データベースシステム 113－14
（1997. 7. 14）

再生時間が未知なメディアを対象としたメディア間同期方式

端山 貴也† 清木 康††

†筑波大学 工学研究科 ††慶應義塾大学 環境情報学部

マルチメディア処理におけるメディア間同期では、メディアやデータの特性に応じて、細かく同期のスケジュールを設定する事が、ジッターやずれの少ない再生を行う上で重要である。これを実現するためには、各メディア・データの再生時間をもとに、同期のスケジュールを設定する必要がある。しかし、マルチメディア・データの再生の際に、ユーザとの対話などを含む場合は、ユーザの反応速度やタイミングに左右されるため、メディア間同期のスケジュールを静的に設定することが困難である。本稿では、メディア・データの再生時間が未知な場合における同期方式について述べる。本稿で述べる同期方式は、協調型処理方式に基づく。

An Inter-Media Synchronization Method for Media with Unknown Duration

Takanari Hayama† Yasushi Kiyoki††

†Doctoral Program in Engineering, University of Tsukuba

††Faculty of Environmental Information, Keio University

While rendering multimedia documents, jitters and lags among media always araise as a common problem. Jitters and lags can be reduced by setting appropriate synchronization points among media. To set the synchronization points, duration of each medium is required. However, there are many multimedia documents with unpredictable duration such as user interaction which makes difficult to create static schedules of synchronizations. In this paper, we present a synchronization method for media whose duration is unpredictable. The synchronization method is based on coordinated computation.

1 はじめに

マルチメディア・ドキュメントは、複数のメディアを時間的、および、空間的に関係付けることにより作成される。複数のメディアを時間的な関係により結びつける時に、メディア間の同期問題が生じる。このようなメディア間の同期問題は、マルチメディア処理における重要な問題の一つである。

メディア間の同期関係は、論理的なメディア・データ間の時間的関係、および、物理的な再生時の時間的関係に分類できる。論理的な関係とは、二つのメディアの同時再生や、連続再生など、メディア・データの内容により決まる時間的関係である。物理的な関係とは、論理的な関係により定まるメディア再生プロセスの時間的関係である。メディア再生プロセス間の時間的関係は、プロセス間の同期問題となる。論理的なメディア間の関係を記述する方法には、幾つかの方式が提案されている [1, 7]。これらの方式では、メディアを一つのインターバルと捉え、そのインターバルの関係により、時間的な関係を表す。この方式で指定された時間的な関係を物理的なプロセス間の同期として解決しようとした時、データとプロセスが時間的な制約を正確に守ることが要求される。そのため、時間的な制約を保証できないシステムでは、論理的な関係と、実際に再現された物理的な関係の間に予測不可能な差異が生

じる。

この問題を解決するためには、複数のメディアの各時間軸上に任意の同期点を設け、各メディアの同期点の関係により時間的関係を記す。これにより、論理的な関係を物理的な関係として再現する際の差異を予測可能にすることができる。マルチメディア処理におけるメディア間同期では、メディアやデータの特性に応じて、同期のスケジュールを設定する事が、ジッターやずれの少ない再生を行う上で重要である。

このように同期のスケジュールを設定した上で、物理的な再生時の時間的関係としてのメディア間同期を実現するために、本稿では、協調型同期方式を用いる方式について述べる。本方式において、メディア・データを再生するプロセスは、独立して動作する。そのため、処理の対象とするメディアに応じて、プロセスは、最適な動作方針を選択できる。また、本方式は、マルチメディア・システムの柔軟な実現を可能とする。しかし、協調型同期方式では、同期のスケジュールが予め静的に設定されていることを想定しているため、再生時間が未知なメディアを対象とすることが難しい。

本稿では、始めに、メディアの論理的な時間的関係の記述方式を述べる。そして、協調型同期方式について述べる。そして、メディア・データの再生時間が未知な場合におけるメディア間同期の問題点とその解決方法について述べる。

2 メディアと同期

メディア間の時間的関係を表す場合、メディアをインターバルと捉え、その時間的関係によって表す[1, 6, 7]。論理的なメディアの時間的関係を表す場合は、インターバルの関係で表すことで、あらゆる同期関係を柔軟かつ簡素に表現することが可能である。しかし、インターバルによる同期関係の記述では、同期の粒度が大きいため、実際に、その情報を元に、同期を実現する際、ジッターやずれなどの問題が発生する。

ここでは、それらの問題を解決するために、インターバル中の任意の時点に同期点を設ける方式を示す。

2.1 インターバルと任意の同期点

インターバルを用いる場合、インターバルの開始点と終了点のみが同期点として指定可能であるため、インターバル内で演奏される曲や動画像の再生時間の調整を予め行う必要がある。同期関係は、インターバル間の相対関係で表されるが、実際に再生される時は、全て実時間で動作することを要求される。リップ・シンクを行う場合、それぞれのメディアは、周期性、かつ、実時間制約の中で再生されることが要求される。演奏時に、動的にリップ・シンクを制御する場合、インターバルの同期関係からでは、同期を取るべきチェック・ポイントに関する情報を得られない。インターバルの関係のみで論理的な時間関係を表す方式は、再生時に必要な情報を十分に記述することができない。

再生時の環境や状況に応じて、演奏時間は変化する。そのため、予め用意された時間的関係を満たす保証はなく、再生時に動的に調整を行う必要がある。そこで、インターバル中の任意の時点の関係により時間的関係を表現することにより、再生時に、柔軟にメディア間の同期処理を行うことが可能とする必要がある[3, 5]。これにより、論理的に同期する必要のある点を同期させることが可能となる。また、インターバル内で再生されるメディアを同期させる時、間延びさせるのか、あるいは、短縮するのかを自由に記述することが可能になる。

ここで、二つのインターバル X と Y があるとする。同期点を任意に設定可能であるとすると、インターバル X と Y の同期点は、次のように定義される。

$$
\begin{aligned}
X &= \{X_{start}, X_1, X_2, ..., X_i, ..., X_n, X_{finish}\}, \\
Y &= \{Y_{start}, Y_1, Y_2, ..., Y_j, ..., Y_m, Y_{finish}\}.
\end{aligned}
$$

X_i や Y_j は、同期点であり、X_{start} を起点とした時の相対時間である。X_{start}、および、X_{finish} は、インターバルの開始と終了点であり、従来の同期関係と同様である。X_i は、新たに設ける任意の同期点である。X_{start} と X_{finish} の間の任意の時点に設定される。時間的関係（同期）は、これらの同期点の関係によって表す。ここで、同期点の関係を表すため、新たに *Meet* 関係を導入する。*Meet* 関係は、インターバルの任意の点である同期点の関係を表す。インターバル X と Y が同時に開始し、終了する場合は、次のように表す。

$$
\begin{aligned}
&Meet(X.X_{start}, Y.Y_{start}), \\
&Meet(X.X_{finish}, Y.Y_{finish}).
\end{aligned}
$$

2.2 未知な再生時間のメディア

インターバルが既知である場合には、任意の同期点をそのインターバル上に設定することが可能である。しかし、メディア・データの再生時間（インターバル）が既知でない場合には、静的に同期点とその関係を設定することが困難である。インターバルという単位で同期を行う場合には、インターバルの長さが既知である必要がない。

ここで、生時間 t のメディア・データに、再生時間 s のメディア・データを同期させて再生することを仮定する。再生時間 t が既知な場合には、あらか

じめ、メディア・データの特性に応じて、同期のタイミングや回数を決定することが可能である。しかし、再生時間 t が未知な場合には、あらかじめ同期の回数を決定することができない。従って、同期のスケジュールを動的に決定していく必要がある。

このような状況は、実時間で再生されないメディアやユーザとの対話と、その他のメディアを同期させる場合に発生する。例えば、ユーザとの対話においては、ユーザの反応がいつ発生するか予測不可能である。ユーザとの対話を一つのメディアと捉え、ユーザからの反応を待つ間、音楽や動画像を流しつづける場合を仮定する。この場合、音楽と動画像は、同期していなくてはならないため、ユーザからの反応があるまでの間、同期点を設け、同期を続ける必要がある。しかし、動的に同期点を設けることは、困難である。これは、それぞれのメディアの時間軸が同じである保証がないため、メディア間の時間軸の差異を動的に埋めるのが難しいことによる。そこで、同期点を動的に設けるのではなく、同期点を必要に応じて設けられる関係により、これを解決する。本稿では、これを *Vote* 関係と呼ぶ。

2.3 投票を行う同期点

投票による同期方式では、新たに同期点とする可能性のある同期点を、投票を行う同期点として、予め設定する。そして、それらの同期点を *Vote* 関係として関係づける。投票を行う同期点に到達したとき、プロセスが次の同期点に進むか否かについて投票を行った結果に応じて、新たな同期点を設定する。全てのプロセスが、次の同期点に進むことに賛成したときは、新たな同期点が設定されずに、次の同期点に進む。これは、従来の *Meet* 関係による同期処理と同様である。一方、一つでも、次の同期点に進む事に賛成しないプロセスがある時は、再度、投票を行うべく、次の投票を行う同期点が設定される。

ここで、二つのインターバル X と Y が次のように定義されているとする。

$$\begin{aligned} X &= \{X_{start}, X_1, X_2, X_{finish}\}, \\ Y &= \{Y_{start}, Y_1, Y_2, Y_{finish}\}. \end{aligned}$$

そして、次のような関係が定義されているとする。

$$\begin{aligned} &Meet(X.X_{start}, Y.Y_{start}), \\ &Meet(X.X_1, Y.Y_1), \\ &Vote(X.X_2, Y.Y_2), \\ &Meet(X.X_{finish}, Y.Y_{finish}), \end{aligned}$$

この時、インターバル X と Y は、同時に開始し、終了する。$X.X_1$ と $Y.Y_1$ の時点では、*Meet* 関係により、同期をする。$X.X_2$ と $Y.Y_2$ の時点においては、次に進むことが可決された場合に、$X.X_{finish}$ と $Y.Y_{finish}$ にそれぞれ進む。否決された場合には、インターバル X においては、$(X.X_2 - X.X_1)$ 単位時間後、また、インターバル Y においては、$(Y.Y_2 - Y.Y_1)$ 単位時間後、再度、投票を行う。

3 メディア間同期の制御方式

メディア・データの論理的な同期関係は、その再生時に、物理的な同期関係に変換される必要がある。メディア・データの再生は、メディア再生アプリケーション（プロセス）により行われる。そのため、この物理的な同期関係の問題は、プロセス間の同期の問題となる。ここでは、メディア・データの再生を対象としたプロセス間の同期制御方式を提案する。

3.1 関連研究

マルチメディア・システムの拡張性を高めるために、複数の比較的小さなアプリケーションを組み合わせ、マルチメディア・システムを構築する方式がある。この方式に基づくシステムには、マスタ・スレーブ方式を用いた MAEstro がある [2]。この方式では、時間や同期情報を管理するマスタが、スレーブである各アプリケーションの実行と終了を管理する。通信量が少ないものの、マスタが実時間制約などを管理する必要があるため、マスタの負担は大きい。

マスタを含め、全てをアプリケーションとして実装する場合、負荷が大きくなるため、マスタをミドルウェアとして実装する方式もある [9]。また、資源予約と実時間の周期スレッドを用いた同期機構についての研究も多くされている [4, 8, 10]。いずれの場合も、スレーブは、与えられた CPU 時間とメモリ資源内に与えられた処理を行う。スレーブ間の同期タイミングは、スケジューラにより制御される。スレーブは、与えられた CPU 時間およびメモリ資源と、行うべき処理の関係を理解する必要がある。

3.2 協調型同期方式

プロセス間の独立性を高めることにより、実行環境に応じて、プロセスは最適な動作方針を選択可能になる。プロセスの動作方針は、その扱うメディアに応じて異なるため、その動作方針も多岐に渡る。本論文では、プロセスの独立性を高める同期処理方式として、協調型処理方式に基づく協調型同期方式を示す [3]。

マルチメディア・ドキュメントの再生は、メディア間の同期関係を表す同期情報を元にアプリケーション(プロセス)が行う。同期情報は、各メディアの時間軸上に設けられた同期点の関係であらわされる。これは、メディアにより、時間の最適な表現方法が異なるためである。例えば、MIDI音楽の場合、MIDI特有のMIDIティック、また、動画像の場合は、フレーム数が最適な時間の単位である。これは、それぞれの単位が、各メディア・データにおいてアトミックな単位であるからである。そこで、メディア固有の時間軸の任意の時点での同期関係でのみ、同期情報をあらわす必要がある。これにより、時間軸が異なる場合であっても、任意の時間軸にあわせて同期させることが可能となる。例えば、実時間動作できないアプリケーションは、このように実時間動作するアプリケーションに同期させることで、実時間動作させることが可能となる。このように、プロセスの動作の独立性を高めることで、プロセス間の同期問題を柔軟に解決することが可能となる。

しかし、全てのプロセスが平等な権限を持った場合、対象とするメディアの組み合わせに応じた最適な同期制御が行うことができない。例えば、動画像と音楽を扱う場合、音楽に動画像を同期させることが多い。これは、人の感覚が、動画像のフレームが飛ばされることよりも、音楽が途切れたり、飛ばされることの方に、人が敏感であるからである。しかし、状況により、音楽より動画像の方が重要な場合もありうる。この場合、動画像が途切れないように再生されなければならない。

この問題を解決するためには、プロセスを優先度により区別化する。優先度を同期の強制のためのヒントとして用いる。ここで、プロセスAとBが同期して動作する場合を考える。プロセスAがプロセスBよりも高い優先度を持つ場合、プロセスBは、プロセスAに追従して動作する。プロセスAが、仮に、プロセスBよりも先に同期点に到着した場合、その到着がプロセスBに知らされる。この時、プロセスBは、可能な限り次の同期点に進む。しかし、同期点に進むことがその時点で不可能な場合、これを無視、あるいは、保留する。これにより、動画像と音楽において、動画像再生プロセスは、フレームを飛ばすが、音楽再生プロセスは、音符を飛ばさない制御を行うことが可能となる。

協調型同期方式は、マスタ・スレーブ方式に比べ、プロセスが互いに直接通信するため、分散環境における通信の複雑さとオーバヘッドが増す。しかし、単一計算機上で実行する場合は、IPCを用いたバリア同期などを用いることにより、通信量をマスタ・スレーブ方式と同等にすることが可能である。また、協調型同期方式では、プロセスの組み合わせ方により、マスタ・スレーブ方式と同様の同期制御を行うことが可能である。この場合、スレーブとなるプロセスの優先度を低くし、必ず、マスタとなるプロセスに従うようにする。

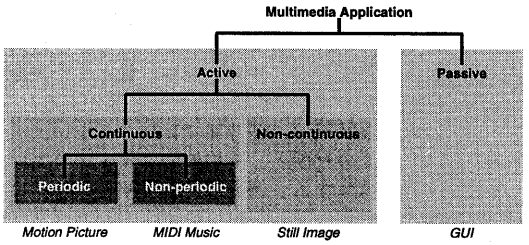

図 1: マルチメディアを対象とするアプリケーションの分類

4 協調型同期方式に基づく同期機構の実現方式

同期機構は、様々なメディアを扱うプロセスを対象とする。プロセスは、扱うメディアにより分類できる。ここでは、プロセスの分類に基づいた同期機構を示し、協調型同期方式に基づく同期機構の実現方式を示す。

4.1 プロセスの分類

マルチメディア・アプリケーションは、図1のように分類可能である。これらのアプリケーションを複数組み合わせることにより、マルチメディア・アプリケーションが構成される。

マルチメディア・アプリケーションのコンポーネントとなりうるプロセスは、二つのタイプ、受動的プロセスと能動的プロセスに大別できる。対話を担うGUIを持つ事象駆動型のものは、受動的プロセスにあたる。また、音楽、動画像、静止画像の再生を行うプロセスは、能動的プロセスにあたる。さらに、能動的プロセスは、扱うメディアが時間の概念を必要とするか否かにより、コンティニュアスとノン・コンティニュアスに分けられる。コンティニュアス・メディアを扱うプロセスには、周期的に毎秒30フレームを再生する動画像再生プロセスと、MIDIシーケンサのように非周期的にコンティニュアス・メディアを扱うプロセスがある。

静止画像のようなノン・コンティニュアス・メディアを扱うプロセスは、時間の概念を持たない。このようなプロセスにも表示の開始と終了はあるが、この開始と終了は、コンティニュアス・メディアを扱うプロセスと同時に動作しない限り、時間的制約を受けることができない。これは、扱うメディ

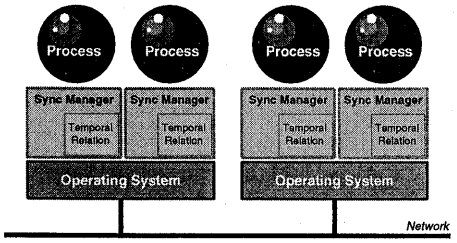

図 2: 同期機構とプロセス

表 1: 同期のための操作プリミティブ

操作プリミティブ	機能
SYNC	同期処理
SAVE_CONTEXT	コンテキストの保存
SET_UPDATE_VAR	強制的な同期時に自動更新する変数
BLOCK_SYNC	強制的な同期のブロック
UNBLOCK_SYNC	強制的な同期のブロック解除
VOTE	投票を行う同期関係での投票

ア自体に、時間の要素がないためである。また、GUI のような受動的なプロセスにおいても、プロセスが受けるイベントの一種としての時間はありうるが、自ら状態遷移をしない。そのため、ノン・コンティニュアス・メディアを扱うプロセス同様、時間の概念を持たない。しかし、画像の表示や、ユーザの入力に対する反応などを指定時間内に行う時間的制約が考えられる。

4.2 同期機構

以上のように分類されたプロセスは、与えられた同期情報に応じて同期する。同期は、プロセスのグループで行われる。グループ内で行われる一連の同期をセッションと呼ぶ。

ここで、プロセスがネットワーク上の計算機に分散することを想定する。複数の計算機上において同時に動画像を再生する場合や、特定の計算機にのみ付随する特殊なデバイスを利用する場合がある。これらの場合、プロセスは、ローカルとリモートに分散するグループ内のプロセスと同期する必要がある。加えて、協調型同期方式では、同期関係の情報をグループ内の全プロセスが知る必要がある。各プロセスは、この情報を取得し、解析し、どのプロセスと同期を行うのかを理解する必要がある。

本方式では、このような複雑な処理をプロセスに代わって処理し、プロセスの同期処理を支援するために、同期マネージャを用いる（図 2）。同期マネージャは、時間的関係を記述した同期情報を元に、各プロセスの同期処理を、他のプロセスと協調し支援する。同期マネージャは、操作プリミティブとして、同期処理支援のための機能を提供する。同期マネージャをプロセスにリンクすることにより、プロセスは、容易に同期に参加可能となる。プロセスは、操作プリミティブを介した同期のために最低限必要な指示を同期マネージャに与えることにより、容易に、同期処理を行う。

同期マネージャは、同期処理時に、他の同期マネージャとの協調、および、プロセスの状態の管理を行う。同期のインターバルがメディアに依存する単位（フレーム数など）の場合、プロセスの同期点への到着の判断は、プロセス自身が行う。そのため、プロセス自身が同期点への到着を判断し、同期マネージャに対し同期処理を依頼する。同期点への到着をプロセスが判断するため、同期マネージャは、次の同期点までのインターバルと同期点の残数を、同期関係の情報より抽出し、プロセスに知らせる。また、次の同期点において要求されている場合には、同期マネージャは、その旨を、プロセスに対して伝える。ただし、同期のインターバルが実時間を単位としている場合、同期マネージャがプロセスの同期点への到着を判断することが可能である。必要に応じて、プロセスは、同期マネージャに同期点への到着の判断を依頼する。

本方式では、同期処理を柔軟に行うため、プロセスの優先度が低い場合、割り込みにより強制的に同期処理を行うことを可能にする。この場合、強制的な同期を要求されるたプロセスは、次の同期点に遷移しなければならない。このような、強制的に同期を行うために、プロセスのコンテキストの保存と更新を行う。強制的な同期処理後、同期マネージャは、保存済みのコンテキストを用いプロセスの同期処理終了直後の状態に戻す。また、ヒープの状態を変化させる必要がある場合は、ヒープの更新を行う。プロセスは、強制的な同期を行う際に、重要な変数の更新中などのクリティカル・リージョンにある場合がある。そのため、クリティカル・リージョンにプロセスがある間は、強制的な同期を一時的にブロック可能とする。

本方式において、これらの機能を実現するための同期マネージャの操作プリミティブを表 1に示す。コンティニュアス・メディアを周期的に処理するプロセスに対し、操作プリミティブは、図 3のように組み込まれる。

```
while (not finished processing) {
  if (arrived at synchronization point)
    SYNC;

  if (next sync point requires to vote)
    VOTE(YES | NO);

  /* save context */
  SAVE_CONTEXT;
  if (no more synchronization)
    exit;

 /* media processing comes here */
 BLOCK_SYNC;
 /* critical region */
 UNBLOCK_SYNC;
 /* rest of the media processing comes here */
}
```

図 3: 操作プリミティブを組み込んだプログラム例

4.3 同期機構とプロセス

受動的なプロセスに操作プリミティブを組み込む場合には、SAVE_CONTEXT() によりイベント・ハンドラを保存し、同期処理後、イベント処理を継続して行うようにする。BLOCK_SYNC() や UNBLOCK_SYNC() を用いることにより、利用者との対話に応じて同期を制御する場合の同期処理全体の制御が可能である。ノン・コンティニュアス・メディアを扱うプロセスの場合、同期は、開始時と終了時にのみ行われる。そのため、例えば、静止画像を表示するプロセスでは、表示直前と表示直後に、SYNC() を用いる。コンティニュアス・メディアを扱う場合は、図3のように、メディア処理を開始する前、メディア処理のループ中、そして、メディア処理後に同期を行うようにする。このように、本方式による同期機構は、先に分類したプロセスの処理の中に、操作プリミティブの形で、柔軟に組み込むことができる。

5 おわりに

本稿では、再生時間が未知なメディアを対象とするための *Vote* 関係について述べた。従来の静的な同期点間の関係だけでは、ユーザとの対話部などの時間的な予測がつかないメディアと他のメディアを同期させるのが困難であった。必要に応じて同期点を増やすことが可能な *Vote* 関係を用いることにより、再生時間が未知なメディアとの同期に柔軟に対応できる。

今後は、プロセス間同期支援システム NAMI に *Vote* 関係を組み込み、より柔軟にマルチメディア・システムの構成を支援する環境の実現を行う。

参考文献

[1] J.F.Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. of ACM*, Vol.26, No.11, pp.832–843, Nov. 1983.

[2] G.D.Drapeau, H.Greenfield, "MAEstro—A Distributed Multimedia Authoring Environment," *1991 Summer USENIX Conference in Nashville, Tennessee*, 1991.

[3] T.Hayama, Y.Kiyoki, "A Distributed Process Coordinator for Rendering Multimedia Resources in a Multimedia System," *Proc. of Multimedia Japan '96*, pp.168–175, Mar. 1996.

[4] K.Kawachiya, H.Tokuda, "QOS-Ticket: A New Resource-Management Mechanism for Dynamic QOS Control of Multimedia," *Proc. of Multimedia Japan '96*, pp. 14–21, Mar. 1996.

[5] Y.Kiyoki, T.Hayama, "The Design and Implementation of a Distributed System Architecture for Multimedia Databases," *Proc. FID '94*, pp.374–379, Oct. 1994.

[6] T.D.C.Little, A.Ghafoor, "Interval-Based Conceptual Models for Time-Dependent Multimedia Data," *IEEE Trans. Knowledge and Data Engineering*, Vol.5, No.4, pp.551–563, Aug. 1993.

[7] Y.Masunaga, "An Object-Oriented Approach to Temporal Multimedia Data Modeling," IEICE Trans. Inf. & Syst., Vol. E78-D, No.11, November 1995.

[8] J.Nieh, M.S.Larm, "Integrated Processor Scheduling for Multiemdia," *Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 215–218, Apr. 1995.

[9] N.Nishio, H.Tokuda, "A Middle-Ware for Continuous Media Processing in the Keio-MMP Project," *Proc. of Multimedia Japan '96*, pp.278–284, Mar. 1996.

[10] R.Yavatkar, K.Lakshman, "A CPU Scheduling Algorithm for Continuous Media Applications," *Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 223–226, Apr. 1995.