マルチメディア通信と分散処理　64－4
（１９９４．　３．　３）

パケットオーディオ・ビデオシステムの同期方法

瀬田直也 清水省悟 柴田義孝

東洋大学工学部 情報工学科

{seta,simi,shibata}@yosemite.cc.toyo.ac.jp

マルチメディア情報ネットワーク上で、動画像と音響データの同期を取りながら効果的なビデオ通信を行なうパケットオーディオ・ビデオシステムを構築するために、動画像及び音響処理方式、ネットワーク処理方式、また同期処理方式の設計・開発を行った。

ここでは特にその同期処理方式に着目し、UNIX-OS のプロセス及び Mach-OS のタスク・スレッドといった２つの異なる実装手段によりパケットオーディオ・ビデオシステムのプロトタイプを構築し、様々な負荷条件下におけるそれぞれの同期性能の評価を行った。

Synchronization Methods for Packet Audio-Video System

Naoya Seta, Shogo Shimizu, Yoshitaka Shibata

Department of Information and Computer Sciences
Toyo University

{seta,simi,shibata}@yosemite.cc.toyo.ac.jp

In order to realize Multimedia Information Network, we developed a efficient Audio/Video transmission system which takes account of each media characteristics and provides flexible synchronization mechanism between them. We have implemented this system on two different operating systems: UNIX-OS using Processes and Mach-OS using Tasks/Threads.

This paper describes synchronization accuracy and performance evaluation of the suggested synchronization schemes by both implementations under various load conditions.

1 はじめに

近年情報システムの発達により、画像や文字、音声といった異なるメディア・データを同時に表示再生可能なマシンを、個人ベースで利用することが可能になりつつある。これにより、それらのメディア・データを意味的に結合された情報としてユーザに提供するような、マルチメディア情報ネットワークを基本とするサービスの必要性が増加してきた。

そこで筆者らはこれまでに、パケットオーディオ・ビデオシステム (以下 PAVS) として、マルチメディア情報ネットワーク上での効果的なビデオ転送、つまり音響及び動画像データ間で同期 ("Lip Sync.") を取りながらの転送を実現するための転送プロトコルと、各メディアの特性を考慮した動画像及び音響処理方式、同期方式の設計及び実装を行ってきた [1, 2]。本稿では特に、UNIX のプロセス及び、Mach のタスク・スレッドといった 2 つの異なる実装手段により PAVS を構築し、様々な負荷条件下における同期性能について、各システムの評価を行ったので報告する。

2 PAVS

本システムは ISO の OSI 参照モデルを適用すると、図 1 に示すような階層構造をしており、クライアント・サーバ方式によって PAVS を構成している。

Synchronization 層では、動画像フレームと対応する音響セグメント間の同期処理を行ない、"Strict","Relaxed","Silence" の 3 種類の同期処理を選択的に利用できる。

Transform 層では、DB に格納されたメディアデータを各クライアント端末の属性に適合させるために、動画像処理としてカラーモード変換、圧縮・伸長などを行い、音響処理としてサンプリングレート・量子化ビット数・変調方式変換、無音検出などを行う。

Media Flow Control 層では、サーバ・クライアント間のビデオフレーム及びオーディオセグメント転送におけるパケット転送レート制御、またバッファの溢れを監視することによるフロー制御を行う。

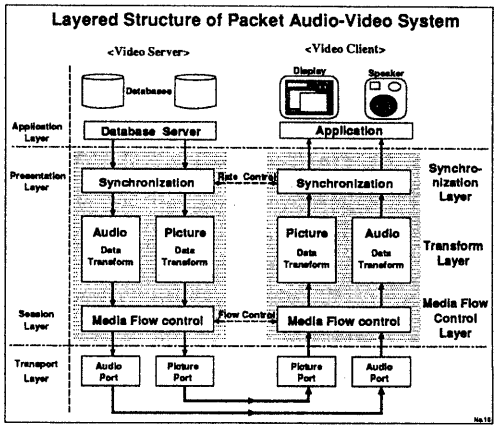

図 1: PAVS の階層構造

2.1 フレームレート制御

本システムでは、クライアント側のレートコントローラーがビデオデータの再生状態を常に監視し、実際の再生レートを設定レートと比較し、もしそれが低下していればサーバ側にレート制御メッセージを送信する。サーバ側ではフレームデータを間引くことにより、クライアントから要求されたフレームレートでの処理を行い、動的なレート制御を行う。またそれに合わせてクライアント側では、同期を行う周期の調整も行なわれる。

2.2 同期方式

本システムでは、動画像フレームと音響セグメント間の同期方式として、同期ポイントの決め方の違いにより以下の 3 種類の方式 [3] を選択できる。

Strict Synchronization: 動画像データのフレームレートと音響データのセグメントレートを完全に一致させ、対応した動画像フレームと音響セグメントの同期をとる方式である。つまり動画像1フレーム毎に、対応する音響1セグメントとの表示・再生の開始時刻を揃える方式である。例えば、N_P [frames/sec] の動画像データの場合は、理論的に $1/N_P$ [sec] の同期間隔となる。

この方式の特徴は、フレーム又はセグメント単位の細かな同期が可能な事である。また、1秒あたりの同期ポイントの数は、ここで提案する3方式のうちで最大である。

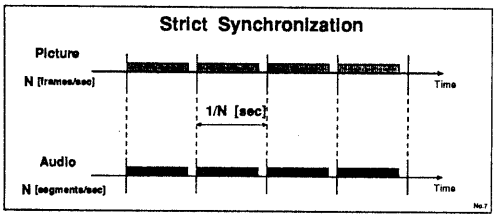

図 2: Strict Synchronization

Relaxed Synchronization: フレーム及びセグメントの合計時間が一致した点で同期をとる方式である。例えば1秒以下の間隔で同期する場合、レート値がそれぞれ N_P[frames/sec], N_A[segments/sec] の動画像及び音響データの時、N_P, N_A の公約数 α でそれぞれのレート値を割ったフレーム及びセグメント毎に同期を取る。この時、理論的な同期間隔は $1/\alpha$ [sec] となる。また1秒以上の同期間隔で行なう場合、それぞれ N_P,N_A の α (α=1,2,3...) 倍のフレーム及びセグメントで同期を行なう。この時の理論的な同期間隔は α [sec] である。

この方式では、先の Strict Sync. に比べ同期ポイント数をかなり抑えることが可能となり、また動画像と音響データのレートを一致させる必要もないため、比較的柔軟にレートを変えることが可能である。

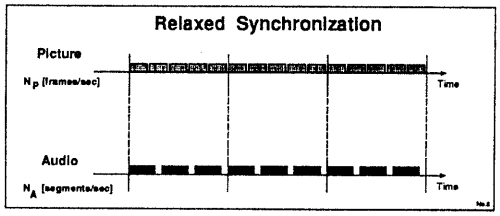

図 3: Relaxed Sychronization

Silence Synchronization: 音響セグメントが無音であるか有音であるかを判定し、有音状態の場合のみ先の2つの同期方式のいずれかを用いて動画像データと同期を取るか、又は有音部分の開始点だけ揃える方式である。無音状態の場合は、動画像を一定レートで表示するだけで、音響との間で同期処理を必要としない。

この方式は、TV 会議のように会話データを多く含むものに対しては非常に有効である。

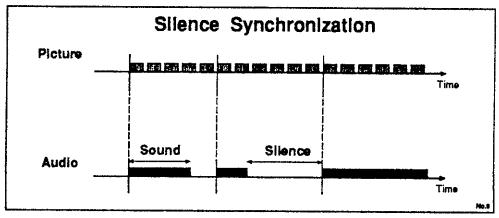

図 4: Silence Sychronization

3 システムの実装

メディアデータの転送を高スループットに維持しながら、正確な同期処理を行うためには、各層の機能モジュールを並列に実行・処理させる必要がある。よって本システムの実装は、複数の UNIX のプロセス、Mach のスレッドにより行った。図5にクライアント側の実装を示す。

UNIX システムでは、各層の機能モジュー

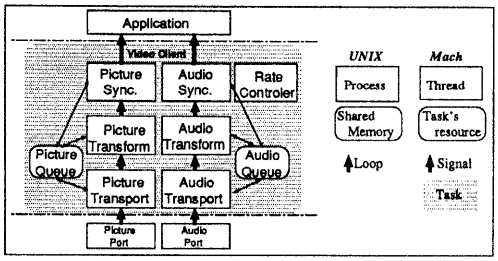

図 5: システムの実装方法

ルそれぞれにプロセスを割当て、メディアデータを格納するリング・キューを共有メモリ上に実現し、その共有メモリに確保されたフラグを常に監視することによって、各機能モジュールが順次処理すべきキューエントリーを区別し、該当するデータ処理を行う。

一方 Mach システムでは、サーバー側、クライアント側でそれぞれ 1 つずつのタスクを生成し、各層の機能モジュール各々にスレッドを割当てることにより、並列処理を可能にした。そのスレッド群は、タスクの資源として確保されたメディアデータ用のリング・キューを共有し、逐次的にデータ処理を行なうためにスレッド用のシグナルを使用している。また UNIX,Mach とも、サーバー側についてもそれぞれのクライアント側と同様な実装を行っている。

Mach による実装では、実行単位がスレッドであるため、各層の機能モジュールの制御が容易になり、しかもその間でのデータコピーの必要性がない。また Mach スレッドのコンテキスト切替えは、プロセスの場合と比べその処理が軽く、仮にマルチ CPU で実行される場合、機能モジュールを割り当てたスレッドごとに分散され、物理的な並列処理が可能であるといった利点が挙げられる。

4 性能評価

4.1 プロトタイプ

本システムのプロトタイプとして、図 6 に示す様に RISC ベースのワークステーションを用いて、UNIX 及び Mach OS による 2 つのシステムを Ethernet(10Mbps) 上に構築した。

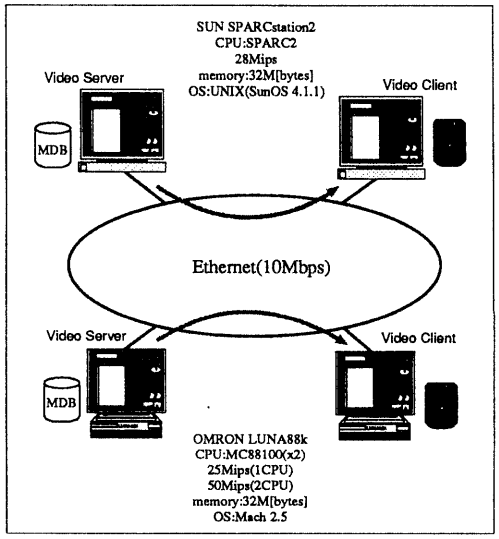

図 6: プロトタイプ

またビデオデータとして以下のものを使用した。

FrameSize[pixel]	80×60, 120×90 160×120, 240×180
FrameRate[fps]	10, 16, 24, 30
ColorDepth	8[bits/pixel]
Bandwidth	8000[Hz]
QuantizeBit	8[bits]
Modulation	PCM, μ-law

4.2 フレームレート制御

図 7 は、UNIX で実装した PAVS により Strict Sync. を行なった際の、フレーム

レート制御とその時の実効フレームレートの実時間遷移を示したものである。この図では、マシンの処理能力の不足が原因となり、ビデオデータが本来なされるべきレートで転送・再生されないことから、レート制御機構が働き、自動的に再生レートが実現可能な実効レートに収束していく様子を示している。この制御により、レートの収束時にはクライアント側での自然な再生が可能であった。

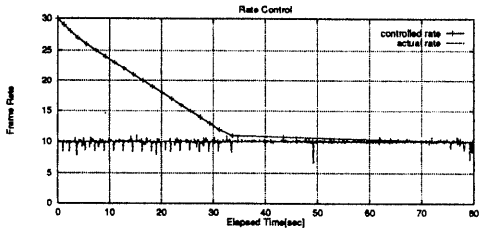

図 7: レート制御と実効フレームレート

4.3 同期精度

同期性能の評価は、その精度のみに重点をおくために、サーバ側でデータをディスクから読み出し、クライアント側で同期処理を行うまでとし、動画像の表示及び音響データの再生は実際には行なわなかった。また評価は、動画フレームデータの表示開始時刻から同期に対応した音響セグメントデータのデバイスへの書き込み開始時刻を引くことにより、その値を本システムの同期誤差として行った。

負荷条件は、サーバ CPU, クライアント CPU , ネットワークのいずれか 1 点に対して与える事とし、そのときの同期精度について測定評価した。CPU 負荷は高速計算を要するアプリケーションを複数起動することにより、CPU に対して 50%~70%の (ps コマンド結果) の負荷を与えた。Network 負荷は通常のネットワーク負荷の他に、故意に評価マシン以外のマシン間でファイル転送を行って実現した。

図 8 は、クライアント側で同期処理を行わなかった場合の結果 (クライアント負荷) を示しており、フレーム毎に前述した同期誤差を示したものである。この図において、同期誤差が不安定でばらつきが大きいことから、特に負荷がかかっている場合においては、クライアント側での同期処理が不可欠であることが分かった。

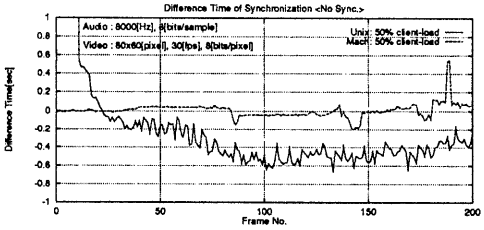

図 8: 同期の時間誤差 (No Sync.)

Strict Sync. の場合では、図 9 の結果 (サーバ負荷) で示すように、UNIX、Mach 両システムとも同期の時間誤差を 1 フレーム間隔以内に抑える事ができ、フレームのサイズやレートに関わらず高精度での同期が可能であった。

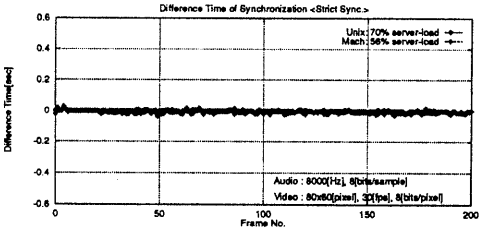

図 9: 同期の時間誤差 (Strict Sync.)

また Relaxed Sync. の場合では、図 10 の結果 (クライアント負荷) で示すように、UNIX,Mach 両システムとも非同期ポイントでは必然的に同期の時間誤差は相対的に大きくなり、特に UNIX でのばらつきが大きかった。しかし、0,30,60... フレーム目といった同期ポイントにおいては、正確な同期が可能であった。

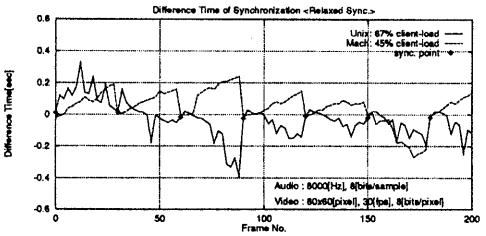

図 10: 同期の時間誤差 (Relaxed Sync.)

Silence Sync. の場合の結果 (無負荷) を、図 11 に示す。この図において、同期誤差 0 の部分は無音部分であり、有音の開始点のみ同期を行なっているので、無音から有音になる点での同期は正確であったが、それ以外の有音状態における非同期ポイントでは、やはり必然的に誤差が増加する。しかしこの場合、Mach による実装では誤差が比較的小さかった。

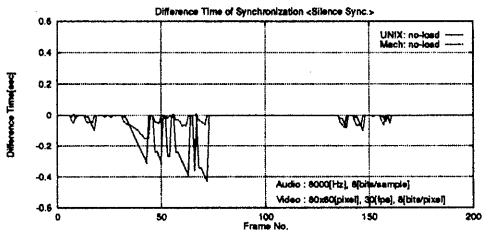

図 11: 同期の時間誤差 (Silence Sync.)

4.4 実効フレームレート

UNIX 実装システムにより Strict 及び Relaxed Sync. を行なった時の平均実効フレームレートは次の通りであった。

Sync. Method	FrameRate [frames/sec]
Strict	9.8
Relaxed	21.0

表より、Relaxed Sync. では Strict Sync. の場合と比較して約 2 倍のフレームレートが得られ、相対的に Strict Sync. がかなりの CPU パワーを必要とすることがわかった。

5 まとめ

アプリケーションの開発に関しては、相互作用する平行処理を考える際、タスク・スレッドを使用する Mach は能率的であった。またプロトタイプを構築して PAVS の同期性能の評価を行なった結果、負荷がない場合は必ずしもクライアント側での同期処理は必要でないが、サーバ, クライアント, ネットワークのいずれかに負荷がかかっているような通常の状態では、同期処理が必要である。また高精度の同期には Strict Sync. が必要であるが、Relaxed Sync. の場合でも UNIX,Mach 双方について、ほぼ満足のいくレベルで、動画像と音響データの同期を取りながらの転送が可能であることが確認できた。

また、同期処理による CPU 負荷への影響という観点から考慮すると、Relaxed Sync が明らかに有利であり、同期精度も含めて実用的であると言える。

今後は、DB に格納された圧縮データや QOS を考慮した同期制御及びフレームレート制御を行う PAVS の設計・開発を行う計画である。

参考文献

[1] 神原久夫, 河野太基, 柴田義孝: パケットビデオシステムのための同期メカニズム, 情報処理学会第 46 回全国大会,1K-05,1993

[2] 清水省悟, 瀬田直也, 神原久夫, 柴田義孝: マルチメディア情報ネットワークのためのパケットビデオシステムの設計と性能評価, 情報処理学会第 47 回全国大会,4V-05,1993

[3] 柴田義孝: マルチメディア情報ネットワークにおける同期転送方式, 情報処理学会第 45 回全国大会,1B-6,1992