4V－3

Real-Time Mach 上での QOS 制御サーバの実験†

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，システムの負荷が動的に変動するため，それに応じて連続メディアのサービスの質 (QOS: Quality of Service) を調整する必要がある．特に，システム内で複数の連続メディアが処理されている場合，この調整は各連続メディアの性質や優先度を反映して行なわれるべきであり，何らかの「調停者」を用意することがのぞましい．本稿では，そのための機構として QOS を集中制御するサーバについて提案し，Real-Time Mach 3.0 (以下 RT-Mach と略す) 上で実現したプロトタイプについて構成と実験結果を報告する．

2 RT-Mach 上での連続メディア処理

我々は現在，RT-Mach[1] をもとにした，分散マルチメディア環境を実現するための基盤ソフトウェアの研究開発評価を進めている [2]．RT-Mach は Mach マイクロカーネル [3] にリアルタイムスレッド，リアルタイムスケジューラ，リアルタイム同期，リアルタイム IPC などの機能を追加した OS である [4]．その上での連続メディア処理は，周期的に起動される RT-Thread 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) を行なうという風に記述することができる．処理が予定した時間 (デッドライン) 内に終了しなかった場合，デッドラインポートを通じてそれを知ることも可能である．

ここで生じる問題点の一つが過負荷状態の解決法で，システムの負荷の変動に応じて処理量すなわちサービスの質 (QOS) を動的に変更していくような機構が必要となる．ネットワークにおける QOS 制御の手法についてはすでに，一定の QOS を保証するプロトコル [5] や，ダイナミックに QOS の調整を行なうモデル [6, 7] などが提案されている．一方，CPU 資源の調整に関しては，セルフスタビライズ方式 [8] などが提案されている．これは，各連続メディアサービスが，そのデッドラインミス情報に従って自分の QOS を変更するという方法である．この方式はシンプルでオーバーヘッドが少なく，また各スレッドが自律的に QOS 制御を行なえるという利点がある反面，他の連続メディアサービスとの協調が難しいという問題点がある．さらに各連続メディアの性質や優先度を反映した制御も難しい．また本来 QOS 制御は，このような局所的方法ではなくシステム全体の状況を把握しながら集中して行なわれるべきであると考えられる．これらのことから，我々は今回 RT-Mach 上で，QOS を集中制御するサーバの試作を行なった．

"A QOS-Control Server on Real-Time Mach,"
Kiyokuni Kawachiya[1], Masanobu Ogata[1], and Hideyuki Tokuda[2]
[1]IBM Research, Tokyo Research Laboratory,
1623-14, Shimotsuruma, Yamato, Kanagawa 242, Japan.
[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ました．

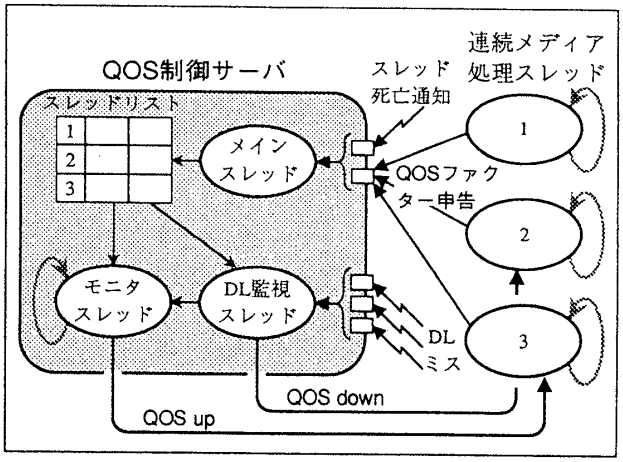

図 1: QOS 制御サーバの構成

3 QOS 制御サーバの構成

QOS 制御サーバの基本構造は次の通りである．各連続メディアサービス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」と，QOS を変更する方法を申告する．サーバは，システムの負荷と申告された QOS ファクターをもとに，各連続メディアサービスの QOS を上げ下げする．

今回 RT-Mach 上に試作した QOS 制御サーバの具体的構成は図 1のようになっている．QOS 制御の方法としては今回，連続メディア処理スレッドの周期を変更するやり方を用いている．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 (大きい方が高い) を Mach IPC を通じてサーバに申告している．システムが過負荷になり QOS を下げる必要が生じた場合，サーバはその時点で周期×優遇度が最も小さいスレッド (で，周期が最大値に達していないもの) を選び，その単位変動量だけ周期を伸ばす．システムに余裕ができて QOS を上げられる場合は逆に，周期×優遇度が最も大きいスレッドを選んで周期を縮めている．

システムの負荷の測定は，サーバ内の 2 つのスレッドを用いて行なっている．デッドライン監視スレッドは，各連続メディア処理スレッドのデッドラインポート (QOS ファクターの申告時にサーバに渡される) を監視しており，デッドラインミスが生じた場合，システムが過負荷になっていると考え上述の QOS を下げる処理を行なう．モニタスレッドは，サーバ内で周期的に起動される RT-Thread で，各周期の間にデッドラインミスが一回も起こらなかった場合，システムに余裕があるとみなし QOS を上げる処理を行なう．

このようにしてサーバはシステムの負荷と申告された QOS ファクターに従って集中的に QOS 制御を行なっている．

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	2	50 (固定)	100
2	0 → 60	10	30 ～ 100	15
3	10 → 40	10	30 ～ ∞	20
4	20 → 50	10	30 ～ ∞	40

表 1: 実験に用いたダミースレッドの QOS ファクター

4 評価実験

次に，今回試作したサーバで，どの程度うまく QOS 制御ができるかを評価する実験を行なった．実験は周期的に起動されて一定量 CPU を消費するダミースレッドを複数用いて行なった．各ダミースレッドの QOS ファクターは表1に示す通りである．スレッド 1 は仕事量は少ないが QOS(周期) を変えられないようなサービスをあらわしている．スレッド 2 は QOS の変動を許しているが，最悪でも 100ms の周期を要求している．スレッド 3,4 は，途中から割り込んでくるサービスで，スレッド 2 よりも高い優遇度が設定されている．これらのダミースレッドを QOS 制御サーバのもとで走らせ，QOS 制御の様子を調べた．実験は 33MHz i486DX** を搭載した IBM PS/V* (2410-Y) 上で行なった． RT-Mach のバージョンは MK78，スケジューリングポリシーは Rate Monotonic を用いている．

実験結果を図2に示す．グラフの横軸は経過時間，縦軸はその時点での各スレッドの周期をあらわしている (つまり下の方が QOS が高い)．最初の 10 秒間は，システムの負荷が軽く要求した最高 QOS 値で処理が行なえている．しかし，スレッド 3,4 の処理が始まるとシステムが過負荷になり，QOS ファクターにもとづいて QOS が下げられている．スレッド 3,4 の処理が終了すると，それに従って再び QOS が上げられている．スレッド 1 は QOS 制御の影響を受けず，全体を通して周期 50ms で実行されている．

このように，QOS 制御サーバによって各スレッドの QOS が指定通りに制御されていることがわかる．ただし，図から分かるように，各スレッドの QOS が定常状態に達するまでにしばらく時間がかかっている．また，定常状態においても QOS が常に振動し固定した値に落ち着いていない．これらは，サーバがシステムの「余裕度」を正確に見積もれないために起こる現象であり，今後の改良が必要な点である．

5 まとめと課題

本稿では，QOS を集中制御するサーバの試作と実験結果について報告した．試作したサーバでも一応の QOS 制御が行なえていることが確認できたが，より適切な制御のためにはサーバとカーネルの連携をもっと密接にし，ユーザレベルスケジューラ [9, 10] のような形で実現する必要があると思われる．QOS 制御の方法も，周期を変更する単純な方法から連続メディアの性質などを利用したより高度な方法 [11] に改良する必要がある．さらに，実際の分散マルチメディア環境では CPU 資源以外の QOS 制御も問題である．今後はこれらの問題も含めた，より統合的な QOS 制御の方法について検討していく予定である．

*米 IBM Corp. の商標，**米インテル社の商標．

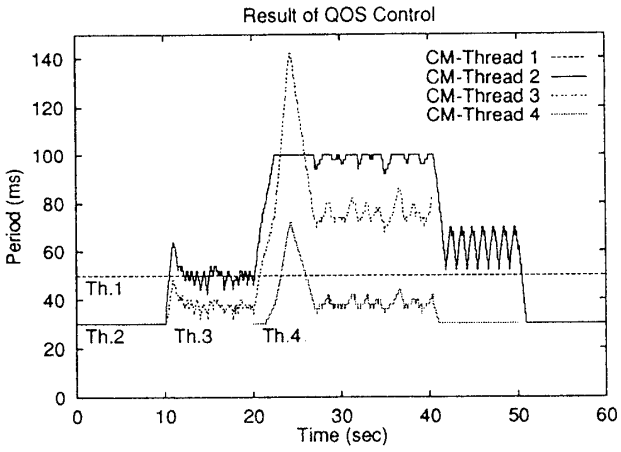

図 2: QOS 制御の実験結果

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] H. Tokuda, et al.: "Real-Time Mach: Towards a Predictable Real-Time System," *Proc. of USENIX Mach Workshop*, pp.73-82 (1990).

[2] 徳田, 斎藤: "マルチメディア統合環境プロジェクトにおけるリアルタイム処理技術," 情処研報 93-ARC-99, pp.9-15 (1993).

[3] D. Golub, et al.: "Unix as an Application Program," *Proc. of USENIX 1990 Summer Conf.*, pp.87-95 (1990).

[4] 緒方, 他: "Real-Time Mach 3.0 のマルチメディア処理に関する性能評価," 情処研報 93-OS-60, pp.67-74 (1993).

[5] C. Topolcic, et al.: "Experimental Internet Stream Protocol, Version 2 (ST-II)," *Internet RFC-1190* (1990).

[6] 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).

[7] A. Campbell, et al.: "A Continous Media Transport and Orchestration Service," *Proc. of ACM SIGCOMM '92*, pp.99-110 (1992).

[8] 舩渡, 徳田: "Real-Time Mach 3.0 における連続メディアサーバの実験," 情処研報 93-OS-60, pp.75-82 (1993).

[9] B. Davis, et al.: "Adding Scheduler Activations to Mach 3.0," *Proc. of USENIX 3rd. Mach Symposium*, pp.119-136 (1993).

[10] S. Oikawa, et al.: "Design and Implementation of Real-Time User-Level Thread," 日本ソフトウェア科学会第 9 回大会, pp.245-248 (1992).

[11] 緒方, 河内谷, 徳田: "動画のフレーム間相関を利用した動的な QOS 制御の実験," 第 47 回情処全大 4V-04 (1993).