連続メディア多重化ストリーム処理における適応的スケジューリング方式とその性能評価

滝沢泰久[†] 瀧本栄二[†] 大久保英嗣[††]

近年の通信と放送の融合を目指す研究において，連続メディア処理は，MPEG 符号化技術に基づき，関連付けられた複数のメディアを多重化したデータのストリーム処理として行われる．一方，我々は，事前にタスクの最悪実行時間を見積もることが困難である動的な実行環境において，ストリーム処理タスクの時間制約を満たすスケジューリング方式 Adaptive Deadline Modification を提案している．Adaptive Deadline Modification 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た数理モデルに基づいたポリシと，Parallel Distributed Processing モデルと熱力学モデルを用いた適応メカニズムから構成される．Adaptive Deadline Modification は，このポリシとメカニズムにより，従来方式より高いスケジューリング可能性を導くが，想定するストリームデータは単一メディアデータであるため，多重化ストリーム処理の特性を考慮できない．本論文では，多重化ストリーム処理において，高いスケジューリング可能性を導くために，Adaptive Deadline Modification に多重化ストリーム処理を適用可能とする改良について述べる．さらに，改良した Adaptive Deadline Modification の性能評価を示し，その有効性について述べる．

An Adaptive Scheduling Method on Continuous Media Multiplexed Stream Processing and Its Performance Evaluation

Yasuhisa Takizawa,[†] Eiji Takimoto[†] and Eiji Okubo[††]

In recent researches to aim at the fusion of communication and broadcasting, continuous media processing with MPEG encoding technology are multiplexed stream processing on continuous media with related information. On the other hand, we have proposed new scheduling method which is adaptable for stream processing tasks with timing constraints and processing delay. The proposed method consists of scheduling policy based on the model which modifies Linear Bounded Arrival Process, and adaptation mechanism applied Parallel Distributed Processing model and thermodynamics model. The proposed method is constructed on the premise that stream processing is a single continuous media processing. Therefore, the proposed policy can not apply to multiplexed stream processing on continuous media. In this paper, modification of the policy and mechanism for the proposed method in consideration of multiplexed stream processing on continuous media and its performance evaluation are described.

1. はじめに

近年のディジタル放送において，映像や音声の連続メディアおよび各種情報を流通する形式として MPEG 符号化方式[3)~5)] が用いられている．MPEG 符号化方式は，映像と音声を高能率符号化し，さらに符号化されたメディア情報を各種関連情報とともにパケット化する．また，MPEG は，放送分野だけでなく，DVD などのパッケージや映像配信などの通信分野でも多く用いられていることから，今後は通信/放送の垣根を越えて，PC，PDA や携帯電話などにおいて，マルチメディア処理の中核技術になると予想される．

このような MPEG において，動画や音声などの連続メディアは，符号化され，さらにメディア間の関連情報（たとえば，同期情報）とともに多重化されて 1 つのデータパケットを形成する．このような多重化データのストリームから連続メディア処理を行う場合，次のようなタスクセットが想定される（**図 1** 参照）．

- 入力装置から生成される多重化データを個々のメ

† 株式会社 ATR 適応コミュニケーション研究所
ATR Adaptive Communications Research Laboratories
†† 立命館大学情報理工学部
Faculty of Information Science and Engineering, Ritsumeikan University

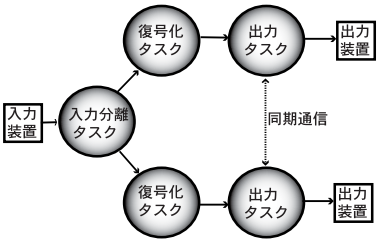

図 1 多重化データストリーム処理のタスクセット
Fig. 1 Task set for multiplexed data stream processing.

ディアに分離するタスク（以降，入力分離タスク）．
- 入力分離タスクよって分離された個々のメディアデータをストリームデータとして復号化するタスク（以降，復号化タスク）．
- 復号化タスクからのデータを出力装置へ出力するタスク（以降，出力タスク）．
- 多重化データに含まれる同期情報に基づいた出力タスク間の同期処理．

したがって，多重化ストリーム処理は，次の特性を持つ．
- 入力分離タスクの処理単位（以降，各タスクの処理単位をメッセージと呼ぶ）である多重化データは，復号化タスクのメッセージ（たとえば，動画の場合，画像 1 フレーム）を複数含むため，入力分離タスクのメッセージ到着レートと復号化タスクのメッセージ到着レートは異なる．また，入力分離タスクのメッセージに含まれる復号化タスクのメッセージ数は一定でないため，復号化タスクのメッセージ到着レートは変動する．
- 入力分離タスクは，複数の復号化タスクのメッセージを生成する．したがって，メッセージを転送する後続タスクが複数存在する．
- 出力タスク間で同期処理が行われるため，異なるメディア間に依存関係が発生する．

一方，我々は，連続メディアストリーム処理において，以下の 5 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方式 Adaptive Deadline Modification（以降 ADM）[9),10)] を提案している．
- システム環境は，最悪実行時間を見積もることが困難な環境である．
- 各タスクのメッセージ処理に必要な CPU 使用時間は，ランダムに変動する．
- 各タスクは，連続メディアメッセージをパイプラインモデルにより転送し，メッセージ受信キューを用いて非同期に処理する．
- 連続メディアメッセージは，その入出力装置から可変量/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．

ADM は，スケジューリングポリシと適応メカニズムから構成される．スケジューリングポリシは，連続メディア資源モデルの Linear Bounded Arrival Process [6)]（以降，LBAP）に VBR（Variable Bit Rate）メッセージ処理のための変更を加えた数理モデルに基づいている．このポリシは，各タスクのメッセージ処理時間が変動する環境で，ストリーム処理タスク間のキューイングメッセージ数を均等にすることにより，単一ストリーム処理において，入出力装置間の処理遅延時間を抑えつつ，その時間制約を満たすポリシ（以降，キューバランシングポリシ）である．適応メカニズムは，Parallel Distributed Processing モデル[2)]（以降 PDP モデル）と熱力学的モデル[1)] を用いて，複数のストリーム処理が混在する環境で，キューバランシングポリシを満たす．ADM は，このポリシとメカニズムによりストリーム処理に高いスケジューリング可能性を導く．しかし，ADM で想定する連続メディアメッセージは，多重化されていない単一連続メディアメッセージであるため，前述の多重化ストリーム処理の特性を考慮できない．

以上のことから，多重化ストリーム処理において高いスケジューリング可能性を導くため，ADM に多重化ストリーム処理を適用可能とする改良を行った．ADM の改良は，前述の 5 つの前提に多重化ストリーム処理の特性を加えた条件で，ストリーム処理の数理モデルを再定義し，この数理モデルからキューバランシングポリシと適応メカニズムを見直すことにより行った．本論文では，改良した ADM の多重化ストリーム処理タスクのスケジューリングポリシと適応メカニズムについて述べる．さらに，多重化ストリーム処理に適用した場合の ADM の性能評価を示し，その有効性について議論する．

以下，2 章で ADM を概説し，3 章では，多重化ストリーム処理のために改良した ADM について述べる．さらに，4 章で ADM の性能評価結果について，5 章で関連研究について述べ，ADM の有効性を議論する．

2. ADM

本章では，ADM を文献 9) に基づき概説する．

2.1 LBAP に基づく従来の処理モデル

従来の方式[7)] 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．したがって，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 message (messages)
C^{max}: maximum processing time for a message (seconds/message)

このような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，LBAP の未処理メッセージ（work ahead message）数は，次のようになる．

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

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．したがって，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．

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

また，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．

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

さらに，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 $D_n^a(m_i)$，論理処理遅延時間 $D_n^l(m_i)$ 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．

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

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

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

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．

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

2.2 ADM のストリーム処理モデル

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 message (messages)
C^{req}: request processing time for a message (seconds/message)

任意の時刻に実測された未処理メッセージ数を実効未処理メッセージ数と呼ぶ．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 $b_n(t)$ と表記する．

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．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．

パイプラインモデルにおいて，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．したがって，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．

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

さらに，メッセージ処理が $1/R^c$ 内で完了すると仮定し，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．

$$f_{n+1}(m_{i-1}) = e_n(m_i) + b_{n+1}(e_n(m_i))/R^c$$

したがって，デッドライン時刻 $d_n(m_i)$ を，次のように定義する．

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

その他の定義は，2.1 節と同様である．

2.3 スケジューリングポリシ

ストリーム処理において，入出力装置間の処理時間を抑えつつ，その時間制約を満たすスケジューリングポリシ（キューバランシングポリシ）は，次のような VBR ストリーム処理モデルの特性[9] に基づく．

[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任意の時間を，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 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．

$$b_{n+1}(e_n(m_i)) \geq \frac{C_n(m_i)}{C_n^{req}} \tag{9}$$

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任意の時間を，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，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．

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

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．

VBR ストリーム処理モデルの特性から，ADM のスケジューリングポリシは，次のようなタスクの優先度変更を行い，タスク間の未処理メッセージ数を均等化する（キューバランシングポリシ）．

- 同一ストリーム処理において，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．
- 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．
- 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．

以上のポリシを，タスクの優先度であるデッドライン時刻に反映させるため，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，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，上記ポリシによる優先度の変更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．

2.4 適応メカニズム

2.3 節のポリシは単一ストリーム処理のポリシである．ADM は，複数のストリーム処理が混在する処理環境を前提としている．したがって，複数のパイプラインのタスク群において 2.3 節の優先度変更ポリシをそれぞれ実施することは，同時に複数の制約を満たす状態を探し出す多重制約問題として考えられる．

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 PDP モデルと熱力学的モデルを組み合わせた適応メカニズム[9] を VBR ストリーム処理環境に適用している．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の出力を入力として受け取る．入力が正の値である場合は，ユニットの状態を高める．負の値である場合は，その状態を低める．さらに，その状態に応じた出力を他のユニットに送る．この相互作用をユニットごとに非同期に繰り返すことにより，ある条件での複数の制約を最大に充足した状態を形成する．

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）に次のように対応付ける．

- 各ユニットの状態に応じた出力（0 から 1 までの連続値）を，各タスクの重要度とする．
- 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．
- 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変更する力とする．すなわち，外部条件は，2.3 節で述べた変更ポリシに基づき，次のような入力として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．
  - 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．

– 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 PDP のネットワークメカニズムにより処理する．ストリーム処理環境において，外部条件はつねに変化する．ネットワークは，この変化に応じて，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外部条件に連続的に動作するようにする．

また，高い充足状態（以降，充足状態）は，文献 9) に示されているように，互いに通信するタスクの重要度が 1 となり，その他の通信しないタスクの重要度が 0 となる状態の近傍にあると想定する．

以下にタスク重要度の更新則を示す．

$$\alpha_n(t+1) = \alpha_n(t) + \begin{cases} (1-\alpha_n(t))net_n(t) & prob(net_n(t)) \geq R \\ \alpha_n(t)net_n(t) & \text{else} \end{cases} \tag{10}$$

$\alpha_n(t)$ はタスク n の時間 t におけるタスクの重要度（$0 \leq \alpha_n(t) \leq 1$），$net_n(t)$ はタスク n の時間 t における総入力，$prob(x)$ は熱力学モデルの熱揺動による振動を擬似するときに用いる確率関数，R は [0,1] での一様乱数である．

タスクへの総入力 $net_n(t)$ は，他のタスクからの入力（他のタスクの出力）の総和と外部条件としての入力により構成される（**図 2** 参照）．タスクへの総入力を以下に示す．

$$net_n(t) = \sum_{j=0, j\neq n}^{N} \omega_{nj}\alpha_j(t) + \sum_{z=0}^{Z} bias_n^z(t) \tag{11}$$

N はタスク数，Z は外部入力数，ω_{nj} はタスク n とタスク j との結合重み，$bias_n^z(t)$ はタスク n の時間 t における外部入力である（Z は，複数の外部からの入力があることを示す）．

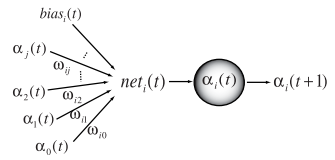

図 2 相互結合ネットワーク
Fig. 2 Inter-connection netowrk.

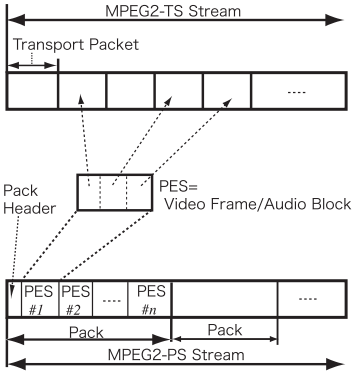

図 3 MPEG2-PS と MPEG2-TS のフォーマット概略
Fig. 3 Outline of MPEG2-PS and MPEG2-TS format.

3. 多重化データストリーム処理を可能とする ADM の改良

MPEG 符号化方式では，複数の連続メディアを符号化し，さらに関連情報とともに多重化したデータを構成する．**図 3** に MPEG2 のデータフォーマットの概略を示す．この図から分かるように，連続メディアは，MPEG2-PS では pack と呼ばれるデータブロックのストリームとして，MPEG2-TS では transport packet のストリームとして構成される．MPEG2-PS における pack は複数のメディア（たとえば，映画において，ある期間に処理する映像，複数言語の音声，複数言語の字幕など）の複数のデータ（動画フレーム，音声ブロック，テキストブロックなど）を含む．pack は入力分離タスクの処理単位（メッセージ）であり，入力分離タスクの処理により，pack から複数の復号化タスクの複数メッセージ（動画フレーム，音声ブロック，テキストブロックなど）が分離生成される．また，pack 内には複数のメディア間の同期情報を含み，この情報に基づき，複数の出力タスク間で同期制御が行われる．MPEG2-TS では，複数の transport packet により，複数の復号化タスクのメッセージを形成する．したがって，連続メディアの多重化ストリーム処理では，LBAP に含まれていない次のような処理が行われる．

- 異なるメッセージ到着レートのタスクによるスト

リーム処理
- 異なるストリーム処理間の同期
- 複数の後続タスク

本章では，上記 3 つの処理に対応する ADM の改良について述べる．

3.1 メッセージ到着レートの相違に基づく改良

ADM で用いている LBAP は，2.1 節と 2.2 節で述べたように，同一ストリーム処理の各タスクは同一のメッセージ到着レート R^c，すなわち同一の処理レートであることを想定している．したがって，数理モデルとして，処理レートが異なるタスクにより構成されるストリーム処理に適用できない．この問題を解決するため，2.2 節のストリーム処理モデルにおける 3 つのパラメータを，各タスクごとに識別して，次のように再定義する．

R_n^a: average message rate for task n (messages/second)

b_n: actual workahead message for task n (messages)

C_n: a message processing time for task n (seconds/message)

すなわち，タスクごとに異なるメッセージ平均到着レート R_n^a，タスクごとの実効未処理メッセージ数 b_n，およびタスクごとのメッセージ処理時間 C_n とする．パックに含まれるメッセージ数は可変であるため，到着レートは変動する．しかし，入出力装置のメッセージ生成/消費レートから各タスクのメッセージ平均到着レート R_n^a は設定可能である．C_n は，メッセージ処理に使用する CPU 時間であるが，CPU 使用時間は未知とする．2.2 節のストリーム処理モデルでは，最悪実行時間より短い任意の CPU 使用時間による予約した場合の特性を示しているが，本論文では CPU 使用時間による定義は用いない．以上のことから，ストリーム処理タスクの数理モデルを変更する．まず，デッドライン時刻は次のようになる．

$$d_n(m_i) = e_n(m_i) + b_{n+1}(e_n(m_i))/R_{n+1}^a \quad (12)$$

その他の属性は 2.2 節と同様である．

この再定義から，VBR ストリーム処理タスクの特性を見直し，スケジューリングポリシを改良する．

特性 2 は予約 CPU 使用時間の定義を用いているため，特性 2 を特性 1 とともに見直す．特性 1 と特性 2 は，ストリーム処理タスクの時間制約を満たすための条件であり，処理実行後の結果から時間制約を満たすように優先度を更新するために用いられている．すなわち，特性 1 は時間制約を満たすように優先度を高めるために用いられ，特性 2 は時間制約を満たすために優先度が過度に高まるのを抑制するのに用いられている．したがって，次のような考え方により，同様の効果を提供する．

- $f_n(m_i) > d_n(m_i)$ の場合
  タスクの時間制約（特性 1）が満たせていないとして，優先度を高める．
- $f_n(m_i) < d_n(m_i)$ の場合
  タスクにおけるメッセージ処理時間に余裕時間があるので，優先度を下げる．

次に，特性 3 を考える．特性 3 はタスクごとにメッセージ到着レートが異なることを考慮する必要がある．まず，文献 9) において特性 3 を導き出したのと同様に，ストリーム処理の終端である出力装置におけるメッセージ到着時刻を考える．ストリーム処理のパイプライン上のタスク数を 0 から n までの $n+1$ とする．出力装置では一定レート R_{n+1}^c でメッセージを消費するとする．また，タスク j において，出力装置における $(i \times R_{n+1}^c + k)$ 番目のメッセージに対応するメッセージを $M_i^{g(j,k)}$ とする．$M_i^{g(j,k)}$ はタスク j における $(i \times R_j^a + g(k,j))$ 番目のメッセージであり，タスクのメッセージ到着レート（処理レート）R_j^a と出力装置のメッセージ消費レート R_{n+1}^c の関係から次のようになる．

$$M_i^{g(j,k)} = m_{i \times R_j^a + g(j,k)}$$
$$g(j,k) = \lceil \frac{R_j^a}{R_{n+1}^c}(k+1) \rceil - 1$$
$$g(0,0) = 0$$
$$0 \le k \le R_j^a - 1$$

出力装置における $M_i^{g(n+1,k)}$ 番目のメッセージ到着時刻は，前方タスク n において，$M_i^{g(n,k)}$ 番目のメッセージの処理完了時刻と一致するので，次のようになる．

$$a_{n+1}(M_i^{g(n+1,k)}) = f_n(M_i^{g(n,k)})$$

処理完了時刻はメッセージ到着時刻に処理遅延時間を加えた時刻であるので，上式は次のようになる．

$$a_{n+1}(M_i^{g(n+1,k)}) = a_n(M_i^{g(n,k)}) + D_n^a(M_i^{g(n,k)})$$

さらに，上の式を n に関して展開し，次の式を得る．

$$a_{n+1}(M_i^{g(n+1,k)}) = a_0(M_i^{g(0,k)}) + \sum_{j=0}^{n} D_j^a(M_i^{g(j,k)})$$

$a_0(M_i^{g(0,k)})$ は，ストリーム処理の先頭タスクにおけるメッセージ到着時刻である．先頭タスクにメッセージを供給するのは入力装置であるので，先頭タスクの

メッセージ到着レートは入力装置のメッセージ生成レートと一致する．したがって，$a_0(M_i^{g(0,k)})$ は，$a_0(m_0)$ から $(1/R_0^c \times M_i^{g(0,k)})$ 経過した時刻であるので，次のようになる．

$$a_0(M_i^{g(0,k)}) = a_0(m_0) + (i + k/R_{n+1}^c)$$

以上から，出力装置における $M_i^{g(n+1,k)}$ 番目のメッセージ到着時刻は，次のようになる．

$$\begin{aligned} &a_{n+1}(M_i^{g(n+1,k)}) \\ &\quad = a_0(m_0) + (i + k/R_{n+1}^c) \\ &\qquad + \sum_{j=0}^{n} D_j^a(M_i^{g(j,k)}) \end{aligned} \tag{13}$$

出力装置の時間制約（レート R_{n+1}^c でメッセージを消費する）を満たすためには，出力装置における最初のメッセージ到着時刻 $a_{n+1}(M_0^0)$（$a_{n+1}(m_0)$ と同じ）と任意のメッセージの到着時刻の関係が次式を満たす必要がある．

$$a_{n+1}(M_i^{g(n+1,k)}) - a_{n+1}(M_0^0) \le i + k/R_{n+1}^c$$

上式に式 (13) を代入すると，次のようになる．

$$\sum_{j=0}^{n}(D_j^a(M_i^{g(j,k)}) - D_j^a(M_0^0)) \le 0$$

さらに，上式の処理遅延時間 $D_j^a(M_i^{g(j,k)})$ を分解すると，次のようになる．

$$\begin{aligned} &\sum_{p=0}^{g(j,k)-1} D_j^a(M_i^{p+1}) - \sum_{p=0}^{g(j,k)-1} D_j^a(M_i^p) \\ &\quad + \sum_{l=0}^{i-1}\left\{ \sum_{p=0}^{R_j^a-1} D_j^a(M_l^{p+1}) \right. \\ &\qquad \left. - \sum_{p=0}^{R_j^a-1} D_j^a(M_l^p) \right\} \le 0 \end{aligned} \tag{14}$$

式 (14) の各メッセージ処理遅延時間の和におけるメッセージ数 $g(j,k)$ と R_j^a は，処理レートに比例した処理メッセージ数である．したがって，次の条件を満たせば，上式を満たす．

$$\begin{aligned} \sum_{l=0}^{i}\left\{ \right. &\sum_{p=0}^{S\times R_j^a-1} D_j^a(M_l^{p+1}) \\ &\left. - \sum_{p=0}^{S\times R_j^a-1} D_j^a(M_l^p) \right\} \le 0 \end{aligned} \tag{15}$$

ただし，S は正の数である．式 (15) から，連続メディア出力装置の時間制約を満たすためには，各タスクにおいてタスクの処理レート R_j^a に比例したメッセージ数の処理遅延時間の総和（以降，メッセージ処理遅延時間和）を均等にすることが求められる．しかし，各メッセージは VBR であることから，メッセージ処理遅延時間和を同一タスク内で均等にすることは困難である．そのため，隣接するタスク間のメッセージ処理遅延時間和の関係を考える．タスク j において k 番目のメッセージから $S \times R_j^a$ 個のメッセージ処理遅延時間和を次のように定義する．

$$Z_j(m_k) = \sum_{p=k}^{k+S\times R_j^a-1} D_j^a(m_p)$$

上記の定義と式 (14) および式 (15) から，次の条件が成立する場合，連続メディア装置の時間制約を満たす．

$$\sum_{j=0}^{n}(Z_j(m_{k+\Delta k}) - Z_j(m_k)) = 0$$

ただし，Δk は正の整数である．さらに，上式をタスク j に関して展開すると次式を得る．

$$\begin{aligned} &\sum_{j=0}^{n-1}(Z_j(m_{k+\Delta k}) - Z_{j+1}(m_k)) \\ &\quad + Z_n(m_{k+\Delta k}) - Z_0(m_k) = 0 \end{aligned} \tag{16}$$

ここで，タスク n におけるメッセージ処理遅延時和 $Z_n(m_{k+\Delta k})$ は，タスク $n+1$ におけるメッセージ処理遅延時間和 $Z_{n+1}(m_k)$ 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，$Z_n(m_{k+\Delta k})$ の比較対象は出力装置におけるメッセージ処理遅延時間和 $Z_{out}(m_k)$ とする．同様に，タスク 0 におけるメッセージ処理遅延時間和 $Z_0(m_k)$ は，入力装置におけるメッセージ処理遅延時間和 $Z_{in}(m_{k+\Delta k})$ と比較する．このことにより，式 (16) は次のようになる．

$$\begin{aligned} &\sum_{j=0}^{n-1}(Z_j(m_{k+\Delta k}) - Z_{j+1}(m_k)) \\ &\quad + Z_{in}(m_{k+\Delta k}) - Z_0(m_k) \\ &\quad + Z_n(m_{k+\Delta k}) - Z_{out}(m_k) = 0 \end{aligned} \tag{17}$$

以上のことから，隣接するタスク間のメッセージ処理遅延時間和を均等にすることにより，連続メディア出力装置の時間制約を満たすことができる．

次に隣接するタスク j と $j+1$ におけるメッセージ処理遅延時間和について考える．タスク j の $k+1$ 番目のメッセージ実効到着時刻 $e_j(m_{k+1})$ において，タスク $j+1$ の未処理メッセージの最後尾は，k 番

目のメッセージである．また，タスク j の未処理メッセージの最後尾は，$(k+1+b_j(e_j(m_k)))$ 番目のメッセージである．それぞれの未処理メッセージの最後尾メッセージの処理遅延時間は，処理レートと未処理メッセージ数に基づき，次のように見積もる．

$$\begin{aligned}\bar{D}_j^a(m_{k+\Delta k}) &= \frac{b_j(e_j(m_{k+1}))}{R_j^a}\\ \bar{D}_{j+1}^a(m_k) &= \frac{b_{j+1}(e_j(m_{k+1}))}{R_{j+1}^a}\\ \Delta k &= k+1+b_j(e_j(m_k))\end{aligned} \tag{18}$$

ただし，$\bar{D}_j^a(m_{k+\Delta k})$ はタスク j における $k+\Delta k$ 番目メッセージのメッセージ処理遅延見積り時間，$\bar{D}_{j+1}^a(m_k)$ はタスク $j+1$ における k 番目メッセージのメッセージ処理遅延見積り時間である．さらに，このメッセージ処理遅延見積り時間から，それぞれのタスクのメッセージ処理遅延時間和を次のように見積もる．

$$\begin{aligned}\bar{Z}_j(m_{k+\Delta k}) &= (S\times R_j^a)\bar{D}_j^a(m_{k+\Delta k})\\ \bar{Z}_{j+1}(m_k) &= (S\times R_{j+1}^a)\bar{D}_{j+1}^a(m_k)\end{aligned} \tag{19}$$

ただし，$\bar{Z}_j(m_{k+\Delta k})$ はタスク j における $k+\Delta k$ 番目メッセージからのメッセージ処理遅延見積り時間和，$\bar{Z}_{j+1}(m_k)$ はタスク $j+1$ における k 番目メッセージからのメッセージ処理遅延見積り時間和である．この見積り時間に関する式 (18) と式 (19) を連続メディア出力装置の時間制約を満たす条件式 (17) に代入すると，次式を得る．

$$b_j(e_j(m_{k+1}))-b_{j+1}(e_j(m_{k+1}))=0$$

したがって，タスク間の処理レートの相違にかかわらず，隣接するタスク間の未処理メッセージ数を均等にすることにより連続メディア出力装置の時間制約を満たす．すなわち，ポリシは，従来の ADM と同様に次のようなタスクの優先度変更を行い，タスク間の未処理メッセージ数を均等化する．

- タスク j の k 番目のメッセージの実効到着時刻における $b_j(e_j(m_k))$ と後続タスク $j+1$ の同じ時刻における $b_{j+1}(e_j(m_k))$ を比較する．前者の値が大きい場合は，タスク j の優先度を上げる．
- 後者の値が大きい場合は，タスク j の優先度を下げる．

3.2 異なるストリーム処理間の同期

動画と音声のようなメディア間に間連がある場合，異なるメディ間で同期処理を行う場合が多く見られる．すなわち，異なるストリーム処理タスク間において同期処理が行われる．同期処理は，処理タスクにブロッキング時間を発生させて，スケジュール可能性を低下させる原因となる．したがって，スケジュール可能性を高めるためには，ブロッキング時間を抑制する制御が必要となる．

一方，3.1 節で示した出力装置の時間制約を満たす条件は，メッセージ処理遅延時間に基づいて，条件付けられている．メッセージ処理遅延時間は，メッセージ到着時刻からメッセージ処理完了時刻までの時間である．すなわち，メッセージ処理遅延時間は，それ自体に，同期処理により発生するブロッキング時間を含むこととなる．したがって，出力装置の時間制約を満たすためには，ブロッキング時間の有無にかかわらず，後続タスクとの未処理メッセージ数を均等にすることが十分条件となる．

3.3 複数の後続タスク

入力分離タスクは，多重化されたストリームから複数の連続メディアメッセージを生成分離し，それぞれの復号化タスクに引き渡す．したがって，入力分離タスクにおいて，未処理メッセージ数の比較対象となる後続タスクが複数存在する．そのため，後続タスクの未処理メッセージ数に依存するデッドライン時刻の定義と適応メカニズムの外部条件の構成を修正する．

まず，デッドライン時刻について述べる．後続タスクが 1 つで処理レートが異なる場合のデッドライン時刻の定義は式 (12) で表される．後続タスクが複数である場合，このようなデッドライン時刻が各後続タスクごとに定義できる．したがって，この複数のデッドライン時刻から，最も早い時刻をデッドライン時刻として，次のように設定する．

$$\begin{aligned}&d_{n[u]}(e_n(m_i)) =\\ &e_n(m_i)+b_{(n+1)[u]}(e_n(m_i))/R^a_{(n+1)[u]}\\ &d_n(e_n(m_i)) =\\ &\min(d_{n[0]}(e_n(m_i)),\cdots,d_{n[U_n-1]}(e_n(m_i))\end{aligned} \tag{20}$$

ただし，$d_{n[u]}(e_n(m_i))$ はタスク n における後続タスク u の未処理メッセージ数から設定されるデッドライン時刻，$b_{(n+1)[u]}(e_n(m_i))$ はタスク n の後続タスク u における i 番目のメッセージ実効到着時刻の未処理メッセージ数，$R^a_{(n+1)[u]}$ はタスク n の後続タスク u の平均処理レート，U_n はタスク n の後続タスク数である．

次に，外部条件について述べる．後続タスクが 1 つで処理レートが異なる場合，3.1 節と 3.2 節で述べたように，変動する処理完了時刻，未処理メッセージ数に応じてネットワークの動作を修正するため，外部条件の入力（以降，外部入力）を次のようにする．

$$bias_n^0(t) =$$

$$\begin{cases} 1-\alpha_n(t) & f_n(m_{i-1}) > d_n(m_{i-1}) \\ 0-\alpha_n(t) & f_n(m_{i-1}) < d_n(m_{i-1}) \\ 0 & \text{otherwise} \end{cases} \quad (21)$$

$$bias_n^1(t) =$$

$$\begin{cases} 1-\alpha_n(t) & b_n(e_n(m_i)) \geq b_{n+1}(e_n(m_i)) \\ 0-\alpha_n(t) & b_n(e_n(m_i)) < b_{n+1}(e_n(m_i)) \end{cases} \quad (22)$$

$bias_n^0(t)$ は特性 1 および特性 2 と同様の効果となる入力であり，$bias_n^1(t)$ は隣接するタスクの未処理メッセージ数を均等にする入力である．

ここで，式 (22) の隣接するタスクの未処理メッセージ数を均等にする入力 $bias_n^1(t)$ について考える．後続タスクが複数ある場合，未処理メッセージ数の比較は，それぞれの後続タスクごとの未処理メッセージ数と比較する必要がある．すなわち，式 (22) の隣接するタスクの未処理メッセージ数を均等にする外部入力は，後続タスクごとに複数発生する．したがって，隣接するタスクの未処理メッセージ数を均等にする外部入力は，後続タスクごとの複数の外部入力から構成され，次のような入力とする．

$$bias_n^1(t) = \sum_{u=0}^{U_n-1} bias_{n[u]}^1(t)$$

$$bias_{n[u]}^1(t) =$$

$$\begin{cases} 1-\alpha_n(t) & b_n(e_n(m_i)) \geq b_{(n+1)[u]}(e_n(m_i)) \\ 0-\alpha_n(t) & b_n(e_n(m_i)) < b_{(n+1)[u]}(e_n(m_i)) \end{cases}$$

ただし，$bias_{n[u]}^1(t)$ はタスク n における後続タスク u との未処理メッセージ数を均等にする外部入力である．

3.4 適応デッドライン時刻の算出

各タスクの重要度は，各タスクのメッセージ実効到着時刻ごとに 3.3 節の外部入力を式 (10) と式 (11) に適用して求める．適応デッドライン時刻（優先度）は，従来の ADM と同様に，このタスク重要度とタスクの時間属性を基に算出する．

したがって，タスク n における i 番目のメッセージの適応デッドライン時刻 $d_n^{adap}(m_i)$ は，次のようになる．

$$v_n(m_i) = h_n - 2h_n\alpha_n(e_n(m_i))$$
$$d_n^{adap}(m_i) = d_n(m_i) + v_n(m_i)$$

ただし，h_n はタスク n における処理遅延許容幅，$v_n(m_i)$ はタスク n における i 番目のメッセージ処理におけるデッドライン時刻を修正する時間（$-h_n \leq v_n(m_i) \leq h_n$）である．

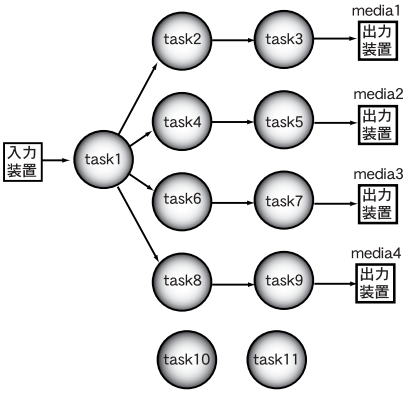

図 4 評価タスクセット
Fig. 4 Task set for evaluation.

4. 性能評価

改良した ADM を，我々が分散 OS の構成法の研究のために開発している分散 OS Solelc[8] 上に実装し，性能評価を行った．本章では，その結果について述べる．

4.1 性能評価環境

ハードウェアの環境としては，次のものを使用した．

- 使用機種　エプソン社製　Endeavor ATX-7000
- プロセッサ　Pentium-S 200 MHz
- キャッシュ　512 KB
- 主記憶　128 MB

今回の評価では，Solelc のディスパッチ機能とメッセージ通信機能のみを使用し，スケジューリング分解能を 1 msec とした．また，コンパイラは gcc（version egcs-2.91.60）を使用した．

4.2 評価方法と評価タスクセット

連続メディアの多重化ストリーム処理を想定し，評価タスクセットを次のように構成する（**図 4** 参照）．

- 周期的に多重化メッセージを生成する入力装置
- 入力装置から多重化メッセージをストリームデータとして受信し，4 つの異なる連続メディアのメッセージを生成する入力分離タスク
- 各連続メディアごとに入力分離タスクからのメッセージをストリームデータとして処理する復号化タスクと出力タスク
- 各連続メディアごとに出力タスクからの連続メディアメッセージを周期的に消費する出力装置
- 周期の異なる 2 つの負荷タスク

入力装置は次のような時間属性を持つ連続メディア多重化メッセージを生成する．

表 1 多重化入力メッセージの構成
Table 1 Construction of multiplex input message.

	average msgs	max msgs	min msgs
media 1	4	6	2
media 2	8	12	4
media 3	2	4	0
media 4	10	14	6

表 2 評価タスクセットの時間属性
Table 2 Timing attributes for taskset.

	$1/R_n^a$	h	C^{ave}	C^{max}	C^{min}
task1	400 ms	40 ms	12 ms	20 ms	4 ms
task2	100 ms	10 ms	13 ms	21 ms	5 ms
task3	100 ms	10 ms	8 ms	12 ms	4 ms
task4	50 ms	5 ms	8 ms	12 ms	4 ms
task5	50 ms	5 ms	5 ms	8 ms	2 ms
task6	200 ms	20 ms	16 ms	26 ms	6 ms
task7	200 ms	20 ms	8 ms	14 ms	2 ms
task8	40 ms	4 ms	6 ms	10 ms	2 ms
task9	40 ms	4 ms	4 ms	7 ms	1 ms
task10	30 ms	–	–	–	–
task11	300 ms	–	–	–	–

h：遅延許容幅 C^{max}：最大実行時間
C^{min}：最小実行時間 C^{ave}：平均実行時間

- 400 msec 周期でメッセージを生成する．
- 生成されるメッセージは 4 つの異なる連続メディア（media1〜4）のメッセージを含み，**表 1** に示されるように，各メディアごとのメッセージ数は最大数と最小数の間でランダムに変動する．

出力装置のメッセージ消費周期は，入力装置の時間属性に基づいて，media1 は 100 msec，media2 は 50 msec，media3 は 200 msec，media4 は 40 msec とする．また，入力装置のメッセージ生成を開始する時刻（以降，入力装置の初期起動時刻）と出力装置のメッセージ消費を開始する時刻（以降，出力装置の初期起動時刻）に時間差を設定する．この時間差を位相と呼ぶ．

入力分離タスクの処理レートは，入力装置のメッセージ生成周期に合わせ，各メディアごとの復号化タスクと出力タスクの処理レートは出力装置のメッセージ消費周期に合わせる（task2〜3 は media1，task4〜5 は media2，task6〜7 は media3，task8〜9 は media4）．また，各タスクの CPU 使用時間は最大時間と最小時間の間をランダムに変動し，適応デッドライン時刻を設定するのに用いる許容幅はレート $1/R_n^a$ の 10% として設定した（**表 2** 参照）．

評価方法は，周期タスクモデルによる方式，LBAP による方式，ADM の 3 つの方式を，上記タスクセットの負荷タスクのスケジューリング成功率と media1〜4 の出力装置の時間制約を満たす割合（以降，ストリームスケジューリング成功率）により比較評価する．また，各タスクの時間属性を表 2 に示す．ただし，タスクの CPU 使用時間および入出力装置間の位相は，スケジューラには未知とする．

4.3 スケジューリング成功率と設定パラメータ

文献 9) におけるスケジューリング成功率の定義は処理数が多いタスクに偏った定義となっている．本論文では，負荷タスクの周期処理および media1〜4 のそれぞれの処理の時間制約の価値は均等と考える．したがって，いずれかのスケジューリング成功率が低下した場合，システム全体としてスケジューリングに失敗していると考える．以上のことから，評価タスクセットのスケジューリング成功率を次のように定義する．

$$SuccessRatio = \min(\frac{Success(task10)}{Arrive(task10)}, \frac{Success(task11)}{Arrive(task11)}, \frac{Success(Str(1))}{Out(Str(1))}, \frac{Success(Str(2))}{Out(Str(2))}, \frac{Success(Str(3))}{Out(Str(3))}, \frac{Success(Str(4))}{Out(Str(4))}) \quad (23)$$

$Success(task10)$，$Success(task11)$ は task10 と task11 のデッドライン時刻までに処理を完了したタスク数，$Arrive(task10)$，$Arrive(task11)$ は task10 と task11 の総到着数，$Success(Str(n))$ は media n の出力装置において処理開始時刻までに出力装置に到着したメッセージ数，$Out(Str(n))$ は media n の出力装置の処理開始回数である．

上記スケジューリング成功率を，前述の周期タスクモデル，LBAP，ADM の 3 つのスケジューリング方式において，負荷と入出力装置間の位相の設定変更により比較する．

負荷は，すべての実行タスクにおける周期（$1/R_n^a$）に占める平均実行時間の割合の総和とした．また，負荷量は，負荷周期タスク（task10 と task11）の実行時間により調整した．

入出力装置間の位相に従い，周期タスクモデルでは，パイプライン上のタスクに入出力装置間の位相を均等に割り振り，入力分離タスク，復号化タスクおよび出力タスクに，入力装置の初期起動時刻からそれぞれ位相の 1/4，2/4，3/4 の時間経過後（以降，初期起動位相）に起動する．また，LBAP においてはパイプライン上の各タスクは入力装置に初期起動時刻と同時に起動されるが，入出力装置間の位相を均等に各タスクの処理遅延時間に配分するため，入出力装置間の位相の 1/4 の時間を，各タスクの平均処理遅延時間として考える．ADM では，LABP と同様に，パイプライン上の各タスクは入力装置に初期起動時刻と同時に起

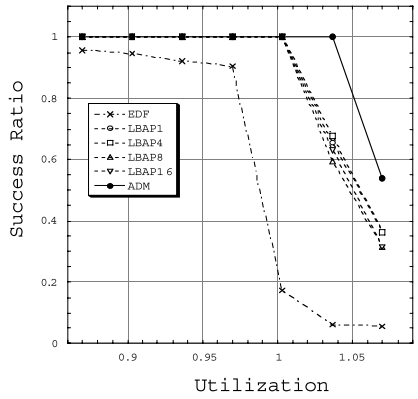

図 5 負荷に応じたスケジューリング成功率
Fig. 5 Scheduling success ratio by utilization.

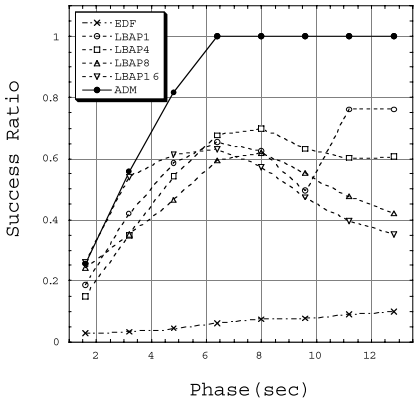

図 6 位相に応じたスケジューリング成功率
Fig. 6 Scheduling success ratio by phase.

動されるが，その他の特別な設定はしない．

キュー容量は，入力装置から出力装置の間の未処理メッセージを保存する領域として用意し，入力装置，タスクおよび出力装置の間にそれぞれ 64 メッセージ数分の容量を設定した．

4.4 多重化ストリームのスケジューリング成功率

本節では，同期処理を含まない多重化ストリーム処理のスケジューリング成功率について述べる．

4.4.1 負荷に応じたスケジューリング成功率

負荷に応じたスケジューリング成功率を**図 5** に示す．入出力装置間の位相は 6.4 msec とした．また，LBAP では，実行状況の変動要因から処理遅延が発生することを考慮して，最大処理遅延時間を平均処理遅延時間に，それぞれ $1/R_n^a$ の 1 倍，4 倍，8 倍，16 倍の時間を加えた時間とする 4 つの場合（それぞれを LBAP1, LBAP4, LBAP8, LBAP16 と呼ぶ）を設けた．評価時間は出力装置の初期起動時刻から 120,000 msec 経過するまでの時間とした．

図 5 から分かるように，平均負荷が 1 以下の場合，LBAP と ADM のスケジューリング成功率は 1 を維持するが，周期タスクモデルでは 1 を維持できない．これは，ストリーム処理の実行時間が実行ごとに変動するため，平均実行時間より長い実行時間のストリーム処理が行われた場合，負荷タスクの実行が遅れ，デッドラインミスを起こすためである．周期タスクモデルではストリーム処理タスクのデッドライン時刻は周期により設定されるため，未処理メッセージ数による処理遅延の可能性を考慮できない．LBAP は，ストリーム処理タスクのデッドライン時刻を位相を考慮した最大処理遅延時間により設定する．また，ADM は実測未処理メッセージ数を基にデッドライン時刻を設定する．したがって，LBAP と ADM は，一時的な過負荷において，出力装置の処理開始に間に合う程度にストリーム処理を遅延させ，負荷タスクのデッドラインを満たす．

一方，平均負荷が 1 を超える場合，LBAP のスケジューリング成功率は急速に減少する．ADM のスケジューリング成功率は，LBAP と比較すると高い値を維持する．LBAP において短い最大処理遅延時間を設定した場合（LBAP1 と LBAP4）では，ストリーム処理タスクに早いデッドライン時刻が設定され，相対的に負荷タスクのデッドライン時刻が遅くなる．そのため，負荷タスクにデッドラインミスが多発し，スケジューリング成功率が劣化する．一方，長い最大処理遅延時間を設定した場合（LBAP8 と LBAP16）では，ストリーム処理タスクに遅いデッドライン時刻が設定され，出力装置の時間制約が満たせなくなる．ADM では，このような偏ったスケジューリングは行われず，多重化ストリーム処理においても高いスケジューリング可能性を示す．

4.4.2 位相に応じたスケジューリング成功率

次に，過負荷時（負荷が 1.037）における位相に応じたスケジューリング成功率を**図 6** に示す．

図 6 から分かるように，ADM は位相を大きくすると，それに比例してスケジューリング成功率が改善され，最終的には 1 に到達する．LBAP では，位相を大きくすると一時的にはスケジューリング成功率が改善されるが，位相量がさらに大きくなるとスケジューリング成功率の改善は鈍化，または劣化し，スケジューリング成功率は 1 には至らない．周期タスクモデルによる方式では，スケジューリング成功率はまったく改善されない．

LBAP による方式における一時的なスケジューリング成功率の改善は，位相量が増えることにより，次の

ような状態となっているためである．

- 出力装置に対して先行処理が多く行われるため，ストリーム処理のスケジューリング成功率が改善される．
- ストリーム処理タスクに比較的遅いデッドライン時刻が設定されることにより，負荷タスクのスケジューリング成功率が改善する．

さらに位相量が増えた場合のスケジューリング成功率の改善が鈍化，または劣化する原因は，ストリーム処理タスクのデッドライン時刻がさらに遅い時刻となり，負荷タスクのスケジューリング成功率は改善されるが，ストリーム処理のスケジューリング成功率が劣化する（特にレートの遅い media 3 の劣化が著しい）ためである．また，LBAP1 では，位相量が 9.6 sec において，著しいスケジューリング成功率の劣化が発生する．これは，先行処理による大量の未処理メッセージ数が一部のタスクに偏った状態で発生し，この状態から一部のストリーム処理タスクにおいてバースト的な未処理メッセージの処理が発生し，負荷タスクのスケジューリング成功率が著しく劣化するためである．他の LBAP による方式では，この現象が発生しないのは，LBAP1 と比較してストリーム処理タスクのデッドライン時刻が遅い時刻に設定されるため，未処理メッセージが徐々に処理されるためである．しかし，この場合は，ストリーム処理のスケジューリング成功率が劣化する．すなわち，いずれの場合も，一部のタスクの時間属性に偏ったスケジューリングとなり，タスクの時間属性に基づいた均等なスケジューリングができていない．

また，周期タスクモデルによる方式では，前節の評価と同様に，ストリーム処理が遅延可能であるにもかかわらず，ストリーム処理タスクを固定の周期で処理を行うため，負荷周期タスクのスケジューリング成功率が改善されない．

ADM では，後続タスクとの未処理メッセージ数を均等にする制御により，LBAP や周期タスクによる方式に見られる偏りの発生が抑えられ，位相量に従いスケジューリング成功率が改善される．以上のことから，多重化ストリーム処理においても，ADM はより少ない位相量（処理遅延時間）で高いスケジューリング可能性を示すことが分かる．

4.5 同期処理をともなう場合のスケジューリング成功率

本節では，同期処理を含む多重化ストリーム処理のスケジューリング成功率について述べる．同期処理は media1 と media2 および media3 と media4 で行われ，それぞれのメディアの出力タスクにおいてセマフォ（test&set）により実現した．同期周期は，media1 および media3 の 20 メッセージごととした．

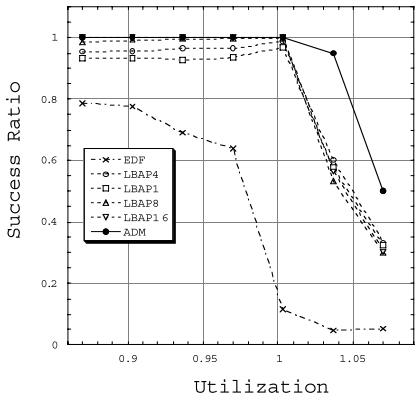

図 7 同期処理をともなう場合の負荷設定に応じたスケジューリング成功率
Fig. 7 Scheduling success ratio by utilization with synchronization control.

4.5.1 負荷に応じたスケジューリング成功率

同期処理をともなう場合の負荷設定に応じたスケジューリング成功率を**図 7** に示す．

図 7 から分かるように，ADM は，同期処理をともなわない場合（図 5 参照）と比較して，負荷が 1.037 において約 5%程度のスケジューリング成功率の劣化が見られるが，ほぼ同等の結果となる．

LBAP による方式は，負荷が 1 以下において，スケジューリング成功率が 1 に到達しない．これは，同期処理により，一部のストリーム処理タスク（特に処理レートの早い media4 を処理する task8 と task9）に偏って未処理メッセージが大量に発生するためである．すなわち，4.4.2 項で述べた位相量 9.6 sec の LBAP1 に見られるスケジューリング成功率の劣化と同様の現象が発生する．負荷が 1 以上になると，ストリーム処理タスクも含め，すべてのタスクの処理が遅延するため，未処理メッセージの偏りは解消する（すべてのストリーム処理タスクに多くの未処理メッセージが発生する）．しかし，4.4.1 項で述べたのと同じ原因によりスケジューリング成功率が劣化する．

周期タスクモデルによる方式は，同期処理をともなわない場合と比較して，より低い負荷状態からスケジューリング成功率が劣化する．これは，ストリーム処理タスクを固定周期で実行するするため，同期処理によるブロッキング時間を考慮できないためである．

また，同期周期が小さい（5 メッセージごとに同期処理）場合は，LBAP に見られる低負荷時でのスケジューリング成功率の劣化が緩和し，同期周期が大き

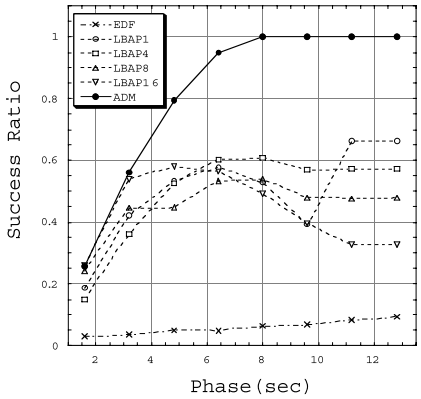

図 8 同期処理を含む場合の位相設定に応じたスケジューリング成功率
Fig. 8 Scheduling success ratio by phase with synchronization control.

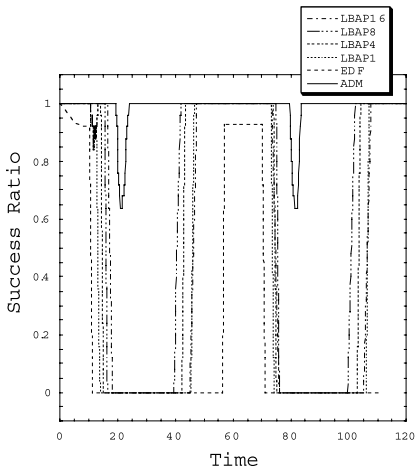

図 9 断続的な過負荷時のスケジューリング成功率の遷移
Fig. 9 Change of scheduling success ratio at intermittent overload.

い（40 メッセージごとに同期処理）場合は，この現象が強く現れる．それ以外は，ほぼ前述と同じ傾向となった．

以上のことから，ADM は，同期処理をともなう多重化ストリーム処理おいても，他の方式より高いスケジューリング可能性を持つといえる．

4.5.2 位相に応じたスケジューリング成功率

同期処理をともなう場合の位相設定に応じたスケジューリング成功率を**図 8** に示す．

図 8 から分かるように，ADM は，同期処理をともなわない場合（図 6 参照）と比較して，ほぼ同じ傾向で位相量に応じてスケジューリング成功率が改善され，1 に到達する．

LBAP による方式は，同期処理をともなわない場合（図 6 参照）と比較して，同様の結果であるが，スケジューリング成功率の最高点が約 10%ほど低下している．これは，レートの早い media4 のスケジューリング成功率が低下していることによる．この低下の原因は，処理レートの早い media4 が，同期処理により処理遅延の影響を受けやすいためである．

周期タスクモデルによる方式は，同期処理の有無にかかわらず，位相量が増えてもスケジューリング成功率は改善されない．これは，ストリーム処理タスクを周期起動するため，位相量にともなう先行処理量（未処理メッセージ数）を考慮できないことが原因である．

ADM では，後続タスクとの未処理メッセージ数を均等にする制御により，LBAP による方式に見られる同期処理にともなう偏りの発生が抑えられ，位相量に従いスケジューリング成功率が改善される．以上のことから，同期処理をともなう多重化ストリーム処理においても，ADM はより少ない位相量（処理遅延時間）で高いスケジューリング可能性を示すことが分かる．

4.5.3 断続的過負荷時のスケジューリング成功率

最後に，連続的な過負荷の期間が断続的に発生する場合のスケジューリング成功率の時間的推移を**図 9** に示す．スケジューリング成功率は，ストリーム処理と負荷タスクのそれぞれ最近 10 回の実行結果を式 (23) に適用した値である．連続的過負荷の期間は，計測開始後 10 秒後に発生し，その長さは 10 秒間，負荷は 1.4 とした．その後，60 秒周期で連続的過負荷の期間を発生させた．それ以外の期間の負荷は 0.9 とした．したがって，平均負荷は 0.98 である．また，入出力装置間の位相は周期の 9.6 sec，同期周期は，media1 および media3 の 5 メッセージごと，キュー容量は 64 メッセージとした．

図 9 から分かるように，周期タスクモデルによる方式は，連続的過負荷が発生すると，ただちにスケジューリング成功率が劣化し，スケジューリング成功率が回復するまでに約 47 sec かかる．LBAP による方式は，周期タスクモデルによる方式よりも劣化は遅く回復が早いが，約 22〜31 sec の期間がスケジューリング成功率が 0 となる．このような両方式のスケジューリング成功率の推移は，連続的過負荷の期間に応じて 60 秒周期で繰り返す．すなわち，平均負荷が 0.98 であるにもかかわらず，周期タスクモデルでは計測全時間の約 8 割，LBAP による方式では計測全時間の 4〜5 割が，スケジューリング成功率 0 となる．

一方，ADM は，周期タスクモデルや LBAP による方式と比較して，スケジューリング成功率が劣化する時刻が遅く，回復する時刻も非常に早い．また，ス

ケジューリング成功率の劣化も少ない．したがって，計測全時間の約 9 割においてスケジューリング成功率が 1 を維持し，劣化量も少ない．すなわち，連続的な過負荷の期間が断続的に発生する処理環境において，ADM は，周期タスクモデルによる方式や LBAP による方式より明白な有効性が認められる．

以上のことから，ADM は，同期処理をともなう多重化ストリーム処理において，従来の方式と比較すると，次のような特性を持つ．

- 初期起動位相や最大処理遅延時間のような処理環境に依存した設定は必要としない．すなわち，自律的に処理環境に適応する．
- 過負荷時でも高いスケジューリング可能性を維持する．このことから，CPU 利用率を高めることが可能である．
- 少ない入出力装置間の位相でスケジューリング成功率が改善される．このことから，入出力装置間において，少ない処理遅延時間でスケジューリングすることが可能である．
- 連続的な過負荷の期間が断続的に発生する処理環境において，高いスケジューリング可能性を示す．

5. 関 連 研 究

実時間システムにおける最悪実行時間に基づく方式の場合，ストリーム処理タスクに割り当てる CPU 利用率が静的に固定されるため，開放型システムでは CPU を効率的に使用できない．このような問題点を解決するため，ストリーム処理タスクの CPU 利用率の割当て方法においていくつかの研究が行われている．

その方法の 1 つとして，発見的手法を用いる研究[6]がある．しかし，これらの手法は，事前処理を行ったうえで CPU 利用率を割り当てるために，動的な環境には向かない．

動的な環境を想定して，その変動をフィードバックして CPU 利用率の割当てを更新する動的手法として BERT [11] や Real-Rate [12] がある．BERT は，計測した過去の実行時間をフィードバックして，動的に優先度（デッドライン）を生成する．しかし，フィードバックの対象は実行時間だけであり，ストリーム処理で用いられるタスク間の中間バッファは考慮していない．そのため，必要以上にストリーム処理タスクに CPU 利用率が割り当てられる．

一方，Real-Rate は，実行時間と中間バッファの変動をフィードバックして CPU 利用率（周期内の CPU 使用時間）を動的に更新する．Real-Rate は，ADM に類似している．Real-Rate では中間バッファの最適な残量をバッファサイズの 1/2 としている．これは，パイプラインのエンドーエンドのストリーム処理の遅延時間がバッファサイズに依存することを意味する．すなわち，バッファサイズが大きくなるとストリーム処理のエンドーエンドの遅延時間が増大する．ADM では，エンドーエンドの処理遅延を可能な限り小さくするように適応デッドライン時間を算出しているため，Real-Rate に見られる問題は発生しない．

以上のことから，関連研究と比較して ADM は優位性が高いと考えられる．

6. お わ り に

本論文では，同期処理をともなう多重化ストリーム処理における ADM のスケジューリングポリシと適応メカニズムを述べた．さらに，ADM の性能評価から，ADM は従来方式と比較して，少ない位相量でスケジューリング可能性を自律的に高めること，および連続的な過負荷期間が断続して発生する処理環境でそのスケジューリング可能性が優位性を持つことを示した．

参 考 文 献

1) Yavatkar, R. and Lakshman, K.: Optimization by Simulated Anneling, *Science*, Vol.220, pp.671–680 (1983).
2) Rumelhart, D.E., McCelland, J.L. and the PDP Research Group: *PARALLEL DISTRIBUTED PROCESSING*, The MIT Press (1986).
3) LeGall, D.: A video compression standard for multimedia applications, *Comm. ACM*, Vol.34, No.4, pp.47–58 (1991).
4) ISO/IEC JTC 1/SC29/N071, Coding of moving pictures and associated audio — for digital storage media at up to about 1.5 Mbits/s: Part 1: Systems, Part 2: video (1992).
5) Krunz, M., Sass, R. and Hughes, H.: Statistical characteristics and multiplexing of MPEG streams, *Proc. IEEE INFOCOM '95 Conference*, pp.455–462 (1995).
6) Anderson, D.P.: Metascheduling for continous media, *Trans Comput Syst ACM*, Vol.11, No.3, pp.226–252 (1993).
7) Govindan, R. and Anderson, D.P.: Scheduling and IPC mechanisms for continuous media, *Proc. 13th ACM on Operating Systems Principles*, pp.68–80 (1991).
8) 芝 公仁，大久保英嗣：分散オペレーティングシステム Solelc の設計と実装，電子情報通信学会論文誌 D-I，Vol.J84-D-I，No.6，pp.617–626

(2001).
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とその性能評価，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).
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) Bavier, A., Peterson, L. and Moseberger, D.: BERT: A scheduler for best effort and realtime tasks, Techical Report TR-587-98, Princeton University (1998).
12) Steere, D.C., Goel, A., Gruenberg, J., McNamee, D., Pu, C. and Walpole, J.: A Feedback-driven Proportion Allocator for Real-Rate Scheduling, *Operating Systems Design and Implementation* (1999).

(平成 16 年 8 月 13 日受付)
(平成 16 年 10 月 4 日再受付)
(平成 16 年 10 月 18 日採録)

滝沢　泰久（正会員）

1983 年京都工芸繊維大学工芸学部機械工学科卒業．同年日本ユニシス（株）入社．1990 年住友金属工業（株）入社．1998 年 ATR 環境適応研究所出向．2002 年 ATR 適応コミュニケーション研究所客員研究員．現在，適応的資源管理方式等の研究に従事．工学博士．電子情報通信学会，IEEE 各会員．

瀧本　栄二（正会員）

1999 年立命館大学理工学部情報学科卒業．2001 年同大学大学院博士課程前期課程修了．2004 年同大学院博士課程後期課程単位取得退学．2004 年 ATR 適応コミュニケーション研究所客員研究員．オペレーティングシステム等システムソフトウェアの研究開発に従事．

大久保英嗣（正会員）

1977 年北海道大学大学院情報工学研究科情報工学修士課程修了．同年（株）日立製作所ソフトウェア工場入所．1979 年京都大学工学部情報工学科助手，1985 年同講師，1987 年同助教授，1991 年立命館大学理工学部情報学科教授となり，現在に至る．工学博士．オペレーティングシステム，データベースシステム，分散システム，実時間システム等の研究に従事．電子情報通信学会，日本ソフトウェア科学会，システム制御情報学会，ACM，IEEE-CS 各会員．