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

2008－ARC－176 (12)
2008－EMB－ 7 (12)
2008／1／16

センサノード向けOSにおける消費電力の低減のための協調型タスクスケジューリング

松尾 英治† 鈴来 和久† 横田 裕介†† 大久保 英嗣††
†立命館大学大学院理工学研究科 ††立命館大学情報理工学部

実世界の状態を取得するための手段としてセンサネットワークが注目されている．センサネットワークは，センサノードと呼ばれる無線端末からなるネットワークであり，各センサノードが取得した環境情報を外部に提供することが主な機能となっている．本研究では，センサノード向けOSにおけるタスクスケジューリングとして，他のノードと協調してスケジュールを決める協調型タスクスケジューリングを実現する．既存のスケジューリングでは，データの送受信処理で消費される電力が多いという問題がある．特に，通信が完了するまでの待ち時間に応じて消費電力が増大する．本論文の提案手法では，予めノード間で送受信のタイミングを同期させることにより，通信における各ノードの待ち時間の低減を実現する．

A Co-operative Task Scheduling Scheme for Power-saving in Sensor Node Operating System

HIDEHARU MATSUO† KAZUHISA SUZUKI† YUSUKE YOKOTA†† EIJI OKUBO††
†Graduate School of Science and Engineering, Ritsumeikan University
††College of Information Science and Engineering, Ritsumeikan University

In order to gather environmental information in the real world, the sensor network technology has been paid attention. Sensor network consists of plural sensor nodes that have a certain wireless communication device. Moreover, main functionality of sensor networks is providing environmental informaion that is gathered by sensor nodes to upstream. In this research, we have developed a co-operative task scheduling scheme for the sensor node operating system. The idea of the proposed scheme is that each sensor nodes decide their task scheduling cooperatively. However, one of the problems of the existing scheduling schemes is that power consumption is high during communication process. Especially, sensor nodes waste additional power due to the increase of waiting time for completing the communication process. The proposed approach described in this paper is that a sender node and a receiver node synchronize the timing of the commnication cooperatively and beforehand in order to reduce the waiting time in the communication process.

1 はじめに

近年，無線デバイスの普及や低コスト化により，ユビキタスコンピューティングなど，無線ネットワークを用いた技術が多く研究されている．このようなネットワークにおいて，実世界の状態を取得するための手段として，センサノードと呼ばれる無線小型端末を複数用いた，無線センサネットワークが注目されている．

現在，センサノードの研究においては，長期間の動作を実現する資源管理手法や，耐障害性の向上についての研究が多く行われている．本研究は，センサノード向けOSにおけるタスクスケジューリングとして，他のノードと協調してスケジュールを決める協調型タスクスケジューリングについて手法の提案，および実装を行う．本スケジューリングでは，従来のセンサノード向けOSにおけるタスクスケジューリングの問題点である，無線デバイスの待機時間や通信とノード間の同期の待ち時間を改善する．また，各ノードの協調動作を可能とすることで，ネットワーク全体における応答性を極力損なうことなく，ネットワーク全体における消費電力の削減を実現する．これにより，無線センサネットワークの高寿命化などを図る．

本論文では，センサノード向けOSにおける消費電力の低減のための協調型タスクスケジューリングについて述べる．まず，2章で，研究背景である無線センサネットワークの概要と既存のスケジューリングにおける問題点を述べる．次に，3章で，提案手法の前提となるネットワークのモデルについて述べ，4章で本論文で提案するスケジューリング方式について述べる．また，5章で，評価と考察について述べ，最後に，第6章にてまとめを述べる．

2 研究背景

本章では，研究背景として，無線センサネットワークと

その関連技術について，OS のタスクスケジューリングを中心に述べる．また，既存技術の問題点として，2 つの代表的なスケジューリングにおける問題点について述べる．

2.1 センサノードと無線センサネットワーク

無線センサネットワークは，無線デバイスを搭載した小型端末である複数のセンサノードから構成される．センサノードの主な機能は，周囲の状況を観測し，無線によって情報を送信することである．センサノードは，電源がバッテリによって供給されることや，低消費電力化やコスト削減のために計算機資源が制限されているなどの特徴を持つ．無線センサネットワークは，センサノードの取得した情報を集約する際に，センサノード同士で伝播させる点が特徴的である．また，回収やメンテナンスが困難な場所においても環境情報の取得を可能とするという特徴を持つ．

しかし，無線センサネットワークは，メンテナンスコストの面から修理などに伴う回収サイクルの長さが要求される．このため，制限の厳しい計算機資源から構成されていたとしても，ネットワークを維持するために少なくとも数年稼動しなければならないといった課題がある．

2.2 通信における電力消費とスケジューリング

無線センサネットワークは，多くのノードが互いに通信することでネットワークを構成している．この通信において，受信ノードは，受信が行えるよう常に無線デバイスを動作させておく必要がある．無線デバイスにより電力が消費されるため，受信の待機時間の増加が電力消費量の増加の要因となる．また，通常は，ノードは複数のノードから同時に受信できないため，通信が衝突する．これによって，通信量が増加し，無線デバイスの電力消費量が増加する．

このように，無線デバイスの受信待機の時間を削減すること，および通信の衝突を回避することが通信時の電力消費量の低減を実現するために重要になる．特に，無線デバイスの動作するタイミングを管理するものとして，タスクや通信のスケジューリングが重要になる．

2.3 既存スケジューリングにおける問題点

既存のセンサノード向け OS では，イベント駆動型スケジューリングや WakeUp 型スケジューリングなどが利用されている．しかし，これらのスケジューリングは，無線センサネットワークにおけるスケジューリングとして，いくつかの問題点を抱えている．本節では，ネットワーク全体の電力消費量を削減する観点から，これらの既存のスケジューリング方式の問題点について述べる．

2.3.1 イベント駆動型スケジューリング

TinyOS [1] などに代表される既存のセンサノード向け OS の多くは，イベント駆動型のスケジューリングを利用している．イベント駆動型スケジューリングは，ハードウェアなどの割込みをイベントとしてとらえ，それを起点として連鎖的に処理を進める．イベントによって呼び出されたタスクは，それまで処理されていたタスクを阻害しないために，迅速に終了する必要があり，複雑な処理を別途

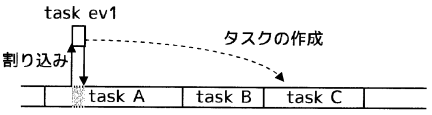

図 1　イベント駆動型スケジューリングの例

作成したタスクに委ねることになる．図 1 では，task A，B，C がユーザタスク，task ev1 がイベントによって呼び出されたタスクに相当する．task ev1 は，task A を阻害しないために，イベントに対する処理を行うタスクとして task C を作成し即時に終了する．通常のユーザタスクは，タスクが数珠繋ぎに終了するモデル (Run to Completion) に基づいているため，割込み以外の処理からは阻害されずに自身の処理を完了できる．イベント駆動型スケジューリングは，I/O 処理などにおいてハードウェアに対する割込み待ちが生じ難く，ソフトウェアの待ち時間である CPU のアイドリング期間を軽減できるといった利点がある．

しかし，イベント駆動型は，センサノード向け OS におけるスケジューリングとしていくつかの問題点を抱える．イベント駆動型スケジューリングでは，受信イベントの発生元である無線デバイスが常に動作していなければならない．この通電期間においては，受信処理だけではなく，受信した信号が自身へのものなのかなど，信号の正当性の判断が常にデバイスレベルで行われる．そのため，無線センサネットワークを長期間運用する場合，この通電による消費電力が問題となる．また，イベント駆動型スケジューリングでは，他のノードの動作状況を踏まえたスケジューリングが行えない．他のノードと同期した処理を行うには，ユーザタスク側でその同期時刻を定めるなど，同期処理やその保証そのものをユーザタスク側において行う必要があるが，イベント駆動型スケジューリング自体がユーザタスクの処理開始，完了時間を保証していない．このため，ユーザタスク側による同期処理も困難となっている．すなわち，イベント駆動型における同期処理は，通信処理の遅延や待ち時間を生じさせる原因となり，無線デバイスなどの消費電力量を増加させる．

このように，イベント駆動型スケジューリングは，おのおののセンサーノードで処理が完結する場合は効率のよい方式であると言える．しかし，無線による他のノードとの通信やネットワーク全体を考慮したスケジューリングとしては，センサノードの消費電力量に問題がある．

2.3.2 WakeUp 型スケジューリング

イベント駆動型スケジューリングの問題点である通信処理の待ち時間を解決する手法として，WakeUp 型スケジューリングが提案されている．WakeUp 型スケジューリングには，Flexible Power Scheduling [2]，Wakeup Scheduling in Wireless Sensor Networks [3] などがある．

WakeUp 型スケジューリングでは，事前に定めたタイミングでのみ無線デバイスを受信待機させることにより，総電力消費量の低減を図る．送信ノードは，受信ノード

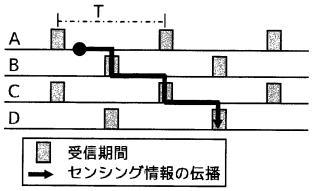

図 2　WakeUp 型スケジューリングにおける送信タイミングの例

の受信タスクが起動するタイミングを見計らい送信する．WakeUp 型スケジューリングにおける受信タイミングと情報の伝播タイミングの関係を図 2 に示す．A，B，C，D は，各ノードの受信タスクの動作を時間軸上に表している．また，T は受信タスクの周期を，矢印はセンシング情報が取得されてから目的のノードに伝播するまでを表している．

しかし，WakeUp 型スケジューリングでは，通信のタイミングを動的に設定する手法がモデル化されていない．例えば，頻繁に送受信を行う見込みのあるノード間を特殊化して扱う際に，ネットワーク全体のタイミングを再構成する必要がある．このように，WakeUp 型スケジューリングは，送受信のタイミングが変化するような動的なネットワークには適用が困難である．さらに，衝突に対する見積り，および衝突に伴う再送処理が他の処理に与える影響の見積りも困難である．このため，衝突しないようにタスクの設計においてユーザが考慮する必要があり，ユーザが行う作業量を増加させる要因となる．

3　ネットワークモデル

本章では，本論文で提案するスケジューリングの前提となる，ネットワークモデルとその他の条件について述べる．本スケジューリングは，ノード内だけではなく，ネットワーク上の他のノードの状態を踏まえてスケジューリングを行う．そのため，スケジューリングにおいては，ネットワークの構成や通信に対して事前に考慮する必要がある．

3.1　前提条件

本スケジューリングは，ネットワークに属する他のノードの状態も踏まえて動的に各ノードの処理のタイミングを定める．これにより，ノード間の無駄な待ち時間の削減を実現する．また，動的な受信タイミングの変化に対応することで，遅延の生じにくいスケジューリングを実現する．なお，ノード数増加に伴う管理コストの増加を防ぐために，クラスタ化されたネットワークにおいて，クラスタ単位で各ノードの情報を共有し，そのクラスタ単位でスケジューリングを行うものとする．

本研究における前提条件として，クラスタ化技術と時刻同期技術が既にあるものとする．また，クラスタ内の各ノードにおいて時刻同期済みであり，センサノードの時刻は，無線センサネットワークの基準時刻との誤差が 10 ミリ秒から 100 ミリ秒程度の精度であると仮定する．

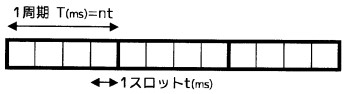

図 3　スロットの例 (n=4)

3.2　クラスタ化ネットワークにおける制限

本スケジューリングでは，クラスタ間において同期を行わないため，クラスタ間の通信がスケジューリングを妨げる恐れがある．そのため，クラスタ間の通信に制限を設け，非同期なクラスタ間の通信に対処する．

本スケジューリングが想定するクラスタ化されたネットワークでは，クラスタ間の通信は，クラスタヘッドのみが通信を行い，クラスタ間をまたいだ一般のノードの通信は，クラスタヘッドを仲介して行うものとする．通信相手の識別には，クラスタヘッドを対象とする場合はクラスタ個別の ID を，ノードを対象とする場合はノード個別の ID を用いる．また，クラスタ化されたネットワークにおいては，クラスタヘッドをスケジューラが任意に変更することが可能であるものとする．

ノード間では，スケジューリングの制御のための通信が適時行われる．このため，ユーザによる通信と制御通信が衝突しないよう，スケジューラによって通信処理が管理されている必要がある．したがって，クラスタ化における処理やノード間の通信処理に対して，スケジューラが介入できるネットワークプロトコルが必要となる．

4　提案手法

本章では，本スケジューリングの設計について述べる．本スケジューリングは，前章で述べたネットワークモデルを用いて，スロットと呼ぶ時間枠によりスケジューリングを決定する．スケジューリングにおいては，送信や受信など，どのような処理を行うのかといったタスクに付加された情報に基づき，送信待ちなどの優先すべきタスクから優先的に処理をする．また，その優先度に基づいて動的なタスクの割り込みを行うことにより，同期処理を実現する．

4.1　スロットとタスク

本スケジューリングは，スケジューリングの指標としてスロットを用いることが特徴である．1 スロットの時間幅を $t_{(ms)}$ とするとき，通信処理やイベント処理などのタスクは，この $t_{(ms)}$ の中で実行される．さらに，3.1 節で述べた時刻同期の仮定から，スロットの開始時刻が，すべてのセンサノードにおいて 10 ミリ秒から 100 ミリ秒の精度で同期していることが保証される．本スケジューリングでは，スロットを n 個まとめたものを 1 セットとして扱い，これを 1 周期 ($T_{(ms)} = nt$) とする (図 3 参照)．

スロットは，タスク配置の指標として，4.3 節で述べるいずれかの状態を 1 つ持つ．状態は，そのスロットにおいてどのようなタスクが優先的に処理されるかを示す．スケジューラは，時間の経過に伴う次のスロットへの遷移を契機として，そのスレットの状態に則した優先すべきタスク

を割り込ませる．

本スケジューリングでは，スロットに設定された状態に基づき，スケジューラ側で受信のタイミングを決定する．これにより，受信期間が明示されるため，無線デバイスへの電力制御をスケジューラで行うことが可能となる．また，ユーザプログラムの開発者は，無線デバイスへの電力制御を意識せずに済む．

4.2 クラスタヘッドとスロット

本スケジューリングでは，クラスタ化されたネットワークにおいて，クラスタヘッドを特殊なノードとして扱う．クラスタヘッドは，1 セットのスロットの周期を m 回 (m は整数) 繰り返した後に同じクラスタ内の他のノードに引き継がれる．また，クラスタヘッドの引継ぎと共にクラスタの管理情報なども引き継ぐ．この引継ぎは，クラスタ内に存在するノードに与えられたノード ID に基づき，昇順で行われる．クラスタ内の全ノードで ID 情報を共有し既知であるものとする．クラスタ内のノード数が増減した場合は，クラスタ内の全ノードへ増減したことが通達された後に引継ぎの順番を変更する．

4.3 スロットの状態

スロットは，優先するタスクの指標として，次の 6 つの状態の内，いずれか 1 つの状態が設定される．

クラスタ遷移状態 CC 前クラスタヘッドとの通信など，クラスタヘッドへの遷移処理が優先される状態．

クラスタ譲渡状態 CS クラスタヘッドとの通信など，クラスタヘッドとの譲渡処理が優先される状態．

システム優先状態 SY その他システムの処理が優先される状態．

送信予約状態 SR 特定相手への送信が優先される状態．

受信予約状態 RR 特定相手からの受信が優先される状態．

受信待機状態 RW 緊急的な通信への予備として，不特定相手からの受信が優先される状態．

通常状態 FR 優先度を考慮しない状態．

なお，CC，CS，SY 状態は，システムから設定される状態であり，変更ができない．

4.4 スロットに対する状態設定

各スロットの状態は，直前の周期の状態を基に周期の開始時に再配置される．この配置は，一般ノード，クラスタヘッドでおのおの異なった配置となる．

4.4.1 クラスタヘッドにおけるスロットの初期状態

クラスタヘッドにおけるスロットの初期状態は，他のクラスタやクラスタ内のノードとの通信処理を優先するために，通信を行うスロットが多くなるように設定される．クラスタヘッドにおけるスロットの初期状態は，CC，SR，RW の 3 つの状態が必ず配置される．CC 状態に設定されたスロットは，クラスタヘッドの遷移に伴う情報の引継ぎ，ならびに，スロット状態の再設置や周期開始における初期化においてスケジューラが利用する．SR 状態に設定されたスロットは，クラスタ内の全ノードと 1 周期内に必ず通信が行えることを保証するために配置する．RW 状態に設定されたスロットは，他のクラスタヘッドから 1 周期内に通信を受け取れることを保証するために配置する．

図 4 スロットへの状態配置の例

CC，SR，RW の各状態のスロットがどのスロットに配置されるかは，次のように決定される．まず，CC 状態は，初期化のために必ず第 1 スロットに配置される．スロット 1 つあたりの時間 t が遷移への処理に要する時間より短い場合は，配置されるスロット数を第 2，第 3 と増やす必要がある．ここで，CC 状態のスロット数を n_{CC} とする．この値は固定であり，スケジューリング中に変更されないものとする．この，CC 状態に関して，第 2 周目からは CC 状態の変わりに SY 状態が設定される．

次に，SR 状態は，$n_{CC}+1$ 番目から $3+j$ スロット毎に nr だけ配置される．j は，配置される SR 状態のスロットの間に FR 状態のスロットをいくつ設けるかを決定する固定された値である．r は，1 周期の内に 1 つの一般ノードに対して何回通信するかを決定する固定された値である．n は，クラスタ内の一般ノードの数を示し，新規ノードの追加や離脱と共に変動する．j を増やせば電力効率の向上と応答性の低下に，r を増やせば電力効率の低下と応答性の向上に繋がる．図 4 は，$n_{CC}=1$，$j=1$，$r=2$，$n=3$ の場合におけるクラスタヘッドのスロットの初期状態の例である．SR 状態において対応する一般ノードは，クラスタヘッドの遷移順を利用し，クラスタ内における識別番号の小さい順になる．

RW 状態は，$n_{CC}+1$ 番目から $2+j+p$ スロット毎に nr だけ配置される．p は，クラスタヘッドの識別番号が偶数なら 1，奇数なら 2 となる．これにより，クラスタヘッド間における RW 状態の重なりを減少させる．

CC，SR，RW の各スロットの配置が決定した後，直前の周期におけるスロットの状態を参考にして SR や RR などの予約済みスロットの配置を決定する．最後に，状態が設定されていないスロットを FR 状態に設定する．

4.4.2 一般ノードにおけるスロットの初期状態

一般ノードにおけるスロットの初期状態は，他のノードとの同期処理を優先するように設置される．一般ノードにおけるスロットの初期状態は，SY，RR，RW の 3 つのスロットが必ず配置される．SY 状態に設定されたスロットは，スロット状態の再設置や周期開始における初期化に

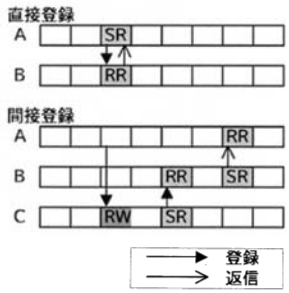

図 5　直接登録と間接登録の例

おいてスケジューラが利用する．RR 状態に設定されたスロットは，クラスタヘッドと 1 周期内に必ず通信が行えることを保証するために配置する．RW 状態に設定されたスロットは，他のノードから 1 周期内に通信を受け取れることを保証するために配置する．この RR 状態と RW 状態を設けることにより，スロット制御における同期通信が 1 周期に 1 回は少なくとも行えることを保証する．

SY，RR，RW の各状態のスロットがどのスロットに配置されるかは，次のように決定される．まず，SY 状態は，初期化のために必ず第 1 スロットに配置される．また，SY 状態のスロットは，前述のクラスタヘッドにおける CC 状態のスロット数 (n_{CC}) と同数でなければならない．

次に，クラスタヘッドに対する RR 状態は，前述の SY 状態のスロットの最後から順に配置される．この配置は，クラスタ内におけるノード ID の小さい順に依存する．自ノードの識別番号が i 番目の場合は，$n_{CC}+1$ 番目から $(3+j)\cdot i$ スロット毎に r だけ配置される．

RW 状態は，通信期間のばらつきを減らすために，配置された RR 状態の中間に位置するスロットに配置される．

SY，RR，RW の各スロットの配置が決定した後，直前の周期におけるスロットの状態を参考にして SR や RR などの予約済みスロットの配置を決定する．

以降，前周期を参考にその他 SR 状態や RR 状態などの予約済み状態が配置され，最後に，状態が設定されていないスロットを FR 状態に設定する．

なお，全周期においてクラスタヘッドであった一般ノードは，最初に配置される状態が SY 状態ではなく CS 状態となる．この CS 状態は，クラスタヘッドへの譲渡処理のために配置される．

4.5　状態の継続と登録

RR，SR，RW 状態のスロットは，次の周期においてもその状態が保持される．ただし，ノードがクラスタヘッドに遷移する場合は，クラスタヘッドの初期状態が優先される．この際，同期相手は，相手がクラスタヘッドに遷移することをその周期中の SY 状態の間に算出し，対応する状態を一時的に抹消する．なお，クラスタヘッドの任期終了を契機として，一時的に抹消した状態を復元する．

新規に RR，SR，RW 状態を登録するには，送信相手への通知が必要となる．登録には，送信元ノードのスロットの状態，予約したい状態の種類，予約したいスロットの位置を送信相手のノードに通知する．また，この登録には，間接登録と直接登録の 2 種類の登録法がある．直接登録は即座に結果が返され，間接登録は非同期に結果が返される．直接登録と間接登録の処理の流れを図 5 に示す．5 の場合，ノード A が登録元，ノード B が登録先となる．直接登録は，登録処理が開始される際に相手ノードが RW 状態か，自身の SR 状態とペアになる RR 状態の場合のみ行われる．間接登録は，登録相手へもっとも早く到着する経路を用いて登録情報を通知する．登録結果の通知も登録処理と同様に，もっとも早く帰着する経路を用いて通知される．図 5 の下段の場合，ノード A から直接登録可能なスロットがないため，ノード C を介してノード B に対して間接登録を実行している．登録が不可能な場合は，互いに空いている FR 状態のスロットを探して配置する．それでも登録が不可能な場合は，再登録のための処理をユーザプログラムに実行させる．

登録した状態は，その利用の終了と共に取り消す必要がある．基本的に，登録の取消しはユーザからの明示的な指示によって行われる．なお，取消しとは，FR 状態への変更を意味するものとする．RR，SR 状態の取消しは，SR 側からの取消しによって行われ，多くはその取り消したいスロットの中で呼び出される．RW 状態の取消しは，受信側送信側共に取消しが可能である．RW 状態の取り消し時は，クラスタヘッドへ取り消す旨を伝えるために，クラスタヘッドへの通達が完了するまで遅延される．

4.6　通信における遅延

本スケジューリングにおいては，スロットの状態に基づいて優先度を決めて同期を実現しているため，状態の登録が行われていない場合は通信に遅延が生じる．

そこで，遅延について考察する．現在のスロットが同期済み状態にない場合，どのタイミングで，どの経路によって通信処理を行うかを次に示す．なお，クラスタヘッド，もしくはその他ノードを中継する場合は，送信元ノードと中継ノードが直接通信を行い，その後に中継ノードから送信先ノードまで直接通信を繰り返すことを意味する．

1. 後続のスロットに通信相手と同期済みのスロット (RR，CC) が存在する場合，そのスロットを利用して通信相手と直接通信する．
2. 後続のスロットに通信相手が RW 状態になるスロットが存在する場合，そのスロットを利用して通信相手と直接通信する．
3. クラスタヘッドを利用する場合，通信相手に到達するまで 2 ホップかかる間接通信となる．
4. その他の一般ノードを中継する場合，中継ノードの個数を n とするとき，通信相手に到達するまで $n+1$ ホップの間接通信となる．

以上より，1 や 2 の場合は直接通信となるため，通信の遅延は通信可能となるスロットまでの待機時間となる．すなわち，1 の場合は，通信相手と同期状態になるまでの時間となる．2 の場合は，通信相手のスロットが RW 状態になるまでの時間となる．また，3 や 4 の場合は間接通信となるため，通信の遅延は直接通信の場合と比較して増大する．特に，中継ノードが通信処理を完了させるまでの待機時間が遅延を増大させる要因となる．

したがって，通信処理に即時性が求められているか否かによって，1 から 4 のいずれかを選択すべきか異なると考えられる．即時性が求められていない場合，電力消費量を削減するために直接通信を優先して選択することが考えられる．これは，1 回の通信処理で済むため，電力消費を抑制できるからである．即時性が求められている場合，スロットの状態に適応して 1 から 4 の中でもっとも遅延が少ない通信方法が選択されると考えられる．

このように，本手法では同期を前提とするため，現在のスロットが同期にある状態でない場合に生じる．そのため，遅延を減らすには，ノード間において同期の状態をユーザが多く予約しておく必要がある．

4.7 タスクの配置

各タスクは，タスクが保有している情報に基づいてスロットの優先度が決定される．この情報は，タスクの実行タイミングに対するユーザからの指示を表すものであり，ユーザがタスクの生成と共に指定しなければならない．なお，この情報は，配置を希望するスロットの状態，処理を期待する時間 (開始時刻，最大遅延時刻)，処理の基本優先度からなる．ユーザは，データ処理，送信，受信などを複数のタスクに分割し，それらタスクの実行結果をまとめ利用する．本論文では，タスクの粒度としてスレッドを想定している．これは，分割されたタスクにおいてコンテキストを共有することを示し，データの引渡し処理やコンテキストスイッチを最小限に抑えるためである．

タスクの優先順位の指標は，次のスロットへの遷移を契機として遷移先の状態に基づいて変更される．本スケジューリングでは，この優先順位の変更に伴い，その状態において優先すべきタスクを割り込ませる．この割込みは，優先すべきタスクがなくなるまで連続して行われる．なお，配置を希望するスロットの状態が同じであるタスク同士は，到着順に実行される．

5 評価と考察

本スケジューリングは，無線デバイスの待機時間を削減し，ネットワーク全体の高寿命化を図る．そのため，本スケジューリングによる効果の評価として，電力消費量の比較を第一の項目として挙げる．次に，同期処理が円滑に行えたか否かの評価として，タスクの待ち時間，ならびに，他ノードがデータを受け取るまでの待ち時間について，その平均時間を比較の項目に挙げる．また，同期処理の通信量への影響を計るために，送受信におけるデータの増加量を項目に挙げる．これら比較評価は，センサノード数やクラスタ構成などのネットワーク構成，通信頻度，スロットのパラメータを変化させ，実装に伴い行っていく予定である．比較対象として，イベント駆動型スケジューリングである TinyOS，Mantis [4] の各スケジューリング．WakeUp 型スケジューリングである Flexible Power Scheduling，Wakeup Scheduling in Wireless Sensor Networks を想定している．

本スケジューリングを実装するにあたり考察すべき事項として，時刻同期の精度とスロット長を挙げる．本論文における提案手法では，スロットの分割に時刻同期による時刻を用いている．この時刻には前提として 10 ミリ秒から 100 ミリ秒の誤差が含まれており，スロットが正確に一致する保証はない．そのため，誤差がスロット長に比べ大きい場合は，同期自体ができず，スケジューリングの破綻や再送処理などの増加の要因となる．したがって，誤差が大きい場合は，スロット長を長くする必要がある．しかし，スロット長を長くした場合，受信時のデバイスの待ち時間が増え，電力削減の効果を低下させる要因となる．スロット長を決めるにあたっては，利用する時刻同期技術の精度から，スケジューリングの精度と電力削減量のどちらを優先するのかを決めなければならない．

6 おわりに

本論文では，センサノード向け OS における消費電力の低減のための協調型タスクスケジューリングについて述べた．本スケジューリングは，クラスタ内において通信などの同期的な処理をノード同士で協調させることにより，無線デバイスなどのハードウェアの稼働時間の低減を図る．これにより，ネットワーク全体における電力消費量を削減し，ネットワーク全体の高寿命化を図る．本スケジューリングを用いることで，WakeUp スケジューリングと同程度の電力効率を保ちつつ，ユーザへの同期処理の容易な実装環境の提供が可能になる．

参考文献

[1] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E. et al.: TinyOS: An Operating System for Sensor Networks, *Ambient Intelligence* (2005).

[2] Hohlt, B., Doherty, L. and Brewer, E.: Flexible power scheduling for sensor networks, *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, New York, NY, USA, ACM, pp. 205–214 (2004).

[3] Keshavarzian, A., Lee, H. and Venkatraman, L.: Wakeup scheduling in wireless sensor networks, *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, New York, NY, USA, ACM, pp. 322–333 (2006).

[4] Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., Torgerson, A. and Han, R.: MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms, *Mob. Netw. Appl.*, Vol. 10, No. 4, pp. 563–579 (2005).