連続メディア処理のためのリアルタイムスレッドモデルの拡張†

1H－1

河内谷 清久仁[1]‡　緒方 正暢[1]‡　徳田 英幸[2]
[1]日本アイ・ビー・エム(株) 東京基礎研究所　[2]慶應義塾大学/カーネギーメロン大学

1　はじめに

対話型のマルチメディアシステムでは，動的に変動するシステムの負荷に応じて連続メディア処理のサービスの質(QOS: Quality of Service)を調整する機能が重要である．そのための機構として我々は，複数の連続メディア処理のQOSを集中制御するサーバを用いたQOS制御モデルの提案と試作を行なった[1]．しかし，この試作版では過負荷時のQOS制御について問題点がみられた．本稿では，この問題点の解決のためのリアルタイムスレッドモデルの拡張と，実験結果について報告する．

2　QOS制御のモデル

我々は現在，Real-Time Mach 3.0(以下RT-Machと略す)をベースにした分散マルチメディアシステムの構築を行なっている[2, 3, 4]．RT-Mach上での連続メディア処理は，周期的に起動されるリアルタイムスレッドを作り，毎回の起動のたびに一定の処理(例えば，ビデオの1フレームの表示)を行なうという形で記述できる．その際，システムが過負荷になって時間内に処理を終えられない場合には，その連続メディア処理のQOSを下げて，単位時間あたりの処理量を減らす必要がある．具体的には，周期を伸ばして時間的解像度を下げる(コマ数を減らす)，一回の処理量を減らして空間的解像度を下げる(画質を下げる)などの手法が考えられる[5, 6]．

システム内に複数の連続メディア処理が存在する場合，QOSの調整は各連続メディアの性質や優先度を反映して集中的に行なわれることが望ましい．そのための機構として我々はQOSを集中制御するサーバを用いたQOS制御モデルの提案と試作を行なった[1]．その構成を図1に示す．各連続メディア処理スレッドは起動時にサーバに対して，その性質や優先度を示す「QOSファクター」を申告する．サーバは，システムの負荷状態と申告されたQOSファクターをもとに各連続メディア処理のQOSを上げ下げする．試作版では連続メディア処理スレッドの周期を変更することでQOS制御を行なっている．QOSファクターとしては，そのスレッドが希望する周期の最小/最大値/単位変動量と優遇度を申告している．システムの負荷状態は，各連続メディア処理スレッドのデッドラインミスを利用して判断している．

このQOS制御モデルで，表1に示すQOSファクターをもつ連続メディア処理群に対してQOS制御を行なってみた結果が図2である．なお実験にはRT-Mach MK83(スケジューリングポリシーはRate Monotonic)，IBM PS/V*(2410-N, 66MHz i486DX2**)を用いた．グラフの横軸は経過時間，縦軸はその時点でQOS制御サーバが指定している各スレッドの周期をあらわしている．各スレッドの周期は，最終的にはほぼ定常状態に落ち着いているが，定常状態になる前の一時的な過負荷時に周期が一旦跳ね上がっていることがわかる．

本稿の残りの部分では，この跳ね上がりを解決するためのスレッドモデルの拡張について検討する．

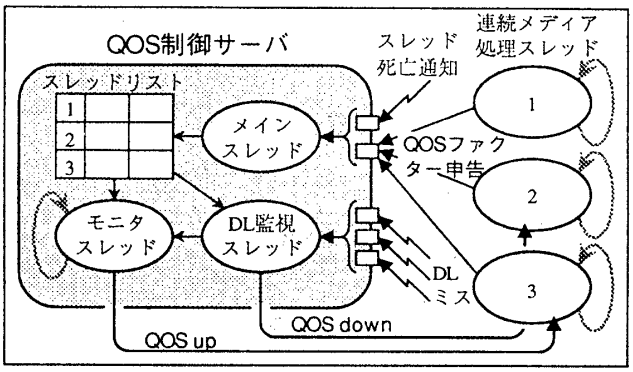

図1: QOS制御サーバによるQOS制御モデル

No.	実行期間(秒) 開始 → 終了	仕事量	周期(ms) 最小 ～ 最大	優遇度
1	0 → 60	10	50 (固定)	100
2	0 → 60	50	30 ～ 100	10
3	10 → 40	50	30 ～ ∞	15
4	20 → 50	50	30 ～ ∞	30

表1: 各連続メディア処理のQOSファクター(実験用)

3　スレッドモデルの拡張と実験

この跳ね上がりの主な原因は，RT-Machの周期的リアルタイムスレッドのセマンティクスが「CATCH UPモード」であることにある．図3(a)はCATCH UPモードの動作を示したものである[1]．リアルタイムスレッドの毎回の起動のタイミングはあらかじめ定められており，システムが過負荷になった場合，その分の「負債」が次回以降にどんどん蓄積されていく．システムが定常状態になるためにはまずこの負債を返し終える必要があるため，図2のように一旦周期が跳ね上がってしまうことになる．CATCH UPのセマンティクスは，ハードリアルタイム的な処理においては重要であるが，連続メディア処理のようなソフトリアルタイム処理には必ずしも向いていない．QOS制御を前提とした連続メディア処理においては，一時的な過負荷状態では正確なタイミングを維持することより，事態の収拾を優先すべき場合があるからである．

"Extending Real-Time Thread Model for Continuous Media Processing,"
Kiyokuni Kawachiya[1], Masanobu Ogata[1] and Hideyuki Tokuda[2]
[1]IBM Research, Tokyo Research Laboratory,
1623-14, Shimotsuruma, Yamato, Kanagawa 242, Japan
E-Mail: <kawatiya@trl.ibm.co.jp>
[2]Keio University / Carnegie Mellon University

†この研究は，情報処理振興事業協会(IPA)が実施している開放型基盤ソフトウェア研究開発評価事業「マルチメディア統合環境基盤ソフトウェア」プロジェクトのもとに行なわれた．

‡開放型基盤ソフトウェア湘南藤沢キャンパス研究室の研究員としてIPAに登録されている．

*米IBM Corp.の商標，**米インテル社の商標．

[1]図3中の網かけの部分は，スレッドがReady-to-Run状態になっている(つまり，その回の処理が残っている)期間を示すもので，この期間中そのスレッドだけが走り続けているわけではない．

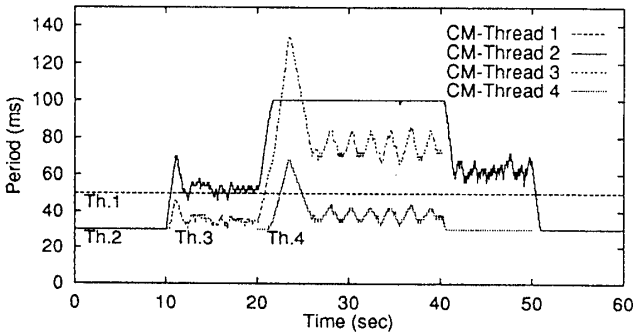

図 2: QOS 制御の実験結果 1

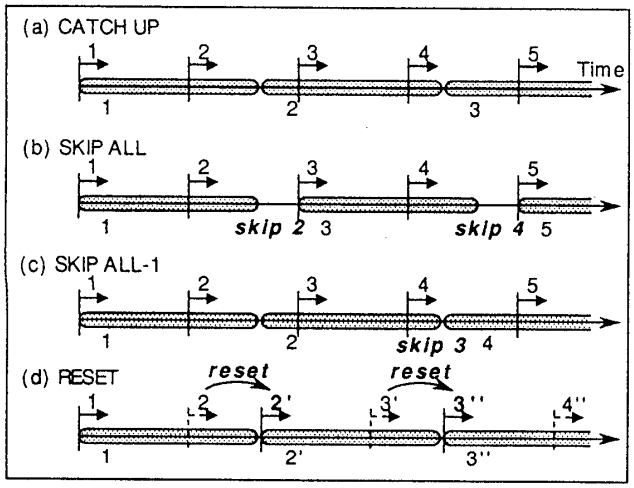

図 3: 周期的リアルタイムスレッドの過負荷時の動作

CATCH UP 以外のセマンティクスとしては，SKIP ALL や SKIP ALL−1 などが提案されている [7]．これらは，スレッドの起動タイミングはそのままで，何回分かの処理をスキップするという方式である．ある回の処理が次回の起動タイミングまでに終わらなかった場合，SKIP ALL では，くい込んだ分の回の起動は全てスキップされ，次回の起動タイミングから新たに処理が行なわれる (図 3(b))．SKIP ALL−1 では，最後の一回分だけはスキップせずに残った時間で処理を行なう (図 3(c))．

これらの SKIP ALL 系のセマンティクスでは，あくまでもスレッドの起動タイミングは一定である．しかし，今回のような連続メディア処理においては，QOS 制御にともなって周期が変動するため，必ずしも起動タイミングにこだわる必要がない．そのため，次の起動タイミングまでに処理を終えられなかった場合，起動タイミング自体をずらすという方式も考えられる (図 3(d))．過負荷時に起動タイミングをリセットするということから，この方式を RESET モードと呼ぶことにする．RESET モードを用いれば，定常状態では周期的リアルタイムスレッドとして動作し，(一時的) 過負荷時には Best-Effort 的に動くという二面性を持ったスレッドを作ることができる．

これらの各モードを RT-Mach 内にインプリメントし，表 1 で示した連続メディア処理スレッドのモードを変更して QOS 制御の様子の変化を調べた．その結果どのモードでも，オリジナルの CATCH UP モードと比べて過負荷状態の解決がより適切に行なわれることがわかった．特に，RESET モードは図 4 に示すように非常に安定した挙動を示した．

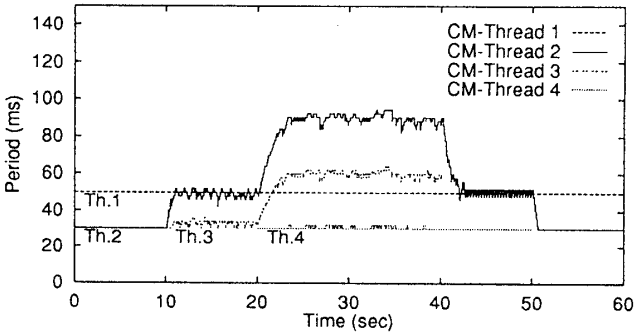

図 4: QOS 制御の実験結果 2 (RESET モード)

4 まとめと課題

本稿では，連続メディア処理のためのスレッドモデルとして，CATCH UP モード以外のセマンティクスを RT-Mach に導入し，その有効性を示した．実験の結果，過負荷時にスレッドの起動タイミングをリセットする RESET モードが特に安定した QOS 制御を行なえることがわかった．

今回の実験はリアルタイムスレッドを拡張して行なったが，さらに効率のよい QOS 制御を考えると，QOS ファクターを属性として持つ新しいスレッドモデルと，それにもとづいて直接スケジューリングを行なう「QOS ベースのユーザレベルスケジューラ」を作るのがよいと思われる．今後はこの方向で改良を続けていく予定である．

謝辞

本研究を行なうにあたり協力/助言していただいている慶応大学「マルチメディア統合環境基盤ソフトウェア」プロジェクトの皆様に感謝いたします．さらに，御指導いただいている慶応大学環境情報学部の斎藤信男教授に感謝いたします．

参考文献

[1] 河内谷, 緒方, 徳田: “Real-Time Mach 上での QOS 制御サーバの実験,” 第 47 回情処全大論文集, pp.2-355–2-356 (1993).

[2] 西尾, 他: “マイクロカーネルによる連続メディア処理の基盤技術,” 第 5 回コンピュータシステム・シンポジウム論文集, pp.17–24 (1993).

[3] 平林, 他: “実時間メディアサーバの設計,” 第 5 回コンピュータシステム・シンポジウム論文集, pp.25–32 (1993).

[4] 多田, 他: “実時間カーネルを用いた連続メディアベースの設計,” 第 5 回コンピュータシステム・シンポジウム論文集, pp.33–40 (1993).

[5] H. Tokuda, et al.: “Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network,” *Proc. of ACM SIGCOMM '92*, pp.88–98 (1992).

[6] H. Tokuda and T. Kitayama: “Dynamic QOS Control based on Real-Time Threads,” *Proc. of the 4th Int. Workshop on Network and Operating System Support for Digital Audio and Video*, pp.113–122 (1993).

[7] H. Tokuda, et al.: “A Real-Time Thread Model for Continuous Media Applications,” *ART Group Tech. Report*, CMU, May 1993 (1993).