オーディオビジュアル複合情報処理 11－3
（1995. 12. 7）

マルチメディアプラットフォームにおける ネットワークプロトコル[1]

木原 誠司　尾上 裕子　盛合 敏　南部 明　徳田 英幸
NTT 情報通信研究所　慶應義塾大学

マルチメディア処理の中でも，動画や音声に代表される，時間軸方向に連続な性質を持つ連続メディアの処理が重要な課題である．本稿では，連続メディア処理のためのプラットフォームにおける，ネットワーク通信プロトコルについて述べる．そして，前もって資源を予約するプロトコルである ST-II の実装と，動的な QOS 制御を行なうプロトコルである MPC の設計と実装について説明する．

Network Protocols for Multimedia Platforms

Seiji Kihara, Yuko Onoe, Satoshi Moriai, Akira Nambu, and **Hideyuki Tokuda**
NTT Information and Communication Systems Laboratories　Keio University
{kihara,yuko,moriai,nambu}@isl.ntt.jp　hxt@sfc.keio.ac.jp

Continuous media, such as digitized audio and video, has timing constraints, so it is important to handle them on multimedia systems. This paper describes network communication protocols suitable for continuous media. It also describes an implementation of ST-II, a protocol which reserves resources in advance, and explains MPC, an original protocol which controls resources dynamically.

[1]この研究の一部は，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」プロジェクトの研究協力として行われたものです．

1 はじめに

近年マルチメディア処理に向いたコンピュータシステムの研究開発が盛んである．様々なメディアの中で，最も処理が難しいのは，連続メディアと呼ばれる，時間軸方向に連続的な性質を持つストリーム状のメディアであり，デジタル化した音声，動画などに代表される．

連続メディアを扱うためには，システムはアプリケーションが要求するQOS(Quality Of Service)を保証しなければならない．アプリケーションから指定されるQOS，例えば動画の色数，ピクセル数，秒あたりのフレーム数などは，帯域幅・パケット長・遅延時間・ジッター・誤り率などの，通信レベルのQOSに変換される．ネットワークプロトコルにおいては，これらのQOSをエンドツーエンドで保証することが要求される．

加えて，動画のように帯域幅の広いメディアを転送するためにはプロトコル処理が高速である必要がある．音声と動画とでは帯域幅が大きく異なるため，様々な帯域幅に対応できることが必要となる．例えば移動ホストに対する対応などの要求から，異種混在のネットワークへの対応が望まれる．さらに，動画や音声などのアプリケーションには1対1ではなく複数のメンバからなるグループ間の通信を必要とし，動画や音声などの巨大になりやすいデータを転送するためには，マルチキャスト機能が必要である．

本稿では，連続メディアのためのネットワーク通信プロトコルに関して述べ，著者らが行なっているプロトコルの実装について説明する．第2節においてネットワーク通信プロトコルに求められる条件について述べ，第3節では物理層からネットワーク通信プロトコルに至るまでのプロトコルアーキテクチャについて述べる．第4節でネットワーク通信プロトコルの実装について述べる．

2 ネットワークプロトコルに求められる条件

第1節で述べたように，マルチメディア処理のネットワークプロトコルには，広帯域ネットワークのサポート，QOSの保証，異種混在ネットワークへの対応，マルチキャスト通信機能などが要求される．

広帯域ネットワークのサポートのためには，高速に処理されること，通信量をうまく制御することが要求される．回線を効果的に用い，かつ輻輳を避けるためには，ウインドウ制御のようなフロー制御機能が必要であるが，ウインドウ制御では広帯域ネットワークでのスループットが劣化する．そこで，ネットワークの負荷やCPUの負荷に応じて動的に最適な帯域幅に合わせることが望まれる．このような適応型フロー制御のひとつとして，モニタリングの結果から自らの使用帯域幅を制御するフロー制御がある．このようなフロー制御はセルフスタビライゼーションの一種ということができる．

連続メディアのように，広帯域を継続して必要とする通信のために，前もった資源予約(advance resource reservation)が重視されている．たとえば圧縮しない音声のように，帯域幅が一定のデータの転送には，前もって帯域幅を予約することにより，効果的に資源を活用することができる．

しかし，資源予約方式では，例えばMPEG圧縮された動画のように，使用する資源が時系列的に変化するような場合，最大値で資源を予約してしまうと，資源の利用率が下がってしまうという問題がある．アプリケーションからのQOS指定において，最良値と最悪値が指定できる場合には，その範囲で実際の割当を動的に変化させることも可能である．したがって，資源予約方式と，適応型フロー制御との組合せも重要となる．

異種混在環境における通信は，マルチキャストの場合に複雑になる．異種のネットワークや通信相手が混在する場合には，送るべき帯域幅も異なる．ひとつのマルチキャストツリーにおいて，複数のQOSを同時に扱えると，送信アプリケーションはひとつのデータを送るだけで済むため，アプリケーションの負荷削減に貢献する．複数のQOSを柔軟に扱うためのひとつの技術に，メディアスケーリングがある．

これらのうち，著者らは，資源予約方式のプロトコルとしてST-IIを実装し，適応型フロー制御と異種混在環境への対応としてMPCを設計および実装している．詳細は第4節で述べる．

3 プロトコルアーキテクチャ

本節においては，まず物理層，データリンク層について説明し，その上で，資源予約をサポートするネットワークプロトコルについて説明する．プロトコルの階層を図1に示す．

3.1 物理層，データリンク層

物理層およびデータリンク層としてよく使われているものに，イーサネット，FDDIがある．また，ATM(Asynchronous Transfer Mode, 非同期転送モード)技術を用いたネットワークが注目されている．

イーサネットは，物理媒体を共有したネットワークで，CSMA/CD方式を採用しているため，ネット

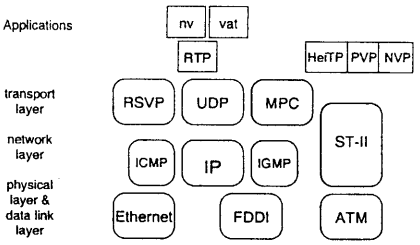

図 1: プロトコル階層

ワークの利用率が上がるにつれて，またノード数が増えるにつれて，全体として使用できる帯域幅が小さくなる傾向がある．最近では，イーサスイッチにより，ノード数の増加による使用帯域幅の減少を防ぐことが可能になっているが，転送にかかる最悪時間を決定することはできず，時間制約を持つ連続メディアデータの転送には向かない．また，100BaseT は 10BaseT の転送速度をちょうど 10 倍にしたものであるため同じ性質を持ち，100VG AnyLAN (IEEE 802.12) はパケット送出の遅延時間を予測可能にしているが，コネクションの概念がないため，上位プロトコルにおける帯域幅の制御をサポートできない．

FDDI は，トークン周回時間の一定部分に特定の通信を割り当てる同期転送モードを使うことにより，特定の通信の最悪転送時間を保証できるが，これはオプションとなっており，サポートするハードウェアはほとんどない．また，物理媒体を共有しているため，接続されているノードすべての合計で帯域幅が 100Mbps という制限がある．

ATM においては，通信のトラフィックパターンに応じて，資源の確保が可能である．資源予約を行なう時には，以下のうちひとつを指定する．

- Constant Bit Rate (CBR)
  常に一定の帯域幅を用いる．例えば無圧縮の動画，音声に用いられる．
- Variable Bit Rate (VBR)
  帯域幅は時系列的に変化するが，サンプル間には固定のタイミング関係がある場合．圧縮された動画などに用いられる．
- Available Bit Rate (ABR)
  可変ビットレートであり，基本的にはベストエフォートであるが，最小帯域幅を予約することができ，輻輳がない時には割当を大きく，輻輳がある時には割当を小さくするような帯域幅制御を行なう．ただし，現時点では実現されていない．
- Unknown Bit Rate(UBR)
  ベストエフォートにて通信する．予約は一切行なわれない．

CBR を除き，統計多重的に予約を行なうため，使用資源が実際の資源を上回る場合には，最悪の場合交換機によりセルを消失させる．

ATM で，あるノードと直結するノードとの間のリンクは，交換機以外のノードについては，それに関係するトラフィック以外流れることがなく (物理媒体を共有しない)，各ノードで独立して使うことができる．したがって，比較的大きな帯域幅が必要なマルチメディアデータ，特に連続メディアに向いているということができる．

以上の性質から，今後マルチメディア向けネットワークとして，ATM が期待される．しかし，通信路 (Virtual Connection) の設定は現在交換機に静的に行なう (Permanent Virtual Connection) か，または交換機とノードとの間の特別の通信 (シグナリング) によって行なわれる (Switched Virtual Connection) が，帯域予約機構とシグナリングとの組み合わせが現在ではなされていない．

3.2 ネットワーク通信プロトコル

連続メディアが持つ時間制約を満たすためには，実時間通信機能を持ち，さらに資源予約機能を持つことが望まれる．これらの例として，ST-II，RSVP について説明する．

3.2.1 ST-II

ST-II(Internet STream protocol, version 2)[7] は，IP と同じ層に属すネットワーク層のプロトコルである．送り手が 1 つ，受け手が任意の数であるマルチキャストツリーの上で，帯域幅をあらかじめ確保し，その帯域幅を守るようにデータの送信を行なう．静的に予約を行ない，予約帯域幅を超えた利用は基本的にはできない．資源予約を行なうために，送信側から，QOS の最良値と最悪値を含んだ予約メッセージを，マルチキャストツリーを通じて受信メンバに向かって送信する．その間に中継ノードやネットワークの資源予約が行なわれる．受信メンバから実際に予約された QOS が返され，送信ノードはすべての QOS のうち最も低いものに合わせて送信を行なう．したがって，ツリー全体の QOS は等しい．

ST-II を改訂した ST2+[2] では，次に説明する RSVP のような，送信側が受信メンバすべてを把握する必要のない，制約の小さなマルチキャストグループを許容できるようになっている．

ST-II は予約機能とともに，予約された帯域幅を用いた通信の機能も持つ．IP よりも小さなヘッダを持

ち，かつ中継ノードにおける転送処理が簡単になるようにコネクション設定時に情報交換を行なう．したがって，高速通信が可能である．

3.2.2 RSVP

RSVP(ReSerVation Protocol)[8]は，マルチキャストツリー上で，単方向にネットワーク資源を予約するためのプロトコルである．主にIPマルチキャストの上位層として使われる．実際のデータの転送はIPによって行ない，RSVPは予約のみを行なう．

RSVPの大きな特徴は予約を受信側から行なうことである．送信側がすべての情報を持つ必要がないため，大規模なマルチキャストツリーの場合など，メンバの追加変更が頻繁に起こるような場合にも適応可能である．もうひとつの特徴は，パケットフィルタ機構である．このことにより，ひとつのマルチキャストグループ内のメンバに対しても，異なる資源を割り当てることができる．資源予約時に，受信側から送信側に向かって渡されるフィルタが中継ノード(ルータ)で保持され，通信時に用いることができる．

RSVPは，IPネットワークでの資源予約のみサポートし，予約された資源の利用(データの通信)は行なわない．

4 プロトコルの実装

本節では，著者らによるST-IIの実装およびMPCの設計と実装について述べる．連続メディアの処理のためには，前もって資源予約をして通信時に予約された資源を用いるプロトコルが必要であり，また動的なQOSの変化に対応できるプロトコルも必要である．そこで，前者としてST-IIを実装し，後者として新たにMPCというプロトコルを設計，実装している．

4.1 プロトコル処理の実装位置

ST-IIを実装するにあたり，Real-Time Mach[6]の機能を使うことにした．Real-Time Machなどのようにマイクロカーネルを採用しているシステムは，サービスはユーザレベルでできるだけ行なうという思想を持っている．そこで，実現する位置によるトレードオフについて説明する．

UNIXをはじめとするオペレーティングシステムではユーザモードとカーネルモードがあるため，プロトコルの処理をどちらで行なうか選択する必要がある．また，ユーザレベルの場合には，別のサーバとして実現するか，アプリケーションにリンクするライブラリとして実現するかの選択があり，合計3通りの

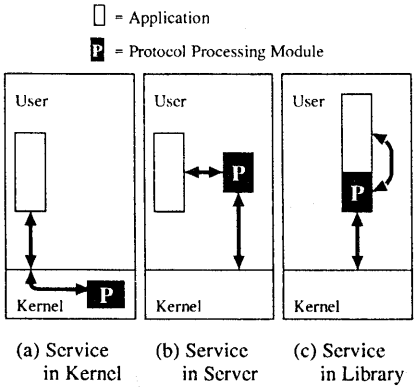

図 2: プロトコル処理の形態

方法があり得る．それらを図2に示す．

これらについて，サービスの実行性能，プログラミングや保守の容易性，サービスの安全性などについて比較する．

カーネル内実装 (図2(a))

アプリケーションからの要求があると，カーネルモードに遷移し，ユーザ空間の切り替えは起こらない．このため，コンテクスト切り替えが不要であり，データの複写は一度で済むため，高速な動作が可能である．しかし，プログラミングや保守が困難であることが問題である．サービスの安全性については，すべてカーネル内に存在するためユーザプログラムから守ることができるため，問題は少ない．

4.3BSDのTCP/IPがカーネル内実装の代表的な例である．しかし，4.3BSDの性質上，複写が2回入るため，高速なネットワークへの対応という面で問題がある．

ユーザレベルのプロトコルサーバ (図2(b))

ユーザレベルで動作する，単一のサーバとして実装されたものである．アプリケーションから要求があると，サーバに制御が移り，多くの場合カーネルに実装されているネットワークデバイスドライバに要求を出すため，カーネルモードに遷移する．タスク間のコンテクスト切替えや，データの複写が起こるため，性能に問題がある．ただし，ユーザレベルでデバイスドライバを実現することにより，性能低下を軽減することも可能である．プログラミングや保守は容易である．サービスの安全性に関しては，カーネルにより，他のアプリケーションからのデータの保護が行なわれるため，例えば無関係なアプリケーションの通信の情報を盗み見られることはない．デバイスなどのシステムの低位の資源を，ユーザ空間にあるサーバに見せる

必要があるが，ケーパビリティによる保護も可能である．

ユーザレベルサーバとしての実装例には，TCP/IPも含む形で実装されているMach 3.0のUXサーバ(4.3BSDのエミュレータ)，Real-Time Machの実時間スレッドと実時間同期機構を用いてネットワーク通信まで含めた優先度逆転の防止をサポートするプロトコルツールキットであるNPS (Network Protocol Server)[10]などがある．

ユーザレベルライブラリ (図2(c))

プロトコル処理がアプリケーションの中で行なわれる．したがって，プロトコル処理のために空間の切替を必要としないため高速動作が可能であり，またカーネル内のプログラミングに比べてプログラミングや保守がしやすい．しかし，デバイスなどのシステムの低位の資源へのアクセスを，アプリケーションに対して許す必要があり，情報の保護の面で問題がある．また，プロトコル処理全体で共有する情報をどこに持たせるか問題があり，アプリケーション内に持たせる場合には，情報の保護の面で問題がある．

このため，すべてをライブラリで行なうのではなく，安全性が必要な情報を扱う機能やプロトコル全体で共有する機能を独立したサーバとし，受信パケットの振り分けをカーネルによって行ない，他の大部分をユーザレベルライブラリとして構成することが多い[3, 5]．

なお，ライブラリによる実現では，アプリケーションのオブジェクトコードの量が増加するが，共有ライブラリを用いることにより解決できる．

4.2 ST-II サーバの実装

著者らは，ST-IIをReal-Time Mach 3.0のユーザレベルサーバとして実装している[9]．ユーザレベルサーバとしたのは，ST-IIはコネクション確立が複雑であり，送受信においてコネクション確立時に得られた情報を必要とするため，ユーザレベルライブラリとしての実装が困難であると判断したためである．

コネクション確立と管理のためのスレッドと，送受信のスレッドを，ひとつのタスク内の別のスレッドとして実装している(図3)．高速化のため，サーバ内のスレッド間では実時間同期機構で保護された共有空間で情報の受渡しを行ない，サーバとカーネル，アプリケーションの間は，仮想ページを交換する機構[11]を用いる．

ST-IIのように帯域を予約するネットワークプロトコルにおいては，ATMのように，ネットワークに帯域予約機構があり，それを管理するネットワーク資源マネージャと協調する必要がある．それ以外に，

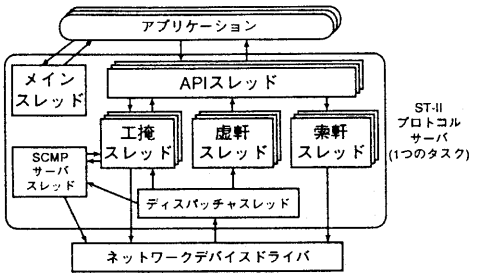

図 3: ST-II サーバの実装

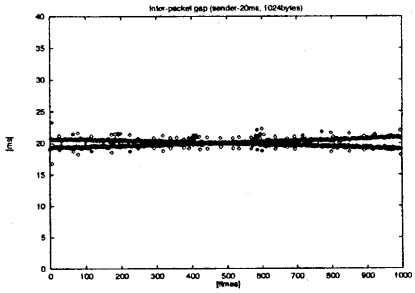

図 4: ST-II サーバの評価実験の結果

ローカルな資源，すなわちCPU資源やバッファ領域などのメモリ資源が必要となる．CPU資源の予約のためには，一定のCPU資源をプロトコル処理に割り当てる必要がある．ST-IIサーバにおいては，Real-Time Machの周期的スレッドを用い，アプリケーションから要求されるパケット間隔と，送信スレッドの実行周期とを合わせることにより，要求されたパケット間隔を守る．

ST-IIの実装を評価するため，簡単な評価実験を行なった[9]．20ミリ秒周期でデータを送出し，受け側でデータの到着間隔を測定した結果を図4に示す．このグラフより，到着間隔のずれが2ミリ秒以内に収まっていることがわかる．

4.3 MPC

著者らは，MPC(Multicast Protocol for Continuous Media)[4, 12]を設計，実装している．第2節で述べた必要条件のうち，マルチキャストツリーで複数のQOSをサポートすることと，さらに資源を有効に利用するために，適応型フロー制御機能を実現する．資源予約にRSVPを利用し，加えてベストエフォートで用いることができる資源を有効に利用するために，動的に使用資源を変更する機構を持つ．具体的には，アプリケーションから，階層的エンコーディン

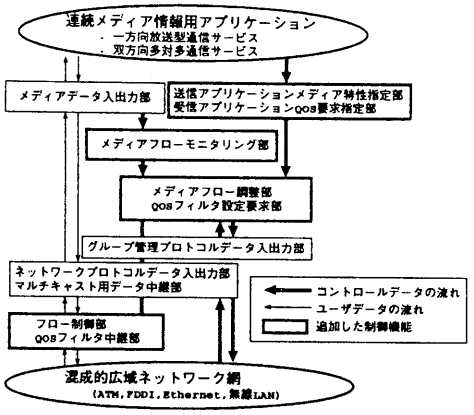

図 5: MPC の実装

グ (MPEG の I フレームだけ, I+P, I+P+B など) に対応させた QOS レベルを受け取り, 送信ノードおよび中継ノードにおいて, 送り先のネットワークや CPU などの資源をモニタリングし, その状況に応じて, QOS レベルを変化させ, 対応したデータを送出することにより, 使用帯域を変化させる.

現在 MPC は, IP, RSVP と組み合わせ, UNIX (主に BSD 系) のカーネル内に実装している (図 5).

IP のように資源予約の概念がないプロトコルを用いる場合には, その下の階層のネットワークが資源予約の機能を持っていても, 使うことはできない. したがって, ST-II のようにデータ転送時にまで予約資源を保証するプロトコルを実現するためには, IP の上に実装するのではなく, 資源予約機能のあるネットワークの上に直接作る必要があり, 例えば, ATM の上に実装することになる. しかし, コストパフォーマンスの問題や使用環境の問題などから, すべて ATM のようなネットワークに移行することは難しい. 一方, IP の上にプロトコルを実現する場合には, 既存の環境との親和性が高く, またネットワークの変化に追従しやすく, また将来 IP に変わるものが資源予約をサポートする場合には, それを使うことが可能である. したがって, MPC のように, IP と RSVP と組み合わせ, 適応制御により実用的に動作させる方法には柔軟性がある. 両者のアプローチとも重要であると考えている.

5 おわりに

本稿では, マルチメディアを扱う上で重要となる, 連続メディア向きのネットワーク通信プロトコルと, その実装上の諸問題について述べた. そして, 前もって資源を予約するプロトコルとして ST-II, 動的な QOS 制御を行なうプロトコルとして MPC について, その設計と実装を中心に述べた. 実装のプロトタイプはフリーソフトウェアであり, そのスナップショットが慶應義塾大学マルチメディア統合基盤ソフトウェア (Keio-MMP) プロジェクトの CD-ROM[2] に統合されている.

今後, ST-II サーバ, MPC とも完成度を高め, ST-II サーバについては ST2+ への対応を検討している. MPC では IP バージョン 6 への対応と, RSVP を含めて Real-Time Mach サーバとして実現することが, 今後の課題である.

参考文献

[1] S. T.-C. Chou and H. Tokuda. System Support for Dynamic QOS Control of Continuous Media Communication. In *Proc. of the 3rd NOSSDAV*, pages 322–327, November 1992.

[2] L. Delgrossi and L. Berger. Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+. RFC 1819, 1995.

[3] C. Maeda and B. N. Bershad. Protocol Service Decomposition for High-Speed Networking. In *Proceedings of the 14th ACM SOSP*, pages 244–255, December 1993.

[4] Y. Onoe, K. Fujii, and H. Tokuda. QOS-based Multicast Communication. In *Abstracts of the 2nd Workshop on Advanced Teleservices and High-Speed Communication Architectures*, 1994.

[5] C. A. Thekkath, T. D. Nguyen, E. Moy, and E. D. Lazowska. Implementing Network Protocols at User Level. In *Proceedings of the ACM SIGCOMM '93*, pages 64–73, October 1993.

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

[7] C. Topolcic. ST-II. In *Proc. of the 1st NOSSDAV (International Workshop on Network and Operating System Support for Digital Audio and Video)*, 1990.

[8] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, pages 8–18, September 1993.

[9] 木原, 盛合, 南部. ST-II プロトコルサーバの Real-Time Mach への実装と評価. 情処研報, 94(OS-65):81-88, July 1994.

[10] 中島, 徳田. 予測可能な通信を提供するユーザレベル・ネットワークシステム. 日本ソフトウェア科学会第 *10* 回大会論文集, 1993.

[11] 盛合 敏. リアルタイムオブジェクトむきの IPC について. 情処研報, 93(OS-60, DPS-61):9–16, 1993.

[12] 尾上, 藤井, 徳田. マルチキャストサーバにおける動的 QOS 制御. 情処研報, 94(DPS-66):1–6, 1994.

[2] `http://www.mmp.sfc.keio.ac.jp/` 参照のこと.