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

滝沢泰久[†] 芝 公仁[††] 大久保英嗣[†††]

連続メディアを扱うアプリケーションを周期タスクとしてスケジュールする場合，その時間制約とタスク間の依存関係を満たすためにリアルタイムスケジューラとリアルタイム同期プロトコルとを組み合わせる方式が有効とされている．しかし，この組合せ方式は，予測が困難な変動する環境には適用できない．また，リアルタイム同期プロトコルは密結合による共有メモリモデルを前提としているため，疎結合モデルによるメッセージ通信にはそのまま利用できない．本論文では，単一プロセッサ上の任意の周期タスク間でメッセージ通信する動的な環境において，Parallel Distributed Processing モデルと熱力学的モデルを用いることにより，その時間制約と通信依存関係に適応するタスクスケジューリングポリシを提案し，その性能評価について述べる．

An Adaptable Task Scheduling for Timing Constraints and Communication Delays on Continuous Media Processing

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

Combination schemes of real-time scheduler and synchronization protocols are useful for scheduling periodic tasks which manipulate continuous media. However, these schemes can not be applied to dynamic environments. Furthermore, these schemes can not be applied to the message passing, because shared memories can not be used. In this paper, a new scheduling policy based on Parallel Distributed Processing model and thermodynamics model, which is adaptable for tasks with timing constraints and communication constraints is proposed, and its performance evaluation is presented.

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，その処理するメディアの特性上，ある時間周期で起動され，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，周期タスクとしての時間制約を持つ．マルチメディアシステムでは，このような時間制約を持つ周期タスクが複数実行され，かつ互いに通信を行う場合が多い．そのため，PC や WS 上で周期タスクをスケジューリングする場合，リアルタイムスケジューラとリアルタイム同期プロトコルとを組み合わせる方式が多く採用されている．この組合せ方式は，起動されるタスクとそのタスクの時間制約および通信関係を事前に正確に知ることで，十分に機能する．また，タスクが入出力処理を行わないことを前提としている．したがって，任意のタスクが実行され，タスク間の通信や任意のタスクで入出力処理が行われる動的な実行環境では，この組合せ方式は十分に機能しない．

一方，PC や WS 上のオペレーティングシステム（以降，OS）は，必要最小限のサービスを提供するマイクロカーネルとマイクロカーネル上で動作する各種システムサービスを提供するサーバプロセス群により構成されている．このような構成では，アプリケーションのシステムサービス要求はシステムサーバプロセスとの通信に置き換えられる．また，アプリケーションのタスク間においても同様の方式が多く利用されてい

† 株式会社 ATR 環境適応通信研究所
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 University
††† 立命館大学理工学部情報学科
Department of Computer Science, Facutly of Science and Engineering, Ritsumeikan University

る．すなわち，タスク間のデータ交換方式は，分散処理環境を想定した疎結合モデルによるメッセージ通信方式が大部分を占める．したがって，このようなデータ交換環境では，リアルタイム同期プロトコルで前提としている密結合（共有メモリ）モデルによる排他制御方式はそのまま利用することができない．

本論文では，連続メディア処理において以上の問題点を解決するために，

- スケジュール環境は，事前に予測が困難な動的な環境である，
- スケジュール問題は，タスクの時間制約と通信依存関係から構成される，
- いくつかのタスクは入出力処理を行う，
- タスク間のデータ交換方式は，疎結合モデルによるメッセージ通信方式である，

の 4 つを前提として，Parallel Distributed Processing モデル（以降，PDP モデル）[6] と熱力学的モデル [3] を用いた適応機能を従来の組合せ方式に付加することにより，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 Adaptive Deadline Modification（以降，ADM）を提案する．また，その ADM の性能評価についても述べる．

以下，2 章で関連研究とそれらの研究における問題点を述べ，3 章でその問題を解決する新たなポリシ ADM を提案する．次に，4 章でその具体的なアルゴリズムを述べる．さらに，5 章では，ADM の性能評価について述べ，その有効性を議論する．

2. 関 連 研 究

2.1 リアルタイムスケジューラ

リアルタイムシステムでは，タスクは時間制約を持つ．リアルタイムスケジューラは，時間制約を持つタスク（以降，リアルタイムタスク）をその時間制約に基づいて実行順を決める．周期タスクに適しているリアルタイムスケジューラには，Earliest Deadline First（以降，EDF）[2] と Rate Monotonic（以降，RM）[2] がある．

EDF は，実行可能な（以降，到着している）タスク集合において処理完了の制限時刻（以降，デッドライン）の早い順に優先度を高くする．EDF は最適なスケジューリングアルゴリズムであり，また，タスクの到着によりタスクの優先度を再設定することから，動的なスケジューリングアルゴリズムでもある．しかし，タスクは互いに独立であることを前提としている．

一方，RM は，周期タスク集合に対してその実行周期が短いタスクほど優先度を高くする．RM は，優先度をタスクの周期に固定するため静的なスケジューリングアルゴリズムであるが，静的なスケジューラ内では最適なスケジューリングアルゴリズムである．RM もまた EDF と同様に，タスクは互いに独立であることを前提としている．

2.2 リアルタイム同期プロトコル

リアルタイムスケジューラは，互いに独立した周期タスクの時間制約からタスクの実行順を決定する．周期タスク間で資源を共有する場合，時間制約に加えて資源制約も考慮する必要がある．資源の共有において，セマフォなどによる排他制御が行われる．この排他制御はタスク間に依存関係を発生させ，優先度逆転（priority inversion）を起こし，タスクの時間制約が満たされなくなる原因となる．リアルタイム同期プロトコルは，この優先度逆転を抑制する手法である．これまでに，Priority Inheritance Protocol（以降，PIP）[1]，Priority Ceiling Protocol（以降，PCP）[1] や Stack Resource Pocily（以降，SRP）[9] などが同期プロトコルとして提案されている．

PIP は，優先度の高いタスクが資源待ちになった場合，当該資源を使用中のタスクの優先度を資源待ちタスクの中で最も高い優先度まで上げる．これにより優先度逆転の発生を抑制するが，デッドロックと連鎖ブロックを発生させる可能性がある．

PCP は，PIP の問題点を解決するためにセマフォに関する優先度シーリングを使用する．しかし，PCP においても，シーリングブロックと呼ばれる不要なブロックを発生させる問題点を持つ．また，タスクの依存関係から事前にセマフォの優先度シーリングを設定する必要があり，予測困難な動的な環境では使用できない．

SRP は，事前に設定したシーリング値と横取りレベルにより PCP で発生するシーリングブロックを回避し，また横取りによるコンテキストスイッチ回数を減少させる．しかし，PCP と同様にシーリング値の事前設定が必要であるため，予測が困難な動的な環境では使用できない．

また，いずれのリアルタイム同期プロトコルも入出力処理は行わないことを前提としている．

2.3 メッセージ通信を考慮したスケジューラ

PC や WS 上の OS は，その多くがマイクロカーネルに基づく構成となっている．このような構成では，資源要求方式やタスク間のデータ交換方式はメッセージ通信に基づくため，周期タスクの時間制約に加えメッセージ通信の依存関係を考慮したスケジューラが必要となる．このようなスケジューラの研究として，Cheng

らの研究[10]やNataleらの研究[11]などある．Chengらの手法は，周期タスクの間の通信依存関係を考慮してスケジュール可能なタスクの実行開始時間を算出するアルゴリズムである．一方，Nataleらの手法は，周期タスク間の通信によるジッタを最小化するアルゴリズムである．しかし，どちらの手法も事前にタスクの時間制約と通信依存関係を正確に知る必要があり，動的な環境には使用できない．

2.4 関連研究における方式の限界

以上の関連研究は，タスクの時間制約および依存関係を事前に正確に知りうることを前提としている．しかし，連続メディアを処理するマルチメディアの実環境ではこのような前提が成立する可能性は低い．このような前提がない予測が困難な動的な環境でのスケジューリングポリシの研究として，AQUAのRAP（Rate-based Adjustable Priority）方式[12]，Lockeのbest-effort法[13]，Buttazzoらの RED（Robust Earliest Dealine）法[14]などがある．しかし，これらはタスクの時間制約のみを考慮し，資源制約，通信依存関係および入出力処理は考慮されていない．

以上のことから，次のことがいえる．リアルタイムスケジューラとリアルタイム同期プロトコルを組み合わせる方式は，複数の制約を処理するために，スケジュール問題を組合せ問題として，総当たり的に処理する．しかし，スケジュール問題の変動には対応できない．一方，動的な環境を想定したスケジューラは，発見的技法を用いて，スケジュール問題の変動に対応する．しかし，複数の制約を扱うことが困難である．すなわち，スケジュール問題が変動し，かつその問題が複数の制約により構成される場合，これまでの研究による方式では処理することができない．動的な環境において周期タスクの時間制約と通信依存関係を満たす問題は，上記の問題に相当する．マルチメディア環境においては，このような問題を処理できることが，強く求められる要件である．

3. 提案スケジューリングポリシ ADM

提案スケジューリングポリシ ADMは，動的な環境において周期タスクの時間制約と通信依存関係を満たす問題を，次の方針で処理する．

- 対象問題は，複数のタスクの時間制約と通信依存関係から構成されている多重制約問題としてとらえ，認知科学におけるPDPモデルを適用し，処理する．
- 対象問題は変動することことから，この変動に追随するために熱力学的モデルを適用する．これにより，変動する問題に連続的にPDPモデルを動作させる．
- 熱力学的モデルの制御を容易にするため，問題に対するいくつかの解の近傍を想定する．

以下，本章ではADMのポリシについて説明する．

3.1 周期タスクモデル

連続メディア処理における周期タスクモデルを次のように定義する．

(1) 横取り可能

横取り可能なタスクとは，その実行中に，より高い優先度を持つタスクが到着した場合，実行権をより高い優先度を持つタスクに引き渡すことが可能なタスクである．

(2) 時間制約

連続メディアを処理する周期タスクは，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る．したがって，従来の周期タスクモデルにデッドラインの許容遅延幅を追加し，各時間属性を**図1**のように定める．

(3) メッセージ通信によるデータ交換

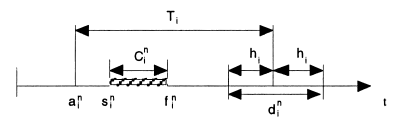
マイクロカーネルに基づく構成において，アプリケーションの各種資源要求呼び出しは資源を管理するサーバプロセスへのメッセージ通信に置き換えられる．また，アプリケーションタスク間のデータ交換も同様の手法を多く利用している．このことから，タスク間のデータ交換はメッセージ通信により実施されると考える．

(4) 入出力処理

周期タスクは，入出力処理を行うために実行権を自ら解放する．入出力処理は入出力デバイスにより処理され，周期タスクはその処理が完了するまで処理待ち状態となる．

(5) ブロッキング時間

周期タスクが到着し完了するまでの期間で，自分より優先度の低いタスクの実行時間とアイドル時間の和をブロッキング時間とする．これは，タスク間通信や入



T_i：タスクiの基準周期
h_i：タスクiの許容遅延時間
C_i^n：タスクiのn周期目のCPU使用時間
a_i^n：タスクiのn周期目到着時刻
s_i^n：タスクiのn周期目実行開始時刻
f_i^n：タスクiのn周期目実行終了時刻
d_i^n：タスクiのn周期目適応デッドライン

図1 周期タスクの時間制約
Fig. 1 Timing constraints for aperiodic task.

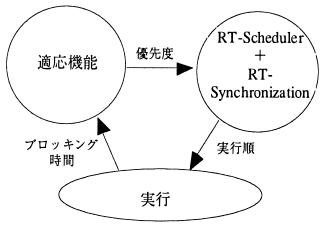


図 2 提案ポリシ ADM の構成
Fig. 2 Configuration for ADM.

出力処理における資源の競合により発生する．

3.2 ADM の構成

ADM は，従来のリアルタイムスケジューラとリアルタイム同期プロトコルの組合せ方式（以降，RT スケジュール機能）に変動環境に対応する適応機能を付加した構成となる（**図 2**）．

適応機能は，資源競合を可能な限り抑制するタスク優先度をタスク到着時（周期開始時）に算出する．一方，RT スケジュール機能は算出された優先度を基に実行順を決め，周期開始後に資源競合が発生した場合，それによるブロッキング時間を抑制する．

3.3 RT スケジュール機能

リアルタイムスケジューラは，高いスケジュール可能性を持ち，また最適なスケジューラである EDF を用いる．

また，リアルタイム同期プロトコルは事前情報が不要な PIP を用いる．しかし，前述のようにリアルタイム同期プロトコルは，メッセージ通信によるデータ交換環境にそのまま適用できない．そこで，RT-Mach [5] で実現されている Real-Time IPC（以降，RT-IPC）[7] と同等の機能を実装して，これを用いる．RT-IPC は，メッセージ通信機能に PIP 機能を追加し，メッセージ通信時の資源競合におけるブロッキング時間を減少させる．

EDF におけるスケジュール可能性判定式は，次式となる．

$$\sum_{k=1}^{i} \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq 1 \qquad (1)$$

ただし，タスク k はタスク i よりも優先度が高いタスクとし，B_i はタスク i の最悪ブロッキング時間，C_k はタスク k の CPU 使用時間，T_k はタスク k の周期，T_i はタスク i の周期である．この式から分かるように，ブロッキング時間を減少させるとスケジュール可能性が高まる．

3.4 適 応 機 能

メッセージ通信によるデータ交換環境において，RT スケジュール機能により発生する資源競合の状況は，メッセージ通信時のブロッキング時間として現れる．適応機能では，タスクの実行により計測されたこのブロッキング時間とタスク間の通信関係から，RT スケジュール機能により発生しうる資源競合のブロッキング時間を抑制するように，タスク優先度の修正を時間制約の範囲内で行う．これにより，高いスケジュール可能性を導く．

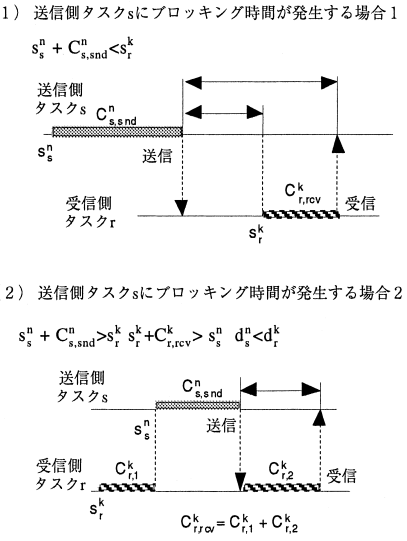


図 3 送信側タスクのブロッキング時間
Fig. 3 Blocking time for sender task.

(1) タスク間通信におけるブロッキング時間

n 周期目の送信タスク s と k 周期目の受信タスク r の 2 つのタスクが通信する場合のブロッキング時間について考える．送信側タスク s の送信タイミング，および受信側タスク r の受信タイミングは任意とする．受信側タスク r の空きバッファ不足などにより送信側タスク s にブロッキング時間が発生する状態を**図 3** に示す．図 3 の（1）は，送信側タスク s の送信時刻が受信側タスク r の実行開始時刻より早い場合を表す．この場合において，送信側タスク s は，送信時刻（$s_s^n + C_{s,snd}^n$）から受信側タスク r の受信時刻（$s_r^k + C_{r,rcv}^k$）までの期間待ち状態となるが，送信側タスク s と受信側タスク r の優先度の関係からブロッキング時間は式 (2) の上段および中段により求められる．一方，図 3 の（2）は，送信側タスク s の送信時刻が受信側タスク r の実行開始時刻より遅い場合を表す．この場合において，ブロッキング時間は，送信側タスク s の優先度が受信側タスク r の優先度より高くかつ送信側タスク s の実行開始時刻が受信側タスク r の受信時刻より早い条件で発生する．また，その

期間は，送信時刻（$s_s^n + C_{s,snd}^n$）から受信側タスクの受信時刻（$s_r^k + C_{s,snd}^n + C_{r,rcv}^k$）までであり，式 (2) の下段により求められる．

$$B_s^n = \begin{cases} (s_r^k + C_{r,rcv}^k) - (s_s^n + C_{s,snd}^n) \\ \quad s_s^n + C_{s,snd}^n < s_r^k, d_s^n < d_r^k \\ s_r^k - (s_s^n + C_{s,snd}^n) \\ \quad s_s^n + C_{s,snd}^n < s_r^k, d_s^n \geq d_r^k \\ (s_r^k + C_{r,rcv}^k) - s_s^n \\ \quad s_s^n + C_{s,snd}^n > s_r^k, \\ \quad\quad s_r^k + C_{r,rcv}^k > s_s^n, d_s^n < d_r^k \end{cases} \quad (2)$$

ただし，B_s^n は送信側タスク s の n 周期目のブロッキング時間，s_s^n は送信側タスク s の n 周期目での実行開始時刻，s_r^k は受信側タスク r の k 周期目での実行開始時刻，$C_{s,snd}^n$ は送信側タスク s の n 周期目での送信開始時刻までの実行時間，$C_{r,rcv}^k$ は受信側タスク r の k 周期目での受信開始時刻までの実行時間，d_r^k は受信側タスク r の k 周期目のデッドライン時刻，d_s^n は送信側タスク s の n 周期目のデッドライン時刻である．

一方，受信側タスク r にブロッキング時間が発生する状態を**図 4** に示す．送信側タスク s のブロッキング時間と同様に場合分けして考えると，図 4 の（1）の受信側タスク r の受信時刻が送信側タスク s の実行開始時刻より早い場合，送信側タスク s と受信側タスク r の優先度の関係から，ブロッキング時間は式 (3) の上段および中段により求められる．また，図 4 の（2）の受信側タスク r の受信時刻が送信側タスク s の実行開始時刻より遅い場合，ブロッキング時間は，受信側タスク r の優先度が送信側タスク s の優先度より高くかつ受信側タスク r の実行開始時刻が送信側タスク s の送信時刻より早い条件で発生し，その時間は，式 (3) の下段により求められる．

$$B_r^k = \begin{cases} (s_s^n + C_{s,snd}^n) - (s_r^k + C_{r,rcv}^k) \\ \quad s_r^k + C_{r,rcv}^k < s_s^n, d_r^k < d_s^n \\ s_s^n - (s_r^k + C_{r,rcv}^k) \\ \quad s_r^k + C_{r,rcv}^k < s_s^n, d_r^k \geq d_s^n \\ (s_s^n + C_{s,snd}^n) - s_r^k \\ \quad s_r^k + C_{r,rcv}^k > s_s^n, \\ \quad\quad s_s^n + C_{s,snd}^n > s_r^k, d_r^k < d_s^n \end{cases} \quad (3)$$

ただし，B_r^k は受信側タスク r の k 周期目のブロッキング時間である．

いずれの場合も，ブロッキング時間はタスクの実行開始時刻，送信開始時刻および受信開始時刻に依存し，式 (2)，(3) から次のように考える．

- 双方の実行開始時刻を近付けるとブロッキング時間が小さくなる．
- ブロッキング時間が発生しているタスクにおいて，実行開始時刻が相対的に遅くなるとブロッキング時間が減少する．
- ブロッキング時間が発生していないタスクにおいて，実行開始時刻が相対的に早くなると，通信相手側タスクのブロッキング時間が減少する．

（2） ブロッキング時間を減少させるデッドライン

前述のブロッキング時間は，式 (2)，(3) より，通信するタスクの実行開始時刻に依存する．一方，EDF では，デッドラインが早いタスクほど優先度が高い．そのため，デッドラインが早い（優先度が高い）タスクは実行開始時刻が早まり，デッドラインが遅い（優先度が低い）タスクは実行開始時刻が遅くなる．すなわち，デッドラインによりタスクの実行開始時刻を制御できる．そこで，タスク実行により計測されたブロッキング時間に基づき，(1) で述べたように実行開始時刻を制御するため，時間制約内でデッドラインを修正する．この修正されたデッドラインを適応デッドラインと呼び，次のように，実行–その結果から修正–再実行を繰り返し，変動するブロッキング時間に対応してタスク到着時に算出する．

- 最初の周期では，ブロッキング時間は不明であるため，互いに通信するタスクの実行開始時刻を時間制約範囲内で，可能な限り近付ける適応デッドラインを算出する．
- 2 周期目以降は，前周期の実行の結果生じたブロッ

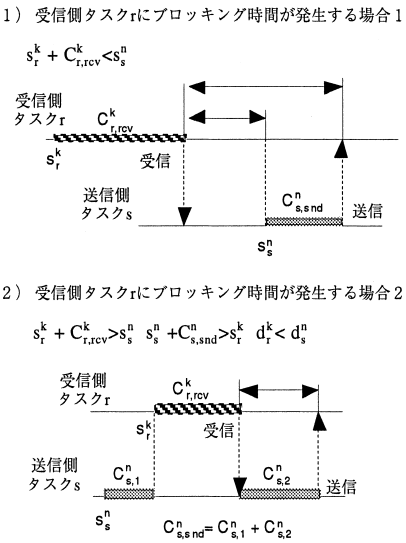


図 4 受信側タスクのブロッキング時間
Fig. 4 Blocking time for receiver task.

キング時間に応じて適応デッドラインを修正算出する．すなわち，ブロッキング時間がある場合は，相対的に開始時刻が早いと判断し，実行開始時刻を遅らす適応デッドラインを算出する．ブロッキング時間がない場合は，相対的に開始時刻が遅いと判断し，実行開始時刻を早くする適応デッドラインを算出する．

以上により，算出された適応デッドラインはブロッキング時間を減少させ，スケジュール可能性を高める．

(3) PDP モデルの応用

以上の処理は 2 つのタスクにおける処理であるが，実際の実行環境ではこのような通信関係が複数混在する．すなわち，扱う問題は，2 つのタスクの通信関係が多重にからみ合う同時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 PDP モデルを応用する．PDP モデルは，相互に関連する複数の制約に対して充足度の高い状態を算出する処理モデルである．この PDP モデルを，タスク間の通信関係と計測されたブロッキング時間から次のような制約に対して動作させる．

- 通信するタスクどうしは協調して，重要度を上げ下げする．
- 通信しないタスクどうしは抑制して，一方が重要度が高い場合は低く，一方が重要度が低い場合は高くする．
- ブロッキング時間がある場合は，重要度を下げる．
- ブロッキング時間がない場合は，重要度を上げる．

これにより，上記制約を充足する各タスクの重要度を探し出し，この重要度と時間制約から適応デッドラインを算出する．

(4) 熱力学モデルの付加

PDP モデルは，初期状態に依存してある充足状態（状態とは各タスクの重要度により表される）に静止（収束）する特徴を持っている．一方，動的なタスク実行環境は，複数の不特定なタスクにより構成され（事前にタスクの特性は知りえない），タスクのブロッキング時間はつねに短い時間で変動する．すなわち，対象問題は短期間に連続的に変動することから，

- 事前情報や比較的長い時間を必要する機能（たとえば学習など）は用いない，
- ブロッキング時間の変化に応じて，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短期間に移動する，

メカニズムが必要となる．このため，PDP モデルに熱力学モデルを付加し，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，充足度の高い状態から遠く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，充足度の高い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 モデルを変動する環境に連続的に動作させる．したがって，ADM は変動するブロッキング時間に適応し，そのブロッキング時間を減少させるタスクの適応デッドラインを探し続けることが可能となる．

4. ADM のアルゴリズム

4.1 相互結合ネットワーク

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（**図 5**）．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ごと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，各ニットで非同期に相互作用の繰返し処理を行う．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，複数のタスクの通信依存関係を充足する適応デッドラインを算出するために，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．

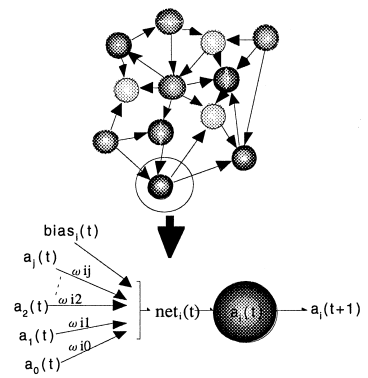


図 5 相互結合ネットワークとユニットの入出力
Fig. 5 Inter-connected network and inpout/output for processing unit.

- 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の最短時刻に，最低値（0）を適応デッドラインの最長時刻に対応づける．これにより，重要度から適応デッドラインを算出する．
- 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．正の重みを持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，負の重みを持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．
- 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，各タスクの前周期で発生したブロッキング時間に応じて，3.4 節の (2) で述べた算出結果を導くような入力とする．

4.2 タスク重要度の更新則

タスク重要度は，PDP モデルに従って，他のタスクからの入力および外部条件としての入力により決める．しかし，ADM では，その決め方が熱力学的モデルの熱揺動により制御され，確率的に決まる．この変更を PDP モデルの状態更新則に加え，タスクの重要度更新則とする．式 (4) にそれを示す．以降，ユニットをタスクに，状態値を重要度に対応づけて述べる．

$$a_i(t+1) = a_i(t) + \begin{cases} (1-a_i(t))net_i(t) & prob(net_i(t)) \geq R \\ a_i(t)net_i(t) & \text{else} \end{cases} \quad (4)$$

$a_i(t)$ はタスク i の時間 t におけるタスクの重要度（$0 \leq a_i(t) \leq 1$），$net_i(t)$ はタスク i 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．

式 (4) の $(1-a_i(t))net_i(t)$ は，タスクの重要度を高める方向へ更新する．一方，$a_i(t)net_i(t)$ は重要度を低める方向へ更新する．どちらの式を適用するかは，$prob(net_i(t)) \geq R$ によって確率的に決める．この確率はタスクの重要度が 1 に向かう確率を意味する．したがって，タスクの重要度更新則は，熱力学モデルの熱揺動による振動を擬似する確率関数に依存して，重要度を高める方向へまたは低める方向へ更新する．

4.3 タスクへの総入力とタスク間の結合重み

タスクへの総入力は，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 5 参照）．タスクへの総入力 $net_i(t)$ を以下に示す．

$$net_i(t) = \sum_{j=1, j\neq i}^{n} \omega_{ij} a_j(t) + bias_i(t) \quad (5)$$

n はタスク数，ω_{ij} はタスク i とタスク j との結合重み，$bias_i(t)$ はタスク i の時間 t における外部入力である．

式 (5) から分かるように，他のタスクからの入力は他のタスクの重要度とそのタスク間の結合重みとの積としている．このため，正の結合重みを持つ，すなわち通信するタスクからはそのタスクの重要度の高さに比例した正の入力が，当該タスクに作用する．一方，負の結合重みを持つ，すなわち通信しないタスクからはそのタスクの重要度の高さに比例した負の入力が当該タスクに作用する．タスクへの総入力が大きな正の値となると，重要度更新則は高い確率でタスクの重要度を高める．その反対の状態では，タスクへの総入力は大きな負の値となり，更新則は高い確率で重要度を低める．すなわち，相互通信するタスクの重要度は連動して更新される可能性が高い．

4.4 外部条件としての入力 $bias_i(t)$

前章で述べたように，変動するブロッキング時間に応じてネットワークの動作を修正する必要がある．その修正を外部条件の入力（以降，外部入力）として対応づける．それを式 (6)，(7) に示す．

$$bias_i(t) = \begin{cases} W_i(t)(0-a_i(t)) & B_i(t) \neq 0 \\ W_i(t)(1-a_i(t)) & B_i(t) = 0 \end{cases} \quad (6)$$

$$W_i(t+1) = \begin{cases} W & |bias_i(t)/W_i(t)| < z \\ W & b_i(t)bias_i(t-1) < 0 \\ W_i(t) + gW & \text{else} \end{cases} \quad (7)$$

$B_i(t)$ はタスク i の時間 t における前周期で発生したブロッキング時間，$W_i(t)$ はタスク i の時間 t における外部入力重み，z は外部入力重みの閾値，W は外部入力重みの初期値，g は外部入力重みのゲイン係数である．

式 (6)，(7) から分かるように，ブロッキング時間が発生している場合，相対的に開始時刻が早いと判断する．そのため，外部入力は，実行開始時刻を遅らす適応デッドラインを算出するように，タスクの重要度を低める負の入力とする．一方，ブロッキング時間が発生していない場合，相対的に開始時刻が遅いと判断し，

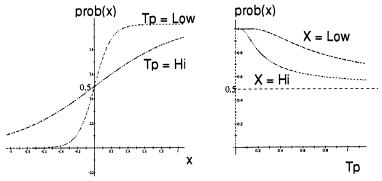


図 6 確率関数と温度
Fig. 6 Probability function and temperature.

タスクの重要度を高める正の入力とする．

また，外部入力の重みは区分線形関数とし，外部入力の作用がタスクの重要度に反映されない場合，重みを増加させ外部入力の作用を強くする．

以上により，タスクの重要度はブロッキング時間を考慮することとなる．

4.5 熱揺動の制御

PDP モデルのネットワークは，初期値に依存してある制約充足度の高い状態に至り，静止する．ADM は，熱力学的モデルによる熱揺動を加えることでこれを回避する．前述したように，タスクの重要度を高めるかまた低めるかは熱揺動の振動により決まるが，このときに使用する確率関数 $prob(x)$ を以下に示す．以下の式の Tp は温度である．

$$prob(x) = \frac{1}{1 + \exp(\frac{-x}{Tp})} \tag{8}$$

式 (8) および**図 6** から分かるように，確率関数 $prob(x)$ は x（タスクへの総入力）が大きいほど 1 になる確率が高くなる．しかし，温度 Tp を高めるとグラフはなだらかになり，入力が正の大きな値であっても確率が 1 より小さな値となる．また，入力が負の大きな値であっても確率が 0 より大きな値となる．したがって，温度 Tp を高くすると更新則が充足度を高める方向だけでなく低める方向にも作用する可能性が高くなる．これにより，様々な状態を動き回り不安定な状態となる．

一方，温度 Tp を下げ 0 に近づいた場合，確率関数はほぼステップ関数と同等なり，更新則が充足度を高める方向に作用し，収束へ向かう．

熱力学的モデルの熱揺動を応用した方式として，焼き鈍し法[3)]がある．焼き鈍し法は，ある固定の外部条件で温度を高い状態（様々な状態を激しく移動する）からゆっくりと温度を下げ，最も安定する状態を探す．この状態は，ある 1 つの条件において安定した状態である．しかし，ADM が想定する動的な環境では，外部条件であるブロッキング時間は短時間に変動するため，焼き鈍し法を利用できない．そこで，単純に温度を下げるのではなく，制約充足度の高い状態からの距離により熱揺動の温度を制御する．これにより，

- 変動する外部条件に PDP モデルを連続的に動作させる，
- 完全な収束を避け，適度な充足度とすることで外部変化に追随する，

を実現する．

(1) 制約充足度の高い状態

PDP モデルにおける制約充足度を次式に示す．

$$G(t) = \sum_{i=1}^{n}\sum_{j=1}^{n} \omega_{ij} a_i(t) a_j(t) + \sum_{i=1}^{n} bias_i(t) a_i(t) \tag{9}$$

外部入力が 0 において制約充足度 $G(t)$ が最大となる状態は，あるタスクとそのタスクと通信するすべてのタスクの重要度が 1 に，通信しないすべてのタスクの重要度を 0 となる状態である．これは，相互に通信するタスクは重要度を同時に高くして実行を早め，その他のタスクは実行を遅らすことを意味する．このような状態が制約充足度 $G(t)$ を高くする．また，制約充足度の高い状態は，相互に通信するタスク集合ごとに存在する．

実環境では，外部入力であるブロッキング時間があるため，実環境の制約充足度の高い状態は前述の制約充足度の高い状態と一致しない可能性が高い．しかし，前章で述べたように，相互に通信するタスクを近い時刻で実行をするとブロッキング時間が減少することから，外部入力なしにおける高充足度状態の近傍に制約充足度の高い状態が存在していると考える．

(2) 温度 Tp の制御

温度は，制約充足度の高い状態からの距離によって制御する．以下にそれを示す．

$$Tp(t+1) = \begin{cases} Tp(t) + r2(T^C - Tp(t)) & D \le D^{Near} \\ Tp(t) + r1(T^C - Tp(t)) & D^{Near} < D < D^{Far} \\ Tp(t) + r3(T^H - Tp(t)) & \text{otherwise} \end{cases} \tag{10}$$

D は，高い充足度となる複数のネットワーク状態と現在のネットワークの状態とのユークリッド距離（以降，距離）において，最小となる値とし，D^{Near} および D^{Far} はネットワーク状態間でとりうる最大距離に対する比率から決定する．ネットワークの状態間の距離は，ある状態におけるタスクの重要度と他の状態におけるタスクの重要度の差が，すべてのタスクに

おいて 1 である場合に最大なる．これをネットワーク状態間の最大距離とし，その値はタスク数の平方根値となる．また，$Tp(t)$ は時間 t の温度，T^C は下限温度，T^H は上限温度，$r1$，$r2$ および $r3$ は制約充足度の高い状態からの距離範囲に応じた温度の変化量を決定するゲイン係数である．

式 (10) から分かるように，タスク間の相互結合ネットワークの状態が制約充足度の高い状態に近い場合は温度を低くする．その結果，確率関数 $prob(x)$ は 1 に近い値を出力し，タスクの重要度は制約充足度の高い状態へ近づく確率が高くなる．しかし，制約充足度の高い状態へ近づきすぎると，温度が小さな値となり，確率関数 $prob(x)$ はほぼ 1 を出力し，タスクの重要度は制約充足度の高い状態に固定化する．ADM の想定環境は，動的な環境であるので，制約充足度の高い状態は時間とともに変わる．そこで下限温度 T^C により，温度を下げ止まりさせ，その近傍で小さく振動させる．これにより，適度な制約充足と外部変化に敏感に反応する動作を実現する．一方，遠い場合は温度を高くし，大きく振動させ充足度の高い状態を発見する可能性を高める．これら制御により，タスクの重要度の初期値に依存することなく，外部環境の変化に応じて複数の制約充足度の高い状態の間を移動する．

4.6 更新則適用タイミングと適応デッドラインの算出

タスクの重要度更新則は，タスクの到着時に各タスクごとに単独に適用する．したがって，タスク到着時のみ，タスク数に依存した計算が行われる．ADM は，タスクの重要度の初期値には依存しないため，タスク起動時の重要度は 0 とし，また，外部入力の初期値に関しても 0 とする．タスクの到着時に算出された重要度は，次式により適応デッドラインに変換する．

$$\begin{aligned} d_i^n &= (n-1)T_i + T_i^{max} \\ &\quad + (T_i^{min} - T_i^{max})a_i(t) \\ T_i^{max} &= T_i + h_i \\ T_i^{min} &= T_i - h_i \end{aligned} \tag{11}$$

以降，算出した適応デッドラインが最も早い順から優先度を高くし，実行順を決定する．

5. 性能評価

提案ポリシを，我々が分散 OS の構成法の研究のため開発している分散 OS Solelc [15] 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．

5.1 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は 1 msec とした．また，コンパイラは gcc（version egcs-2.91.60）を使用した．

5.2 評価方法と評価タスクセット

評価はメッセージの到着順に処理する FIFO，メッセージの優先度順に処理する PRIQ およびメッセージの優先度順に処理し，受信タスクの優先度をメッセージキュー内の最も高い優先度にする PIP の 3 つの RT-IPC に適応機能を付加した場合と付加しない場合の合計 6 つのケースを比較する．

タスクの通信依存関係と時間制約は，タスクが周期起動する直前に，接続先，基準周期および許容遅延時間として OS に通知する．ただし，CPU 使用時間および入出力遅延時間は未知とする．これらタスクからの情報により，ADM は動的に通信依存関係のタスク間相互ネットワークを構築する．ネットワークの結合重みは通信関係がある場合は 5.0，通信関係がない場合は -5.0 とした．温度制御パラメータにおいて，D^{Near} および D^{Far} はそれぞれネットワーク状態間の最大距離 × 0.2，ネットワーク状態間の最大距離 × 0.5 とし，$r1$，$r2$ および $r3$ はそれぞれ 0.5，0.9，0.9 とした．また，各タスクは，Solelc のスレッドに対応づける．

(1) 評価タスクセット 1

評価タスクセット 1 は，クライアント/サーバモデルに基づく要求時処理型である．これのタスクセットは，1 つの負荷タスク，6 つのクライアントタスク，入力データサーバタスク，出力データサーバタスク，入力および出力の擬似ドライバにより構成される（**図 7**）．クライアントタスクは入力データサーバタスクと通信

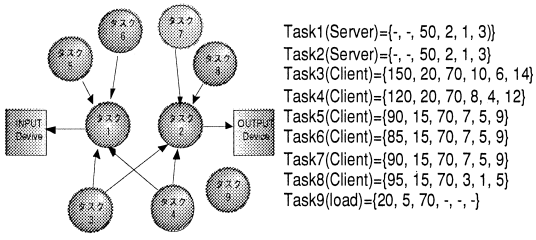


図 7 評価タスクセット 1 の通信関係と時間制約
Fig. 7 Communication constraints and timing constraints for taskset 1.

するタスク，出力データサーバタスクと通信するタスクおよび両サーバタスクと通信するタスクの 3 つの形態とする．サーバタスクはクライアントタスクの要求の到着により起動され，擬似ドライバへ処理要求を行う．擬似ドライバは，サーバからの要求を 1 つごとに到着順に処理し，データ量に応じて遅延を発生させる．サーバタスクのデッドラインは，適応機能を付加しない場合は要求クライアントタスクのデッドラインと同一とし，適応機能を付加した場合は要求クライアントタスクの時間制約から要求の到着時にサーバタスクの適応デッドラインを算出し，これを用いる．また，クライアントの各周期における CPU 使用時間はランダムに変動させる．また，サーバタスクの CPU 使用時間および擬似ドライバでの遅延時間もクライアントの CPU 使用時間に連動し変動させる．

各タスクの通信関係と時間制約を図 7 に示す．図中のはタスクの時間制約を示す．内の各項目は，左から基準周期，許容遅延時間，起動時刻，平均 CPU 使用時間，最小 CPU 使用時間，最大 CPU 使用時間を示す．単位はすべて ms である．

(2) 評価タスクセット 2

評価タスクセット 2 は，パイプラインモデルによる前処理型/予約処理型である．このタスクセットは，連続メディアストレージサーバ[8]などの処理において使用されるモデルである．最初のタスクがデータを生成し，次のタスクに渡して別の処理をう．データはタスクからタスクへと流れ，順次処理される．クライアントタスクと入力サーバタスクの関係からみた場合，クライアントタスクとサーバタスクが 1 対 1 に対応し，サーバタスクは前処理を行い，これをクライアントタスクに引き渡す．出力サーバタスクの場合，同様にクライアントタスクと 1 対 1 に対応し，クライアントタスクの処理後に処理を開始する．すなわち，サーバの処理は複数タスク（マルチスレッド）により構成される．このタスクセットは，1 つの負荷タスク，3 つのクライアントタスク，3 つの入力サーバタスク，3 つの出力サーバタスク，1 つの入力擬似ドライバ，1 つ出力擬似ドライバにより構成される（**図 8**）．

5.3 スケジューリング成功率

各評価タスクセットに関して，負荷状況，入出力遅延状況と起動位相状況に応じて周期タスクのスケジューリング成功率の変動を前述の 6 つのケースについて比較評価した．スケジューリング成功率は，到着総周期タスク数に対するデッドラインまでに処理が完了したタスク数の割合とし，評価期間は 180,000 msec とした．

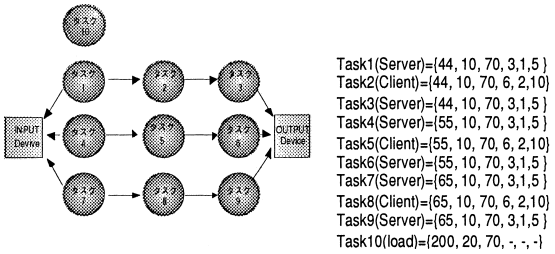


図 8 評価タスクセット 2 の通信関係と時間制約

Fig. 8 Communication constraints and timing constraints for taskset 2.

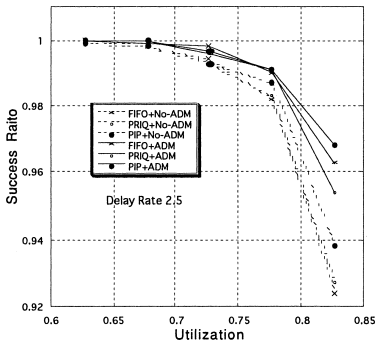


図 9 評価タスクセット 1 の負荷状況に応じたスケジューリング成功率

Fig. 9 Scheduling success raito by utilization for taskset 1.

(1) 評価タスクセット 1 の結果

6 つのケースの負荷状況に応じたスケジューリング成功率を**図 9** に示す．同一の RT-IPC（●点は PIP，○点は PRIQ，×点は FIFO）において，適応機能を付加したケース（実線表示）の方がスケジューリング成功率が高い．また，RT-IPC の種類にかかわらず，適応機能を付加したケースがスケジューリング成功率が良い結果となっている．負荷が高くなるに従い，これが顕著になる．これは，次のように考える．すなわち，負荷が高くなるとクライアントタスク間でサーバタスク競合が発生する可能性が高まる．それに従い，適応機能なしの FIFO のケースでは，サーバタスクは到着順に処理を行うため比較的デッドラインが早いクライアントタスクに実行権が渡らなくなる．適応機能なしの PRIQ および PIP のケースでは，サーバタスクはデッドラインの早いタスクの処理を優先するが，高負荷状態ではこの傾向が強まり，比較的デッドラインの遅いタスクに実行権が渡らなくなる．それぞれ，実行権が渡らなくなったタスクはブロッキング時間が増大しデッドラインミスを発生させる．したがって，スケジューリング成功率が低下する．

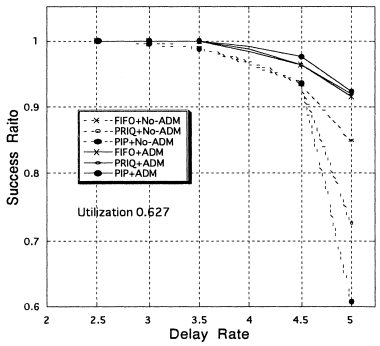


図 10 評価タスクセット 1 の遅延状況に応じたスケジューリング成功率
Fig. 10 Scheduling success raito by delay for taskset 1.

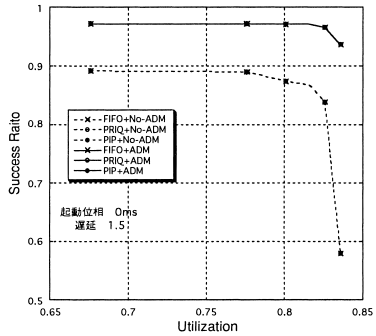


図 11 評価タスクセット 2 の負荷状況に応じたスケジューリング成功率
Fig. 11 Scheduling success raito by utilization for taskset 2.

一方，適応機能を付加したケースでは，それぞれの RT-IPC の種類で発生するブロッキング時間から，サーバタスク競合を回避するようにデッドラインを修正する．そのため，適応機能なしのケースで発生するブロッキング時間の偏り傾向を抑制し，スケジューリング成功率が改善されると考える．

6 つのケースの入出力遅延に応じたスケジューリング成功率を**図 10** に示す．入出力遅延時間はサーバタスクの実行時間に比例して設定し，横軸の Delay Rate はその比例定数を示す．この図から分かるように，負荷状況の場合と同様に，適応機能を用いたケースでスケジューリング成功率が高くなる．また，その差異が著しい．入出力遅延時間の増大によりサーバタスクの処理完了が大きく遅れるため，高負荷時よりクライアントタスク間でのサーバタスク競合の可能性が高くなる．したがって，適応機能の有無によりスケジューリング成功率が大きく異なると考える．

(2) 評価タスクセット 2 の結果

6 つのケースの負荷状況に応じたスケジューリング成功率を**図 11**，入出力遅延に応じたスケジューリング成功率を**図 12** に示す．この両図から分かるように，スケジューリング成功率は RT-IPC の種類にまったく依存せず，適応機能を用いたケースでつねに高い．スケジューリング成功率が RT-IPC の種類にまったく依存しないのは，サーバタスクとクライアントタスクが 1 対 1 に対応している（マルチスレッドサーバ）ため，サーバタスクとの通信によりクライアントタスク間での競合が発生しないことによる．したがって，RT-IPC の 3 つの機能は同一の結果となる．一方，サーバタスクと擬似ドライバとの通信で競合は発生する．しかし，擬似ドライバはカーネルタスクであり，つねに最も高い優先度であることから，RT-IPC は擬似ドライバには機能しない．適応機能は，擬似ドライバとの通信により発生したブロッキング時間からデッドラインを修正することから，このような資源競合にも機能する．したがって，適応機能を用いたケースでつねにスケジューリング成功率が高くなると考える．

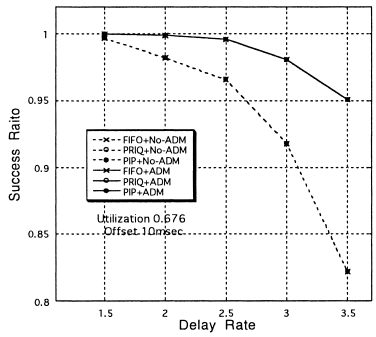


図 12 評価タスクセット 2 の遅延状況に応じたスケジューリング成功率
Fig. 12 Scheduling success raito by delay for taskset 2.

6 つのケースの起動位相に応じたスケジューリング成功率を**図 13** に示す．この図から分かるように，図 11 や図 12 と同様に，RT-IPC の種類にまったく依存せず，適応機能を用いたケースでスケジューリング成功率がつねに高い．したがって，適応機能を用いたケースは各タスクの間の起動位相が少なく済むため，エンド–エンドでの処理遅延が小さくなる．これは，図 11，図 12 の場合と同じ理由と考える．

以上の (1) と (2) の結果から，適応機能は，従来の方式では考慮されていない入出力遅延やタスク間通信による競合に対して有効に機能し，高いスケジュール可能性を導くことが分かる．

5.4 ブロッキング時間

評価タスクセット 1 における適応機能なしの PIP と

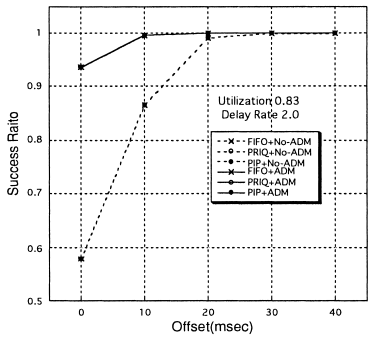


図 13 評価タスクセット 2 の起動位相に応じたスケジューリング成功率
Fig. 13 Scheduling success raito by offset for taskset 2.

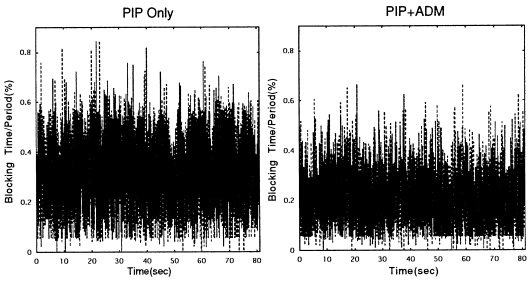


図 14 ブロッキング時間の推移
Fig. 14 Blocking time.

適応機能あり PIP のブロッキング時間の推移を，**図 14** に示す．縦軸は各タスクの実行終了時における周期に占めるブロッキング時間の割合である．また，負荷は 0.627，遅延レートは 2.5 である．

2 つのグラフを比較すると，適応機能あり PIP のケースの周期に占めるブロッキン時間の割合が適応機能なしのケースよりも全時間を通して低いことが分かる．このことから，適応機能によりブロッキング時間を抑制していることが分かる．したがって，ブロッキング時間が少ないことからも，ADM は高いスケジュール可能性を持つスケジューリングポリシであることが判断できる．

5.5 スケジューリングオーバヘッド

ADM の計算量は，PDP モデルに依存し，スケジュール対象のタスク総数に比例する．したがって，ADM のスケジューリングオーバヘッドは，タスク総数が増えると高くなる．そのオーバヘッドの評価結果を**図 15** に示す．この図から分かるように，タスク総数に応じて，直線的にオーバヘッドは高くなる．プロセッサが Pentium-S 200 MHz を用いた場合，タスク総数 10 で平均約 72 μsec である．現在の一般的なプロセッサのクロックは評価環境の数倍である．このことから，一

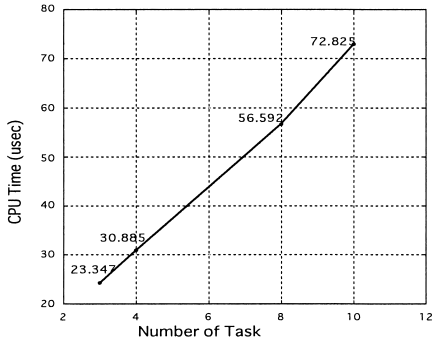


図 15 スケジューリングオーバヘッド
Fig. 15 Scheduling overhead.

般的な PC は，評価環境の 2 倍のクロックを持つと仮定すると，10 個のタスクのスケジューリングで約 36 μsec の計算時間が必要となる．ただし，この負荷は，適応デッドデッドラインを計算するタスクの到着時のみに必要であり，それ以外のコンテキストスイッチ時には不要である．一方，マルチメディア環境で扱う音声，アニメーションおよびビデオは文献 4) によると，最短周期が 75 Hz，すなわち約 13 msec となっている．この最短周期のタスクを 10 個スケジュールする場合，一般的な PC 環境において 1 周期の適応デッドライン計算に必要となる時間は，上記で示した約 36 μsec である．この時間のタスクの周期に占める割合は，たかだか 0.28%にすぎない．また，タスクを 100 個スケジュールする場合でも，2.8%ほどである．

以上のことから，ADM はタスク総数に比例してオーバヘッドは高くなるが，タスクのスケジューリング周期に占める割合は極微量であるといえる．したがって，ADM は，周期的タスクのスケジューリングに十分に適用できると考える．

6. おわりに

本論文では，従来方式において問題であった，1）スケジュール環境が事前に予測が困難な動的な環境である，2）スケジュール問題がタスクの時間制約と通信依存関係により構成される，3）いくつかのタスクは入出力処理を行う，4）タスク間のデータ交換方式は疎結合モデルによるメッセージ通信方式である，といった 4 つの問題に適用可能な新たなポリシ ADM を提案し，さらに，その性能評価から ADM は高いスケジュール可能性を導くポリシであることを示した．

参考文献

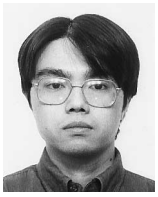
1) Sha, L., Rajikumar, R. and Lehoczky, J.P.: Priority Inheritance Protocols: An Approach to

Real-Time Synchronization, *IEEE Trans. Comput.*, Vol.39, No.9, pp.1175–1185 (1990).
2) Liu, C.L. and Layland, J.W.: Scheduling algorithms for multiprograming in a hard real time enviroment, *J. ACM*, Vol.20, No.1, pp.46–61 (1973).
3) Yavatkar, R. and Lakshman, K.: Optimization by Simulated Anneling, *Science*, Vol.220, pp.671–680 (1983).
4) Furth, B.: Multimedia Systems: An Overview, *IEEE Multimedia*, Vol.1, No.1, pp.47–59 (1994).
5) Tokuda, H., Nakajima, T. and Rao, P.: Real-time mach: Towards a predictable real-time system, *Proc. USENIX Mach Workshop*, pp.1–10 (1990).
6) Rumelhart, D.E., McCelland, J.L. and the PDP Research Group: Parallel Distributed Processing, The MIT Press (1986).
7) Tokuda, H., Nakajima, T. and Tokuda, H.: RT-IPC An IPC Extension for Real-Time Mach, *Proc. 2nd Microkernel and Other kernel Architectue*, USENIX (1993).
8) Anderson, D.P., Osawa, Y. and Govindan, R.: A File System for Continuous Media, *ACM Trans. Comput. Syst.*, Vol.10, No.4, pp.311–337 (1992).
9) Baker, T.P.: Stack-Based Scheduling of Realtime Processes, *J. Real-time Systems*, Vol.3, No.1, pp.67–99 (1991).
10) Cheng, S.T. and Agrawala, A.K.: Allocation and Scheduling of Real-Time Periodic Tasks with Relative Timing Constraints, *Proc. International Workshop on Real-Time Computing and Applications*, pp.210–217 (1995).
11) DiNatale, M. and Stankovic, J.A.: Applicability of Simulated Annelaling Methods to Real-Time Scheduling Jitter Control, *Proc. IEEE Real-Time Systems Symposium*, pp.190–199 (1995).
12) Kirkpatrick, S., Gelatt, Jr., C.D. and Vecchi, M.P.: A CPU Scheduling Algorithm for Continuous Media Application, *Proc. 5th Intl. Workshop on Network and Operating Systems Support Digital Audio and Video*, pp.210–215 (1995).
13) Locke, C.D.: Best-effort decision making for real-time scheduling, Ph.D. Thesis, CMU (1986).
14) Buttazzo, G.C. and Stankovic, J.A.: *RED: Robust Earliest Dealine scheduling*, University of Messachusetts (1993).
15) 芝 公仁，大久保英嗣：分散オペレーティングシステム Solelc の構成，情報処理学会研究会報告，2000-OS-84, pp.237–244 (2000).

(平成 12 年 8 月 17 日受付)
(平成 12 年 10 月 13 日再受付)
(平成 12 年 12 月 13 日採録)

滝沢 泰久（正会員）

昭和 34 年生．昭和 58 年京都工芸繊維大学工芸学部機械学科卒業．同年日本ユニシス（株）入社．平成 2 年住友金属工業（株）入社．平成 10 年より（株）ATR 環境適応通信研究所に出向．現在，実時間オペレーティングシステム等の研究に従事．



芝 公仁（学生会員）

昭和 49 年生．平成 11 年立命館大学大学院理工学研究科博士課程前期課程修了．現在，同大学大学院理工学研究科博士課程後期課程に在学中．オペレーティングシステム，分散システム，実時間システム等に興味を持つ．

大久保英嗣（正会員）

昭和 26 年生．昭和 52 年北海道大学大学院工学研究科情報工学専攻修士課程修了．同年（株）日立製作所ソフトウエア工場入所．主として FORTRAN コンパイラの開発に従事．昭和 54 年京都大学工学部情報工学科助手．昭和 60 年同講師．昭和 62 年同助教授．平成 3 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．