システムＬＳＩ設計技術 105－11
（２００２．３．５）

I/O ブロッキングを考慮したスケジュール可能性の解析手法に関する研究

遠藤 友悟　高田 広章
豊橋技術科学大学 情報工学系
〒 441-8580 豊橋市天伯町雲雀ヶ丘 1-1
{yugo,hiro}@ertl.ics.tut.ac.jp

あらまし　従来のハードリアルタイムスケジューリング理論では，I/O 待ちなどで自ら実行を中断するタスクを扱いにくかったが，このようなタスクをマルチフレームタスクの枠組みで扱うことで，より正確にスケジュール可能性を評価できることが示されている．この手法では，I/O ブロッキングを含むタスクのデッドラインを各フレームへ割り振り，それぞれの優先度を決定する必要がある．本稿では，各フレームに任意の優先度を持つマルチフレームタスクの critical instant 定理を示し，スケジュール可能性の解析手法を提案する．さらにフレームに異なる優先度を割り付けることでスケジュール可能性が向上するかどうかを検討し，マルチフレームタスクモデルにおける最適な優先度割り付けの近似解として，Effective Deadline Monotonic Scheduling (EDMS) の手法を提案する．

キーワード　リアルタイムシステム, I/O ブロッキング, マルチフレームタスク, スケジュール可能性解析

Studies on the Schedulability Analysis of Task Sets with I/O Blockings

Yugo Endo　Hiroaki Takada
Dept. of Information and Computer Sciences,
Toyohashi Univ. of Technology
1–1 Hibarigaoka, Tempaku-cho, Toyohashi 441-8580, Japan
{yugo,hiro}@ertl.ics.tut.ac.jp

Abstract　A task that suspends itself in order to wait for an I/O completion is difficult to handle with conventional hard real-time scheduling theory. The schedulability of the task set can be analyzed more accurately compared to conventional approaches when the tasks, including I/O blockings, are handled with multiframe task model. With this approach, allocating the deadlines of tasks to each frame and determining each of the task's priority is necessary. In this paper, we show the critical instant theorem for multiframe task set assigned arbitrary priority for each frame and propose a schedulability analysis based on the theorem. Also, we discuss whether or not the schedulability is improved in assigning different priorities for each frame and propose the Effective Deadline Monotonic Scheduling (EDMS) Approach as an approximate solution to assigning priorities on a multiframe task model.

Key words　real-time system, I/O blocking, multiframe task, schedulability analysis

1 はじめに

リアルタイムシステムにおいて，その振る舞いが時間的制約を満たせることを確認し，保証するために，スケジュール可能性解析が広くおこなわれている．これまでに発表されているスケジュール可能性の解析手法は，Liu と Layland の理論 [1] と，そこで述べられているタスクモデル（L&L タスクモデル）をベースに展開されている．

L&L タスクモデルはタスクモデルとして基本的なもので，一定の周期をもって定期的に起動するタスク（周期タスク）を扱う枠組みであり，主に次のような仮定が置かれている．

1. 各周期起動は，全て等しい最悪計算時間を持つ
2. 各周期起動は，周期に等しい相対デッドラインを持つ
3. 全てのタスクは独立である
4. タスクは自ら待ち状態になることはない

これらの性質は，L&L タスクモデルを実際のシステムに適用しにくいものとし，悲観的な解析結果を生み出す要因となっていた．これらの過程を緩め，スケジュール可能性解析の適用範囲をより広げる方向で研究がおこなわれている．とりわけ，rate monotonic analysis（RMA）と呼ばれる，静的優先度割り付けのもとでの優先度ベースのスケジューリングモデルにおけるスケジュール可能性の解析手法に関する研究が盛んにおこなわれており，実際のシステム設計への適用も進んでいる．

しかしながら，従来の RMA の枠組みでは，外部記憶装置などの I/O 待ちなどの理由で，タスクが自ら実行を中断するような状況を扱いにくい．これは，L&L タスクモデルの条件 (4) を満たしておらず，RMA における基本的な定理である critical instant 定理がそのままの形では成り立たなくなり，それをベースに構築された解析手法が使えなくなるためである．

RMA におけるタスクモデルの拡張の 1 つとして，マルチフレームタスク（multiframe task）モデルが提案され，そのスケジュール可能性に関する研究がおこなわれている [2]．これは，タスクの実行時間があるパターンにしたがって大きく変動する状況を効率的に扱うための枠組みであり，上で述べた L&L タスクモデルの仮定 (1) を緩めたものである．例えば，MPEG の解凍タスク [2] やエンジン制御の回転角に同期したタスク [3, 4] を有効にモデル化できることが指摘されている．

さらに，I/O ブロッキングを含むタスクを，ブロッキングの前後でフレームに分割してマルチフレームタスクとしてモデル化することで，I/O ブロッキングを含むタスクより低優先度のタスクについて，従来のスケジュール可能性の解析手法と比較してより正確なスケジュール可能性解析がおこなえることが示されている [5]．I/O ブロッキングを含むタスクをマルチフレームタスクモデルに当てはめる場合，I/O ブロッキングを含むタスクのデッドラインを各フレームへ割り振り，各フレームへの優先度割り付けを決定する必要があるが，両者は密接に関連するため，最適解を求めるにはこれらを同時に決定しなければならない．しかしながら，この 2 つの決定にはそれぞれ任意性があるため，最適解を求めることは困難である．そのため，(1) 各フレームの優先度割り付けが与えられているものとして，より低い優先度を持ったタスクのスケジューリングを最大にするようなデッドラインの割り振り方について検討する，(2) 逆にデッドラインの割り振りが与えられているものとして，優先度割り付けの決定方法について検討する手法が解決の糸口と考えられる．

本稿では，静的優先度割り付けのもとで，I/O ブロッキングを含むタスクをマルチフレームタスクとしてモデル化する場合に，デッドラインの割り振りが与えられているものとして，各フレームへの適切な優先度の割り付け方を考える手法をとる．ところがこれまで，マルチフレームタスクのスケジューリングに関する理論においては，あるタスクのすべてのフレームは同一の優先度を持つことが前提であった．各フレームへの適切な優先度割り付けを議論するためには，スケジュール可能性の解析手法が必要である．しかし，従来のマルチフレームタスクにおけるスケジュール可能性の解析手法のベースとなっている critical instant 定理は，各フレームに任意の優先度割り付けを許す場合にはそのまま適用できない．そのため，まず各フレームに任意の優先度割り付けを許すマルチフレームタスクにおける critical instant 定理を示し，それに基づいたスケジュール可能性の解析手法を考える必要がある．

本稿は，I/O ブロッキングを含むタスクのスケジュール可能性の解析手法を導く過程として，各フレームに異なる優先度を割り付けるマルチフレームタスクモデルのスケジュール可能性解析手法を導き，それぞれのフレームへの適切な優先度の割り付け方を議論するものと位置付けることができる．まず 2 章で，マルチフレームタスクモデルの形式化をおこない，本研究でおかれている仮定を述べる．3 章では，マルチフレームタスクの各フレームに異なる優先度を割り付ける場合には，従来のマルチフレームタスクモデルにおける critical instant 定理が成り立たないことを示し，新たな critical instant 定理を導く．4 章では，従来のスケジュール可能性の解析手法を拡張する形で，新たな critical instant 定理に基づいたスケジュール可能性の解析手法を提案する．5 章では，マルチフレームタスクの各フレームに異なる優先度を割り付けることがスケジュール可能性を向上させるかどうかを検討し，6 章で最適な優先度割り付け手法に対するよい近似解の候補として，effective deadline monotonic scheduling (EDMS) の手法を提案する．

なお，本稿は研究の途中経過の報告であり，現在進行中の部分が多く残ることをお断りしておく．

2 マルチフレームタスクモデル

マルチフレームタスクモデルは，タスクの実行時間があるパターンにしたがって大きく変動する状況を効率的に扱うための枠組みで，タスクの周期をいくつかの同じ長さのフレームに分割し，タスクの最大実行時間をそれぞれのフレームに対して与える．各フレームの最大実行時間が大きい周期で変動するようにモデル化される．タスクは各フレームの始めに起動され，そのフレームの終わりをデッドラインとする [2].

一般化されたマルチフレームタスク (generalized multiframe task) とは，マルチフレームタスクの前提を，(1) 各フレームの長さが異なる場合を許す (2) 各フレームにおけるデッドラインがそのフレームの終わりに一致しない場合を許す，という点において緩めたものである [6]. 本稿ではこれ以降，単にマルチフレームタスクといった場合，一般化されたマルチフレームタスクのことを指すものとする．

2.1 マルチフレームタスクモデルの形式化

本稿で扱うマルチフレームタスクモデルを形式化して示す．タスク番号 n で N_n 個のフレームを持つマルチフレームタスク τ_n の j 番目のフレームを 3 つ組 $(C_{n,j}, D_{n,j}, P_{n,j})$ であらわすことにすると，長さ N_n の 3 つ組の列 $((C_{n,0}, D_{n,0}, P_{n,0}), \cdots, (C_{n,N_n-1}, D_{n,N_n-1}, P_{n,N_n-1}))$ のように表すことができる．また，τ_n の j 番目のフレームの優先度を $p_{n,j}$ と書く．ここで，$C_{n,j}$ は τ_n の j 番目のフレームにおける最大実行時間，$D_{n,j}$ は j 番目のフレームの (相対的な) デッドライン，$P_{n,j}$ は j 番目と $j+1$ 番目のフレームの最小間隔 (minimum separation) を示す．フォーマルに述べると，τ_n の (通算して) k 番目のフレームの起動時刻を a^k，(絶対) デッドラインを $a^k + d^k$，実行時間を c^k とすると，次の関係が成り立つ．

$$a^0 \geq 0,\ a^{k+1} \geq a^k + P_{n,k \bmod N_n}$$
$$d^k = D_{n,k \bmod N_n}$$
$$c^k \geq C_{n,k \bmod N_n}$$

マルチフレームタスクが FS 性 (Frame Separation property) を持つとは，各フレームのデッドラインが次のフレームが始まるより前であることをいう．言い換えると，任意の j $(0 \leq j \leq N_n - 1)$ に対して $D_{n,j} \leq P_{n,j}$ が成り立つ場合に，そのマルチフレームタスクは FS 性を持つという．

本稿では，FS 性を持つマルチフレームタスクのみを扱う．また議論を簡単にするために，同じ優先度を持つフレームは 1 つだけであることを仮定する．

3 Critical Instant 定理

L&L タスクモデルでは，critical instant 定理があることで各タスクのデッドラインまでのスケジュールをチェックするだけで，スケジュール可能性を必要十分にチェックできる．FS 性を持つマルチフレームタスクモデルに対しても，L&L タスクモデルにおける critical instant 定理と同様に critical instant 定理に対応するものが示されている [3, 4]．しかしこれは，あるタスクのすべてのフレームが同一優先度であることを前提としているため，これをそのまま，異なる優先度割り付けを許すマルチフレームタスクに適用することができない．

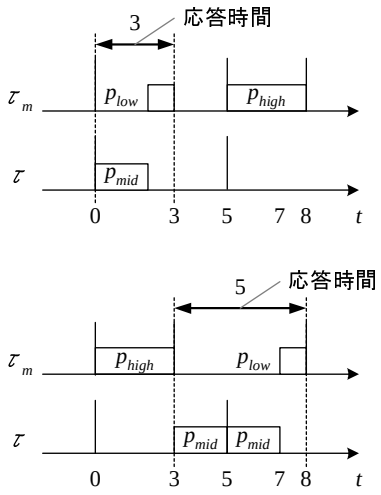

図 1: 従来の critical instant 定理が成り立たない例

例 1 (従来の critical instant 定理が成り立たない例) 2 つのフレームからなるマルチフレームタスク $\tau_m = ((3,3,3),(1,5,5)$ と，最大実行時間が 2 で，周期が 5 のタスク τ があり，τ_m の 1 つめのフレームが高い優先度，τ が中間の優先度，τ_m の 2 つめのフレームが低い優先度を持つものとする．このとき，τ_m の 2 つめのフレームが最悪応答時間を持つ状況を考える．

GMF タスクモデルにおける critical instant 定理によれば，あるフレームが最悪応答時間を持つのは，そのタスクより優先度の高いタスクがすべて同時にいずれかのフレームを起動した場合であるから，この例に適用すると τ_m の 2 つめのフレームが最悪応答時間を持つのは，τ_m の 2 つめのフレームが τ と同時に起動した場合である．このときの応答時間は 3 である（図 1上）．

ところが，τ_m の 1 つめのフレームが τ と同時に起動した場合を考えると，この場合の応答時間は 5 である．この例の τ_m の 2 つめのフレームは，これまで言われてきたマルチフレームタスクモデルにおける critical instant には最悪応答時間を持たない（図 1下）． □

この例から，従来の critical instant 定理が，各フレームに任意の優先度を許す場合には適用できないことがわかる．そこで FS 性を持つマルチフレームタスクにおける critical instant 定理を，各フレームに異なる優先度割り付けを許す場合にも適用できる形に書き換える．

定義 2 (critical instant candidates) あるマルチフレームタスクのあるフレームに対する *critical instant candidates* とは，そのフレームあるいはそれの前により高い優先度を持って続くフレームのいずれかが起動されると同時に，そのフレームより優先度の高いフレームを含むすべてのタスクの，そのフレームより優先度の高いいずれかのフレームが同時に起動した状況で，以降続くフレームを最小間隔で起動し，いずれのフレームにおいても最大実行時間を使い切る場合をいう．

この定義を用いて，マルチフレームタスクに対する critical instant 定理は，次のように書ける．

定理 3 (critical instant 定理) あるマルチフレームタスクのあるフレームに対する *critical instant* は，そのタスクに対する *critical instant candidates* のいずれかである．

証明. タスク τ の優先度 i をもつフレーム F^i が時刻 t に到着したとき，F^i が最悪応答時間を持ち，時刻 t_{end} に終了すると仮定する．t と t_{end} のあいだは常に F^i 以上の優先度を持つフレームが実行されているはずである．t 以前で，より高い優先度を持つフレームが実行されていないようなもっとも遅い時間を t_0 とする．時刻 0 ではどのフレームも実行されていないので，t_0 は必ず存在する．ここで F^i は t_0 後の最初の τ の到着であると仮定する．F^i の到着時刻を t_0 に変更したとしても，t_0 と t の間は少なくとも *1* つの i より高い優先度を持つフレームが実行されているため，依然として t_{end} に終了する．

それぞれのより高い優先度を持つフレームの t_0 後の最初の到着時刻が t_0 に移動したとする．この場合，F^i の終了時刻は $t_{end'}$ に遅延するかもしれない．また，それらの後に続くフレームの到着時刻が早まるように，それぞれのフレームがその最小間隔で要求され，それぞれのフレームがその最大実行時間を消費すると仮定する．これは F_i の終了時刻を $t_{end''}$ にさらに遅延させる．今，t_0 は τ の *critical instant candidates* の *1* つであり，F^i は最初の状況と同じか，より長い応答時間を持つ．

F^i が最初の状況で最悪応答時間を持つという仮定から，F^i がその *critical instant candidates* に要求されたときも最悪応答時間を持つ．

次に， F^i が t_0 後の最初の τ の到着であるという仮定をはずす．これを帰納法によりおこなう．

1. F^i が t_0 後の 1 番目の τ の到着であるとき，F^i がその *critical instant candidates* に要求されたときに最悪応答時間を持つ．
2. タスク τ のフレーム F^i が時刻 t に到着したとき，F^i が最悪応答時間を持ち，時刻 t_{end} に終了すると仮定する．t と t_{end} のあいだは常に F^i 以上の優先度を持つフレームが実行されているものとする．t 以前で，より高い優先度を持つフレームが実行されていないようなもっとも遅い時間を t_0 とする．時刻 0 ではどのフレームも実行されていないので，t_0 は必ず存在する．ここで F^i は，後ろのフレームより高い優先度を持つ t_0 後の最初の τ の到着 F より後の，$k+1$ 番目の τ の到着であると仮定する．F の到着時刻を t_0 に変更したとしても，t_0 と t の間は少なくとも *1* つの i より高い優先度を持つフレームが実行されているため，依然として t_{end} に終了する．
   それぞれのより高い優先度を持つフレームの t_0 後の最初の到着時刻が t_0 に移動したとする．この場合，F^i の終了時刻は $t_{end'}$ に遅延するかもしれない．また，それらの後に続くフレームの到着時刻が早まるように，それぞれのフレームがその最小間隔で要求され，それぞれのフレームがその最大実行時間を消費すると仮定する．これは F^i の終了時刻を $t_{end''}$ にさらに遅延させる．今，t_0 は τ の *critical instant candidates* の *1* つであり，F^i は最初の状況と同じか，より長い応答時間を持つ．
   F^i が最初の状況で最悪応答時間を持つという仮定から，F がその *critical instant candidates* に要求されたときも最悪応答時間を持つ．

(1), (2) より，証明できる． □

マルチフレームタスクの各フレームに異なる優先度を割り付けることで，例 1 に示したように，対象としているフレームが同一タスクの手前のフレームの実行の影響を間接的に受けることが起こり得る．そのため対象フレームだけでなく，対象フレーム以前の高優先度フレームがすべての（他タスクの）高優先度フレームと同時に起動した場合についても考慮しなければならず，同一タスク内で対象フレームよりも低い優先度を持つフレームがあらわれる時刻まで遡って調べなければならない，という点が従来の critical instant 定理と大きく異なる．対象としているフレームが，そのタスク内で最も優先度が低い場合は，対象フレームの前回の起動があらわれる時刻まで遡ればよい．すべてのフレームは，その前回の起動に間接的に邪魔をされることがなく（優先度が同じであるため），FS 性の仮定から同一タスク内のそれぞれのフレームが直接的に邪魔しあうことはないためである．

この critical instant 定理を用いて，従来のマルチフレームタスクセットの場合と同様にして，異なる優先度割り付けを許すマルチフレームタスクセットのスケジュール可能性の必要十分条件式を記述することができる．まず，必要十分条件式を記述するのに必要となる定義を行う．なお，優先度をあらわす数字が小さいほど高い優先度を持つこととする．

$$C_{n,k}^{(p)} = \begin{cases} 0 & (p_{n,k} \geq p) \\ C_{n,k} & (p_{n,k} < p) \end{cases}$$

これを用いて，タスク τ_n の k 番目のフレームが起動されてから，時間 $t(>0)$ が経過するまでの間に，τ_n が起動する p より高い優先度を持つ処理の実行時間の合計の最大値を $E_{n,k}^{(p)}(t)$ と書く．式で書くと，

$$E_{n,k}^{(p)}(t) = \sum_{h=k}^{k+J-1} C_{n,h \bmod N_n}^{(p)}$$

ここで J は次の式を満たす最小の整数である．

$$\sum_{h=k}^{k+J-1} P_{n,h \bmod N_n} \geq t$$

J は，k 番目のフレームが起動されてから，時間 t が経過するまでの間に起動されるフレームの最大数をあらわしている．また，$E_{n,k}^{(p)}(0) = 0$ と定義しておく．

これを用いて，GMF タスクセットに含まれるタスク τ_n がスケジュール可能な必要十分条件を記述する．$h_1, \cdots, h_{n-1}, h_{n+1}, \cdots, h_N$ をそれぞれ，タスク $\tau_1, \cdots, \tau_{n-1}, \tau_{n+1}, \cdots, \tau_N$ に含まれる $p_{n,k}$ よりも高い優先度を持つフレームの番号であるとし，H は，タスク τ_n の k 番目のフレームよりも前に $p_{n,k}$ よりも高い優先度を持って続くフレームの総数であるとしたとき，次のように記述することができる．

$$\begin{aligned}
&\forall k,\ 0 \leq k \leq N_n - 1; \\
&\quad \forall h_n,\ k - H \leq h_n \leq k; \\
&\quad\quad \forall h_1;\ \cdots;\ \forall h_{n-1};\ \forall h_{n+1};\ \cdots;\ \forall h_N; \\
&\quad\quad\quad \min_{0<t\leq D_{n,k}} \frac{\sum_{m=1}^{N} E_{m,h_m}^{(p_{n,k})}(t) + C_{n,k}^{(p_{n,k})} - \sum_{l=k-h_n}^{k-1} P_{n,l \bmod N_n}}{t} \\
&\quad\quad\quad\quad \leq 1 \quad (1)
\end{aligned}$$

この条件は，τ_n の各フレームに対して，すべての critical instant candidates におけるスケジュール可能性をチェックする形になっている．

なおこれらの式は，t を連続的に変化させて最小値を取る形になっているが，実際には [7] で述べられている maximum interfference function（後述）の飽和演算のアイデアを用いることにより，離散的な点のみのチェックで必要十分条件を調べることができる．

なお，本節で示した critical instant 定理において，1 つのマルチフレームタスクのすべてのフレームが同一優先度を持つものとすれば，従来のマルチフレームタスクにおける critical instant 定理と本質的に同じであり，これの一般化になっていると言える．

4 スケジュール可能性解析

前節で示した critical instant 定理のもとでは，従来の critical instant 定理を前提に構築されたスケジュール可能性の解析手法をそのまま利用することはできない．そのため，従来の手法を新たな critical instant 定理にあてはめることを考える．

基本的には，タスクの応答時間を正確に，かつ離散的な点のみをチェックすることで効率よく求めることができる response time analysis[8] を適用すればよい．ここで，(従来の）必要十分条件を各フレーム数に対する多項式オーダの時間で効率よくチェックする方法として提案された MIF のアイデア [3, 4] を用いる．MIF は，あるタスクが，時間 t の間にそれより優先度の低いタスクの実行を邪魔する最大時間の関数として定義される．MIF を，あるタスクのそれぞれのフレームが任意の優先度を持つ場合に当てはめると，タスク τ_n の MIF $M_n^{(p)}(t)$，すなわち τ_n が長さ t の期間に p 以下の優先度を持つタスクの実行を邪魔する最大時間は，次の式で求めることができる．

$$M_n^{(p)}(t) = \max_{0 \leq k \leq N_n - 1} E_{n,k}^{\prime(p)}(t) \quad (2)$$

ここで $E_{n,k}^{\prime(p)}$ は，タスク τ_n が k 番目のフレームから実行開始してから長さ t の期間に優先度 p 以下のフレームの実行を邪魔する最大時間 (これを，interference function と呼ぶ) で，次の式であらわすことができる．

$$\begin{aligned}
E_{n,k}^{\prime(p)}(t) = &\sum_{h=k}^{k+J-1} C_{n,h \bmod N_n}^{(p)} \\
&+ \min\left(C_{n,(k+J) \bmod N_n}^{(p)}, t - \sum_{h=k}^{k+J-1} P_{n,h \bmod N_n} \right)
\end{aligned}$$

ここで J は次の式を満たす最大の整数である．

$$\sum_{h=k}^{k+J-1} P_{n,h \bmod N_n} \leq t$$

これを用いると，タスクセットがスケジュール可能になる必要十分条件は，次のように書くことができる．

$$\begin{aligned}
&\forall k,\ 0 \leq k \leq N_i - 1; \\
&\quad \max_{\forall h_n,\ k-H \leq h_n \leq k} \left(\min_{0<t\leq D_{n,k}} \right. \\
&\quad\quad \left. \frac{\sum_{m=1}^{N} M_m^{(p_{n,k})}(t) + C_{n,k}^{(p_{n,k})} - \sum_{l=k-h_n}^{k-1} P_{n,l \bmod N_n}}{t} \right) \leq 1 \quad (3)
\end{aligned}$$

例として，マルチフレームタスク $((1^{(1)}, 2^{(2)}, 3^{(3)}), 4)$ の $E_0^{\prime(2)}(t)$, $E_1^{\prime(2)}(t)$ を図 2に示す．このタスクの MIF は，式 (2) からもわかるように，$E_0^{\prime(2)}(t)$, $E_1^{\prime(2)}(t)$ の 2 つのグラフの，最も上の線をつないだものとなる．

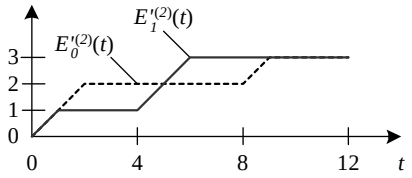

図 2: interference function

ここで定義した MIF に response time analysis を適用して，タスク τ_n の優先度 p を持つ第 k 番目のフレームのスケジュール可能性をチェックするためには，次の式において初期値 $R_h = 0$ とし，

$$R_h = \sum_{m=1}^{N} M_m^{(p)}(R_h) + C_{n,k}^{(p)} \tag{4}$$

の右辺を計算して出てくる R_h の値が計算に使った R_h に一致するまで繰り返し適用する．ここで，

$$R = \max_{k-H \le h \le k} \left(R_h - \sum_{l=k-h}^{k-1} P_{n,l \bmod N_n} \right) \tag{5}$$

が着目しているフレームの最悪応答時間をあらわす．R がそのフレームのデッドラインを越えて大きくなる場合には，そのフレームはスケジュール可能ではない．

この方法を，最高優先度を持つフレームから最低優先度を持つフレームまで，すべてのフレームについて順に繰り返し適用する．すべてのフレームについて最悪応答時間がそのデッドラインに間に合っていれば，そのタスクセットはスケジュール可能であると言える．

例 1 のタスクセットを考える．すなわち，$\tau_1 = ((3,3,3),(1,5,5))$，$\tau_2 = (2,5,5)$，$p_{1,0} = 1$, $p_2 = 2$, $p_{1,1} = 3$ における $\tau_{1,1}$ のスケジュール可能性をチェックする．$\tau_{1,1}$ の前に続く高優先度フレームは $\tau_{1,0}$ ひとつなので，$0 \le h \le 1$ である．$h = 0$ のとき $R_0 = 0$ とし，式 (4) に入れて計算すると，$R_0 = M_1^{(3)}(0) + M_2^{(3)}(0) + 1 = 1$ が出てくる．以下，出てくる値を繰り返し 式 (4) に入れると，$R_0 = M_1^{(3)}(1) + M_2^{(3)}(1) + 1 = 2$，以下，3, 3 となり $R_0 = 3$ を得る．次に $h = 1$ として同様に，$R_1 = M_1^{(3)}(0) + M_2^{(3)}(0) + 1 = 1$ から始めて，$R_0 = M_1^{(3)}(1) + M_2^{(3)}(1) + 1 = 3$，以下，6, 7, 8, 8 となり $R_1 = 8$ を得る．式 (5) を計算すると，3 と $8-3$ の max をとるので，$\tau_{1,1}$ の最大応答時間 $R = 5$ を得る．ここで $R(=5) \le D_{1,1}(=5)$ であるので，$\tau_{1,1}$ はスケジュール可能である．

5 異なる優先度割り付けの有効性

この節では，マルチフレームタスクの各フレームに異なる優先度割り付けをおこなうことで，タスクセットのスケジュール可能性が向上するかどうかを検討する．

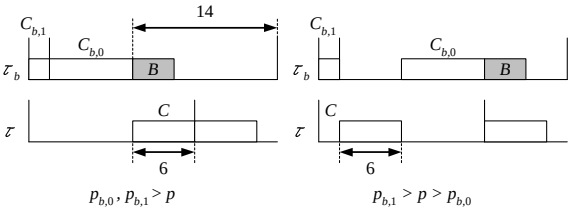

図 3: 例 4 の (1) の場合

例 4 (例題) I/O ブロッキングを 1 回含むタスク τ_b とタスク τ の 2 つのタスクからなるタスクセットを考える．τ_b は，最初に最大実行時間 $C_{b,0} = 8$ の処理をおこなってから，最大待ち時間 $B_b = 4$ の I/O 待ちをおこない，最後に最大実行時間 $C_{b,1} = 2$ の処理をおこなうタスクであるとし，全体の周期（= デッドライン）を 24 とする．τ の周期が 16 であり，それぞれのフレームあるいはタスクに任意に優先度割り付けをおこなえるとき，このタスクセットがスケジュール可能な τ の実行時間 C の最大値はいくらか． □

この例題に対し，デッドライン割り振りが次の 3 つのように与えられた場合を考えてみる．

1. I/O ブロッキング後のフレームの余裕時間が 0．すなわち τ_b の後ろのフレームにとって一番厳しい状況
2. I/O ブロッキング前後のフレームの余裕時間が均等．すなわち（I/O ブロッキングが 1 回なので）I/O ブロッキングを含むタスクよりも優先度が低いタスクにとってスケジュール可能性が最適である [5]
3. I/O ブロッキング前のフレームの余裕時間が 0．すなわち τ_b の前のフレームにとって一番厳しい状況

このそれぞれの場合に対し，3 つの処理 $\tau_{b,0}$，$\tau_{b,1}$，τ のそれぞれに異なる優先度 $p_{b,0}, p_{b,1}, p$ を割り付け，すべての優先関係の組み合わせについて検討する[1]．それぞれの場合で，最低優先度を持つタスクにとっての critical instant から実行を開始する状況を考える．

(1) の場合は $\tau_{b,1}$ にとって最も厳しい状況であるため，$p_{b,1}$ が高優先度のときにタスクセットのスケジュール可能性が大きい（図 3）．ここで $p_{b,0}, p_{b,1} > p$ の場合と $p_{b,0} > p > p_{b,1}$ の場合に差は見られない．両方の場合のタスクセット全体の MIF が一致するため，この 2 つのタスクよりも低優先度のタスクに対して与える影響も同一であり，少なくともこの例においては，異なる優先度を割り付けた場合のスケジュール可能性が同一優先度を割り付けた場合を超えて大きくならない．

(2) の場合は $p_{b,0}, p_{b,1} > p$ と $p_{b,0} > p > p_{b,1}$ の場合に τ のスケジュール可能性が大きく，さらにこの例で考えた範囲で最大である（図 4）．この状況でも (1) の場合

[1] FS 性の前提から，$p_{b,0} > p_{b,1} > p$ と $p_{b,1} > p_{b,0} > p$，$p > p_{b,0} > p_{b,1}$ と $p > p_{b,1} > p_{b,0}$ のそれぞれを区別する必要はない

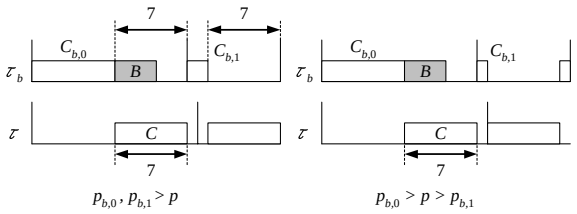

図 4: 例 4 の (2) の場合

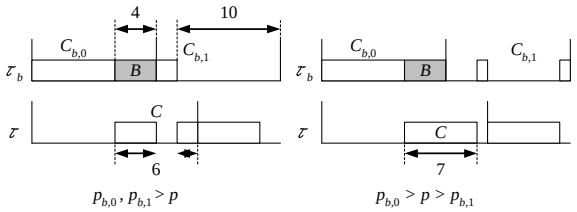

図 5: 例 4 の (3) の場合

と同様に，異なる優先度を割り付けた場合のスケジュール可能性の方が大きくはならない．

(3) の場合は $\tau_{b,0}$ にとって最も厳しい状況であるため，$p_{b,0}$ が高優先度のときにタスクセットのスケジュール可能性が大きい（図 5）．この状況では，従来のマルチフレームタスクモデルで扱うことができる $p_{b,0}, p_{b,1} > p$ の場合より，異なる優先度割り付ける $p_{b,0} > p > p_{b,1}$ の場合の方がタスクセットのスケジュール可能性が大きい．

デッドラインは与えられる一般のマルチフレームタスクにおいては異なる優先度を割り付けることでスケジュール可能性を向上させることができる．しかし，I/O ブロッキングを含むタスクをマルチフレームタスクに当てはめる場合は各フレームへのデッドラインの割り振り方には任意性があり，よりスケジュール可能性を向上させるのは (1) または（I/O ブロッキングの長さによっては）(2) の状況である [5]．つまり，少なくともこの例からは，I/O ブロッキングを含むタスクをマルチフレームタスクモデルとして扱う場合に，それぞれのフレームへ異なる優先度を割り付けることがスケジュール可能性を向上させるために有効であるとは言えない．

6 優先度割り付け方法の検討

本稿の最初で述べたように，デッドラインの割り振りと優先度の割り付けは相互に関連するため，最適解を求めるにはこれらを同時に決定しなければならないが，この 2 つの決定にはそれぞれ任意性があり，最適解を求めるためにはそれぞれを連続的に変化させてすべての組み合わせを調べなければならず，非常に困難である．また，これまで L&L タスクモデルをはじめとする周期タスクモデルのもとでの優先度割り付け方法は知られているが，マルチフレームタスクモデルを考慮した優先度割り付け方法は存在しなかった．そこで，最適解になるべく近づくような効率の良い近似解を求めることを目標とし，その過程としてマルチフレームタスクにとってより効率の良い優先度割り付け方法を議論する．

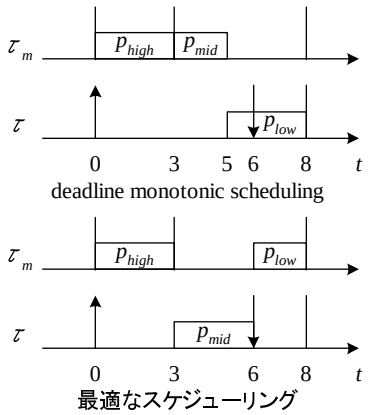

図 6: DM が最適でない例

周期タスクモデルにおいては，相対デッドラインが短いタスクほど高い優先度を割り付ける deadline monotonic scheduling[9] (以下，DM) と呼ばれる方法が，最適な静的優先度割り付けの方法として知られている．マルチフレームタスクにおいて各フレームに優先度割り付ける場合，単純には，この方法を各タスクの各フレームごとに適用する方法が考えられる．しかしながら DM による優先度の割り付けは，マルチフレームタスクの場合には必ずしも最適ではない．

例 5 (DM が最適でない例) 2 つのフレームからなるマルチフレームタスク $\tau_m = ((3,3,3),(2,5,5)$ と，最大実行時間が 3 で，(相対) デッドラインが 6，周期が 8 の周期タスク τ を考える．

この時，DM に従うと，τ_m (の両方のフレーム) に高い優先度，τ に低い優先度を割り付けることになる．ところが，この優先度割り付けでは，τ がスケジュール可能にならない．スケジュール可能にならないスケジュール例を図 6上 に示す．

それに対して，$\tau_{m,0}$ に高い優先度，τ に中間の優先度，$\tau_{m,1}$ に低い優先度を割り付けた場合，このタスクセットはスケジュール可能となる．スケジュール可能となるスケジュール例を図 6下 に示す． □

$\tau_{m,1}$ は相対デッドラインが τ よりも小さいため，DM に従えば $\tau_{m,1}$ の方が緊急性が高いと考えられ，こちらが先にスケジュールされる．しかしながら $\tau_{m,1}$ は $\tau_{m,0}$ と同時に起動することはない．すなわち，中間の優先度を持つタスクを通して間接的に邪魔される可能性はあるが，直接的に邪魔されることはないため，$\tau_{m,1}$ の方がスケジュールしやすく，実際には τ の方が緊急性が高い．そのため，マルチフレームタスクセットの場合は，他のタスクにどの程度実行を邪魔されるかを考慮して各フレームに優先度を割り付けることで，スケジュール可能性が向上すると考えられる．

6.1 Effective Deadline Monotonic Scheduling (EDMS)

ここで提案する EDMS では，他のタスクに直接的に実行を邪魔される影響を考慮した仮想的なデッドライン (effective deadline) を基準に優先度を決定する．effective deadline は具体的には，対象となるフレームの相対デッドラインから，すべてのタスクの高優先度フレームについての MIF の総和を減じたものである．EDMS のおおまかなアルゴリズムは次のようになる．

既に優先度を割り付けたフレームの集合 Γ_p，優先度が割り付けられていないフレームの集合 Γ_0 を考え，高い優先度から順番に割り付けるものとする．

1. Γ_p に含まれるフレームのみを考慮し，すべてのタスクについての MIF の総和を求め，effective deadline を決定する
2. Γ_0 に含まれるフレームのうち，effective deadline が最小のものに優先度 p を割り付け，そのフレームを Γ_0 から除き Γ_p の要素とする
3. p を 1 つ低い優先度に更新する．以上を繰り返す

なおこの手法は，前述したように自タスクのより高優先度のフレームに実行を直接的に邪魔される可能性を考慮しているものの，他タスクのフレームの実行を通して間接的に邪魔される可能性を考慮していないため，マルチフレームタスクにおける最適な優先度割り付け手法とはなっておらず，近似解のひとつであると考えている．また，effective deadline が等しいフレームが同時に存在した場合の振る舞いについては検討中である．

7 まとめと今後の課題

本稿では，静的優先度スケジューリングのもとで，I/O ブロッキングを含むタスクをマルチフレームタスクとして扱う方法を取り上げ，タスクセットのスケジュール可能性を向上させる優先度割り付けを検討した．タスクセットに FS 性の仮定をおいて，各フレームへ異なる優先度を割り付けた場合の critical instant 定理を導き，それをベースとするスケジュール可能性の解析手法を提案した．各フレームへのデッドライン割り振りが与えられている場合は，それぞれに異なる優先度を割り付けることが有効であることが確認できた．ただし，デッドライン割り振りが任意の場合についてはさらに検討する必要がある．従来の優先度割り付け方法では，マルチフレームタスクを効率よくスケジュールできないため，それに代わる Effective Deadline Monotonic Scheduling を提案した．EDMS と DM との比較評価を現在進めている．

今後の課題としては，EDMS の完成，スケジュール可能なデッドライン割り振りと優先度割り付けの同時決定が NP- 完全になるとすれば，その証明が挙げられる．また，文献 [7] で提案された MIF の飽和演算を用いて，マルチフレームタスクをより簡潔に扱うことができると考えている．さらには，FS 性の制約をはずすなど，マルチフレームタスクの前提を緩める方向に発展させることが考えられる．

参考文献

[1] C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” *JACM*, vol. 20, no. 1, pp. 46–61, 1973.

[2] A. K. Mok and D. Chen, “A multiframe model for real-time tasks,” in *Proc. Real-Time Systems Symposium*, pp. 22–29, Dec. 1996.

[3] 高田広章, 坂村健, “マルチフレームタスクセットの静的優先度スケジューリングによるスケジュール可能性,” 信学技報 *(1997* 年実時間処理に関するワークショップ *RTP'97)*, vol. 96, no. 596, pp. 29–35, 電子情報通信学会, Mar. 1997.

[4] H. Takada and K. Sakamura, “Schedulability of generalized multiframe task sets under static priority assignment,” in *Proc. Real-Time Computing Systems and Applications*, pp. 80–86, Oct. 1997.

[5] 高田広章, “I/O ブロッキングを考慮に入れたスケジュール可能性の解析手法について,” 信学技報 *(1998* 年実時間処理に関するワークショップ *RTP'98)*, vol. 97, no. 569, pp. 57–63, 電子情報通信学会, Feb. 1998.

[6] S. Baruah, D. Chen, S. Gorinsky, and A. Mok, “Generalized Multiframe Tasks,” (unpublished manuscript,
available from “http://www.emba.uvm.edu/~sanjoy/Papers/papers.html”), 1996.

[7] 飯山真一, 遠藤友悟, 高田広章, 菅沼英明, “RMA 手法のエンジン制御システムへの適用に関する研究,” 信学技報 *(2001* 年実時間処理に関するワークショップ *RTP2001)*, vol. 100, no. 655, pp. 7–14, 電子情報通信学会, Mar. 2001.

[8] Neil C. Audsley and Alan Burns and Robert I. Davis and Ken W. Tindell and Andy J. Wellings, “Fixed Priority Pre-emptive Scheduling: An Historical Perspective,” *Real-Time Systems*, vol. 8, pp. 173–198, 1995.

[9] Joseph Y.-T. Leung and Jennifer Whitehead, “On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks,” *Performance Evaluation*, vol. 2, no. 4, pp. 237–250, 1982.