，ADV-REQ などのコントロールメッセージ送信を抑制することを目標とする．そこでメッセージ送信を抑えるためにセグメン

ト分割数を少なくする．セグメント1つにはコントロールメッセージが付随するため，セグメントの数が少なければコントロールメッセージの送信回数を減らすことができる．

第2.3章でネットワーク全体で必要とする更新完了時間について解説したが，動的セグメント分割を適用した場合の変化について考える．動的セグメント分割を適用した場合は，ネットワークを複数のエリアに分割し，それぞれのエリアにおいて異なるセグメント分割数を使ってパイプライン方式を適用した場合と同等になる．図5では，ノードAからEまでのエリアとノードEからIまでのエリアそれぞれにパイプライン方式を適用したことになる．したがって，全体の更新完了時間は，それぞれのエリアの更新完了時間の和であらわされることがわかる．

$$T_{dseg} = \sum_{i=1}^{p} (n_i + 3(m_i - 1)) \cdot T_i \qquad (3)$$

この式では，n_i が各エリアのホップ数，m_i が各エリアのセグメント分割数，T_i が1セグメントを配送する時間である．pはエリアの数である．ただしパイプライン方式による恩恵を受けるためには，各エリアのホップ数は3以上でなければならない(3はパイプライン方式なしの場合とパイプライン方式ありの場合で更新完了時間が等しくなるときのホップ数)．各エリアのホップ数が3以上ならば，各エリアはパイプライン方式なしの場合と比較して小さい更新完了時間ですむ．したがってネットワーク全体から見ても，パイプライン方式なしの場合よりも更新完了時間を短くしつつ，各ノードの状況をパイプラインに反映させることが可能である．

3.2 セグメントバースト方式

2つ目の方式は，コントロールオーバヘッド自体を削減することを試みる．本章で提案するセグメントバースト方式は，連続する複数のセグメントを一括して要求し，配送する方式である．従来のパイプライン方式は，最初にコントロールメッセージを使い，1つのセグメントを要求する．1つのセグメントを受信し終えたら，再びコントロールメッセージを使って次のセグメントを要求する．この方法は，パイプライン方式の配送リズムを崩さず信頼性の高い配送を可能にしている一方で，コントロールメッセージによるオーバヘッドが付随する．そこでセグメントバースト方式では，1回のコントロールメッセージのやり取りでセグメントを1つ要求するのではなく，連続する複数のセグメントを一括で要求するようにする．こうすることでコントロールオーバヘッドを減らすことが可能である．

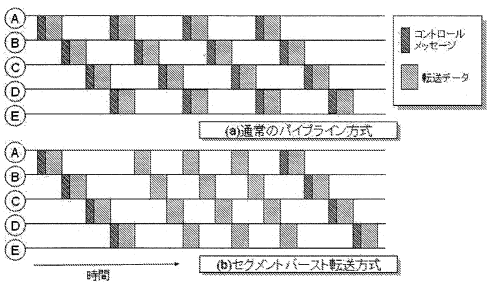

図6: セグメントバースト方式

図6は従来の方式とセグメントバースト方式の違いを示している．この図を見るとわかるように，従来の方式がコントロールメッセージを毎回つけて配送しているのに対して，セグメントバースト方式は毎回ではなく，定期的にコントロールメッセージのやり取りを行っている．このような連続配送を実現するために，コントロールメッセージの中身を少し変更する必要がある．従来の方式におけるコントロールメッセージは，2.3章で述べたADVメッセージとREQメッセージである．ADVメッセージには現在何番のセグメントが含まれているかということを示すメタデータが含まれている．対してREQメッセージには，そのセグメントのどのパケットが必要かということが記されている．

今回検討する拡張は，ADVメッセージに連続して送信可能なセグメント数の情報を付加する．具体的には現在利用可能なセグメント番号から連続して転送可能なセグメント番号までの有効範囲を指定する．例えば現在B番のセグメントが利用可能であり，$B+\alpha$ 番まで連続転送したいとすると，ADVメッセージには基準となるセグメント番号B番と，連続転送範囲 α という値を通知する．一方でREQメッセージにも同様の機構を組み込む．ADVメッセージを受信したノードは，相手側のノードがどれだけ連続転送が可能か(α の値)ということを知り，それを参照にして自身がどれだけのセグメントを連続して受信したいかということを検討する．検討した結果得られた連続転送範囲(β)をREQメッセージに付加してADVメッセージの送信元へ返信する．REQを受信したノードは，REQメッセージ中に含まれる連続転送範囲 β を取り出し，基準となるセグメント番号Bから初めて，$B+\beta$ までのセグメントを連続して配送する．こうすることでセグメント1つをやり取りするのに，毎回コントロールメッセージを付加する必要がないため，コントロールオーバーヘッドを軽減することが可能である．

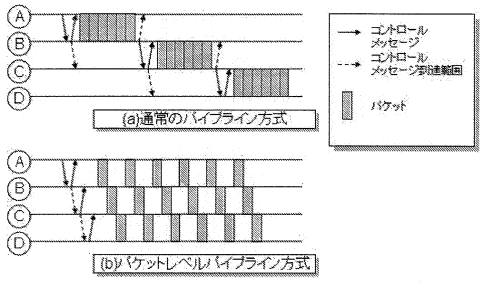

図 7: パケットレベルパイプライン

3.3 パケットレベルパイプライン

3つ目の方式は，より高速な配送の実現を目指す方式である．従来の方式は，セグメント単位でパイプライン方式を適用していた．セグメントは図2のように複数のパケットで構成されており，それらのパケットを受信し，1つ分のセグメントが完成してから，近隣ノードへセグメントを配送するということを行っていた．パケットレベルパイプライン方式では，1つのセグメントが完成する前に受信したパケットを近隣ノードに転送する．セグメントが完成する前にパケットを広く拡散させることができるため，セグメントレベルでパイプライン化を行うよりも高速な配送が可能である．

従来のパイプライン方式と，パケットレベルパイプライン方式の違いはコントロールメッセージ有無である．図7は従来のパイプライン方式とパケットレベルパイプラインのメッセージのやり取りの違いを示している．従来の方式でセグメント分割数を最大にした場合，セグメント1つ1つ(この場合パケット1つ1つ)にコントロールメッセージが付加される．一方でパケットレベルパイプライン方式では，パケット1つ1つにコントロールメッセージを付加しない．コントロールメッセージのやり取りを行うのはセグメント単位であり，パケットレベルパイプライン方式では，データを配送する前にコントロールメッセージのやり取りを行う．このようにパケットレベルパイプライン方式のほうが，コントロールメッセージ送信回数が少ないため，従来のパイプライン方式でセグメント分割数を最大にするよりも消費電力を抑えつつ，高速な配送を行うことが可能だと考えられる．

4 まとめ

本稿では無線通信を利用したソフトウェア更新手法を扱い，既存の高速化手法であるパイプライン方式の課題を整理した．1つ目の課題は各ノードの状況が全体の配送に反映されないというものであり，2つ目の課題はコントロールメッセージによるオーバヘッドである．1つ目の課題に対しては状況に応じてセグメント分割数を変化させる動的セグメント分割方式を提案し，2つ目の課題に対しては一括してセグメントを要求することで，コントロールメッセージ数を削減するセグメントバースト方式を提案した．また現在のパイプライン方式を更に高速化するパケットレベルパイプライン方式を提案した．今後検討すべきことは，方式の効果を確認するための評価である．そのために本方式を Mica2[2] 上で実装することを目指す．評価の方法は，数台の Mica2 による実環境評価と，TOSSIM[8] によるノード台数が多い場合のシミュレーション評価を検討する．

参考文献

[1] Deborah Estin 他 ” Environmental Cyberinfrastructure Needs For Distributed Sensor Networks ” A Report from a National Science Foundation Sponsored Workshop, 12-14 August, 2003,

[2] Crossbow:http://www.xbow.com/

[3] Pedro Jose Marron 他”TinyCubus: A Flexible and Adaptive Framework for Sensor Networks”, Proceedings of the 2nd European Workshop on Wireless Sensor Networks, pp. 278-289, 2005.

[4] Joanna Kulik 他 ” Adaptive Protocols for Information Dissemination in Wireless Sensor Network ” ,Mobicom1999

[5] Jonathan W. Hui, David Culler ” The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale ” ,SenSys ’04

[6] Sandeep S. Kulkarni, Limin Wang ”Multihop Network Reprogramming Service for Sensor Networks ” ,SenSys ’04

[7] Qiang Wang 他 ”Reprogramming Wireless Sensor Networks: Challenges and Approaches” IEEE Network May/June 2006 p48-55

[8] TOSSIM: http://www.cs.berkeley.edu/ pal/research /tossim.html