Vol.2011-SLDM-149 No.42
Vol.2011-EMB-20 No.42
2011/3/19

リアルタイムアプリケーション向け タスク処理定義可能なスケジューリングシミュレータ

佐野泰正[†1] 松原豊[†1]
本田晋也[†1] 高田広章[†1]

様々なタスクスケジューリングアルゴリズムを用いて，既存のリアルタイムアプリーションをスケジュールしたときの振る舞いを，容易に確認することを目的に，スケジューリングに特化したシミュレータが開発されている．既存のシミュレータでは，複雑なアプリケーションの振る舞いを正確にモデル化することが困難であるため，シミュレーション結果が，実機での動作と大きく異なるという問題がある．本論文では，タスク処理を詳細に定義可能なアプリケーションモデルと，そのモデルを扱うシミュレータを提案する．モデリング性能を評価するため，実際のエンジン制御アプリケーションをモデル化し，タスクの応答時間を測定した．その結果，エンジン制御アプリケーションの実行ファイルを命令セットシミュレータ上でシミュレーションした結果に対して，タスク応答時間の平均誤差を 7.4%に抑えることができた．

A Scheduling Simulator with task modeling capabilities for Real-time Applications

Yasumasa Sano,[†1] Yutaka Matsubara,[†1] Shinya Honda[†1]
and Hiroaki Takada [†1]

There are several real-time scheduling simulators to verify the behavior of real-time applications under different task scheduling algorithms. Current simulators cannot model the application accurately and, consequently the results of the simulation differ considerably from the actual behavior on a real computing system. This paper presents a scheduling simulator with task modeling capabilities for real-time applications. The proposed approach supports modeling of complex task control flows and dependency relations between tasks. In order to evaluate the modeling capabilities, we modeled a real engine control application and simulated it. We measured the average response times of the application model running on our scheduling simulator and compared them with the ones obtained by running the real engine application binaries on an instruction set simulator. The average of the percentage error between both simulations was only 7.4%.

1. はじめに

これまでに，リアルタイムシステム向けのタスクスケジューリングアルゴリズムが数多く提案されている．しかし，これらの多くは，理論的なアルゴリズムの提案に留まっており，実際のリアルタイム OS（RTOS）に実装されているものは少ない．そのため，様々なスケジューリングアルゴリズムを用いて，RTOS 上で動作する既存のアプリケーションの振る舞いを検証する場合には，RTOS のスケジューラを変更する必要がある．RTOS のスケジューラを変更することは，RTOS の内部構造を詳細に把握し，ハードウェアに依存する高度なプログラミングスキルを要求されるため容易ではない．

スケジューリングアルゴリズムの動作を検証する手法として，スケジューリングアルゴリズムに特化したシミュレータ（スケジューリングシミュレータと呼ぶ）が開発されている．スケジューリングシミュレータは，モデル化したアプリケーションの情報を入力すると，実装されているスケジューリングアルゴリズムの動作をシミュレーションし，入力したアプリケーションをスケジュールした結果を出力するツールである．

しかし，これまでに開発されたスケジューリングシミュレータは，タスクの実行時間や起動周期などを固定値でモデル化するため，実際の複雑なアプリケーションの振る舞いを正確にモデル化することは困難である．例えば，条件分岐の結果により，実行時間が大きく変動するようなタスクをモデル化するのは難しい．そのため，シミュレータの出力結果は，実機上での動作と大きく異なってしまうという問題がある．

本論文では，この問題を解決する手段として，タスク処理定義可能なスケジューリングシミュレータ *schesim* を提案する．*schesim* の主な特徴は，処理内容を詳細に定義できるタスクモデルを導入することで，実際のアプリケーションをより正確にモデル化できることである．加えて，RTOS の API と同等の機能を実現する関数群や，外部割込みやメッセージ受信などの非周期的に発生するイベントを定義できる機能を持つ．特定の RTOS やハードウェアアーキテクチャを前提として動作するスケジューリングアルゴリズムをシミュレーションすることもできる．例えば，マルチコアプロセッサ向けスケジューリングや，階層型スケジューリングをサポートしている．

†1 名古屋大学 大学院情報科学研究科
Graduate School of Information Science,Nagoya University

Vol.2011-SLDM-149 No.42
Vol.2011-EMB-20 No.42
2011/3/19

本論文の構成は，以下の通りである．まず，第 2 章で，スケジューリングシミュレータに対する機能要件を整理する．第 3 章では，*schesim* について詳細に述べる．第 4 章では，既存のエンジン制御アプリケーションのモデルを作成し，*schesim* でシミュレーションした事例について述べる．第 5 章で，関連研究を述べる．最後に，第 6 章で，本論文のまとめと，今後の展望について述べる．

2. スケジューリングシミュレータの要件

スケジューリングシミュレータの要件を以下にまとめる．

- 環境実現性：実機に近い環境を実現するための要件
  - (1) 非同期処理をシミュレーションできること
    外部割込みやメッセージ受信などの，発生間隔に周期性のないイベントをモデル化するために必要な要件である．
  - (2) タスクから RTOS の API を呼び出しできること
    タスクの起動や，タスク間の排他制御などをモデル化するために必要な要件である．
  - (3) マルチコアプロセッサ向けスケジューリングアルゴリズムをシミュレーションできること
- 柔軟性：複雑なタスク設計に対応するための要件
  - (4) タスクの処理に制御構造を記述できること
    条件分岐やタスク内の内部変数を用いて，処理内容を分けることで，詳細にタスクをモデル化するために必要な要件である．この要件を満たすことで，実行時間の変動するタスクもモデル化できる．
  - (5) タスク間の依存関係を記述できること
    タスク間での共有変数を用いて，タスクの処理において処理内容の変更や，条件分岐をモデル化するために必要な要件である．
- 拡張性：シミュレータを容易に拡張するための要件
  - (6) ユーザ定義のスケジューリングアルゴリズムを容易に実現できること
- 使用性：容易に使用するための要件
  - (7) アプリケーションモデルを容易に作成できる機能を持つこと
  - (8) シミュレーション結果を容易に利用できる機能を持つこと

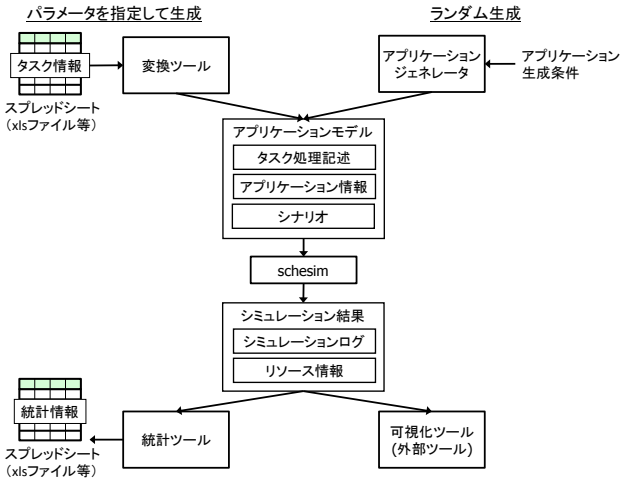

図 1 シミュレーションのフロー図
Fig. 1 Simulation flow diagram.

3. スケジューリングシミュレータ *schesim*

3.1 *schesim* の概要

第 2 章の要件を踏まえて，*schesim* の機能概要について述べる．要件（1）は，シナリオファイルにより実現する．要件（2），(4)，(5）は，タスク処理記述ファイルにより実現する．要件（3）は，標準でサポートしている．要件（7）は，モデル作成を支援するツールを提供することで実現する．要件（8）については，シミュレーション結果を可視化するツールに対応し，さらに，統計情報を計算するツールを用意することで実現する．要件（6）については，シミュレータを拡張することで，新しいアルゴリズムを実装できる．しかし現状では，シミュレータの内部構造を理解する必要がある．従って，要件（6）を十分に実現できているとは言い難い．より容易にシミュレータを拡張する方法は，今後の課題とする．

3.2 シミュレーションの流れ

schesim でのシミュレーションの流れを**図 1** に示す．まず，ユーザは，アプリケーションのモデルを作成する．アプリケーションモデルは，アプリケーション情報ファイル，タスク処理記述ファイル，シナリオファイルの 3 つで構成される（ただし，シナリオファイルは必

Vol.2011-SLDM-149 No.42
Vol.2011-EMB-20 No.42
2011/3/19

```
"task": [{
        "period": 36,       /* 起動周期 */
        "priority": 1,      /* 優先度 */
        "id": 1,            /* タスク ID */
        "deadline": 36,     /* 相対デッドライン */
        "attr": "cyclic",   /* 起動属性 */
        "offset": 5}],      /* 初期起動オフセット */
```

図 2 アプリケーション情報ファイルでのタスク定義の例
Fig. 2 Example of Task Model in Application Information File.

須ではない）．アプリケーション情報ファイルとタスク処理記述ファイルは 2 つの支援ツールを用いて作成することができる．1 つ目は，タスクのパラメータが入力されたスプレッドシート形式のファイルから生成する方法である．2 つ目は，タスクのパラメータを乱数を用いて決定し，生成する方法である．アプリケーションモデルをシミュレータに入力すると，アプリケーション情報ファイルにて選択したスケジューリングアルゴリズムをシミュレーションし，アプリケーションのスケジュール結果を出力する．シミュレーション結果は，シミュレーションログとリソース情報で構成される．シミュレーションログとリソース情報を可視化ツールに入力することで，シミュレーション時のタスクの切り替えや，API 呼び出しの情報を可視化することができる．また，シミュレーションログを統計ツールに入力することで，システム全体のタスク数と CPU 利用率，タスク毎の起動回数，実行時間，応答時間，CPU 利用率の統計情報を得ることができる．

3.3 アプリケーションモデル

3.3.1 アプリケーション情報ファイル

アプリケーション情報ファイルは，システムのプロセッサコア数や，アプリケーションの構成，スケジューリング方式などを JSON（JavaScript Object Notation）形式で記述したテキストファイルである．

設定できるパラメータ情報は，コアについては，そのコアで動作するアプリケーションをスケジューリングするアルゴリズム（グローバルスケジューリングアルゴリズム）とコア ID である．アプリケーションについては，所属するタスクをスケジューリングするアルゴリズム（ローカルスケジューリングアルゴリズム），プロセッサ利用率，起動周期，アプリケーション ID，優先度である．タスクについては，起動周期，優先度，タスク ID，相対デッドライン，起動属性，初期起動オフセットである．起動属性は，周期，非周期，離散から選択できる．**図 2** に，アプリケーション情報ファイルにおけるタスク情報の記述例を示す．

選択できるスケジューリングアルゴリズムについて述べる．シングルプロセッサ向けスケジューリングアルゴリズムとして，FPS（Fixed Priority Scheduling），EDFS（Earliest Deadline First Scheduling）がある．FPS での優先度の決め方は RM（Rate Monotonic），DM（Deadline Monotonic），ランダムから選択できる．階層型スケジューリングアルゴリズムとしては，BSSA（Bandwidth Sharing Server Algorithm）[1]，TPA（Temporal Protection Algorithm）[2] を選択できる．マルチコアプロセッサ向けスケジューリングアルゴリズムとしては，各プロセッサコア毎にシングルプロセッサ向けスケジューリングアルゴリズムを選択できる．すなわち，AMP 型マルチコアプロセッサアーキテクチャのみを想定している．

3.3.2 タスク処理記述ファイル

タスク処理記述ファイルには，個々のタスクの処理内容を Ruby の関数定義と同様の方法で記述する．処理内容を記述する際には，内部変数や，if 文，while 文といった Ruby の制御構文を使用できる．さらに，API として以下の関数を呼び出すことができる．

- exc：指定した単位時間分のプロセッサ時間を消費する
- act_tsk：指定した ID を持つタスクを起動する（μITRON 仕様準拠[3]）
- wai_sem/sig_sem：セマフォを獲得，解放する（μITRON 仕様準拠）
- GetResource/ReleaseResource：リソースを獲得，解放する（OSEK OS 仕様準拠[4]）
- SetEvent/WaitEvent/ClearEvent：イベントをセット，待つ，クリアする（OSEK OS 仕様準拠）

図 3 にタスク処理記述ファイルの例を示す．この例では，タスク 1 の処理内容を定義している．2 行目で，アプリケーションで使用するリソースを定義する．5 行目から 14 行目までがタスク 1 の処理内容である．6 行目から 9 行目は，内部変数 mode1 が 0 のときのみ，タスク ID が 2 のタスクを起動するという処理をする．10 行目では，リソース res1 を獲得し，12 行目でリソース res1 を解放する．その間に，11 行目でプロセッサ時間 1 を消費する．13 行目でプロセッサ時間 1 を消費し，処理を終了する．*schesim* では，このような多彩な処理記述方法により，RTOS 上で動作しているアプリケーションを，より正確にモデル化したシミュレーションモデルが作成できる．

3.3.3 シナリオファイル

リアルタイムアプリケーションには動作するアプリケーションには，割込みのように非同期に発生するイベントや，外部の環境に応じて処理が変化するタスクが存在することが多い．これらをモデル化するために，シナリオファイルを用いる．**図 4** にシナリオファイルの例を示す．左から順に，イベントの発生時刻，コア ID，発生するイベント，イベントへの

Vol.2011-SLDM-149 No.42
Vol.2011-EMB-20 No.42
2011/3/19

```
 1 class TCB
 2     @@res1 = RESOURCE.new(1)            # リソース 1 の定義
 3     @@mode1 = 0                         # モード変数の定義
 4     # タスク ID が 1 のタスク処理記述 #
 5     def task1
 6         if @@mode1 == 0
 7             act_tsk(2)                  # タスク ID が 2 のタスクを起動
 8             @@mode1 = 1
 9         end
10         GetResource(@@res1)             # リソース 1 を獲得
11         exc(1)                          # 1 単位時間実行
12         ReleaseResource(@@res1)         # リソース 1 を解放
13         exc(1)                          # 1 単位時間実行
14     end
       ### 以下省略 ###
```

図 3 タスク処理記述ファイルの例
Fig. 3 Example of Task Modeling File.

```
12:1:chg_mod:@@mode2:2          # 変数@@mode2 の値を 2 に変更
15:1:act_tsk:4                  # タスク ID4 のタスクを起動
```

図 4 シナリオファイルの例
Fig. 4 Example of Scenario File.

引数を表している．イベントとしては，3.3.2 項で挙げた関数に加えて，タスクの内部変数の値を変更する “chg_mod” を使用できる．

3.4 *schesim* によるシミュレーション

schesim は，スクリプト言語 Ruby で実装したシミュレータであり，Ruby1.8.7 以降で動作する．

シミュレータ内部では離散的なシステム時刻を管理し，システム時刻を更新するタイミングでタスクスケジューリングや API の処理を実行する．例えば，実行中のタスクが exc(2) を呼び出すと，呼び出した時刻からシステム時刻が 2 単位時間経過した後に，次の処理を実行できる．その他の関数や制御構文は，同一のシステム時刻内で処理される．すなわち，タスクスケジューリングや，タスク切替えの処理時間は 0 となる．

3.5 シミュレーション結果

シミュレーションが終了すると，シミュレーションログとリソースファイルが出力される．シミュレーションログは，タスクの状態遷移や API 呼び出しの履歴を時系列で出力したものである．**図 5** に，シミュレーションログの例を示す．左から順に，システム時刻，コア ID，履歴を表す．このフォーマットは，TOPPERS/FMP カーネル[5] が出力するログと同様であるが，階層型スケジューリングのための独自の情報として，アプリケーションの状態遷移や，バジェットの残量を追加している．1 行目は TASK1 が ReleaseResource を呼び出した履歴で，4 行目は，TASK1 が DORMANT 状態に遷移した履歴である．2，3 行目と 5 行目は，それぞれアプリケーションのバジェットの残量と，状態遷移を表す履歴である．リソースファイルは，可視化ツールの入力として必要なファイルであり，アプリケーション情報ファイルから自動で生成される．

```
1  [100]:[1]: applog strtask : TASK 1 : ReleaseResource().
2  [100]:[2]: budget of application 2 is 10.
3  [110]:[1]: budget of application 1 is 0.
4  [110]:[1]: task 1 becomes DORMANT.
5  [110]:[1]: application 1 becomes EXPIRED.
```

図 5 シミュレーションログの例
Fig. 5 Example of Simulation Log.

3.6 可視化ツール

可視化として，TLV（Trace Log Visualizer）[6] を使用できる．TLV でシミュレーション結果を可視化するためには，*schesim* によって出力されたシミュレーションログとリソースファイルを TLV に入力する．

図 6 に，TLV によるシミュレーション結果の可視化例を示す．可視化しているアプリケーションでは，2 つのコアで 3 つずつタスクが存在している．1 つ目のコアでは，TASK1 から TASK3 が動作しており，優先度は，TASK3，TASK2，TASK1 の順に高い．2 つ目のコアでは，TASK4 から TASK6 が動作し，優先度は，TASK6，TASK5，TASK4 の順に高い．どちらのコアでもタスクは FPS でスケジューリングされる．TASK1 と TASK2 は，処理の一部でリソースを用いて排他制御を行っている．TASK4 は実行開始直後に，act_tsk を呼び出して TASK5 を起動している．

4. エンジン制御アプリケーションへの適用

4.1 エンジン制御アプリケーションの特徴

schesim のモデリング性能の評価として，エンジン制御アプリケーションをモデル化した

Vol.2011-SLDM-149 No.42
Vol.2011-EMB-20 No.42
2011/3/19

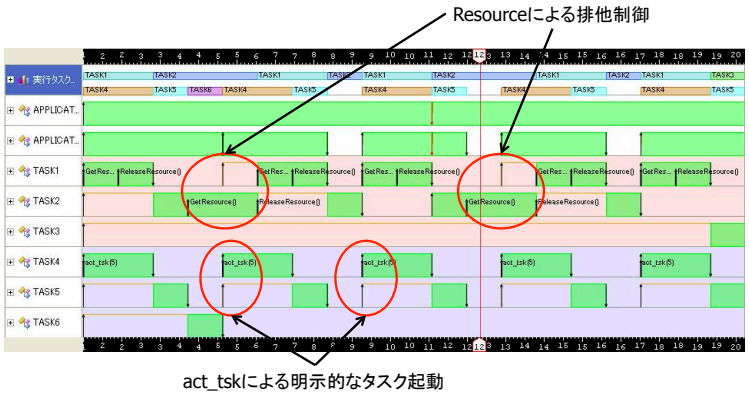

図 6 TLV によるシミュレーション結果の可視化
Fig. 6 Example of TLV Screenshot.

事例について述べる．モデリングの対象とするエンジン制御アプリケーションは，タスク数が 63 個，割込み処理の数が 62 個で構成されており，RTOS 上で動作する．タスクは FPS でスケジューリングされる．エンジンの回転数によって，タスクの実行時間や，割込み処理の発生時刻は変化する．このように複雑なアプリケーションを既存のスケジューリングシミュレータでシミュレーションすると，起動間隔や実行時間の変動をモデルに反映できないため，実機上での動作と大きく異なってしまう可能性がある．

4.2 エンジン制御アプリケーションのモデル化

今回は，エンジン制御アプリケーションの実行ファイルを命令セットシミュレータ上でシミュレーションした結果から，タスク，割込み処理に関する情報を得ている．これらの情報を用いてアプリケーションモデルを作成した．

schesim のアプリケーションモデルでは，割込み処理とタスクを区別しないため，割込み処理を全てのタスクより高い優先度を持つタスクとして考えた．タスク，割込み処理の実行時間は数 μs 以上であることから，シミュレータのシステム時刻の単位を μs と設定した．割込み禁止区間による割込み処理の遅延，API 実行時間，タスクの切り替えや，スケジューリング等の OS の処理時間はタスクの実行時間の比べて非常に短いため，モデルに反映していない．タスク，割込み処理の実行時間が内部変数の値により規則的に大きく変化する場合のみ，タスク処理記述によりモデル化した．

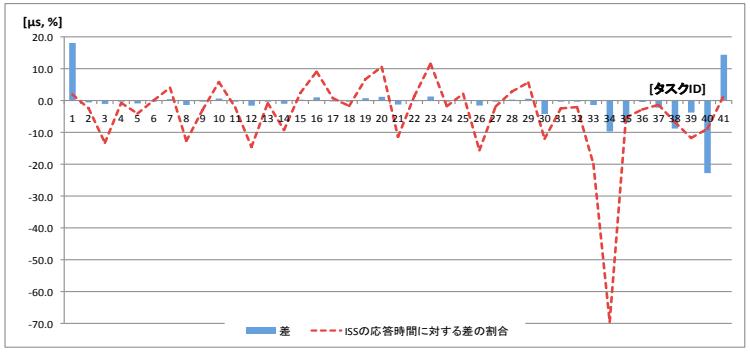

図 7 ISS での動作と，*schesim* での動作のタスク平均応答時間の差
Fig. 7 Error of Average Response Time between ISS and *schesim* simulations.

4.3 評　　価

モデリング性能を評価するため，*schesim* によるシミュレーション結果と，エンジン制御アプリケーションの実行ファイルを命令セットシミュレータ上でシミュレーションした結果において，タスクの平均応答時間を比較した．**図 7** に，タスクの平均応答時間の差と，ISS の平均応答時間に対する差の割合を示す．差の絶対値は，平均で 2.7μs，最大で 22.8μs となった．割合の絶対値は，平均で 7.4%，最大で 69.5%となった．タスク 34 において，割合の絶対値が最大値 69.5%となった理由は，タスクの実行時間と応答時間が非常に短いため，モデル誤差の影響による，平均応答時間の差の絶対値は 10μs と小さいにもかかわらず，差の割合は大きくなってしまったためと考えられる．このアプリケーションには，実行時間が数十 μs のタスクから，千数百 μs のタスクまで存在し，起動間隔も数千 μs のタスクが多いため，差の平均が 2.7μs，割合の平均が 7.4%という結果は十分に小さいといえる．エンジン制御アプリケーションのように複雑なアプリケーションに対しても，*schesim* を適用することで，正確なシミュレーション結果を得ることができた．

5. 関 連 研 究

リアルタイムスケジューリングアルゴリズムの研究においては，その動作の妥当性を検証するために，独自のスケジューリングシミュレータを開発することが多い．RTMultiSim[7] は，マルチコアプロセッサ向けのスケジューリングアルゴリズムをシミュレーションする．しかし，このようなスケジューリングシミュレータは，多様なスケジューリングアルゴリズ

Vol.2011-SLDM-149 No.42
Vol.2011-EMB-20 No.42
2011/3/19

ムのサポートをしていないことや，タスクのパラメータを定数値でモデリングすることが多いため，要件（4），(5），(6）を満たしていない．

モデル化したアプリケーションを複数のスケジューリングアルゴリズムでスケジューリングして結果を比較することや，スケジューリングアルゴリズムの教育を目的として，FORTISSIMO[8]，AURTSS[9]，Realtss[10]，RTsim[11] などが開発されている．これらのシミュレータは，共通して，タスクの具体的な処理内容を記述する機能がないため，要件（4）と要件（5）を満たすことができず，分岐による実行時間の変動や，タスク間の依存関係をモデルに反映できない．

C/C++言語を用いて，要件（2）を満たすことのできるシミュレータも開発されている．ARTISST[12] は，タスク間の排他制御を実現するための API が用意されているが，非同期処理やタスクの制御構造を記述できないため，要件（1）と要件（4）を満たさない．RTSSim[13] は，絶対時刻を指定したイベント処理機能や，マルチコアプロセッサ環境に対応していないため，要件（1）と要件（3）を満たすことができない．RTSIM[14] は，オープンソースであるため，シミュレータを容易に拡張することもできる[15]．しかし，RTSIM のタスク処理記述方法は，実行したい命令列をコンフィギュレーション時に列挙する必要があり，タスク処理内での条件分岐や，他のタスクと変数を共有できないなどの制限がある．さらに，RTSSim と同じく，絶対時刻を指定したイベント処理機能や，マルチコアプロセッサ環境に対応していないため，要件（1）と要件（3）を満たすことができない．

6. おわりに

本論文では，タスク処理を詳細に定義可能なアプリケーションモデルと，そのモデルを扱うシミュレータを提案した．実際のエンジン制御アプリケーションをモデル化し，タスクの応答時間を測定した結果，実行ファイルを命令セットシミュレータ上でシミュレーションした結果に対して，タスク応答時間の平均誤差は 7.4%であった．エンジン制御アプリケーションのように複雑なアプリケーションに対しても，*schesim* を適用することで，正確なシミュレーション結果を得ることができることを確認した．今後の展望は，SMP 型マルチプロセッサアーキテクチャを想定したスケジューリングアルゴリズムのサポートや，新しいスケジューリングアルゴリズムを容易に実装するためのインタフェースの定義などの改良を重ね，オープンソース化する予定である．

参考文献

1) G.Lipari and K.Baruah: Efficient Scheduling of Real-Time Multi-Task Applications in Dynamic Systems, *In Proceedings of IEEE Real-Time Technology and Applications Symposium* (2000).
2) 松原豊，本田晋也，高田広章：時間保護のためのタスク起動遅延付き階層型スケジューリングアルゴリズム，情報処理学会研究報告， Vol.2010-EMB-19 (2010).
3) (社) トロン協会 ITRON 仕様検討グループ：μ ITRON4.0 仕様 Ver.4.03.03 (2008).
4) OSEK/VDX: OSEK/VDX Operating System Specification 2.2.3 (2005).
5) TOPPERS Project: TOPPERS/FMP kernel, http://www.toppers.jp/fmp-kernel.html.
6) 後藤隼弐，本田晋也，長尾卓哉，高田広章：トレースログ可視化ツール TraceLogVisualizer (TLV)， コンピュータ ソフトウェア， Vol.27, No.4, pp.8–23 (2010).
7) Hangan, A. and Sebestyen, G.: Simulation-based Evaluation of Real-Time Multiprocessor Scheduling Strategies, *In Proceedings of IEEE International Conference on Intelligent Computer Communication and Processing*, pp.375–378 (2010).
8) Kramp, T., Adrian, M. and Koster, R.: An Open Framework for Real-Time Scheduling Simulation, *In Proceedings of International Parallel and Distributed Processing Symposium 2000 Workshops*, pp.766–772 (2000).
9) Yaashuwanth, C. and Ramesh, R.: Web-Enabled Framework for Real-Time Scheduler Simulator: A Teaching Tool, *In Proceedings of 2nd International Conference on Computer Research and Development*, pp.826–830 (2010).
10) Diaz, A., Batista, R. and Castro, O.: Realtss: a real-time scheduling simulator, *In Proceedings of 4th International Conference on Electrical and Electronics Engineering*, pp.165–168 (2007).
11) Jr., A.M., Miola, M.B. and Nabuco, V.A.: Teaching Real-Time with a scheduler simulator, *In Proceedings of 31st Annual Frontiers in Education Conference*, pp.15–19 (2001).
12) Decotigny, D. and Puaut, I.: ARTISST: An Extensible and Modular Simulation Tool for Real-Time Systems, *In Proceedings of IEEE International Symposium on Object-oriented Real-time Distributed Computing*, pp.365–372 (2002).
13) Kraft, J.: RTSSim - A Simulation Framework for Complex Embedded Systems, *Mlardalen University, Technical Report* (2009).
14) Casile, A., Buttazzo, G., Lamastra, G. and Lipari, G.: Simulation and Tracing of Hybrid Task Sets on Distributed Systems, *In Proceedings of IEEE International Conference on Real-Time Conputer Systems and Applications*, pp.249–256 (1998).
15) Bartolini, C. and Lipari, G.: RTSIM, http://rtsim.sssup.it/.