Vol.2014-SLDM-165 No.2
Vol.2014-EMB-32 No.2
2014/3/15

重要タスクの応答時間を短縮するリアルタイムスケジューリング

田中 清史[1,a)]

概要：周期タスクに対する代表的なリアルタイムスケジューリング法として，Rate Monotonic（RM）とEarly Deadline First（EDF）がある．RM はタスクの重要度を周期として反映させることにより重要タスクのジッタや応答時間を短く保つことが可能であるが，重要度とは無関係に周期を設定することは困難であり，さらにプロセッサを 100%まで使用することが不可能である．EDF は 100% のプロセッサ使用率を可能とするが，タスクの重要度を反映することが不可能であり，特定の重要なタスクのジッタや応答時間を短く保つことができない．本稿では，各タスクが周期と独立した重要度を持つ場合に，重要タスクの応答時間を短縮するスケジューリング方法である適応型 EDF に対し，2 つの改良方法を提案する．シミュレーションによる評価では，2 つの提案改良方法を組み合わせることにより，従来の適応型 EDF に対して平均応答時間が最大で 29.2% 短縮した．

Real-Time Scheduling for Shortening Response Times of Important Tasks

Kiyofumi Tanaka[1,a)]

Abstract: Rate monotonic (RM) and earliest deadline first (EDF) are representative scheduling algorithms for real-time periodic tasks. RM has a merit that tasks with high-priority (short-period) have small jitters and short response times. However, it is impossible for processor utilization to reach 100% or that tasks are given importance independent of their periods. On the other hand, EDF can utilize processors by 100% with schedulability, while it cannot give tasks fixed priorities or importance, therefore, it is difficult to keep jitters or response times of particular tasks short. This paper proposes two improvement techniques for the adaptive EDF which is a scheduling method that shortens response times of tasks with importance independent of their periods. In the evaluation with simulation, by combining the two improvement techniques, the average response times were reduced by 29.2%, at maximum, compared to the adaptive EDF.

1. はじめに

リアルタイムシステムシステムにおいて，周期タスクを対象とする代表的なスケジューリングアルゴリズムとしてRate Monotonic（RM）と Earliest Deadline First（EDF）がある [1]．RM は固定優先度アルゴリズムの一つであり，周期の短いタスクが高い優先度を持つとみなされ，優先的に実行される．RM では優先度が高い（周期の短い）タスクのジッタや応答時間が小さいという特長があるが，スケジューラビリティを確保しつつプロセッサ時間を 100%使用することが不可能である．EDF は動的優先度アルゴリズムの一つであり，絶対デッドラインの近いタスクを優先的に実行する．100%のプロセッサ使用率の下でスケジューラビリティが保証されるという特長があるが，タスクに静的な優先度を与えることができないため，特定の重要度の高いタスクのジッタや平均応答時間を小さく保つことが困難である．

ハードリアルタイムシステムでは，各タスクがデッドライン制約を満たすことが最重要であるが，加えて閉ループ制御系システム等では，ジッタの発生がシステム性能の低

[1] 北陸先端科学技術大学院大学
JAIST, Asahidai 1-1, Nomi, Ishikawa 923–1292, Japan
[a)] kiyofumi@jaist.ac.jp

© 2014 Information Processing Society of Japan

下や不安定性を引き起こす可能性がある [2], [3]．（ここで“ジッタ”とは，応答時間の揺れを意味する．言い換えると，繰り返される実行の間での応答時間の変動量である．）また，将来的には，異なる重要度（クリティカリティ）のタスクが混在するシステムが一般的になることが予想されるため [4], [5]，本研究では 100%のプロセッサ使用率を達成する EDF に基づき，スケジューラビリティを確保しつつ，システムにとって（周期とは無関係に）重要な周期タスクの応答時間を短縮することでジッタを抑えることを目的とする．

著者は過去の研究で，特定の周期タスクの応答時間を短縮する方式である適応型 EDF を提案した [6], [7]．本方式は最悪実行時間（WCET）に加えて予測実行時間を導入し，特定タスクの実行を 2 つのサブインスタンスに分解する．第一サブインスタンスに予測実行時間に基づく早いデッドラインを与えることにより，EDF によってより早くスケジュールされ，実際の実行時間が予測実行時間内で終了する場合に応答時間を短縮可能である．本稿では適応型 EDF に対する 2 種類の改良方法である余剰バンド幅占有法と漸進デッドライン更新法を提案し，評価において平均応答時間を更に短縮することを示す．

2. 適応型 EDF

本節では，著者が過去に提案した適応型 EDF [6], [7] の概要を述べる．

2.1 最悪実行時間と予測実行時間

従来のリアルタイムスケジューリングアルゴリズムおよびスケジューラビリティ解析の説明では，タスクの実行時間として最悪実行時間（WCET）が想定されてきた [8]．しかし，実際の実行ではほとんどの場合で実行時間は WCET よりも短い．特にプロセッサアーキテクチャやプログラム構造の大規模化／複雑化により，正確な WCET の取得は極めて困難となり [9], [10]，結果的に WCET は悲観的に見積もられることになり，実際の実行時間との差は拡大する傾向にある．

適応型 EDF はこの差を利用し，特定タスクの応答時間を短縮する方法である．すなわち，重要タスクについて実際の実行時間の予測を行い，予測実行時間に基づいた仮の（短い）デッドラインを設定し，EDF における当該タスクの優先順位を向上させる．（デッドライン設定については，2.2 節参照.）

実行時間の予測方法は適応型 EDF の定義の対象外であり，様々な方法が採られる可能性がある．文献 [6] では以下の実行時間予測方法を使用した．

$$C_{i_0}^{PET} = C_i^{WCET}$$
$$C_{i_k}^{PET} = \alpha \times C_{i_{k-1}}^{PET} + (1-\alpha) \times C_{i_{k-1}}^{AET} \quad (k>0) \quad (1)$$

ここで，$C_{i_k}^{PET}$ は周期タスク τ_i の k 回目 $(k = 0, 1, \ldots)$ の実行のための予測実行時間である．$C_{i_{k-1}}^{AET}$ は同一タスクの前回の実行で実際に費やした実行時間を意味する．初期値 $C_{i_0}^{PET}$ はそのタスクの WCET（C_i^{WCET}）とする．この式は重み係数を α として，前回の実行で予測値として使用した $C_{i_{k-1}}^{PET}$ と前回の実際の実行時間との加重平均を計算し，実行時間の予測値としている．

2.2 適応型 EDF の定義

適応型 EDF は周期タスクセットを対象とする．タスクは互いに独立であり，各タスクは周期の長さと等しい相対デッドラインを持つものとする．また，各タスクは（周期と独立した）重要度を持つ．

適応型 EDF では，重要度の高い周期タスクのインスタンスは 2 つに分解される．すなわち，周期タスク τ_i の重要度が高いと仮定すると，その k 番目のインスタンス J_{i_k} の実行を 2 つのサブインスタンス（$J_{i_k}^{PET}$ と $J_{i_k}^{REST}$）に分割する．$J_{i_k}^{PET}$ は J_{i_k} の最初から予測された終了時刻までの実行に相当する．$J_{i_k}^{REST}$ は予測された終了時刻から後の実行に相当する．τ_i の最悪実行時間を C_i^{WCET}，J_{i_k} の予測実行時間を $C_{i_k}^{PET}$，$J_{i_k}^{REST}$ の実行時間を $C_{i_k}^{REST}$ とする．$C_{i_k}^{REST}$ は最大となる $C_{i_k}^{REST} = C_i^{WCET} - C_{i_k}^{PET}$ の値とする．すなわち，$J_{i_k}^{REST}$ の実行時間は $0 \le C_{i_k}^{REST} \le C_i^{WCET} - C_{i_k}^{PET}$ であるが，以下ではその最大の値とする．

$J_{i_k}^{PET}$ と $J_{i_k}^{REST}$ には，以下の式によりそれぞれ絶対デッドライン $d_{i_k}^{PET}$ と $d_{i_k}^{REST}$ が設定される．

$$d_{i_k}^{PET} = k \times T_i + \frac{C_{i_k}^{PET}}{U_i} \quad (2)$$

$$d_{i_k}^{REST} = d_{i_k}^{PET} + \frac{C_{i_k}^{REST}}{U_i} = (k+1) \times T_i = d_{i_k} \quad (3)$$

式の中で T_i は τ_i の周期であり，U_i は τ_i による WCET に基づいたプロセッサ使用率（$U_i = C_i^{WCET}/T_i$）である．また，d_{i_k} は J_{i_k} の本来の（周期タイミングと等しい）デッドラインである．

デッドラインが与えられると，2 つのサブインスタンスは EDF アルゴリズムにしたがってスケジュールされる．式 (2) と式 (3) の関係から，$d_{i_k}^{PET} \le d_{i_k}^{REST}$ であるため，$J_{i_k}^{PET}$ は $J_{i_k}^{REST}$ に先行して実行される．$J_{i_k}^{PET}$ が予測終了時刻あるいはその前に終了した場合は $J_{i_k}^{REST}$ は存在しない（実行されない）ことになる．そのような場合は，小さい $d_{i_k}^{PET}$ の効果により，EDF アルゴリズムにより絶対デッドラインが d_{i_k} である場合よりも早くスケジュールされ，応答時間が短縮することが期待できる．

2.3 適応型 EDF のスケジューラビリティ

適応型 EDF のスケジューラビリティは次のように論じることができる．

© 2014 Information Processing Society of Japan

重要度の高い周期タスク τ_i による，WCET に基づいたプロセッサ使用率を U_i とすると，適応型 EDF は τ_i から分割された 2 つのサブインスタンスを，U_i をサーババンド幅とする Total Bandwidth Server (TBS) [11] によって実行する状況と同じものと見なすことができる．（TBS は周期タスクと非周期タスクの混在するタスクセットを対象としたスケジューリングアルゴリズムであり，非周期タスクは WCET に基づくデッドラインが与えられ，タスクセット全体が EDF によってスケジュールされる．）すなわち，2 つのサブインスタンスの実行時間は $C_{i_k}^{PET}$ と $C_{i_k}^{REST}$ であり，TBS により与えられるそれぞれのデッドラインは式（2）と式（3）に等しい．したがって文献 [11] における TBS のスケジューラビリティの性質が保たれるため，全タスクによるトータルプロセッサ使用率 U に関して $U < 1$ であることがタスクセットがスケジュール可能となる必要十分条件となる．

2.4 適応型 EDF の実装の複雑さ

適応型 EDF ではタスクのインスタンスは 2 分割されるが，OS 管理においてはタスク情報は一つの情報セット（タスク制御ブロック）として存在するべきである．このことは，予測実行時間が経過した際，タスクの実行が終了していない場合はデッドラインを再設定し，レディキューに挿入し直すことで実現可能である．したがって EDF との相違点は，予測実行時間経過の出検とデッドラインの再設定，およびレディキューへの再挿入のみである．予測実行時間経過の検出を可能とするために，ティックタイミング毎にスケジューラの実行が必要となるが，これは通常のタイマー／ティック割込みと連動することで可能である．（これは OS が通常採る自然な手続きである．）また，式（2）の計算における定数による除算は乗算に置き換え可能であり，式（3）の計算は加算に過ぎないため，デッドラインの計算は大きなオーバヘッドにはならない．

3. 適応型 EDF の 2 つの改良方法

適応型 EDF に対する 2 つの改良方法である，余剰バンド幅占有法と漸進型デッドライン更新法を提案する．

3.1 余剰バンド幅占有法

適応型 EDF の基本的な考え方は，重要タスクのインスタンスを 2 つに分割して TBS で実行することである．適応型 EDF では重要タスクのデッドラインを求める際に使用するタスクのバンド幅として，式（2）からわかるように当該タスクのプロセッサ使用率である U_i を使用する．一方，TBS では周期タスクセットの使用率を U_p，サーバのバンド幅を U_s とした場合，$U_p + U_s \leq 1$ によってスケジューラビリティが保証される性質がある．したがって，式（2）におけるバンド幅として U_i の代わりに $1 - (U_p - U_i)$ を使用することが可能である．このことは，U_p が 1 より小さいときに，その余剰バンド幅である $1 - U_p$ を対象重要タスクの実行に使用可能であることを意味する．

結果的に，式（2）は以下のように変更可能である．

$$d_{i_k}^{PET} = k \times T_i + \frac{C_{i_k}^{PET}}{1 - (U_p - U_i)} \tag{4}$$

$U_p < 1$ ならば，$1 - (U_p - U_i) > U_i$ であることから，これにより計算されるデッドラインは式（2）で得られる値よりも小さくなり，EDF によって当該タスク実行が前倒しになることが期待できる．

図 1 に余剰バンド幅占有法によって応答時間が短縮する例を示す．図では 2 つの周期タスク τ_1, τ_2 の周期がそれぞれ $T_1 = 4$, $T_2 = 6$ であり，実行時間が $C_1^{WCET} = C_{1*}^{AET} = 2$, $C_2^{WCET} = C_{2*}^{AET} = 1$ である．（実行時間は変動しない．）τ_2 が重要タスクであると仮定する．EDF でスケジュールした結果，図中 (1) のように τ_2 の最初のインスタンスの応答時間は 3 となる[*1]．一方，(2) は余剰バンド幅占有法を適用した場合であり，τ_2 の使用可能なバンド幅が $1 - 2/4 = 0.5$ となり，式（4）により当該インスタンスには破線矢印のデッドラインが与えられ，(1) のスケジュールよりも前倒しでスケジュールされ，応答時間は 1 となる．

3.2 漸進デッドライン更新法

文献 [6] における評価では，本稿 2.1 節の実行時間の予測方法（$\alpha = 0.5$）を使用しており，結果としては実行時間が既知の場合（すなわち理想的に完全に予測可能な場合）と比較し，平均応答時間に大きな差があった．このことは，実行時間の予測方法を改善することで大きな改善効果が期

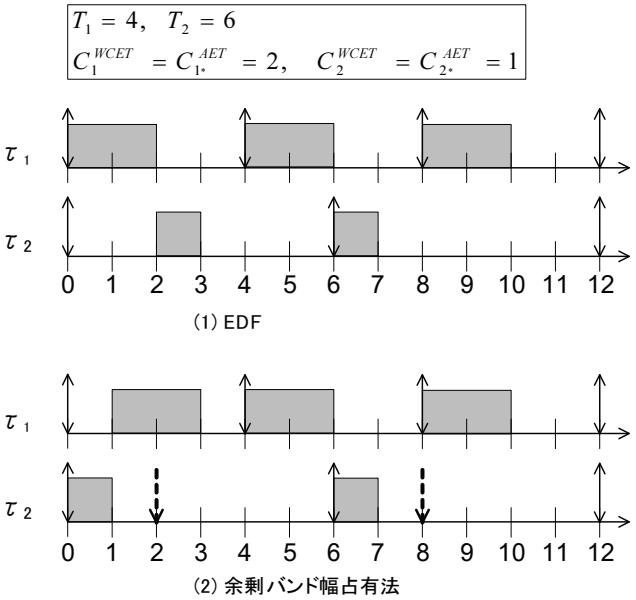


図 1 余剰バンド幅占有法による応答時間の短縮例．

[*1] この例のように実行時間が常に WCET と等しい場合は，既存の適応型 EDF でスケジュールした場合も同じ応答時間となる．

© 2014 Information Processing Society of Japan

待できることを意味する．

適応型 EDF の性質を考慮すると，予測が過大見積りとなった場合は十分に短いデッドラインが得られず，応答時間の改善が期待できない．逆に過小見積りの場合は，予測実行時間を経過した後の残りの実行は WCET に基づくデッドラインにしたがってスケジュールされることになり，やはり短い応答時間は期待できない．実行時間の予測に基づく適応型 EDF では，完全な予測方法が存在しない限り，最適な応答時間は達成されない．

適応型 EDF のこれらの問題点は次のように解決可能である．予測実行時間として，実行要求後の最初は最小のもの，すなわち 1 ティックの長さを使用する．予測時間が経過したとき，従来の適応型 EDF では残りの実行に対して WCET に基づいたデッドラインを割当てていたが，これを改良し，残りの実行時間が 1 ティックと仮定した場合のデッドラインを与える．以下，実行が完了するまで同様にデッドライン更新を繰り返す．この手法を漸進デッドライン更新法と呼ぶ．この方法により，実行時間の過大見積りに起因する過大デッドラインの問題，および過小見積り時に残り実行として WCET を仮定することによるデッドライン過大延長の問題を解決可能である．

漸進デッドライン更新法は以下のように定式化される．周期タスク τ_i の k 番目のインスタンス J_{i_k} の実行をサブインスタンス $J^1_{i_k}, J^2_{i_k}, J^3_{i_k}, \ldots$ に分割する．$J^1_{i_k}$ は J_k の最初から 1 ティック実行後までの実行に相当する．$J^j_{i_k}$ は $j-1$ ティック実行後から j ティック実行後までの実行に相当する．J_{i_k} が j ティックで終了した場合は，$J^{j+1}_{i_k}$ 以降のサブインスタンスは存在しないことになる．サブインスタンス $J^j_{i_k}$ には以下のデッドラインが与えられる．

$$
\begin{aligned}
d^1_{i_k} &= k \times T_i + \frac{1}{U_s} \\
d^j_{i_k} &= d^{j-1}_{i_k} + \frac{1}{U_s} \quad (j > 1)
\end{aligned} \tag{5}
$$

図 2 は（1）EDF，（2）従来の適応型 EDF で実行時間を過大見積りした場合，（3）適応型 EDF で実行時間を過小見積りした場合，および（4）漸進デッドライン更新法を適用した適応型 EDF のスケジュールを示している．τ_3 が重要タスクであり，その WCET は 4，実際の実行時間は 2 であるとする．EDF では τ_3 の応答時間は 6 ティックとなっている．実行時間過大見積り（実行時間=2 に対し，$C^{PET}_{3*} = 3$）の適応型 EDF では，デッドラインがティック 9 に設定され，応答時間は同じく 6 ティックとなっている．実行時間過小見積り（実行時間=2 に対し，$C^{PET}_{3*} = 1$）の適応型 EDF では，最初のサブインスタンスのデッドラインがティック 3 に設定されるが，予測実行時間内で当該インスタンスは終了せず，再度残り実行のためのデッドラインがティック 12 に設定され，結果的に応答時間は同じく 6 ティックとなっている．これらに対し，漸進デッドライン更新法では，最初のサブインスタンスのデッドラインがティック 3 に設定され，（3）と同じくこのサブインスタンス実行では終了しないが，次のサブインスタンスのデッドラインがティック 6 に設定されるため，結果的に応答時間は最少の 4 ティックとなっている．

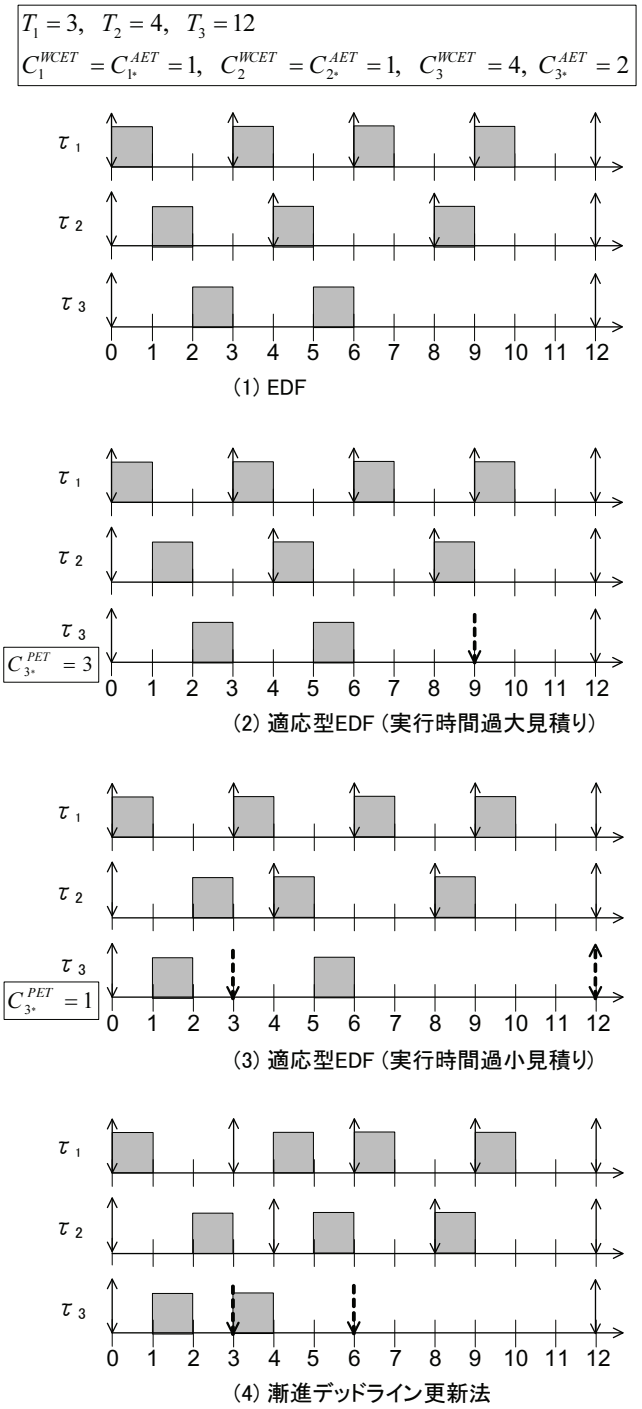


図 2 スケジュール例.

4. 評価

本節において，シミュレーションにより提案する改良版の適応型 EDF の効果を EDF，RM，DM[12]，および従来の適応型 EDF との比較によって示す．

© 2014 Information Processing Society of Japan

4.1 シミュレーション条件

比較対象の一つとして使用する Deadline Monotonic（DM）[12] は，RM と同様，固定優先度アルゴリズムの一つであるが，RM に対し，デッドラインが周期と一致するという条件を緩和するものである．すなわち，対象タスクは周期よりも短い相対デッドラインを持つことが可能である．この条件の緩和を利用し，余剰バンド幅占有法と同様に，余剰バンド幅から求められるデッドラインを重要タスクに設定する方法を採ることができる．したがって，重要タスク τ_i のデッドラインは以下で与えられる．（U_{ub} は RM にとってのプロセッサ使用率の上限値であり，シミュレーションでは 0.9 を使用した[*2]．）

$$d_{i_k} = k \times T_i + \frac{C_i^{WCET}}{U_{ub} - (U_p - U_i)} \quad (6)$$

なお，重要タスク以外のタスクは周期と等しいデッドラインを持つものとする．

比較対象の従来の適応型 EDF において，予測実行時間の算出における加重平均の重み係数（2.1 節における α）として 0.5 を使用した．

WCET に基づく CPU 使用率（U_p）が 70%から 100%までの 5%おきとなる周期タスクセットを複数用意した．各周期タスクに関して周期は 0〜100 ティックの範囲の一様分布乱数で決定し，最悪実行時間は周期の 10 分の 1 以上かつ 3 分の 1 以下の範囲の一様分布乱数で決定した．タスクの各実行インスタンスの実際の実行時間は WCET の 1/3 以上かつ WCET 以下の範囲の一様分布に従った．（同一タスクの実行は実行毎に実行時間が変動するモデルである．）観測時間は 100,000 ティックである．

各 U_p に対して，20 個の周期タスクセットに対してシミュレーションを行った場合の応答時間の平均値を示す．なお，全てのタスクセットのシミュレーションにおいてデッドラインミスは発生しなかった．

4.2 結果

図 3 は，最短周期のタスクを重要タスクとみなした場合の重要タスクの平均応答時間を示している．“適応型 EDF（R）” は余剰バンド幅占有法を付加した適応型 EDF，“適応型 EDF（I）” は漸進バンド幅更新法による適応型 EDF，“適応型 EDF（RI）” は両改良方法を付加した適応型 EDF である．横軸は U_p，縦軸は RM の結果で正規化した平均応答時間の値である．

図では，RM と DM が最短の応答時間となっている．このことは当然であり，RM と DM は常に（この場合は最も周期の短い）重要タスクを常に優先して実行するため，当該タスクの平均応答時間は最も短くなる．一方，EDF はタスクの重要性を考慮しないため応答時間が長くなっている．これに対し，適応型 EDF は EDF よりも応答時間を改善していることがわかる．

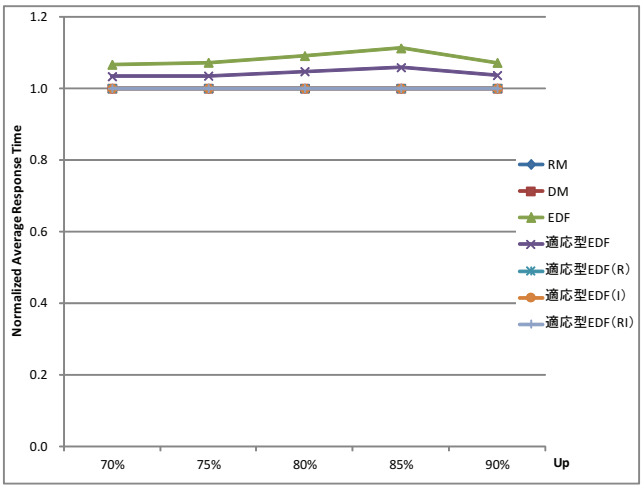


図 3 結果 – 最短周期のタスクを対象．

適応型 EDF（R）と適応型 EDF（I）は RM とほぼ同じ値となり，両改良方法が有効であることがわかる．（RM に対して適応型 EDF（R）は最大で 0.0056%，適応型 EDF（I）は 0.0061%の差であった．）最後に，適応型 EDF（RI）は RM と全く同じ値となった．このことから，重要度を周期に反映させた場合，提案方式は RM と同等の応答時間を達成可能であるといえる．

続いて**図 4** に周期の長さがタスクセット内で中間であるタスクを重要タスクとみなした場合の結果を示す．図からわかるように，RM と EDF はほとんど同じ値となった．DM は RM に対して大きく改善するが，U_p が大きくなるにしたがい RM の性能に近づいている．これは，U_p が大きくなるほど U_{ub} から得られる余剰バンド幅が小さくなるためである．（U_p が 90%のとき余剰バンド幅はゼロになる．）適応型 EDF は RM よりも良い結果であるが，適応型 EDF（R）と適応型 EDF（I）によって更に改善されている．適

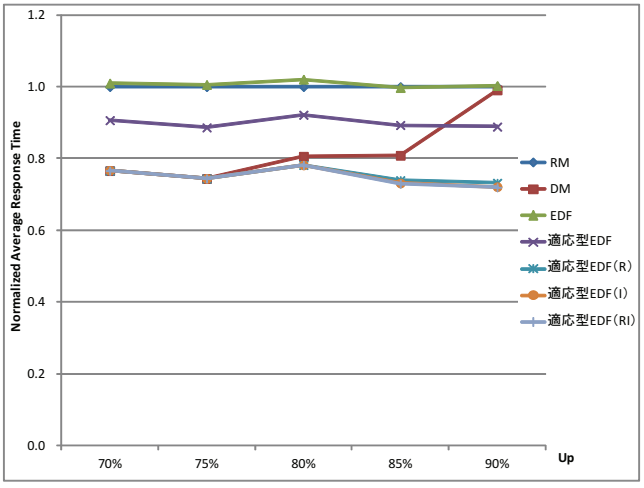


図 4 結果 – 平均的周期のタスクを対象．

[*2] $U_{ub} = 0.9$ は一般に知られている RM の上限値よりも高いが，ほとんどの場合で実際の実行時間が WCET よりも短いため，実際の使用率も小さくなり，デッドラインミスは発生しなかった．

© 2014 Information Processing Society of Japan

応型 EDF（RI）は適応型 EDF に対し，U_p が 90%のときに約 19.1%の削減率を示した．

図 5 は最長周期のタスクを重要タスクとみなした場合の結果である．EDF が RM よりも良い結果であることを除けば，図 4 と類似した傾向を示している．また，RM に対する改善率はより高くなっている．適応型 EDF（RI）は適応型 EDF に対し，U_p が 90%のときに約 29.2%の削減率を示した．

以上の結果から，提案する 2 つの改良方法は，周期と独立した重要度を設定した場合に（特に周期が長いタスクが重要であるときに）重要タスクの応答時間を削減し，評価対象方式の中で最も短い平均応答時間になることが示された．

5. おわりに

本稿では，周期と独立した重要性を持つタスクの応答時間を短縮する手法である適応型 EDF に対して，2 つの改良方法である余剰バンド幅占有法と漸進デッドライン更新法を提案した．余剰バンド幅占有法は，EDF がプロセッサ使用率 100%を達成可能であることを利用し，タスクセットが 100%未満の使用率の場合に，その余剰バンド幅を重要タスクに占有させることで短いデッドラインを設定可能とする方法である．漸進デッドライン更新法は，デッドラインを段階的に更新することにより，従来の適応型 EDF における実行時間の過大見積り，過小見積りに起因する応答時間削減の限界を除去するものである．本手法は，著者が過去に提案した非周期タスクの応答時間を短縮する方法である適応型 TBS [6], [13] に対する改良方法 [14] を，周期タスクを対象とする手法として再定義したものである．シミュレーションによる評価では，提案する 2 つの改良方法を組み合わせることにより，重要タスクの応答時間が最大で 29.2%短縮することが示された．

今回の評価では実行時間に関して確率的に生成したタスクセットを使用したシミュレーションを行ったが，実際の実行時間の変動を表現するためには，実プログラムを使用した評価が望まれる．加えて，デッドライン計算を含めたスケジューリングオーバヘッドを考慮した評価を行う予定である．

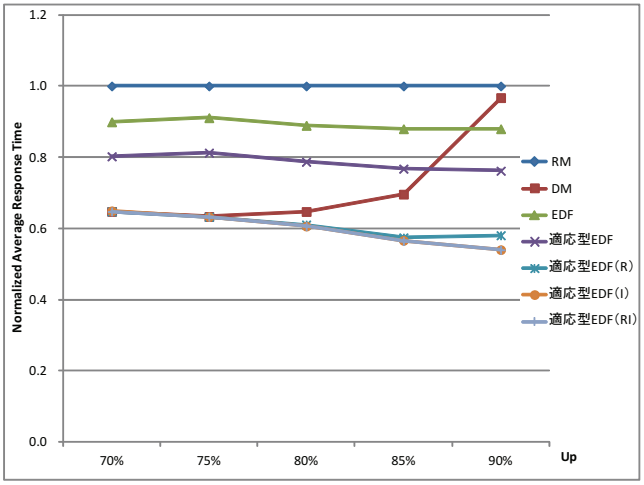


図 5 結果 – 最長周期のタスクを対象．

参考文献

[1] Liu, C.L., Layland, J.W.: *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment*, Journal of the Association for Computing Machinery, Vol.20, No.1, pp.46–61 (1973).
[2] Marti, P., Fuertes, J.M., Fohler, G., Ramamritham, K.: *Jitter Compensation for Real-Time Control Systems*, Proc. of IEEE Real-Time Systems Symposium, pp.39–48, IEEE Computer Society, London (2001).
[3] Buttazzo, G.C.: *Rate Monotonic vs. EDF: Judgment Day*, The Journal of Real-Time Systems, Vol.29, No.1, pp.5–26, (2005).
[4] de Niz, D. Lakshmanan, K., Rajkumar, R.: *On the Scheduling of Mixed-Criticality Real-Time Task Sets*, Proc. of IEEE Real-Time Systems Symposium, pp.291–300, IEEE Computer Society, Washington, DC, (2009).
[5] Baruah, S., Li, H., Stougie, L.: *Towards the Design of Certifiable Mixed-Criticality Systems*, Proc. of IEEE Real-Time and Embedded Technology and Application Symposium, pp.13–22, IEEE Computer Society, Stockholm, (2010).
[6] 田中清史: 実行時間の変動を利用するリアルタイムスケジューリング，情報処理学会研究報告，Vol.2013-EMB-28, No.25，対馬 (2013).
[7] Tanaka, K.: *Adaptive EDF: Using Predictive Execution Time*, Proc. of 5th Workshop on Adaptive and Reconfigurable Embedded Systems (APRES), pp.2–5, Philadelphia (2013).
[8] Buttazzo, G.C.: *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Third Edition, Springer, (2011).
[9] Lundqvist, T., Stenström, P.: *Timing Anomalies in Dynamically Scheduled Microprocessors*, Proc. of IEEE Real-Time Systems Symposium, pp.12–21, IEEE Computer Society, Phoenix (1999).
[10] Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., Bernat, G., Ferdinand, C., Heckmann, R., Mitra, T., Mueller, F., Puaut, I., Puschner, P., Staschulat, J., Stenström, P.: *The Worst-Case Execution Time Problem – Overview of Methods and Survey of Tools*, ACM Trans. on Embedded Computing Systems, Vol.7, No.3, pp.1–53 (2008).
[11] Spuri, M., Buttazzo, G.C.: *Efficient Aperiodic Service under Earliest Deadline First Scheduling*, Proc. of IEEE Real-Time Systems Symposium, pp.2–11, IEEE Computer Society, San Juan (1994).
[12] Leung, J., Whitehead, J.: *On the Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks*, Performance Evaluation, Vol. 2, No. 4, pp.237–250, 1982.
[13] Tanaka, K., *Adaptive Total Bandwidth Server: Using Predictive Execution Time*, Proc. of 4th IFIP TC 10 International Embedded Systems Symposium (IESS), Springer, pp.250–261, Paderborn (2013).
[14] 田中 清史：適応型スケジューリングによる平均応答時間の短縮法 — 実行時間見積り方法の影響 —，組込システムシンポジウム 2013（ESS2013）論文集，pp.87–94，東京 (2013).

© 2014 Information Processing Society of Japan