システムソフトウェアとオペレーティング・システム 86−19
(2001. 3. 6)

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

滝沢泰久[†], 芝公仁[††], 大久保英嗣[†††]

†（株）ATR環境適応通信研究所
††立命館大学大学院理工学研究科
†††立命館大学理工学部情報学科
E-mail: takizawa @acr.atr.co.jp
E-mail:shiba@sol.cs.ritsumei.ac.jp
E-mail:okubo@cs.ritsumei.ac.jp

あらまし 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をスケジュールする場合，その時間制約を満たすために，最悪処理時間に基づいてリアルタイムスケジューラを利用することが有効とされている．一方，多くの連続メディアデータは圧縮／伸張処理や差分処理などにより，処理時間はランダムに変動する．このような連続メディアデータを従来の方式で処理すると，過度なCPU利用率予約のため大きくCPU利用率が低下する．本稿では，連続メディア資源モデルであるLinear Bounded Arrival Processに変動処理量の変更を加え，その処理モデルに基づき，Parallel Distributed Processingモデルと熱力学的モデルを用いることにより，ストリーム処理タスクの時間制約と処理量の変動に適応するスケジューリングポリシを提案する．

キーワード スケジューリングポリシ リアルタイムシステム マルチメディア

An Adaptive Scheduling Policy for VBR Stream Processing

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

† ATR Adaptive Communication Research Laboratories
†† Graduate School of Science and Engineering, Ritsumeikan Univ.
††† Department of Computer Science, Facutly of Science and Engineering, Ritsumeikan Univ.
E-mail: takizawa @acr.atr.co.jp
E-mail:shiba@sol.cs.ritsumei.ac.jp
E-mail:okubo@cs.ritsumei.ac.jp

Abstract Real-time scheduling policies based on worst-case execution time are useful for scheduling stream processing tasks which manipulate continuous media such as voice, audio, video and animation. On the other hand, processing time for continuous media data dynamically fluctuate as result of compression, extension and deferences between data. Therefore, the above scheme causes excessive reservation of processing time because processing time for continuous media data is shorter than worst-case execution time.In this paper, a new scheduling policy based on Linear Bounded Arrival Process and applied Parallel Distributed Processing model and thermal dynamics model, which is adaptable for stream processing tasks with timing constraints and processing delay is proposed.

key words scheduling policy, real-time system, multimedia

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[4]は，それらの処理するメディアの特性上，ある時間周期でデータが発生し，次の周期までに処理を完了しなければならない．すなわち，連続メディアデータを処理するタスクは，周期タスクモデルに基づいた時間制約を持つ．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，このような時間制約を持つ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．このようなストリーム処理タスクをスケジュールする場合，その時間制約を満たすために，最悪実行時間と周期タスクモデルに基づいてCPU利用率を予約した上で，リアルタイムスケジューラを用いる方式が有効とされている．

一方，連続メディアデータは，入出力装置から固定周期で生成／消費され，そのデータ量は動画データなどの圧縮／伸張や差分処理[6]に見られるように可変量である．すなわち，連続メディアデータは，Variable Bit Rate（以降VBR）データである．このような連続メディアデータのストリーム処理は，順次処理されるパイプラインモデルにより行われる（図1）．このパイプライン上のタスクはVBRデータを処理するため，その処理遅延も可変となる．すなわち，VBRデータを処理するタスクの処理遅延は可変となり，パイプライン上のタスクにおけるデータの到着は周期性を失う．このため，周期タスクモデルをそのままストリーム処理に適用すると，タスクにデータ未到着による待ち時間が発生しスケジュール可能性が低下する[1]．これを回避するためには，パイプライン上の隣接するタスク間に十分な初期位相が必要であり，そのため，エンドーエンドの処理遅延が大きくなる．また，タスクの時間制約を満たすために，最悪実行時間に基づきCPU利用率を予約することは，VBRデータ処理環境では過度な予約量となり，システム全体でのCPU利用率が大きく低下する．

本論文では，以上の問題点を解決するために，以下の4つを前提として，従来の方式より高いスケジュール可能性を導く適応的スケジューリングポリシを提案する．

- 最悪実行時間より短い時間によりCPU利用率を予約する．
- 各タスクのメッセージ処理に必要なCPU使用時間は，ランダムである．
- 各タスクは，連続メディアデータをパイプラインモデルにより処理する．
- 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．
- 複数のストリーム処理が混在する．

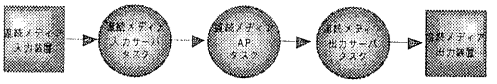

図1 連続メディアデータのストリーム処理
Fig. 1 Stream processing for continuous media data.

提案ポリシでは，ストリーム処理モデルを連続メディア資源モデルLinear Bounded Arrival Process[7]（以降LBAP）にVBRデータ処理のための変更を加えたモデルとし，Parallel Distributed Processingモデル（以降PDPモデル）[5]と熱力学的モデル[3]を動作メカニズムとしている．これにより，VBRデータに動的に適応するタスクスケジューリングが可能となる．

以下，2章でVBRストリーム処理モデルを定義する．次に，3章で，2章の定義に基づき，従来の方式の問題点を解決するための新たなスケジューリングポリシを提案する．

2. VBRストリーム処理モデル

本章では，LBAPについて説明し，LBAPに基づいた従来の処理モデルについて議論する．その上で，VBRストリーム処理モデルをLBAPに基づいて定義する．

2.1 LBAP

LBAPでは，処理すべきデータを以下の3つのパラメータにより特徴付けたメッセージのストリームしている．

R^{max} maximum message rate (messages/second)
W^{max} maximum workahead (messages)
S^{max} maximum message size (bytes)

LBAPでは，時間間隔tにおいて到着する最大メッセージ数は$R^{max}t+W^{max}$であると定義している．すなわち，長期的なデータ処理レートは$R^{max}S^{max}$以下であるが，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，データレートは短期間$R^{max}S^{max}$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．

$$w_n(m_0) = 0$$
$$w_n(m_i) = max(0, w_n(m_{i-1}) - (a_n(m_i) - a_n(m_{i-1}))R^{max} + 1)$$

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．

$$W_n^{max} = \max_i(w_n(m_i)) \quad (1)$$

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．

$$l_n(m_i) = a_n(m_i) + w_n(m_i)/R^{max}$$

$$l_n(m_0) = a_n(m_0)$$
$$l_n(m_i) = max(a_n(m_i), l_n(m_{i-1}) + 1/R^{max})$$

また，LBAP では，タスク n における i 番目のメッ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，メッセージ到着時刻とメッセージ論理到着時刻を用いて，次のように定義する（**図 2** 参照）．

$$D_n^a(m_i) = a_{n+1}(m_i) - a_n(m_i) \quad (2)$$

$$D_n^l(m_i) = l_{n+1}(m_i) - l_n(m_i) \quad (3)$$

$$d_n(m_i) = l_n(m_i) + D_n^{max} \quad (4)$$

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る．すなわち，D_n^{max} は次にように定義できる．

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

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[2] 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，メッセージの到着時刻に起動する．
- 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起動する．

2.2 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．

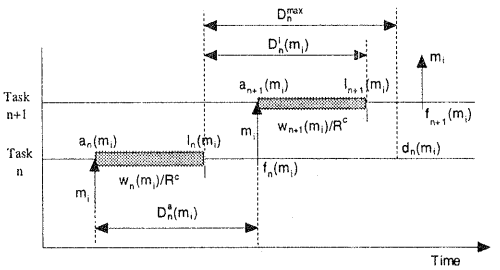

図 2 LBAP における時間属性
Fig. 2 timing attribute on LBAP.

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

この定義により，LBAP を用いた最悪実行時間に基づく CBR ストリーム処理モデルにおける未処理メッセージ数，およびメッセージ論理到着時刻は，次のようになる．

$$w_n(m_0) = 0$$
$$w_n(m_i) = max(0, w_n(m_{i-1}) - (a_n(m_i) - a_n(m_{i-1}))R^c + 1) \quad (5)$$

$$l_n(m_i) = a_n(m_i) + w_n(m_i)/R^c \quad (6)$$

$$l_n(m_0) = a_n(m_0)$$
$$l_n(m_i) = max(a_n(m_i), l_n(m_{i-1}) + 1/R^c) \quad (7)$$

その他の定義は，2.1 と同様である．上記のように，定義された時間属性において，パラメータ C^{max} と R^c により CPU 利用率を予約している場合，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，周期タスクモデルと同様に，隣接するタスク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．

$$a_n(m_i) \geq a_0(m_0) + \sum_{j=0}^{n-1}(W_j^{max} + 1)/R^c$$

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においても，大きなエンドーエンドの遅延時間が必要となる．

2.3 VBRデータにおけるストリーム処理モデル

前述のように，最悪実行時間に基づくCBRストリーム処理モデルは，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．また，出力装置の時間制約を満たすためには，十分な初期位相が必要である．そのため，エンドーエンドの処理遅延時間が大きくなる．

上記の問題を解決するために，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} とする．本節では，C^{req} に基づいて，次のパラメータのように，CPU利用率を予約した場合の処理モデルについて考える．

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 (5)，(6)，(7) のような予測は困難である．そこで，VBRデータ処理モデルおける時間属性は，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 $l_n(m_i)$ に代わり，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．

$$\begin{aligned} e_n(m_0) &= a_n(m_0) \\ e_n(m_i) &= max(a_n(m_i), f_n(m_{i-1})) \end{aligned} \tag{8}$$

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．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．

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

しかし，時刻 $e_n(m_i)$ において，$f_{n+1}(m_{i-1})$ は，$D_n^l(m_i)$ と同様に不明である．そのため，時刻 $e_n(m_i)$ におけるタスク $n+1$ の実効未処理メッセージ数から，$f_{n+1}(m_{i-1})$ を次のように想定する．

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

従って，デッドライン時刻 $d_n(m_i)$ は，次のように定義する．

$$d_n(m_i) = e_n(m_i) + \frac{b_{n+1}(e_n(m_i))}{R^c} \tag{9}$$

その他の定義は，2.1と同様である．

2.4 VBRストリーム処理モデルにおける特性

本節では，最悪実行時間より短い処理時間に基づきCPU利用率を予約している場合におけるVBRストリーム処理モデルの特性に関して考察する．

［定理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，パラメータ C^{req} と R^c により CPU利用率として予約している場合，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．

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

［証明］メッセージの処理完了時刻が，デッドライン時刻を満たすためには，次のような条件となる．

$$f_n(m_i) \leq d_n(m_i) \tag{10}$$

上式を，式 (8) のメッセージ実効到着時刻の定義に従い，$a_n(m_i)$ が $f_n(m_{i-1})$ より大きい場合と，そうでない場合とに分けて考える．

$a_n(m_i) \geq f_n(m_{i-1})$ の場合，$e_n(m_i)$ は $a_n(m_i)$ となるので，式 (8) により，式 (10) は次のようになる．

$$f_n(m_i) \leq a_n(m_i) + \frac{b_{n+1}(e_n(m_i)}{R^c}$$

また，メッセージ処理完了時刻 $f_n(m_i)$ は，メッセージ到着時刻 $a_n(m_i)$ に処理遅延時間 $D_n^a(m_i)$ を加えた時刻である．従って，上式は次のようになる．

$$\frac{b_{n+1}(e_n(m_i))}{R^c} \geq D_n^a(m_i) \tag{11}$$

従って，$a_n(m_i) \geq f_n(m_{i-1})$ の場合，上式を満たす

場合，処理モデルはデッドライン時刻を満たす．

次に，$a_n(m_i) < f_n(m_{i-1})$ の場合について考える．この場合，$e_n(m_i)$ は $f_n(m_{i-1})$ となるので，式 (8) により，式 (10) は次のようになる．

$$f_n(m_i) \leq f_n(m_{i-1}) + \frac{b_{n+1}(e_n(m_i))}{R^c}$$

パイプラインモデルにおいて，タスク n のメッセージ処理完了時刻 $f_n(m_i)$ は，タスク $n+1$ のメッセージ到着時刻 $a_{n+1}(m_i)$ と一致する．従って，上式は，次のようになる．

$$\frac{b_{n+1}(e_n(m_i))}{R^c} \geq a_{n+1}(m_i) - a_{n+1}(m_{i-1}) \quad (12)$$

また，場合分けの条件も同様に，次のようになる．

$$a_n(m_i) < a_{n+1}(m_{i-1}) \quad (13)$$

ここで，式 (12) の右辺の式，$a_{n+1}(m_i) - a_{n+1}(m_{i-1})$ について考える．この式を $D_n^a(m_i)$ を用いて表すと，次のようになる．

$$\begin{aligned} &a_{n+1}(m_i) - a_{n+1}(m_{i-1}) \\ &\quad = (a_{n+1}(m_i) - a_n(m_i)) \\ &\qquad + a_n(m_i) - a_{n+1}(m_{i-1}) \\ &\quad = D_n^a(m_i) + a_n(m_i) - a_{n+1}(m_{i-1}) \end{aligned}$$

上式は，式 (13) により，次のようになる．

$$a_{n+1}(m_i) - a_{n+1}(m_{i-1}) = D_n^a(m_i) - x$$

ただし，x は正の数である．

このことから，式 (12) は次のようになる．

$$\frac{b_{n+1}(e_n(m_i))}{R^c} \geq D_n^a(m_i) - x \quad (14)$$

従って，$a_n(m_i) > f_n(m_{i-1})$ の場合，上式を満たす場合，処理モデルはデッドライン時刻を満たす．

以上の 2 つの場合において，両方の場合共，デッドライン時刻を満たすには，式 (11)，(14) を同時に満たす必要がある．従って，次のようになる．

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

□

［定理 2］定理 1 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．

$$b_{n+1}(e_n(m_i)) \geq \frac{C_n(m_i)}{C_n^{req}} \quad (15)$$

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．

［証明］タスク n において，1 メッセージ処理当たりに予約した CPU 使用時間を C_n^{req} とすると，CPU 予約量は $C_n^{req}R^c$ である．従って，タスク n における i 番目のメッセージの処理ために使用した CPU 利用率 $U_n(m_i)$ は，次のようになる．

$$U_n(m_i) = \frac{C_n(m_i)}{d_n(m_i) - a_n(m_i)} \quad (16)$$

処理モデルは定理 1 を満たすことから，メッセージ処理完了時刻はデッドライン時刻より小さい．従って，上記の $[a_n(m_i), d_n(m_i))$ の時間間隔は，次のようになる．

$$d_n(m_i) - a_n(m_i) \geq f_n(m_i) - a_n(m_i) = D_n^a(m_i)$$

上記を式 (16) に適用すると $U_n(m_i)$ は次のようになる．

$$U_n(m_i) \leq \frac{C_n(m_i)}{D_n^a(m_i)}$$

タスク n において，処理に必要な CPU 利用率 $U_n(m_i)$ は，予約した CPU 利用率 $C_n^{req}R^c$ を超えないためには，次式を満たす必要がある．

$$U_n(m_i) \leq \frac{C_n(m_i)}{D_n^a(m_i)} \leq C_n^{req}R^c$$

上式と定理 1 より，次式が得られる．

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

□

［定理 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，パラメータ C^{req} と R^c により，CPU 利用率として予約している場合，タスク n の初期位相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．

$$a_n(m_0) \geq a_0(m_0) + \sum_{j=0}^{n-1} D_j^a(m_i) \quad (17)$$

［証明］出力装置の時間制約を満たすには，出力装置は $1/R^c$ 時間毎にメッセージを必要とすることから，次式が成り立つ必要がある．

$$a_{out}(m_i) \leq a_{out}(m_0) + i/R^c$$

ただし，$a_{out}(m_i)$ は出力装置における i 番目のメッセージの到着時刻である．

出力装置をパイプライン上の終端タスク n の次のタスク，すなわちタスク $n+1$ として考えると，上式は，次のようになる．

$$a_{n+1}(m_i) - a_{n+1}(m_0) \leq i/R^c \quad (18)$$

パイプラインモデルにおいて，隣接するタスクのメッセージ到着時刻 $a_{n+1}(m_i)$ とメッセージ処理完了時刻 $f_n(m_i)$ が一致することから，$a_{n+1}(m_i)$ は，次のようになる．

$$a_{n+1}(m_i) = f_n(m_i) = a_n(m_i) + D_n^a(m_i)$$

この式を，n に関して展開すると，次式が得られる．

$$a_{n+1}(m_i) = a_0(m_i) + \sum_{j=0}^{n} D_j^a(m_i)$$

上式の $a_0(m_i)$ は，タスク 0 におけるメッセージ到着時刻であるが，タスク 0 はパイプラインの先頭のタスクである．従って，タスク 0 のメッセージは入力装置から得られる．入力装置のメッセージの到着時刻は周期的であることから，$a_0(m_i)$ は次のようになる．

$a_0(m_i) = a_0(m_0) + i/R^c$

以上より，$a_{n+1}(m_i)$ は次のようになる．

$$a_{n+1}(m_i) = a_0(m_0) + i/R^c + \sum_{j=0}^{n} D_j^a(m_i)$$

上式を，式（18）に適用すると，次の式が得られる．

$$a_{n+1}(m_0) \geq a_0(m_0) + \sum_{j=0}^{n} D_j^a(m_i)$$

□

3. 提案スケジューリングポリシ

提案スケジューリングポリシは，複数の VBR ストリーム処理において，タスクのスケジュール可能性を高める．以下，本節では提案ポリシについて説明する．

3.1 スケジュール可能性を高める制御

本節では，前章で述べた VBR ストリーム処理モデルを前提として出力装置の時間制約を満たすタスクの制御について考える．

定理 3 により，タスクの初期位相が式（17）を満たすなら，最悪実行時間より短い時間を CPU 利用率として予約する場合においても，出力装置の時間制約を満たすことができる．しかし，出力装置の時間制約を満たすためには，各タスクに十分な初期位相が必要である．そのため，エンド－エンドの処理遅延時間が大きくなる．ライブによるシステム，例えば，テレビ会議システムでは，エンド－エンドの処理遅延時間が少ないことが求めるれる．このことから，十分な初期位相がない場合において，出力装置の時間制約を満たす可能性を高める制御について考える．

定理 3 の式（17）を見直す．この式の $a_n(m_i)$ に，前述のパイプラインモデルにおけるメッセージ到着時刻と処理完了時刻の関係を適用すると，次式が得られる．

$$\sum_{j=0}^{n}(D_j^a(m_i) - D_j^a(m_0)) \leq (a_0(m_0) - a_0(m_i)) + i/R^c$$

ここで，$(a_0(m_0) - a_0(m_i))$ は，タスク 0 のメッセージ到着時刻の差である．タスク 0 はパイプラインの最初のタスクであるため，メッセージの到着時刻は，入力装置からのメッセージ出力時刻である．従って，メッセージの到着時刻は周期的であり，$(a_0(m_0) - a_0(m_i))$ は $-i/R^c$ である．このことから，上式は次のようになる．

$$\sum_{j=0}^{n}(D_j^a(m_i) - D_j^a(m_0)) \leq 0 \qquad (19)$$

さらに，0 番目のメッセージ到着時刻から i 番目のメッセージ到着時刻までの時間間隔を，メッセージの到着間隔ごとに分割した場合，式（19）の $D_j^a(m_i) - D_j^a(m_0)$ は，次のように分解できる．

$$\begin{aligned} &D_j^a(m_i) - D_j^a(m_0)) \\ &\quad = D_j^a(m_i) - D_j^a(m_{i-1}) \\ &\qquad + D_j^a(m_{i-1}) - D_j^a(m_{i-2}) \\ &\qquad\quad \vdots \\ &\qquad + D_j^a(m_2) - D_j^a(m_1) \\ &\qquad + D_j^a(m_1)) - D_j^a(m_0) \end{aligned}$$

従って，式（19）は次のようになる．

$$\sum_{j=0}^{n}\sum_{k=0}^{i}(D_j^a(m_{k+1}) - D_j^a(m_k)) \leq 0 \qquad (20)$$

式（20）から分かるように，各タスクにおいて，メッセージ間での処理遅延時間を均等にすると，時間制約が満たされる．しかし，各メッセージの処理遅延時間は，そのメッセージ毎に変動するため，単一のタスク内でメッセージ間での処理遅延時間を均等にすることは難しい．そこで，各タスクのメッセージ間での処理遅延時間の差異を，隣接するタスク間で吸収することを考える．先ず，式（20）をタスク j に関して展開すると次のようになる．

$$\begin{aligned} &\textstyle\sum_{j=0}^{n}\sum_{k=0}^{i}\{D_j^a(m_{k+1}) - D_j^a(m_k))\} \\ &\quad = \textstyle\sum_{k=0}^{i}\{D_0^a(m_{k+1}) - D_0^a(m_k) \\ &\qquad + D_1^a(m_{k+1}) - D_1^a(m_k) \\ &\qquad\quad \vdots \\ &\qquad + D_{n-1}^a(m_{k+1}) - D_{n-1}^a(m_k) \\ &\qquad + D_n^a(m_{k+1}) - D_n^a(m_k)\} \qquad (21) \\ &\quad = \textstyle\sum_{k=0}^{i}\{D_0^a(m_{k+1}) - D_1^a(m_k) \\ &\qquad + D_1^a(m_{k+1}) - D_2^a(m_k) \\ &\qquad\quad \vdots \\ &\qquad + D_{n-1}^a(m_{k+1}) - D_n^a(m_k) \\ &\qquad + D_n^a(m_{k+1}) - D_0^a(m_k)\} \end{aligned}$$

ここで，タスク n における $k+1$ 番目のメッセー

ジ処理遅延時間 $D_n^a(m_{k+1})$ は，タスク $n+1$ における k 番目のメッセージ処理遅延時間 $D_{n+1}^a(m_k)$ と比較されるべきであるが，タスク $n+1$ は存在しない．しかし，タスク $n+1$ は出力装置として考えられるので，$D_n^a(m_{k+1})$ の比較対象は出力装置における k 番目のメッセージ処理遅延時間 $D_{out}^a(m_k)$ とする．同様に，タスク 0 における k 番目のメッセージ処理遅延時間 $D_0^a(m_k)$ 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 $D_{in}^a(m_{(k+1)})$ と比較する．このことより，式 (21) は次のようになる．

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

従って，隣接するタスク間のメッセージ処理遅延時間を均等にすることにより，出力装置の時間制約を満たすことが可能となる．従って，次のような制御を行う．

- 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 $k+1$ 番目のメッセージにおける $b_n(e_n(m_{k+1}))$ と後続タスク $n+1$ の k 番目のメッセージにおける $b_{n+1}(e_n(m_k))$ を比較し，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，処理を速める．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．
- メッセージ処理完了後に，実測された処理遅延時間が定理 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，実測された処理遅延時間が定理 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．

以上により，予約 CPU 利用率に基づき，連続メディア出力装置の時間制約を満たし，かつ，エンドーエンドの処理遅延時間を小さくすることが可能となる．

3.2 適応デッドライン時刻

ストリーム処理するタスクは，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リアルタイムタスクと異なり，その処理完了時刻には許容される遅延幅があるソフトリアルタイムタスクとして考えられる．従って，デッドラインに許容遅延幅を持たせる．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 (9) で求められる時刻に，許容遅延幅内で，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 $d_n^{adap}(m_i)$ と呼び，次のように定義する．

$$-h_n \leq v_n(m_i) \leq h_n$$
$$d_n^{adap}(m_i) = d_n(m_i) + v_n(m_i)$$

$v_n(m_i)$ はタスク n の i 番目のメッセージにおける修正時間であり，h_n はタスク n の許容遅延時間である．

3.3 複数のストリーム処理が混在する環境と多重制約

3.1 節では，スケジュール可能性を高める制御について述べた．しかし，この制御は単一のストリーム処理における制御である．マルチメディア環境においては，多地点のテレビ会議システムに見られるように，複数のストリーム処理が混在する場合が多い．すなわち，マルチメディア環境におけるストリーム処理は，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として考えられる．このようなタスクセットにおいて前節における制御を行うスケジュール問題は，複数のパイプラインのタスク群において前節における制御，すなわち制約を同時に満たすような状態を探し出す多重制約問題である．

提案ポリシは，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 [5]（以降 PDP モデル）を応用する．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正の重みを，通信しないタスク間の場合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，3.1 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 $k+1$ 番目のメッセージにおける $b_n(e_n(m_{k+1}))$ と後続タスク $n+1$ の k 番目のメッセージにおける $b_{n+1}(e_n(m_k))$ 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．
  - 実測された処理遅延時間が定理 1 を満たさないならば，正の入力とする．
  - 実測された処理遅延時間が定理 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 [3] 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 モデルを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し，定理 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．

4. おわりに

本稿では，VBR ストリーム処理タスクのスケジューリング方式において，予約 CPU に基づき，連続メディア出力装置の時間制約を満たし，かつ，エンド－エンドの処理遅延時間を小さくするポリシを提案した．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) L. Sha, R. Rajikumar, and J. P. Lehoczky: Priority Inheritance Protocols: An Approach to Real-Time Synchronization, *IEEE Trans. on Computer*, Vol. 39, No. 9, pp. 1175–1185 (1990).
2) 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).
3) R. Yavatkar and K. Lakshman: Optimization by Simulated Anneling, *Science*, Vol. 220, pp. 671–680 (1983).
4) R. Steinmetz and K. Nahrstedt: *Multimedia: Multimedia Applications*, Prentice Hall, pp. 709–765 (1995).
5) D. E. Rumelhart, J. L. McCelland and the PDP Research Group: *PARALLEL DISTRIBUTED PROCESSING*, The MIT Press (1986).
6) D. LeGall: A video compression standard for multimedua applications., *Commun ACM*, No. 34, pp. 47–58 (1991).
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).

盛光印刷所