システムソフトウェアと
オペレーティング・システム 75－14
（1997．6．5）

帯域保証型通信環境における連続メディアQOS制御

戸辺 義人[1]　徳田 英幸[2]

[1] 慶應義塾大学 SFC 研究所　[2] 慶應義塾大学環境情報学部

あらまし

動画像や音声等の連続メディア・フローのＱＯＳ (QoS: Quality of Service) をOSで制御する研究が行われている．これらの研究においては，システム負荷に応じて，CPU時間やメモリ等の資源を予約し，場合によっては予約する量を動的に変更することを基本としている．一方，ネットワークにおいては，ATM(Asynchronous Transfer Mode) ネットワークや，RSVP (Resource Reservation setup Protocol) プロトコルにより帯域の保証される通信環境が整ってきた．そこで，帯域が保証される通信環境でのエンドホストにおける処理について考察し，QOSパラメータを順次保証していく「逐次QOS保証方式」と，マイクロカーネル・アーキテクチャにおいて，トラフィックの特性が確定してからプロトコル処理をサーバから帯域保証済みのチャネルに切り替えるチャネル切り替え方式について述べる．トラフィックを特定する段階においては，VBRトラフィックに対応できるようにする．スレッドへ連続メディアデータを送ることによりQOSを保証する．本稿では，トラフィック特性の特定とメモリおよび要求帯域値との関連を中心に，RT-Mach3.0を用いて，ATMチャネル切り替えに関する実装方式について述べる．

キーワード　マルチメディアシステム，分散リアルタイムカーネル，リアルタイムスレッド，マイクロカーネル，QOS，ATMネットワーク

QOS Control of Continuous Media in a Bandwidth-Guaranteed Communication Environment

Yoshito Tobe[1]　Hideyuki Tokuda[2]

[1]tobe@sfc.keio.ac.jp Keio University, 5322, Endo, Fujisawa-shi, Kanagawa, 252 Japan,
[2] hxt@sfc.keio.ac.jp, Keio University, 5322, Endo, Fujisawa-shi, Kanagawa, 252 Japan,

Abstract.

In this paper, we describe a design of Successive QOS Control (SQC) of continuous media control under a banswidth-reserved communication environment, such as an ATM network. We, first, explain usuall QOS Control model done on continuous media server on top of RT-Mach3.0 Microkernel. In the model. period of a periodic thread that handles continuous media is stabilized after it runs for some period. Therefore characteristics of continuous media is obtained after the period of the thread. We then show the design of traffic measuremement and a swithing mechanism of VCCs (Virtual Channel Connections) with RT-Mach3.0.

Keywords: Multimedia System, Distributed Real-Time Kernel, Real-Time Thread, Microkernel, QOS, ATM Network

1 はじめに

音声や動画など時間に従って変化していく連続メディアデータを中心としたマルチメディアシステムをコンピュータ上で扱う際には，動的に負荷が変化する．特に対話的なユーザ操作によってシステムが一時的に過負荷な状況に陥ってしまうことがある．そのため，サービスの質(QOS)を定義し，実行時にシステムの負荷や，アプリケーションの優先度に応じた資源割当てや実行単位のスケジュール制御を動的に行う必要があり，そのための研究が行なわれてきた[1, 2, 3, 4, 5, 6]．ここで，資源とは，CPU時間，メモリおよびネットワークの帯域を指す．従来のモデルでは，図1における送信側のCPU時間，メモリ，ネットワークの帯域，受信側のCPU時間，メモリを確保した上で，転送を始めるというものであった．この方法では，複数の要求パラメータが同時に満足されるまで転送が開始できないし，1つの変動要因が他へ影響を与えることになる．例えば，送信側で確保できたCPU時間で，トラフィック特性が決定され，確保できたCPU時間の如何によらずに，ネットワーク帯域を決定してしまうと，実際に連続メディアデータを転送する段階によって，過不足が生じうる．そこで，順次，パラメータを決定していき，段階的にQOSを保証する「逐次QOS保証方式」を提案する．

ネットワークにおいては，ATMネットワークが実用になっており，従来型の媒体においてもRSVP等により，ネットワーク帯域の保証が可能となってきている．そのため，送信ホストと受信ホストでのQOS保証により，送信スレッド-受信スレッド間でのQOS保証が可能となる．なお，ルータやスイッチ等の資源予約は本稿では扱わず，エンド・エンドで帯域が保証されたチャネルが提供された環境での，エンドホストに焦点を当てる．

本論文では，第2章で，連続メディアのOS内での取り扱いについて述べた後，第3章で，提案する逐次QOS保証方式，トラフィック特性切り替え，チャネル切り替えについて説明する．第4章で，本方式のRT-Machへの実装方法，第5章で送信遅延時間の測定結果を示した後，第6章で議論と今後の予定に触れ，第7章でまとめる．

2 連続メディア・フローのOS内での取り扱い

連続メディアオブジェクトは，通常のオブジェクトと異なり，ディジタル化された音声やビデオデータなどのように時系列で連続的に変化するデータを呼び，オブジェクトの正当性が値の正当性だけでなく時間的な正当性にも依存している．

連続メディアオブジェクトをデータストリームで表し，ストリームの構成要素である個々のデータオブジェクトに

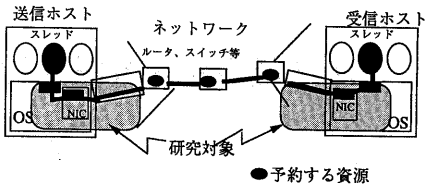

図 1: 連続メディア通信における予約資源

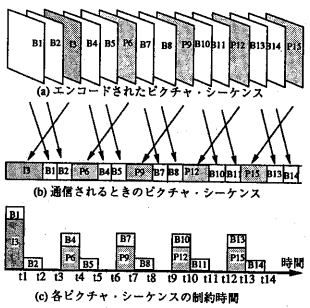

図 2: MPEG-2のシーケンス

時間的制約が付随しているものと定義する．個々のデータオブジェクトに与えられた時間的制約が同一の場合，そのストリームを周期的データストリームと呼び，そうでない場合を非周期的データストリームと呼ぶ．

例えば，ビデオ，動画，音声データなどは，周期的データストリームで表わせ，オンラインのデスクトップ・プレゼンテーションは，非周期的データストリームで表すことができる．

QoS-A[1]モデルにしたがい，連続メディアの送信端から受信端までの流れをフローと呼ぶ．フローそのものをFlow Spec[8]で指定する試みもあるが，ここでは，送信ホスト，ネットワーク，受信ホストの能力に応じてフローの特性も変化するものとする．一方，セッションは，フローのサービス開始から終了までを指す．

連続メディアの代表例であるMPEG-2ビデオでは，図2に示すような周期的なバーストトラフィックが発生する．

一方，帯域予約型のネットワークでは，ATMに見られるようにVBRのトラフィックをそのままネットワークへ流し出すよりも，送信ノードにおいてトラフィック・シェーピングを行なって流し出した方が，信頼性が高まる．また，周期的なリアルタイムスレッドから送信する場合，システムの負荷に依存して周期も決定されるので，定められた周期によりトラフィックも変わる．送信すべきデータが周期的で，周期当たりの転送データ量が固定値ないしは変動が

少なく，確定した周期的なリアルタイムスレッドを用いることができれば，確保すべきネットワークの帯域を決定しやすいが，発生するトラフィック，スレッドと帯域予約型ネットワークにどう適合させるかという問題が生じる．連続メディアデータの種類によっては，周期性も見られないバースト性を生み出すこともあったり，周期性があったとしても時間の経過と共に，トラフィック特性が変化するという問題もある．

MPEG-2のデータ・ストリームは，単独でチャネルを設けるよりも複数本束ねる方が帯域を有効に利用できることが確かめられており [13, 14]，総帯域を共有するチャネルにMPEG-2のデータ・ストリームを通すことも考えなければならない．

3 逐次QOS保証方式

先に，周期が決定されてから発生トラフィックが決定されるというQOSパラメータを同時に決定するのが難しい例を挙げたが，他にも既にリモートホストおよび総帯域割り当て量が定められていて，総帯域割り当て量をオーバしないように，後からCPU使用量を決定しなければならないこともある．

明示的にQOSパラメータ・セットが与えられたとしても，全体を同時に満たすのではなく，QOS-Control Server [3] 等により，送信側のスレッドの周期，計算時間が決定すると，セッションを開始し，セッションが発生するトラフィックの統計と要求遅延時間とに基づき伝送帯域量，送信バッファ量を決定し，その結果を見て，受信側のスレッドの周期，受信バッファ量を決定して，要求QOSを順次QOSを保証していく．または，伝送帯域量を最初に決定し，残りのパラメータを後から決定する等のサイクルを実行して，部分的に満たしていく方法を逐次QOS保証方式と呼ぶ．

システム資源に応じて，QOSパラメータを変更する必要性も生じる．そのため，周期的に上記サイクルを繰り返すことが必要となってくる．本方式では，ATM等のように具体的に数Mbpsと帯域保証される通信チャネルを生成できる環境下で，送信側のスレッドの周期，計算時間が決定された後の必要な伝送帯域量と送信バッファ量の特定を行う．特に，トラフィックがMPEG-2のようなVBRである場合にも適用可能である．

3.1 トラフィック特性の特定およびチャネル切替え

トラフィック特性を特定するための観測点として，ユーザスレッドが送信するためのシステムコールを発行する点，プロトコル処理を実行するモジュール (マイクロカーネルを用いる場合，サーバ・タスク) の入口および出口の3種類が考えられる．しかし，後者2つにおいては，送信スレッドが送出する純粋なトラフィック特性を捉えることができないので，ユーザスレッドが送信するためのシステムコールを発行する点を観測点とする．

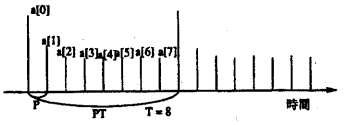

図 3: トラフィック特性特定のモデル化

マイクロカーネル・アーキテクチャでは，常にスレッドにリンクされるライブラリで通信プロトコル処理をしてもよい．Best-Effortの通信路には，この方法でもよいが，帯域が保証される通信路では，通信開始時に要求帯域量をいくらにするか，という問題が生じる．そこで，逐次QOS保証方式においては，QOSパラメータ未確定時に送信側では，送信スレッドがプロトコル処理サーバを介して，Best-Effort通信路に送出し，受信側ではプロトコル処理サーバを介して受信スレッドが受信データを受けとり，トラフィック特性特定によりQOSパラメータ確定後，プロトコル処理サーバを介さず，両スレッドの間で帯域が保証された通信路を用いて通信することとする．

ここにエンド・エンドで一意の識別子で認識されるエンド・エンドの通信路をチャネルと呼び，上記の通信路の切替えをチャネル切替えと称する．

3.2 通信に必要な資源の予約

トラフィック特性特定結果は，帯域量C[bit per second]，メインメモリ割当て量M[byte]，NIC (Network Interface Card) メモリ割当て量N[byte]が直接計算できるものでなければならない．通常NICの搭載するメモリの容量には限界があり，Nの値をさほど大きくすることはできない．

また，2章で述べたように，連続メディアには制約時間があり，制約時間として送信スレッドから受信スレッドに届くまでの時間Dを考えたときに，Dの要求値が資源の予約量に影響を与える．時間Dは，送信ホストでの処理遅延時間 D_s，伝送遅延時間 D_t，受信ホストでの処理遅延時間 D_r の和からなる．図4,5に示すように，D_sの要求値が大きければバーストをならして全体のスループットを確保するだけでよいが，D_sの要求値が小さいと，バーストを完全に吸収するだけの資源が必要になる．

送信スレッドの周期をP[s]としたときに，バースト周期がT*P[s]で出現するものとし，各周期毎に発生するデータ量を $a[i]\ i = 0, 1, 2, ..., T-1$ とする．このバース

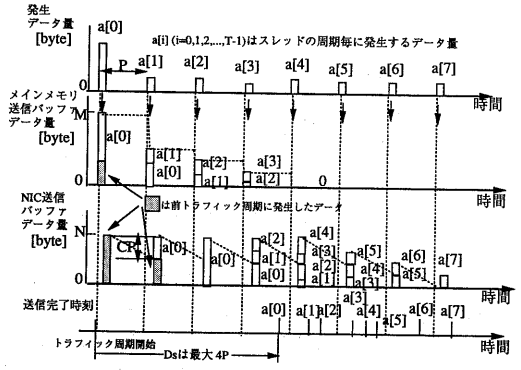

図 4: スループットを満たす最低の帯域量とした場合の送信バッファおよび遅延

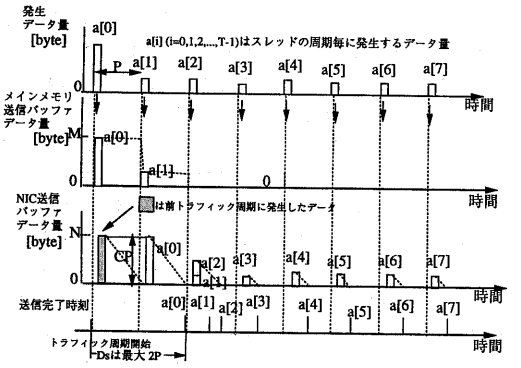

図 5: 帯域量を大きくした場合の送信バッファおよび遅延

ト周期を送信スレッド，受信スレッドの周期と区別して，バースト周期と呼ぶ．本方式においては，以下のアルゴリズムで，送信側の C, M, N を決定する．

$$C = \text{Average}\left(\frac{\sum_{i=0}^{T-1} 8^* a[i]}{TP}\right) + \alpha\ ;$$

while (D_s を満たせない AND
C,N が限界値に達していない) {
$N = 2CP/8^*$;
$C+ = \beta$; }

$$M = \text{Max}_{i=0}^{T-1} \sum_{j=0}^{i-1} (a[j] - CP/8^*)\ ;$$

ここに，α は余裕パラメータ，8^* は，byte を bit へ換算するための値である．C または，N が限界値に達したら，限界の値を基準に資源を割り当てる．受信側のバッファ量 L は，受信スレッドの周期 P' に依存し，$L = 2(CP/8^*)(P'/P) + \gamma$ とする．γ はジッタ等変動要因を加味して決定する．

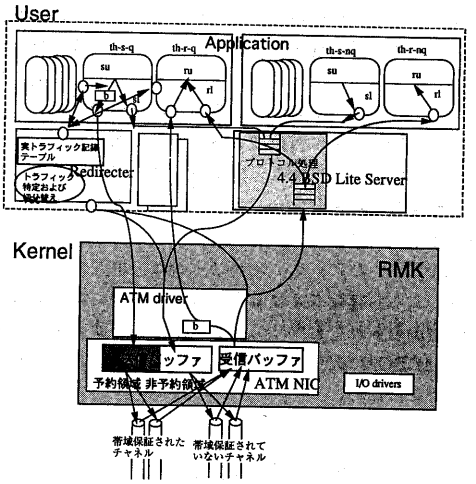

図 6: RT-Mach への実装モデル

3.3 ソケット・ライブラリ API

Winsock2.0 のように Native ATM Interface を用いることも考えられるが，ATM 以外への拡張，特に，IP ネットワークのスケーラビリティを重視して，UNIX および Windows で用いられている socket の形式を保持し，既存のアプリケーションが動作できるようにする．そのため，*socket*(), *bind*(), *sendto*(), *send*(), *receive*(), *recvfrom*(), *close*() 等のソケット・ライブラリ API をユーザ API として用いる．

4 RT-Mach への実装方法

第 3 章で述べた逐次 QOS 保証方式を実行するためのトラフィック特性特定，チャネル切替を行なうためのモジュールとして，Redirector を設け，RT-Mach のユーザ・タスクとして実装する．

4.1 RT-Mach3.0

RT-Mach 3.0[9] は，カーネギーメロン大学で開発された Mach 3.0 マイクロカーネルをベースに ARTS カーネルで開発したリアルタイムスレッド，リアルタイムスケジューラ，同期操作，タイマオブジェクト，クロックオブジェクト，リアルタイムプロトコルモジュールなどを取り入れてリアルタイム処理用に拡張したマイクロカーネルである．

RT-Mach3.0 上では，4.4 BSD Lite Server (Lites) や X ウインドウシステムを動作させることができる．

ネットワーク処理サーバとして，NPS(Network Protocol Server)[10] を用いることも考えられるが，X ウイドウシステム上で，連続メディアを扱う既存アプリケーションを動作させることも視野に入れて，本実装モデルにおいては，Lites を用いる．

4.2 ENI ATM ボード

ATM ボードとして，Efficient Networks Inc. (ENI) の PCI バス・ATM ボードを用いる．ENI の ATM ボードでは，最大 1024 個までの VC(Virtual Channel) がサポートされ，VCC 毎のトラフィック・シェーピングが可能である．本 ENI ATM ボードの FreeBSD 用ドライバコード [11] を RT-Mach に移植し，さらに逐次 QOS 保証方式を実現するための機能を付加した．

4.3 Redirector

図 6 に，RT-Mach への実装モデルを示す．Redirector というタスクを設け，通信路の切り替えを制御する．QOS 制御にしたがう送信スレッド th-s-q, 受信スレッド th-r-q と，QOS 制御にしたがわないスレッド 送信スレッド th-s-nq, 受信スレッド th-r-nq とでは，リンクされるライブラリが異なり (前者では l-q, 後者では l-nq)，th-s-nq, th-r-nq は常に Lites を介して，通信を行なう．

th-s-q では，socket 生成から，実行周期が確定するまでは，RT-Mach の UNIX エミュレーション機能を利用して，Lites へ処理を渡した後，ATM NIC 送信バッファの非予約領域を通して，帯域保証されていないチャネルに送信される．この間，Redirector は，th-s-q から，送信タイミングと，送信データ量を IPC (Inter Process Communication) により，収集する．Redirector は，実行周期が確定したと判断すると，ATM NIC 内の送信バッファに専用領域を獲得し，必要帯域のチャネルを生成し，相手 Redirector に対して，チャネル切替えの準備を依頼する．相手 Redirector で準備完了すると，Redirector は，th-s-q に対して，チャネル切り替えの指示をする．

このとき，アプリケーションは，切替えを認識する必要はなく，リンクされるライブラリ l-q 内で切り替える．前章で見た，N[byte] のバッファは l-q で保持し，送信バッファの予約領域が万杯になる分だけ転送し，残りは，次の周期の起動時に，転送を試みる．

受信側では，特に ATM NIC に予約バッファは設けないが，帯域保証されたチャネルから受けとったデータは，カーネル内でバッファリングし，th-r-q に渡す．

以上のようにして，Redirector は，送信スレッドのトラフィック特性の特定，送信チャネルの切り替え指示，相手 Redirector との間の制御情報の交換を行う．

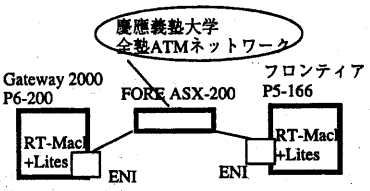

図 7: ATM ネットワーク実験システム

5 送信遅延時間の実測

評価項目として，チャネル切り替え時のオーバヘッド，トラフィック特性特定の精度，が挙げられるが，現在実装は進行中であり，ここでは，チャネルを切り替えたときの送信処理遅延時間の短縮効果について調べた結果を示す．図 7 の構成とし，2 台の計算機間で FORE 社 ASX-200 を介してデータの授受を行い，送信処理の時間をフロンティア社の Pentium 166MHz の計算機上で Pentium Counter [12] の値を読むことで測定を行った．送信処理の開始は，Lites を経由させる場合には sendto システムコールを発行する時点，Lites を経由させない場合には RT-Mach の device_write を呼び出す時点とし，終了は ATM ドライバに送信完了割込みが入るときとして，この間を送信処理時間とした．両者ともに伝送帯域量は 155 Mbps とした．

表 1 に結果を示す．送信ノードだけにおいても，送信トラフィックが特定できた後に，Lites を通さないことにより，送信処理時間を 1/4 ほどに縮まることが期待される．実際の実装においては，送信スレッドが直接 device_write を呼び出すのではなく，sendto に対応したライブラリで行うので，表 1 に示す値以外のオーバヘッドが生じる．

送信処理時間 (単位 ms)

データ量 [Byte]	64	1024
Lites を経由する場合	1.503	1.593
Lites を経由しない場合	0.489	0.572

6 議論および今後の予定

ATM ネットワークを用いてメモリ資源を予約する点に関して，本研究に関連する研究として Chorus による QOS 制御 [2] の研究がある．[2] においては，必要な受信バッファを見積もっているが，送信バッファでは 1 バッファとして次の周期までにすべて送りきるものとしている．この方法には，MPEG-2 のような周期的バーストと要求遅延との関連，チャネルの帯域量獲得に関する考慮がなされていない．

マイクロカーネル・アーキテクチャにおいて，プロト

コル処理をスレッドで直接行なう方法は，既に行なわれているが[15]，帯域保証型のネットワークにおいては，帯域が確定しない限り直接ネットワークへ送出することができず，トラフィック特性が確定するまではサーバで処理する方が自然である．

今後の予定として，トラフィック特性特定，トランスポート層相当プロトコルの実装，他入出力との連係が挙げられる．本稿では，MPEG-2でエンコードされたバースト性に周期性のあるデータを対象として，トラフィック特性の特定を想定したが，特定として実際の他のVBRデータでの有効性を確認する必要がある．

Litesから処理を切り替えたときのトランスポート層相当プロトコルを考える必要がある．UDP (User Datagram Protocol) データの切り替えに対しては，チェックサムを確認するだけでよいが，TCP (Transmission Control Protocol) データの切り替えに対しては，いくら信頼性が高い帯域保証されたチャネルを使用したと謂えども，伝送誤りは皆無ではない．TCPほど頑強でなくとも，トランスポート層相当プロトコルが必要になる．

さらに，カメラでキャプチャしたデータをMPEG-2フォーマットへエンコードするための送信側での入力処理，デコードしたデータを表示するを表示したりディスクへ保存する

7 まとめ

以上，ATMを例にとり，OSとしてRT-Machを使用したときの，帯域保証される通信環境でのQOS制御について述べた．マイクロカーネルOSで，トラフィック特性が確定し次第，プロトコル処理をサーバで行うよりも，スレッドで実行する方が，処理遅延時間が短くなるのみならず，連続メディア転送のQOSを保証するという観点からも重要であり，ライブラリ部分における処理オーバヘッドが短くなれば，かなりの効果が期待できる．ATMのチャネルを切り替えたときのオーバヘッドおよび問題点について，慶應義塾大学全塾ATM実験ネットワーク[16]により，明らかにしていく予定である．

謝辞

本研究を行うにあたっては，MKngプロジェクトの皆さまから，多大な助言をいただきました．ここに感謝申し上げます．

参考文献

[1] C. Aurrecoechea, A. Campbell, and L. Hauw, "A Survey of QoS Architectures," In *Multimedia Systems Journal, Special Issues on QoS Architecture*,1997.

[2] G. Coulson, A. Campbell, P. Robin, G. Blair, M. Papathomas and D. Hutchison, "The Design of a QoS Controlled ATM Based Communications System in Chorus," In *IEEE JSAC Special Issue on ATM Local Area Networks*, 1994.

[3] K. Kawachiya, K. Ogata, N. Nishio, and H. Tokuda, "Evaluation of QOS-Control Servers on Real-Time Mach," In *Proc. of the 5th Int. Workshop on Network and Operating Systems Suport for Digital Audio and Video 123-6*, 1995.

[4] N. Nishio, H. Tokuda, "Simplified Method for Session Coorditaion Using Multi-level QOS Specification and Translation," In *Proc. of IWQoS'97*, May 1997.

[5] H. Tokuda, Y. Tobe, S.T.-C. Chou and J. M. F. Moura, "Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network," In *Proceedings of ACM SIGCOMM '92*, August, 1992.

[6] 橋本浩二, 渡辺光輝, 柴田義孝, "連続メディアサービスのためのQoS保証及び交渉機能," In 情処研報 *96-DPS-74, Vol. 96, No.12*, Jan, 1996.

[7] L. Zhang, S. Deering, E. Estrin, S. Shenker, and D. Zappala, "A New Resource ReSerVation Protocol," In *IEEE Network, Vol. 7, No. 5, pp. 8-18*, Sep., 1993.

[8] C. Partridge, "A Proposed Flow Specification," *RFC 1363* Sep., 1992.

[9] H. Tokuda, T. Nakajima, and P. Rao, "Real-Time Mach: Towards a Predictable Real-Time System", In *Proceedings of USENIX Mach Workshop*, October, 1990.

[10] T. Nakajima, T. Kitayama and H. Tokuda, "Experiments with Real-Time Servers in Real-Time Mach," In *Proc. of 3rd USENIX Mach Symposium*, Arp. 1993.

[11] Cranor, http://www.ccrc.wustl.edu/pub/chuck/bsdatm/wucs.html.

[12] T. Mathisen, "Pentium Secrets," In *Byte magazine, http://www.byte.com/art/9407/sec12/art3.htm*

[13] N. Matsui, H .Tokuda, "VoR:Network Subsystem Framework for VBR over Reserved Network," *submitted for publication* , 1997.

[14] N. Matsui, "Network Control for Variable Bit Rate Continuous Media," 慶應義塾大学大学院政策メディア研究科修士論文, Mar., 1997.

[15] C. Maeda and B. N. Bershad, "Protocol Service Decomposition for High-Performance Networking," In *Proc. of the 14th Symposium on Operating Systems Principles*, 1993.

[16] 南部明, 小野定康, 徳田英幸, "超高細画像マルチメディアと超高速コンピュータネットワーク実験," In 画像電子学会誌, *No. 25, Vol. 4*, 1996.