1X-8

TinyOS へのマルチタスクスケジューラの実装

山口 翔一† 宮崎 敏明†
会津大学コンピュータ理工学部†

1 はじめに

メモリおよび計算資源が限られたセンサノードでは高機能なオペレーティングシステム(OS)を搭載することは難しい．そのため，センサノード専用の OS が開発され，使用されている．TinyOS[1]はその一つである．TinyOS は少ない資源で動作するが，タスクスケジューリングに First-In-First-Served(FIFS)方式のノンプリエンプティブなシングルタスクスケジューラを用いているため，複雑な処理を実行すると，パケット受信処理など緊急性を要するタスクの実行が困難となる場合がある．本稿では，TinyOS へプリエンプティブなマルチタスクスケジューラを実装し，上記問題の解決を図ったので報告する．

2 TinyOS

TinyOS は，研究開発用途として最も普及しているセンサノードの一つである Crossbow 社 MOTE IRIS[2]の OS として良く知られている．MOTE IRIS は，8bit CPU (Atmel ATmega1281)を使用しており，uCLinux[3]や ITRON 仕様準拠の OS[4]など，一般に広く知られた組み込みシステム用 OS が必要とする計算能力を持たない．そのため，簡素な機能で構成される TinyOS が開発された．TinyOS のタスクスケジューラも例外ではなく，単純な構成となっている．OS において，タスクスケジューリングは重要な機能であり，これまで多くのアルゴリズムが研究されてきた．たとえば，Priority Scheduling(PS)，Shortest-Job-Next(SJN)，Round-Robin(RR)は，良く知られたタスクスケジューリング手法である．しかし，TinyOS では優先度を決定する機構や処理時間を解析する機構を持たないため PS や SJN を実装することは難しい．一方，RR 方式はそのような機構を必要としない．RR 方式でも，一般にタスク間での資源競合問題を解決する必要があるが，各タスクが完全に独立している FIFS では，大きな問題とならない．

3 マルチタスクスケジューラの構成

3.1 全体構成

2 章の考察より，本稿では TinyOS に RR 方式のマルチタスクスケジューラを実装する．図 1 に TinyOS のタスクスケジューラと提案タスクスケジューラの全体構成を示す．提案タスクスケジューラは，処理のスレッド化と，リスト構造と割り込みによるタスク切り替え機構を用いて実装した．実行中のタスクをリストで連結し，定期的に発生するタイマ割り込みによって実行中のタスクリストを回転する．

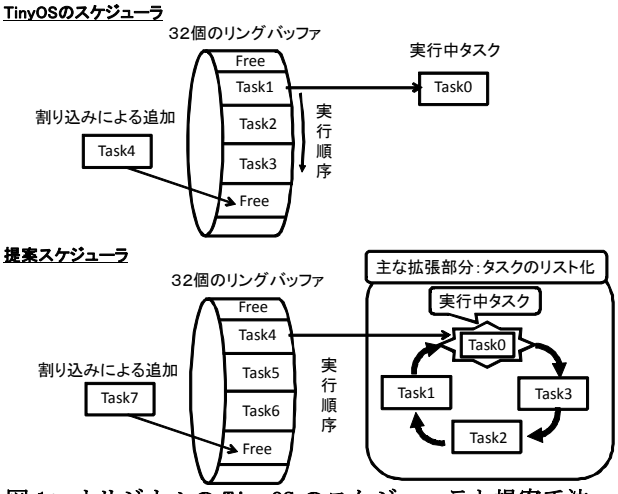

図 1: オリジナルの TinyOS のスケジューラと提案手法のスケジューラの動作

3.2 スレッドの実装

TinyOS では，二つの処理を繰り返すことでタスクスケジューリングを行なっている．すなわち，(1)FIFS 方式による次のタスクの選択と(2)その実行，を交互に繰り返す．スレッドを実装する際もスレッド内部で同じ手順を行うようにした．これにより，変更箇所を少なく出来るだけでなくスレッドの生成と終了処理を，タスクの種類に依存せずスレッド間で共通化できた．

3.3 Round-Robin 方式

RR 方式は，実行中のタスクと実行待ちのタスクをある一定時間間隔毎に切り替えるタスクスケジューリング手法である．その時間間隔をタイムスロットと呼ぶ．また，切り替えに必要な時間はオーバーヘッドと呼び，実際のタスクスケジューリングでは，それも考慮する必要がある[5]．今回提案したスケジューラの実装では，オーバーヘッドは 195 クロックであった．そこで，タイムスロットは，オーバーヘッドが全体の 0.1%程度になるように，200000 クロックに設定した．

An implementation of a multi-task scheduler for TinyOS
†Shoichi Yamaguchi, †Toshiaki Miyazaki.
†School of Computer Science and Engineering, The University of Aizu

4 メモリ使用量

TinyOS のタスクスケジューラを変更する前後の OS 全体が使用するメモリ量の変化を表１に示す．表１より提案スケジューラ導入よるメモリ増加量は，プログラムメモリにおいて 1 KB であり，データ RAM において 0.03 KB であった．また，データ ROM では拡張による変化はなかった．これは，センサノードに搭載された各メモリの総容量の 0.8%と 0.4%の増加に相当し，本スケジューラ導入によるメモリ資源への影響は，充分に小さいことが分かる．

表１： メモリ使用量

項目	プログラムメモリ	データ ROM	データ RAM
総容量	128 KB	4 KB	8 KB
TinyOS スケジューラ	8.7 KB (6.8%)	0.014 KB (0.35%)	0.36 KB (4.5%)
提案スケジューラ	9.7 KB (7.6%)	0.014 KB (0.35%)	0.39 KB (4.9%)

（ ）内は，メモリ使用率.

5 検証

オリジナルの TinyOS と本提案スケジューラを実装した TinyOS（提案スケジューラ）の動作検証を，２つのタスク TaskA および TaskB を用いて行った．TaskA は，ある計算処理と一つの LED の点滅動作を実行する．TaskB は，別の LED の点滅動作を実行する．図 2 に，各スケジューラの動作検証シーケンスを示す．センサノード起動後，タイマ割り込み毎に TaskB を起動する．その後，TaskA を起動する． TinyOS のスケジューラでは，TaskA が実行中は，タイマ割り込みが発生しても TaskB の処理が実行されるとはなかった．一方，提案スケジューラでは，TaskA が実行中であっても，TaskB の処理を行うことができた．また，本提案スケジューラを導入した TinyOS は，既存アプリケーションに何ら手を加えることなく動作させることができることも確認した．

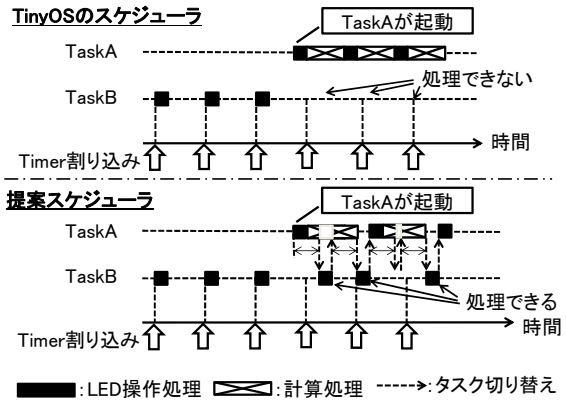

図 2： 動作検証シーケンス

6 性能評価

提案手法の性能を評価するために，一つのセンサノードと受信基地局を用意し，センサノードで計算処理による負荷をかけながら，１秒間に１回パケットを送信し，その時の転送にかかる遅延と受信基地局でのパケットの取得率を計測した．ここで用いた計算処理とは，for(i=0;i<0xffff;i++);という単純ループを実行するものである．図３に実験結果を示す．図３の棒グラフは転送遅延を示し，折れ線グラフはパケット取得率を示す．横軸は計算処理の実行回数を表す．棒グラフから分かるように，TinyOS では計算処理の実行回数が増えと転送遅延が著しく増加するが，提案スケジューラでは低遅延のままである．また，折れ線グラフが示すように，基地局でのパケット取得率は，TinyOS では，実行回数が７回以上となると急激に低下しているが，提案スケジューラではそのような低下は見られない．これらの結果より，提案スケジューラを導入することにより，高負荷状態でも，パケット送信処理を遅滞なくて一定間隔で実行できることが分かった．

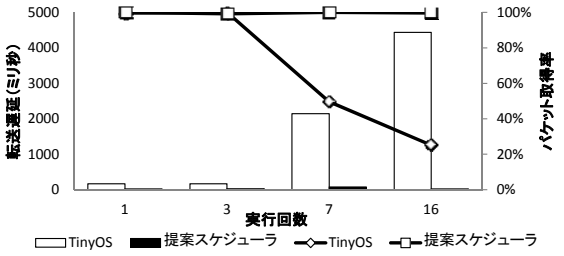

図 3: 送信遅延とパケット取得率

7 おわりに

TinyOS へプリエンプティブなマルチタスクスケジューラを実装し，実行時間の長いタスクを実行中でも，割り込み処理など，他のタスクを遅滞なく処理できるようにした．本スケジューラ導入によるメモリ増加量は微小であり，既存アプリケーションもそのまま動作する．今後は，各種アプリケーションを用いて，タスク実行のタイムスロット間隔の最適化を図る．

文献

[1]J Hill, R Szewczyk, A Woo, S Hollar, D Culler, and K Pister, “System Architecture Directions for Networked Sensors,” GPLAN Notices, Vol.35, pp. 93-104, Nov. 2000
[2]“IRIS MOTE Datasheet,” http://www.memsic.com
[3] “uCLinux,” The Linux/Microcontroller Project. http://uclinux.org
[4]“uITRON4.0 仕様,” http://www.ertl.jp/ITRON/SPEC/mitron4-j.html
[5]アンドリュー・S・タネンバウム(水野忠則，太田剛．最所圭三，福田晃，吉澤康文訳), “モダンオペレーティングシステム,” 原著第２版，ピアソンエデュケーション，pp. 141-142 (2004)