1K−1

MMC(Multimedia Control) アーキテクチャの提案*

佐藤 龍雄　田中 賢一朗　上野 幹大　林 正薫　米田 健　松下 温†
慶応大学 理工学部‡

1 はじめに

近年マルチメディアを応用した様々なアプリケーションが開発されている。マルチメディア情報を扱う以上、音声や動画像の処理が不可欠であるが、技術的な面やコストの面で制約が多く、いまだ一般的なワークステーションで自由にマルチメディアアプリケーションが構築できるレベルに至っていない。

しかし、課題は多いながらも、近い将来、マシンの性能向上と超高速大容量ネットワークの出現によって、一般的なワークステーションでもマルチメディア情報を扱えるようになると思われる。

実際には、EtherNet(10Mbps) を介して 20〜30MIPS 程度のワークステーション間で音声処理 (8kHz サンプリングの場合) を行なう場合はほとんど問題はなく、動画像では 320x240pixel のカラー画像 (8bit/pixel) を毎秒 5〜8 フレーム程度の表示できることを確認している。

現状では、マルチメディア処理のための開発環境 (記述言語やライブラリ) は存在していないため、アプリケーションを構築するたびに多大な時間とコストがかかってしまう。そこで、マルチメディア処理の共通基盤の整備が望まれるようになってきた。

我々は、人間にとってわかりやすいマルチメディア処理のモデル化をおこない、多様なマルチメディアの要素に対応したアプリケーションを比較的簡単に開発できることを目的とした共通基盤として MMC(Multimedia Control) を提案する。実際に、MMC の機能を実現するために、クライアントサーバモデルに基づいたシステムを実装した。このシステムを用いることにより、分散環境に対応したマルチメディアアプリケーションの開発が容易に行なえるようになった。

2 MMC が提供する機能

2.1 continuous data の性質

音声や動画のような情報 (continuous data) は一定時間内に処理されるべき量が決まっている点で通常のデータとは扱いが異なる [1,2]。また、通常のデータだと 1 ビットの狂いも許されないが、continuous data の場合は、人間が知覚できなければ多少のビット誤りや欠落が許される。

2.2 デバイス

動画や音声など様々なメディアを処理する実体を “デバイス” とよぶ。デバイスは、continuous data を生成するデバイス (provider) と消費するデバイス (consumer) とに分類できる。

我々は以下のようなデバイスを実装した。

MMC_MIC,MMC_SPEAKER: 音声の取り込み、再生。
MMC_AUDIO_MIXER: 複数の音声を 1 つにミックスする。
MMC_AUDIO_FILE: 音声ファイルから読み込む。
MMC_CAMERA: ビデオカメラの映像を取り込む。
MMC_WINDOW: ウィンドウに画像を表示する。
MMC_VIDEO_FILE: 動画像ファイルから読み込む。
MMC_CHANNEL: ネットワークを通じたデータの授受。

2.3 バッファ管理

MMC では、全ての continuous data を “frame” という単位で扱う。provider デバイスは frame 単位でデータを生成し、バッファへ移動する。consumer デバイスはバッファからデータをとりだし、処理する。単位時間あたりの consumer デバイスの処理量が provider デバイスの処理量を越えると、frame 単位で continuous data を欠落させることにより速度の調整を行なう。この仕組みは、リングバッファ構成のため、非常に単純となった。

---

*An Advanced Multimedia System Architecture in the Distributed Environment

†Tatsuo SATO, Ken-ichiro TANAKA, Mikihiro UENO, Joung-hoon LIM, Takeshi YONEDA, Yutaka MATSUSHITA
‡Faculty of Science and Technology, KEIO UNIVERSITY

2.4 デバイスの接続

2つのデバイスを “接続 (connect)” することにより、デバイス間で continuous data の移動が開始される。データの移動は、接続時に2つのデバイス間でとり決めたパラメタに従って、定期的に自動的になされる。例えば、MMC_SPEAKER に MMC_MIC を “connect” すると、マイクへしゃべった声がスピーカから流れる。

2.5 デバイス間のスケジューリング

各デバイスは独立に動作すると考えるが、通常の計算機では完全に並列に動作させることができないので、デバイスを適当にスケジューリングしながら動作させる必要がある。基本的にはデッドラインスケージューリングを行なう。

3 クライアントサーバモデルによる MMC の実現

MMC 機能の提供者をサーバ、利用者をクライアントとして、別々のプロセスとして実行することにした (図1)。サーバとクライアントは同一のマシンに存在する必要はないが、1つのマシンにつき1つのサーバプロセスしか許さない。クライアントプロセスはいくつ存在してもよい。

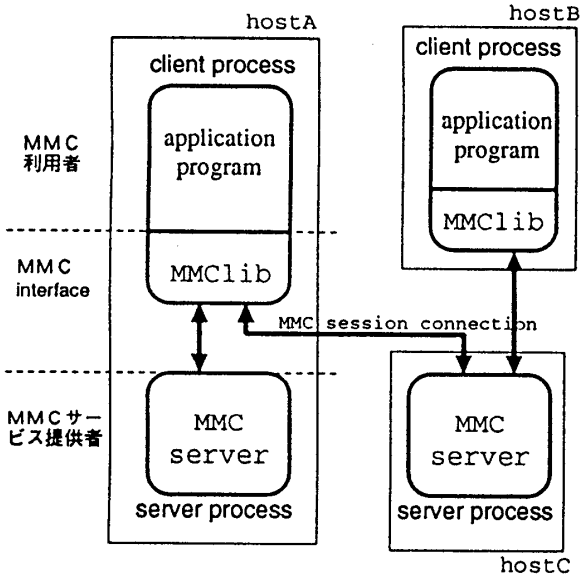

図 1: MMC システムの構成

MMC 機能を提供する MMC server は UNIX 上のプロセスであり、I/O をすべて non-block で使用することで1プロセス内での複数処理の並行動作を実現している。音声処理は SUN Sparc ワークステーションに標準装備しているオーディオデバイスを使用した。また、画像関係の処理には X Window システムを利用した。

4 MMClib: C 言語インタフェースライブラリ

クライアント側では MMClib という C 言語のライブラリを用いてサーバに対して要求を行なう。クライアントアプリケーション作成者は MMClib の関数を使用して MMC の機能を利用する。

MMClib の基本的な関数として次のようなものが挙げられる。

mmc_open(), mmc_close(): サーバとの接続、解除。

mmc_create(), mmc_destroy(): デバイスの生成、削除。

mmc_set(), mmc_get(): デバイスの属性設定、取得。

mmc_connect(), mmc_disconnect(): デバイスの接続、解除。

5 アプリケーションの試作

MMClib を用いてテレビ会議システムを作成した。この会議システムの特徴は、自分の端末である映像をみているときに、その映像をそのまま他人の端末に転送する（情報のリダイレクション）ことが直観的な操作で行なえる。

6 今後の課題

よくいわれるように、UNIX はリアルタイム OS ではない。したがって、他のジョブが実行中であると極端に性能が低下する。これは、MMC server がユーザプロセスの1つである以上避けられない。今後マシンの性能がさらに向上しても避けられない問題であり、OS がある程度のリアルタイム性を保証する必要がある。

参考文献

[1]T.D.C.Little, A.Ghafoor: “Multimedia Object Models for Synchronization and Databases,” Proc. IEEE 6th international conference on DATA ENGINEERING,pp.20-27,1990

[2]D.P.Anderson and G.Homsy,“Synchronization Policies and Mechanisms in a Continuous Media I/O Server”, Tech.Report No. UCB/CSD 91/617, UC Perkeley, 1991.