システムＬＳＩ設計技術 105－10
（２００２．３．５）

# ストリーム処理環境における実時間および非実時間タスクの適応的スケジューリングポリシ

滝沢泰久[†] 芝 公仁[††] 大久保英嗣[†††]

マルチメディア処理環境は，動画や音声に代表される連続メディアを扱うストリーム処理タスクと非連続メディアを扱うイベント駆動タスクが混在する処理環境である．ストリーム処理タスクは，扱うメディアの特性上，その処理において時間制約を持つソフト実時間タスクである．一方，イベント駆動タスクは，時間制約を持たないが，より早い応答時間を要求する非実時間タスクである．このようなソフト実時間タスクと非実時間タスクを同時に処理するため，我々が提案しているストリーム処理における適応的スケジューリングポリシ Adaptive Deadline Modification をイベント駆動タスクへ適用する．本稿では，Adaptive Deadline Modification に基づき，任意の処理環境においてストリーム処理タスクの時間制約を満たし，かつイベント駆動タスクに高い応答性を提供するスケジューリングポリシを提案する．

# An Adaptive Scheduling Policy for Real-Time and Non Real-Time Tasks in Stream Processing Enviroment

Yasuhisa Takizawa,[†] Masahito Shiba[††] and Eiji Okubo[†††]

Multimedia processing enviroment is a hybrid enviroment that manipulates stream processing tasks and event-driven tasks. Each stream tasks is a soft real-time task,which manipulates continuous media such as video and audio, has timing constraints according to characteristcs of media. On the other hand, event-driven task is a non real-time task which manipulates discrete media with no timing constraint, and requires better response time. We have been an adaptive scheduling policy called ADM(Adaptive Deadline Modification) which dynamically adaptable to various processing enviroments. In this report, we apply ADM to scheduling of stream processing tasks and event-driven tasks in the hybrid enviroment. Namely we propose scheduling policy which meets timing constraints for stream processing tasks and provide better response time for event-driven tasks.

## 1. はじめに

マイクロプロセッサの急速な進歩により，パーソナルコンピュータ（以降 PC）やワークステーション（以降 WS）上で動画や音声に代表される連続メディアを処理するアプリケーション（以降連続メディアアプリケーション）が数多く出現している．連続メディアアプリケーション[5]は，それらの処理するメディアの特性上，ソフト実時間タスクとしての時間制約を持つ．PC や WS 上のマルチメディアシステムにおける連続メディア処理は，前述のようなタスクが複数実行され，かつ直列にデータ通信を行うストリーム処理により行われる場合が多い．一方，マルチメディアシステムにおいては従来の非連続メディアを扱うイベント駆動タスクも存在する．イベント駆動タスクはストリーム処理タスクとは異なり，時間制約を有しない非実時間タスクであるが，高い応答性を要求する．

実時間システムにおいては，このようなソフト実時間タスクと非実時間タスクを同時にスケジューリングする場合，リアルタイムスケジューラ[1]とスポラディックサーバ[2)3)]を用いる方式が多く用いられている．この方式は，実時間タスク群の CPU 利用率が既知であることを前提として，スポラディックサーバの容量に上限を設定することにより，実時間タスクのスケジューリング可能性と非実時間タスクの応答性を保証する．

しかし，マルチメディア処理環境では任意のアプリケーションが生成／実行されるため，事前に上記のような実時間タスクに関する情報を得ることは困難であ

† （株）ATR 適応コミュニケーション研究所
ATR Adaptive Communications Research Laboratories
†† 立命館大学大学院理工学研究科
Graduate School of Science and Engineering, Ritsumeikan Univ.
††† 立命館大学理工学部
Facutly of Science and Engineering, Ritsumeikan Univ.


**図 1** 連続メディアデータのストリーム処理
Fig. 1 Stream processing for continuous media data.

る．また，任意のデータを処理するため実時間タスクの実行時間に上限を設定することが難しい．従って，実時間タスクの CPU 利用率が未知で変動する環境において，スポラディックサーバの容量を固定的に設定すると次のような問題点が発生する．

- CPU 利用率が低い場合，スポラディックサーバの上限容量により非実時間タスクの処理が抑制され，非実時間タスクの応答性が向上しない．
- 一時的な過負荷において，スポラディックサーバの上限容量により実時間タスクのスケジューリング可能性が大きく低下する．

上記の問題点を解決するため，ストリーム処理の適応的スケジューリングポリシ Adaptive Deadline Modification[9)11)]（以降 ADM）をストリーム処理タスクだけではなく，同一処理環境のスポラディックサーバに拡張適用する．

本稿では，以下の 6 つを前提として，スポラディックサーバを用いたストリーム処理環境におけるソフト実時間タスクおよび非実時間タスクの適応的スケジューリングポリシを提案する．

- 連続メディア処理はストリーム処理により行われる．
- 連続メディアデータはその入出力装置から VBR／固定周期で生成／消費される．
- ストリーム処理タスクの実行時間は，ランダムに変動する．
- 非連続メディアデータの到着はポアソン到着とする．
- 非連続メディアを扱う非実時間タスクの実行時間は指数分布とする．
- 複数のストリーム処理と複数の非実時間処理が混在する．

以下，2 章で ADM の適応ポリシと適応メカニズムを説明する．次に，3 章で，2 章の ADM を拡張し，ストリーム処理環境における実時間および非実時間タスクの新たな適応ポリシを提案する．

## 2. ADM の適応ポリシと適応メカニズム

本章では，提案ポリシの基となる ADM の適応ポリシとそのメカニズムの概要を述べる．

### 2.1 適応ポリシ

ADM の適応ポリシは，連続メディア資源モデル Linear Bounded Arrival Process[7)]（以降 LBAP）に Variable Bit Rate（以降 VBR）データ処理のための変更を加えたモデルに基づいている．本節では，LBAP とそれに基づいた従来のストリーム処理モデルを説明する．さらに，従来モデルに変更を加えた VBR ストリーム処理モデルを説明する．その上で，ADM の適応ポリシを概説する．

#### 2.1.1 LBAP に基づく従来の処理モデル

従来の方式[8)]では，ストリームデータを一定のメッセージの到着レートと最悪実行時間により特徴付け，CPU 利用率を予約している．すなわち，各タスクの時間制約を満たすために，すべてのメッセージ処理時間が最悪実行時間である場合でも，十分に処理できる CPU 利用率を予約している．従って，従来の方式は，最悪実行時間に基づいた Constant Bit Rate（以降 CBR）ストリーム処理として考えられる．このような CBR ストリーム処理を，LBAP のパラメータを用いて表すと，次のようになる．

$R^c$: constant message rate (messages/second)
$W^{max}$: maximum workahead (messages)
$C^{max}$: maximum processing time for a message (seconds/message)

LBAP では，パラメータ $W^{max}$ により示されるバースト的なデータ到着により，データ到着レートは短期間 $R^{max}$ を超えることを許容している．このバースト的なデータ到着に伴う未処理メッセージ数を次のように定義する．

$$\begin{aligned} &w_n(m_0) = 0 \\ &w_n(m_i) = max(0, w_n(m_{i-1}) \\ &\qquad - (a_n(m_i) - a_n(m_{i-1}))R^c + 1) \end{aligned} \tag{1}$$

ただし，$w_n(m_i)$ はタスク $n$ における $i$ 番目のメッセージ到着時の未処理メッセージ数，$a_n(m_i)$ はタスク $n$ における $i$ 番目のメッセージ到着時刻である．従って，タスク $n$ における $W^{max}$，すなわち $W_n^{max}$ は次のように定義できる．

$$W_n^{max} = \max_i(w_n(m_i)) \tag{2}$$

到着したメッセージは，未処理メッセージをすべて処理完了した後に，初めて処理対象となる．LBAP では，メッセージが処理対象となる論理的時刻をタスク $n$ における $i$ 番目のメッセージのメッセージ論理到着時刻 $l_n(m_i)$ として，次のように定義する．

$$l_n(m_i) = a_n(m_i) + w_n(m_i)/R^c \tag{3}$$

また，LBAP では，タスク $n$ における $i$ 番目のメッ

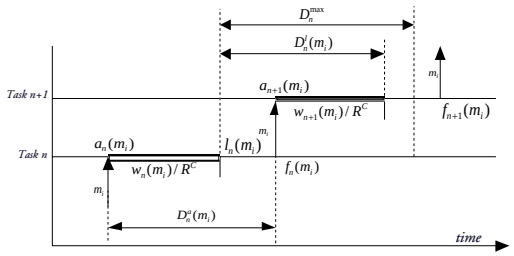


**図 2** LBAP における時間属性
Fig. 2 Timing attributes on LBAP.

セージにおける処理遅延時間 $D^a(m_i)$，論理処理遅延時間 $D^l(m_i)$ およびデッドライン時刻 $d_n(m_i)$ を，次のように定義する（**図 2** 参照）.

$$D_n^a(m_i) = a_{n+1}(m_i) - a_n(m_i) \tag{4}$$

$$D_n^l(m_i) = l_{n+1}(m_i) - l_n(m_i) \tag{5}$$

$$d_n(m_i) = l_n(m_i) + D_n^{max} \tag{6}$$

ただし，$D_n^{max}$ は，前述のパラメータにより資源予約した場合の最大の論理処理遅延時間であり，次のように定義できる.

$$D_n^{max} = \max_i(D_n^l(m_i))$$

LBAP では，このように特徴付けられたメッセージストリームの処理を，次のように行う.

- タスクのデッドライン時刻に基づき，EDF により実行順を決定する.
- 前述のように定義されたタスクがパイプラインモデルにより複数接続される.
- タスクは，メッセージの到着により実行可能となる.
- タスクは，メッセージ処理完了時刻に未処理メッセージがある場合，処理完了したメッセージのデッドライン時刻を待たずに直ちに実行可能となる.

上記のように，定義された時間属性において連続メディアデータ出力装置の時間制約を保証するには，入力装置と出力装置間に十分な初期位相が必要である．その初期位相は，次のようになる.

$$a_{out}(m_i) \geq a_{in}(m_0) + \sum_{j=0}^{n}(W_j^{max} + 1)/R^c$$

ただし，$a_{out}(m_i)$ は出力装置における $i$ 番目のメッセージ到着時刻，$a_{in}(m_0)$ は入力装置における最初のメッセージ発生時刻，$n$ は入出力装置間のパイプライン上のタスク数である.

従って，出力装置の時間制約を保証する場合，最悪実行時間に基づく CBR ストリーム処理モデルは，大きなエンドーエンドの遅延時間が必要となる．また，最悪実行時間に基づく CPU 使用量の予約は，過度な CPU 予約量となり，システム全体での CPU 利用率が低下する.

### 2.1.2 VBR ストリーム処理モデル

前節の問題を解決するために，ADM では，メッセージ処理時間を最悪実行時間 $C^{max}$ より短い任意の時間 $C^{req}$ とし，また，最大未処理メッセージ数 $W^{max}$ の代わりに実測の未処理メッセージ数 $b$ を用いる．そのパラメータを次に示す.

$R^c$: constant message rate (messages/second)
$b$: actual workahead messages (message)
$C^{req}$: request processing time for a message (seconds/message)

タスク $n$ において，レート $R^c$ で最悪実行時間より短い時間 $C^{req}$ を，CPU 利用率として予約している場合，すべてのメッセージの処理時間が $1/R^c$ 以下であることは保証されない．従って，式（1）, (3）のような予測は困難である．そこで，VBR データ処理モデルにおける時間属性は，変動する処理時間を考慮して定義する.

未処理メッセージ数の定義は，$w_n(m_i)$ に代わり，任意の時刻に実測された未処理メッセージ数を用いる．タスク $n$ において，時刻 $t$ に実測された未処理メッセージ数を $b_n(t)$ と表記し，実効未処理メッセージ数と呼ぶ.

メッセージ論理到着時刻 $l_n(m_i)$ の代わりに，メッセージ実効到着時刻 $e_n(m_i)$ を定義する．$l_n(m_i)$ は，メッセージが初めて処理対象と予測される論理時刻である．このことから，メッセージ実効到着時刻 $e_n(m_i)$ は，タスク $n$ において，$i$ 番目のメッセージが，初めて処理対象となった実際の時刻とし，次のように定義する.

$$\begin{aligned} e_n(m_0) &= a_n(m_0) \\ e_n(m_i) &= max(a_n(m_i), f_n(m_{i-1})) \end{aligned} \tag{7}$$

ただし，$f_n(m_{i-1})$ は，タスク $n$ における $i-1$ 番目のメッセージの処理完了時刻である.

次に，デッドライン時刻について定義する．デッドライン時刻を算出するために，論理処理遅延時間 $D_n^l(m_i)$ を知る必要がある．しかし，時刻 $e_n(m_i)$ において，$i$ 番目のメッセージは処理を完了していないため，$D_n^l(m_i)$ を知ることができない．そこで，パイプラインモデルの隣接するタスク $n$ と $n+1$ について考える．パイプラインモデルにおいて，タスク $n+1$ における $i$ 番目のメッセージが処理対象となる時刻は，タスク $n+1$ における $i-1$ 番目のメッセージの処理完了時刻 $f_{n+1}(m_{i-1})$ である．すなわち，タスク $n$ は，$i$ 番目のメッセージ

の処理を，$f_{n+1}(m_{i-1})$ までに完了すればよいと考えられる．従って，タスク $n$ における $i$ 番目のメッセージのデッドライン時刻は，次のように考える．

$d_n(m_i) = f_{n+1}(m_{i-1})$

時刻 $e_n(m_i)$ において，$f_{n+1}(m_{i-1})$ は

$\bar{f}_{n+1}(m_{i-1}) = e_n(m_i) + b_{n+1}(e_n(m_i))/R^c$

従って，デッドライン時刻 $d_n(m_i)$ を，次のように定義する．

$$d_n(m_i) = e_n(m_i) + b_{n+1}(e_n(m_i))/R^c \qquad (8)$$

その他の定義は，前節と同様である．

#### 2.1.3 VBR ストリーム処理タスクにおける特性

VBR ストリーム処理モデルにおけるタスクの特性は，著者らの文献 11) と報告 9) により，次のようになる．

**[特性 1]** VBR ストリーム処理モデルにおいて，最悪実行時間より短い時間を予約する場合，メッセージの処理完了時刻は，以下の条件で，デッドライン時刻を満たす．

$b_{n+1}(e_n(m_i))/R^c \geq D_n^a(m_i)$

**[特性 2]** 特性 2 を満たす VBR ストリーム処理モデルにおいて，以下の条件を満たすならば，メッセージ処理に必要な CPU 利用率は，予約した CPU 利用率を超えない．

$$b_{n+1}(e_n(m_i)) \geq \frac{C_n(m_i)}{C_n^{req}}$$

ただし，$C_n(m_i)$ は，タスク $n$ における $i$ 番目のメッセージ処理に使用する CPU 時間である．

**[特性 3]** VBR ストリーム処理モデルにおいて，最悪実行時間より短い時間を予約している場合，パイプライン上の隣接するタスクのメッセージ処理遅延時間が以下を満たすならば，連続メディアデータ出力装置の時間制約が満たされることが保証される．

$$\begin{aligned}\sum_{k=0}^{i}\{\sum_{j=0}^{n-1}(&D_j^a(m_{k+1}) - D_{j+1}^a(m_k)) \\ &+ (D_{in}^a(m_{k+1}) - D_0^a(m_k)) \\ &+ (D_n^a(m_{k+1}) - D_{out}^a(m_k))\} \leq 0\end{aligned}$$

ただし，$D_{out}^a(m_k)$ は出力装置における $k$ 番目のメッセージ処理遅延時間，$D_{in}^a(m_{k+1})$ は入力装置における $k+1$ 番目のメッセージ処理遅延時間である．

#### 2.1.4 ブロッキング時間

ストリーム処理において，パイプライン上の隣接するタスク間でのメッセージ通信は非同期通信により行われる．そのため，隣接するタスク間にはメッセージを保存する FIFO キューが用意される．この各タスク間のキューもまた，使用量には限度がある．タスク間のキューが使用量の限度に達している場合，送信側タスクのメッセージ送信はキューが空くまで待機する．これにより，タスクにはブロッキング時間が発生する．

ADM では，ブロッキング時間をメッセージ実効到着時刻 $e_n(m_i)$ からメッセージ処理完了 $f_n(m_i)$ までの期間で，当該タスクの優先度より低い優先度のタスクが実行した時間とアイドル時間の和をブロッキング時間とする．このブロッキング時間は優先度逆転（priority inversion) を引き起こし，スケジューリング可能性を低下させる．

ADM では，著者らが文献 10) に示したブロッキング時間を抑制する制御を用いる．すなわち，前メッセージ処理でブロッキング時間が発生したタスクは相対的に実行開始時刻が早いと考えられるので，実行開始時刻を遅くするために優先度を低くする．これにより，スケジュール可能性の低下を防ぐ．

#### 2.1.5 スケジュール可能性を高めるポリシ

スケジュール可能性を高めるため，ストリーム処理タスクの特性とブロッキング時間の抑制を満たすように優先度を次のように制御する．

- パイプライン上のタスクの実行機会が同じになるように，各タスクの優先度を連動させる．
- メッセージ処理遅延時間は実効未処理メッセージ数 $b_n(t)$ に依存することから，タスク $n$ の $i$ 番目のメッセージの実行到着時刻における $b_n(e_n(m_i))$ と後続タスク $n+1$ の同じ時刻における $b_{n+1}(e_n(m_i))$ を比較する．前者の値が大きい場合は，タスク $n$ に大きな遅延が発生することが予想されるのでタスク $n$ の優先度を高め，処理を早める．また，後者の値が大きい場合は，タスク $n$ の処理遅延は小さいことが予想されるので，優先度を低め，処理を遅らせる．
- 前メッセージ処理においてブロッキング時間が発生している場合は，実行開始時刻を遅くするため，優先度を低くする．
- メッセージ処理完了後に，ブロッキング時間が未発生であり，かつ，実測された処理遅延時間が特性 1 を満たさないならば，処理遅延時間を小さくするため，優先度を高める．
- メッセージ処理完了後に，ブロッキング時間が未発生であり，かつ，実測された処理遅延時間が特性 2 を満たさないならば，処理遅延時間を大きくするため，優先度を低める．

#### 2.1.6 適応デッドライン時刻

ストリーム処理タスクは，航空システムや原子力システムに見られる処理完了時刻に厳密なハード実時間タスクと異なり，その処理完了時刻には許容される遅

延幅があるソフト実時間タスクとして考えられる．従って，デッドライン時刻に許容遅延幅 $h_n$ を持たせる．これにより，タスクの優先度として用いるデッドライン時刻は，式（8）で求められる時刻に，許容遅延幅内 $[-h_n, h_n]$ で，スケジュール可能性を高める制御による優先度の修正を行った時刻とする．この時刻を適応デッドライン時刻 $d_n^{adap}(m_i)$ と呼ぶ．

### 2.2 適応メカニズム

2.1.5 節では，スケジューリング可能性を高めるポリシについて述べた．しかし，このポリシは単一のストリーム処理タスクにおけるポリシである．前提としている問題は複数のストリーム処理が混在する処理環境である．すなわち，スケジューリング問題は，パインラインモデルにより接続されたタスク群が複数混在するタスクセットにおいて，各タスクのポリシを同時に満たす状態を探し出す多重制約問題である．

ADM は，この多重制約問題を処理するために，認知科学における Parallel Distributed Processing モデル [6]（以降 PDP モデル）と熱力学的モデル [4] を組み合わせた適応メカニズム [10] を VBR ストリーム処理環境に適用する．以下に概要を示す．

PDP モデルでは，多くの単純な情報処理ユニットが相互に結合し，それぞれが他のユニットから信号を受け取る．その入力信号が正の値である場合は，ユニットの状態値を高める．負の値である場合は，その状態値を低める．さらに，その状態値に応じた信号を他のユニットに送る．この相互作用をユニット毎に非同期に繰り返すことにより情報処理を行う．各ユニットの相互結合により構成されるネットワークは，多重制約の構造を表している．このネットワークに，何らかの外部条件を与え，各ユニットで非同期に相互作用の繰り返し処理を行う．すると，各ユニットはある状態に落ち着く．その状態がある条件での制約を最大に充足した状態を表す．

以上のことから，複数のパイプラインのタスク群における制約を充足する状態を算出するために，タスク間の依存関係を PDP モデルの相互結合ネットワーク（以降，ネットワーク）に次のように対応づける．

- 各ユニットの状態値（0 から 1 までの連続値）は，各タスクの重要度とする．タスクの重要度は，最高値（1）を適応デッドライン時刻 $d_n^{adap}(m_i)$ の最短時刻 $d_n(m_i) - h_n$ に，最低値（0）を適応デッドライン時刻の最長時刻 $d_n(m_i) + h_n$ に対応づける．これにより，重要度から適応デッドライン時刻を算出する．
- ユニット間の結合は，通信するタスク間の場合は正の重みを，通信しないタスク間の場合は負の重みを持たせる．正の重みをもつタスク間の結合では，ユニットの状態値に比例した正の入力が他のタスクに作用し，相互に近い状態値になる．一方，負の重みをもつタスク間の結合では，ユニットの状態値に比例した負の入力が他のタスクに作用し，互いに遠い状態値になる．これにより，同一パイプライン上のタスク群の重要度が連動し，タスクの実行機会が同じになる．
- 各ユニットに与えられる外部条件は，変動するタスク実行状況に対応してネットワークの動作を修正する力とする．すなわち，外部条件は，各タスクの変動する処理遅延時間に応じて，2.1.5 節で述べた制御を，次のようにネットワークに作用させる．
  - タスク $n$ の $i$ 番目のメッセージ実効到着時刻における $b_n(e_n(m_i))$ と後続タスク $n+1$ の同じ時刻における $b_{n+1}(e_n(m_i))$ を比較し，前者の値が大きい場合は，正の入力とする．また，後者の値が大きい場合は，負の入力とする．
  - メッセージ処理でブロッキング時間が発生しているならば，負の入力とする．
  - メッセージ処理でブロッキング時間が未発生であり，実測された処理遅延時間が特性 1 を満たさないならば，正の入力とする．
  - メッセージ処理でブロッキング時間が未発生であり，実測された処理遅延時間が特性 2 を満たさないならば，負の入力とする．

以上により，複数のパイプラインのタスク群における制約をネットワークを動作させて処理する．しかし，PDP モデルは，ネットワークを動作させると，その状態は初期値に依存してある制約充足度の高い状態に至り，動作は静止する．一方，ストリーム処理環境において，各タスクの処理遅延時間は，処理するメッセージにより，常に変化する．ネットワークは，この変化に応じて，1 つの状態に静止することなく，いくつかの制約充足度の高い状態間を移動する必要がある．このため，PDP モデルに熱力学的モデルを加える．熱力学的モデルは，温度による物質の熱揺動（高温で分子間の結合が弱まり不安定，低温で分子間の結合が強まり安定する性質）を PDP モデルの処理において擬似する．すなわち，高い充足状態から離れた場合は温度を高くし，PDP モデルの処理動作を大きく振動させ新しい状態を見つける可能性を高める．一方，高い充足状態の近傍にある場合は温度を低くし，PDP モデルの処理動作を現在の状態の近傍で小さく振動させ，そ

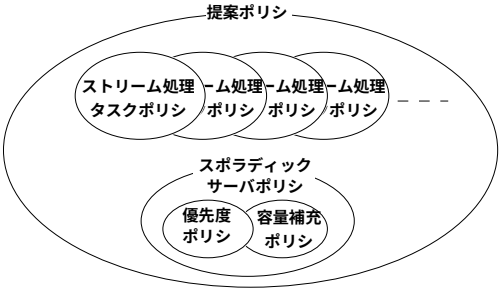


**図 3** 提案ポリシの構成
Fig. 3 Construction of proposed policy.

の状態を維持する．この温度による熱揺動制御により，PDP モデルは各タスクの変動する処理遅延時間に連続的に動作するようになる．従って，ブロッキング時間を抑制し，特性 1 と 2 を満たし，かつ，タスク間の処理遅延時間を均一にする適応デッドライン時刻を探し続けることを可能とする．

## 3. 実時間および非実時間タスクの提案ポリシ

本章では，連続メディアをストリームデータとして扱うストリーム処理タスク（ソフト実時間タスク）と非連続メディアを扱うイベント駆動タスク（非実時間タスク）が混在する環境におけるスケジューリングポリシを提案する．

### 3.1 提案ポリシの構成

提案ポリシは，スポラディックサーバを用いて，非実時間タスクであるイベント駆動タスクを ADM のソフト実時間タスクスケジュール環境に取り込む．イベント駆動タスクはスポラディックサーバ内の任意のスケジューリング方式より実行順が決められる．さらに，スポラディックサーバは，ストリーム処理タスクと同様に 1 つのソフト実時間タスクとして，ADM によりスケジューリングされる．すなわち，提案ポリシは前章のストリーム処理タスクのポリシとスポラディックサーバのポリシから構成される（**図 3** 参照）．また，スポラディックサーバのポリシは，容量補充ポリシと優先度ポリシの 2 つから構成される（図 3 参照）．

### 3.2 スポラディックサーバのポリシ

本節ではスポラディックサーバのポリシを構成する容量補充ポリシと優先度ポリシについて述べる．

#### 3.2.1 容量補充ポリシ

非連続メディアデータの到着により実行可能状態となったイベント駆動タスクは，スポラディックサーバのランキューに接続される．スポラディックサーバはランキューに接続されたタスクをスケジューリングし処理する．このキューの待ち行列の平均長は，非連続

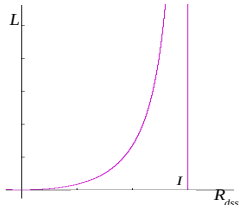


**図 4** 容量補充時間によるランキューの長さ
Fig. 4 Length of run queue by replenishment period.

メディアデータの到着がポアソン分布で処理時間が指数分布であると仮定すると，待ち行列理論の M/M/1 により次のようになる．

$$L = \frac{\rho^2}{1-\rho}$$

ただし，$\rho$ はトラフィック強度であるが，ここでは CPU 利用率であり，次のように定義される．

$$\rho = \frac{\lambda}{\mu}$$

$\lambda$ は平均到着率，$\mu$ は平均サービス率である．

ここで，スポラディックサーバにおけるイベント駆動タスクの平均サービス率 $\mu$ について考える．スポラディックサーバの最大補充量をイベント駆動タスクの平均 CPU 使用時間とし，またデッドライン時間 $T_{dss}$，容量補充時間を $R_{dss}$ とすると，イベント駆動タスクの平均サービス時間は $T_{dss} \leq R_{dss}$ の条件で $R_{dss}$ となる．従って，$\mu$ は次のようになる．

$$\mu = \frac{1}{R_{dss}}$$

また，非連続メディアデータの平均到着間隔を $I$ とすると，$\lambda$ はつぎのようになる．

$$\lambda = \frac{1}{I}$$

従って，スポラディックサーバにおけるイベント駆動タスクのランキュー（実行可能タスクの待ち行列）の平均長は次のようになる．

$$T_{dss} \leq R_{dss}$$

$$L = \frac{\left(\frac{R_{dss}}{I}\right)^2}{1-\frac{R_{dss}}{I}} \qquad (9)$$

イベント駆動タスクの平均応答時間は，スポラディックサーバの任意のスケジューリング方式において，スポラディックサーバにおけるイベント駆動タスクのランキューの平均長に依存して変動する．すなわち，ランキューが長くなると平均応答時間が大きくなり，またランキューが短くなると平均応答時間が減少する．従って，式（9）および**図 4** から，イベント駆動タスクの

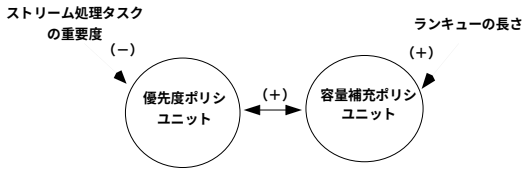


**図 5** スポラディックサーバポリシ内部のメカニズム
Fig. 5 Mechanism inside policy of dynamic sporadic server.

平均応答時間を小さくするために，次のようなポリシで容量補充時間 $R_{dss}$ を制御する．

- ランキューの長さが大きくなる場合，$T_{dss} \leq R_{dss}$ の条件においてスポラディックサーバの容量補充時間 $R_{dss}$ を小さくする．

#### 3.2.2 優先度ポリシ

一方，スポラディックサーバはストリーム処理タスクと CPU において競合関係にある．そのため，優先度において次のようなポリシで制御する．

- CPU 利用率が高く過負荷の場合，ストリーム処理タスクのスケジューリング可能性を維持するためスポラディックサーバの優先度を下げる．すなわち，スポラディックサーバのデッドライン時間 $T_{dss}$ を大きくする．
- CPU 利用率が低い場合，ストリーム処理タスクのスケジューリング可能性は十分に高いのでスポラディックサーバの優先度を上げる．すなわち，スポラディックサーバのデッドライン時間を $T_{dss}$ を小さくする．

#### 3.2.3 2 つのポリシの協調メカニズム

スポラディックサーバ内部の 2 つのポリシは，スポラディックサーバの容量補充時間 $R_{dss}$ とデッドライン時間 $T_{dss}$ の関係から協調関係と考える．この協調関係を ADM のネットワークメカニズムにより処理するため，次のようにネットワークを構成する（**図 5** 参照）．

- ネットワークは容量補充ポリシのユニットと優先度ポリシのユニットから構成する．
- 上記 2 つのユニット間の結合は正の重み付けとする．
- 容量補充ポリシのユニットの外部制約として，ランキューの長さが大きくなる場合正の入力を与える．
- 優先度ポリシのユニットの外部制約として，各ストリーム処理タスクの重要度に応じて負の入力を与える．

このネットワークメカニズムにより求められた 2 つのユニットの状態値を，ADM と同様にそれぞれの許容時間幅に重ねて適応容量補充時間 $R_{dss}^{adap}$ と適応デッドライン時間 $T_{ass}^{adap}$ を算出する．すなわち，スポラディックサーバの 2 つの時間は固定とせず，時間幅を持たせて実行状況に応じて変動させる．

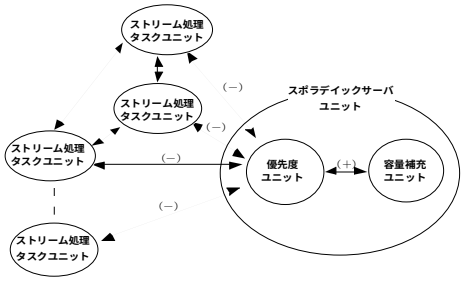


**図 6** 提案ポリシのメカニズム
Fig. 6 Mechanism for proposed policy.

任意のスケジューリングポイント $t$ における適応デッドライン時間 $T_{dss}^{adap}(t)$ は，当該ユニットの状態値の最高値 (1) を最小デッドライン時間 $T_{dss}^{min}$ に，最低値 (0) を最大デッドライン時間 $T_{dss}^{max}$ に対応付けて次のように求める．

$$T_{dss}^{adap}(t) = T_{dss}^{max} - v_{dss}^{T}(t)(T_{dss}^{max} - T_{dss}^{min})$$

ただし，$v_{dss}^{T}(t)$ は任意のスケジューリングポイント $t$ におけるスポラディックサーバポリシ内の当該の優先度ポリシユニットの状態値である．

また，任意のスケジューリングポイント $t$ における適応容量補充時間 $R_{dss}^{adap}(t)$ は，当該ユニットの状態値の最高値を適応デッドライン時間に，最小値を適応デッドライン時間に許容時間 $h_{dss}$ を加えた時間に対応付けて次のように求める．

$$R_{dss}^{adap}(t) = T_{dss}^{adap}(t) + (1 - v_{dss}^{R}(t))h_{dss}$$

ただし，$v_{dss}^{R}(t)$ は任意のスケジューリングポイント $t$ におけるスポラディックサーバポリシ内の当該の容量補充ポリシユニットの状態値である．

### 3.3 提案ポリシにおけるメカニズム

提案ポリシはスポラディックサーバのポリシと各ストリーム処理タスクのポリシから構成される．この複数のポリシを同時に満たすため，2 章で述べた多重制約を処理するメカニズムにスポラディックサーバのポリシを取り込む．すなわち，ストリーム処理タスク間の相互接続ネットワークにスポラディックサーバを次のように接続する（**図 6** 参照）．

- スポラディックサーバの当該ユニットと各ストリーム処理タスクのユニットとの結合は負の重み付けをする．すなわち，ストリーム処理タスクの重要度が高くなるとスポラディックサーバの重要度が

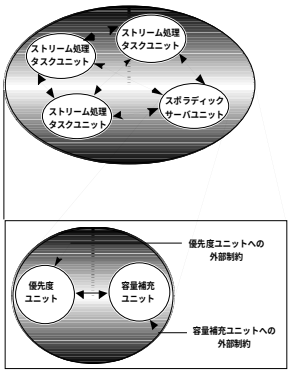


**図 7** 提案ポリシのメタレベル構造
Fig. 7 Meta-level architechure for proposed policy.

低くなり，ストリーム処理タスクの重要度が低くなるとスポラディックサーバの重要度が高くなる．

- 各ストリーム処理タスクのユニットからのスポラディックサーバのユニットへの入力は，スポラディックサーバのポリシ内部における優先度ポリシの当該ユニットの外部制約とする．
- スポラディックサーバにおいて，内部ポリシである優先度ポリシのユニットの状態値をスポラディックサーバの当該ユニットの状態値とする．

上記のように，提案ポリシはスポラディックサーバのポリシを含む複数のポリシを処理するため，ADM のメカニズムを動作させる．また，スポラディックサーバのポリシは 2 つのポリシを処理するため，その内部で ADM を動作させる．すなわち，提案ポリシはスポラディックサーバにおいて ADM メカニズムのメタレベル構造を形成する（**図 7** 参照）．

このような ADM のメカニズムをスケジューリングポイント毎に非同期に動作させて，ストリーム処理タスクの適応デッドライン時間，スポラディックサーバの適応容量補充時間と適応デッドライン時間を求める．

## 4. おわりに

本稿では，任意のタスクが生成されるストリーム処理環境において，実時間タスクのスケジューリング可能性を高め，また非実時間タスクの応答性を改善することを目的とするスケジューリングポリシを提案した．今後は，提案ポリシの実装および評価を行う予定である．

## 参考文献


1) C. L. Liu and J. W. Layland: Scheduling algorithms for multiprograming in a hard real time enviroment, *J. ACM*, Vol. 20, No. 1, pp. 46–61 (1973).
2) M. Spuri and G. C. Buttazzo: Efficent aperiodic service under earliest deadline scheduling, *Proc. IEEE Real-Time Systems Symposium*, (1994).
3) M. Spuri and G. C. Buttazzo: Scheduling aperiodic tasks in dynamic priority systems, *Journal of Real-Time Systems*, Vol.10, No.2, (1996).
4) R. Yavatkar and K. Lakshman: Optimization by Simulated Anneling, *Science*, Vol. 220, pp. 671–680 (1983).
5) R. Steinmetz and K. Nahrstedt: *Multimedia: Multimedia Applications*, Prentice Hall, pp. 709–765 (1995).
6) D. E. Rumelhart, J. L. McCelland and the PDP Research Group: *PARALLEL DISTRIBUTED PROCESSING*, The MIT Press (1986).
7) D. P. Anderson: Metascheduling for continous media, *Trans Comput Syst ACM*, No. 11, pp. 226–252 (1993).
8) R. Govindan and D. P. Anderson: Scheduling and IPC mechanisms for continuous media, *Proceeding of the 13th ACM on Operating Systems Principles*, pp. 68–80 (1991).
9) 滝沢泰久，芝公仁，大久保英嗣:VBR ストリーム処理のための適応的スケジューリングポリシ, **情報処理学会研究会報告** *2000-OS-86*, Vol. 2001, No. 21, pp. 123–130 (2001).
10) 滝沢泰久，芝公仁，大久保英嗣:連続メディア処理における時間制約と通信遅延に適応するタスクスケジューリング, **情報処理学会論文誌：数理モデル化と応用**, Vol. 42, No. Sig5, pp. 29–41 (2001).
11) 滝沢泰久，芝公仁，大久保英嗣:VBR ストリーム処理のための適応的スケジューリングポリシと性能評価, **情報処理学会論文誌：数理モデル化と応用**, Vol. 42, No. Sig14, pp. 50–63 (2001).