マルチメディア通信と分散処理 64－1
（１９９４．３．３）

周期スレッドを用いたマルチメディアデータの同期処理

吉川耕平† 岡村耕二 荒木啓二郎
奈良先端科学技術大学院大学 情報科学研究科
（†シャープ株式会社技術本部より派遣留学中）

分散環境において、処理の連続性及び同期という時間的制約を保証する機構をハードウェアに依存しないマルチメディア処理機能として実現することを目指している。本稿では、まず実時間同期と、イベントによる同期という枠組で連続メディアと非連続メディアを統合する手法を示す。次にこれを実現する機構として周期スレッドを提案し、その API について考察を行なう。周期スレッドは、アプリケーションから時間依存のプログラムを分離させたもので、１）ネットワークやデバイスの制御、２）周期的な処理に対して定義される QoS をもとにした処理の連続性の管理、３）複数処理間の実時間同期、４）ユーザが設定するイベントを用いた開始／終了／切替えといったイベント型同期、などを行なう。

Multimedia Synchronization by Periodic Threads

Kouhei YOSHIKAWA† Koji OKAMURA Keijiro ARAKI
Graduate School of Information Science
Nara Institute of Science and Technology
†Student from SHARP Corporation

Next generation Parallel and Distributed Environment must support multimedia composed by multiple continuous media streams which need to be synchronized each other. We have already defined QoS, i.e., the degrees of continuity and synchronization of streams. In this paper, we discuss about two types of synchronization mechanisms. One is a real-time synchronization that monitors and adjusts degrees of continuity and synchronization according to QoS. The other is an event-driven synchronization that controls each time interval. Then we propose periodic threads for multimedia data processing and transmission. It provides these synchronization mechanism and interface like socket to application programs. Periodic thread helps application programmer by simplifying time dependent program, integrates continuous and discrete media retrieving, and is a mechanism for hardware independent multimedia application.

1 はじめに

マルチメディアとは、映像・音声といった時間的制約を持つ複数の連続メディアと、従来の非連続メディアとを統合したものであり、既存の計算機ハードウェアによる分散環境でそれらを統合することを目指した研究が盛んに進められている[1]。

我々は、分散環境において複数の連続メディアデータの交換、または処理の際の、処理の連続性と複数処理間の同期に、処理の質(QoS:Quality of Service)を定義し、その保証機構を、ハードウェアに依存しないマルチメディア処理機能としてOSに取り込むことを目的とした研究を進めている。これを目指し、ネットワークをはさんだ複数のエンティティ(アプリケーションやデバイス)間に張られたコネクション上を、表示の連続性及び同期という時間的制約を持つ表現メディアデータを伝送し、コネクション上に置かれたQoSポイント(図1.)でその制約を管理するモデルを先に提案している[2]。

本稿では、まず連続メディアを扱うために必要な同期の機構として実時間同期とイベント駆動型同期を提案する。特に後者は、連続メディアと従来の非連続メディアとの統合を可能とする重要な機構と考えている。続いて先に提案したモデルの構成要素であるQoSポイントを実現する一手法となる「周期スレッド」を提案し、それが提供する実時間同期及びイベント駆動型同期のプログラムインタフェースについて述べる。

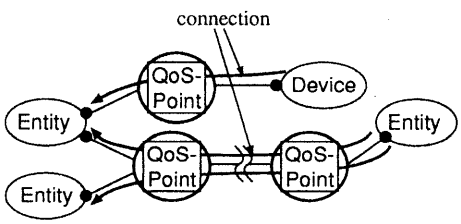


図1: QoSポイント

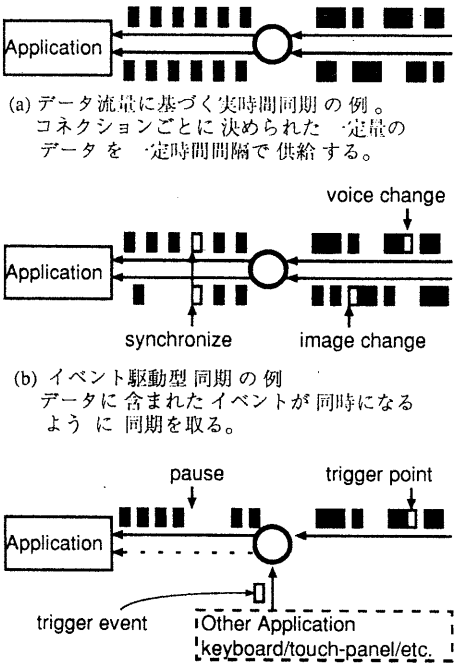
2 マルチメディアにおける同期

一般にメディアというと、音や映像そのもの(知覚メディア)、それらを計算機処理可能な形に表現したデータ(表現メディア)、音や映像を再生するデバイス(表示メディア)などを指す。また表現メディアのうち、時間的制約を持ち連続的に再生する必要のあるものを連続メディアと呼んでいる。我々は連続メディアを含む異種複数の表現メディアの入出力/蓄積/圧縮/伸長/交換などをマルチメディア処理と考えている。特にその中で、音や映像など連続メディアの処理における時間的制約の充足と、連続/非連続メディアの処理の統合が課題となっている。

分散環境における連続メディアデータの交換に関しては、転送の連続性を保証する各種実時間プロトコルの開発が進められる一方、連続性の保証に加えて関連する複数のストリームを実時間上で同期させる機構も提案されている。

例えばACME[3]では、実時間での同期のため以下のような機構を取っている。まず制御コネクションを使ってRPCによりデバイスを抽象化した論理デバイスと、連続メディアを流すデータコネクションを生成する。データコネクション上の連続メディアデータを時間軸上でユニット化し、各ユニットにタイムスタンプを付ける。次にデバイスからのコネクションやネットワークによるコネクションの中の一つを親と考え、そのコネクションのタイムスタンプを大域的な論理時刻(Logical Time)とみなす。親以外の各コネクションは、論理時刻との間で、パラメータで設定した数値以上の差が生じれば、skip/pauseにより自身の進行を制御する。これは、図2(a).に表現されるような、データ流量をベースとした実時間での同期には適切な機構となっているが、非連続メディアとの統合に関する考察は行なわれていない。

次に、データベース化を目指したマルチメディアデータの概念モデルの提案[4]では、マルチメディアプレゼンテーションのデータベース化などを対象に、「一かたまりの連続メディアデータ」



(a) データ流量に基づく実時間同期の例。
コネクションごとに決められた一定量の
データを一定時間間隔で供給する。

(b) イベント駆動型同期の例
データに含まれたイベントが同時になる
ように同期を取る。

(c) イベント駆動型同期の例
外部のシステムからの入力をイベントとみな
してデータを供給するタイミングを制御する。

図 2: 同期

を単位として、それらの前後関係、重なり関係などを時区間 (time interval) をベースとした関係で表現している。これは、実時間同期とは別の、イベントドリブンな同期の必要性を指摘したもので、筆者らはこれを拡張することで、連続／非連続メディアの統合が可能になると考えた。

例えば実時間での同期として、テレビ電話における口の動きと声との同期、マルチメディアサーバのデータ再生における背景音、効果音、映像、テロップ間の同期等が考えられる。これらをコネクションの優先度と、データ流量だけをベースに実時間で同期させた場合、初期状態での同期を別途制御する必要があったり、またシーン切替えの瞬間に一時的にシーンを構成する音や映像といった各メディアを、優先度とは無関係にバリア同期させるような手法が必要となる。

この問題に対して、表現メディアの交換において必要な時に付随する情報を別途イベントとして交換し、その解釈によってデータ供給のタイミングを制御することが考えられる。例えば図2(b). に示すように、シーン切替えの瞬間に、各コネクションのシーン切替えイベントが揃うまですべてのコネクションの動作を待機させたり、図2(c). に示すように連続メディアを扱わない他の通常のアプリケーションとの間で情報を交換するなど、より柔軟な同期の制御が可能になる。

イベント送信には、データコネクションとは別のコネクションを設けることもできるが、利用するプロトコルを制限しないため、モデル上データコネクションを使って送信するのが妥当と考えている。

ここで、簡単な例として図3に示す、シーン切替えを含むマルチメディアプレゼンテーションをもとに、イベント駆動型同期の枠組を紹介する。

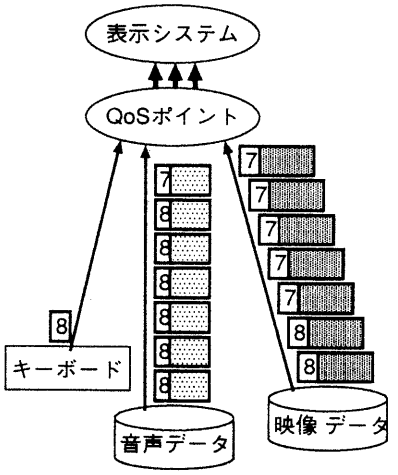


図 3: イベント駆動型同期を用いたアプリケーション例

図は、蓄積された音声／映像で構成されるマルチメディアデータの再生、特にシーン切替えのタイミングをキーボードからの指示で行なう例を

示している。QoS ポイントに入るデータを構造化し、その1エントリーがシーン番号を表すとする。このシーン番号をイベントと考え、それが常に同一でなければならないという制約を QoS ポイントに与える。

図は、シーン7から8への移行状態で、音声データはほぼ送信を終了しかかっている一方、映像にはまだ送信されるべきデータがある状況を表している。しかし、上記制約からシーン8用の音声データは他の全てのコネクションにシーン8用のデータが揃うまで待ち状態となり、表示システムに対しては供給されない。ここで例えば、最後にキーボードの入力が来れば、その瞬間をとらえてシーンが切替わることになる。

なお、前記待ち状態の間にシーン7用とシーン8用のデータが前後入れ替わって到着した場合も、シーン7用のデータが先に表示システムに渡される。

3 周期スレッド

QoS ポイントは、マルチメディアアプリケーションから時間依存のデータ処理を行なう部分だけを分離し、これを OS の機能として提供するものである。これにより、アプリケーションプログラムのうち時間に依存しない処理を行なう部分だけがエンティティ（デバイスドライバを含む）として独立したものとなり、プログラマは QoS ポイントとの間のＡＰＩを用いてエンティティをプログラムすることで連続メディアを扱うアプリケーションを記述できることになる。

ＡＰＩでは、QoS ポイントを、エンティティが持つコネクションごとの通信ポートとは別の、コネクション上に独立に存在するポートとみなしたインタフェースを提供する。また、QoS ポイント自身は、相手エンティティ側の QoS ポイントとの間で通信し、またはファイル／メモリ／デバイスなどの操作を行ない、かつ連続性の監視や同期の修正を行なう自律的なプロセスとして動作する。よってこれを、図4. に示すように、固有のバッファ領域と動作プログラム、タイマを持つ周期スレッドによる実現を提案する。

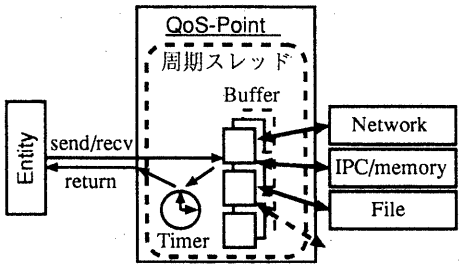


図 4: 同期ポイントの周期スレッドによる実現

以下、周期スレッドの基本的な動作を説明する。

- 相手となる QoS ポイントとの間での通信や、ファイル／メモリ／デバイス等の操作を行なう。通信に関しては実時間プロトコル等を前提としない。

- データの連続性と同期を監視するとともに可能な範囲でその調整を行ない、実時間での同期を行なう。ここでは一つの QoS ポイントを通る複数のコネクション間の依存関係を木構造で管理して互いの同期をとるため、大域的な論理時刻は使用しない。

- 指定されたイベントの解釈ルールをもとに、イベント駆動型同期を行なう。イベントは内部では単なる数値またはラベルによって表現される。データ読み出し側のエンティティは必要に応じてイベントで表される情報を利用することもできる。

- 処理のサイクル時間を持ち、エンティティからの呼び出し（読み出し／書き込み）に対し、あとに述べる周期的な動作を行なう。

- エンティティからの呼び出しの返り値として連続性と同期の程度を報告する。これは、コネクションごとの時間的解像度、空間的解像度で表された連続性と、そのコネクションが取るべき同期との「ずれ」によって表現する各々の QoS を表すものである。

- 読み出し/書き込みのデータが足りない場合、必要に応じてダミーデータや補間データを生成する。

4 周期スレッドインタフェース

周期スレッドは以下の形で呼び出される。ここではメッセージパッシングの標準化案[5]を参考とした。

$$\begin{bmatrix} \sqcup \\ a \\ v \end{bmatrix} \begin{bmatrix} p \\ b \\ n \end{bmatrix} - \begin{bmatrix} send \\ recv \end{bmatrix}$$

上の組合せは関数名を表しており、例えば“ap_send()”のような関数が用意される。それぞれの動作を下で説明する。

1. 読み出し／書き込み

   send: エンティティから QoS ポイントへのデータの書き込み。

   recv: エンティティの QoS ポイントからのデータの読み出し。

2. 読み出し／書き込み時のデータ構造

   ⊔ 指定なし。バッファに順番に書き込む。

   a 配列指定。同じ型、サイズのデータを等間隔（引数指定）を取ってバッファに書き込む。

   v ベクトル指定。任意の構造（引数指定）に従って、バッファに書き込む。この構造を用いてイベントを定義する。

3. 実時間同期

   p periodic: 処理のサイクル終了時刻にのみリターンする。その時点で、必要なデータ量に対して供給し又は供給されたデータ量の割合、他のコネクションとの同期の程度を QoS として返す。なお、呼び出しの有無に関わらずサイクルは進むものとする。

   b block: 通常のソケットインタフェースと同様、サイクル時間とは無関係にスレッドの処理完了までリターンしない。通信の場合は相手からの ack を待つ。

   n non-block: サイクル時間とは無関係に、呼び出しの直後にリターンする。

図5に periodic 指定による同期をとったデータ読み出し (p_recv) の動作例を示す。図は、呼び出しの時点に関わらず、サイクル終了時点にのみリターンすることで、結果としてエンティティも

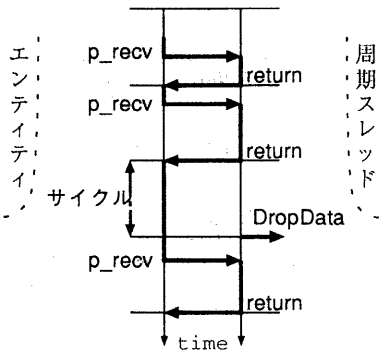


図 5: p_recv の動作

周期的に動作し、またサイクル内に p_recv が呼ばれなかった場合はそのサイクルのデータが捨てられることを表している。このほかに、block/non-block 指定の処理（通信）により、従来の非連続メディアの交換を記述することもできる。

イベント駆動型同期については、応用目的ごとに送受信の各エンティティ間でプロトコルを規定する必要があるが、一般的と考えられるイベント解釈ルールを以下に示す。

- identical
  イベントが同一である時のみエンティティに対してデータを供給する。よって、全コネクションの持つイベントが同じになる瞬間に同期する。

- increment
  数値型のイベントにおいて、その値が増えた直後にデータを供給する。例えばテレビ電話等の応用において、定期的に increment を起こすことにより、声と映像との極端な「ずれ」を防ぐことができる。

- change
  イベントが変化した直後にデータを供給する。変化の種類は問わない。

  これら以外に、ライブラリ等を用いて解釈のルールを任意に設定できるようにする予定である。

これらの設定は、受信側のエンティティが、QoS ポイントを生成する際に行なう。後に述べ

るように、受信側のエンティティは送信側のエンティティの存在と、その仕様を知った上で自身との間にコネクションを張るため、いかに設定するかは単独で決定することができる。

以上述べた周期スレッドにより、以下のような利点が得られる。

1. ハードウェアに依存しない実時間同期処理の枠組を提供する。アプリケーションプログラムが、ハードウェアの進歩による処理の高速化の影響を受けない。

2. 実時間同期処理の記述が容易である。

   - アプリケーションがQoSを管理する必要がない。プログラマは、実時間性が守られているか、他との同期が守られているかを、send/recvの返り値によって知ることができる。
   - 周期スレッドが、サイクル終了時刻までリターンしないため、エンティティも周期的な動作を行なう。

3. マルチメディアシステムを、イベント駆動型同期を用いて柔軟に構築することができる。

4. 従来の非連続メディアの処理との統合ができる。

5 QoS ポイントの管理

モデル上、QoSポイントは実際のデータ転送の有無とは独立に存在しうる。例えば、デバイスからの出力に周期性を与えるものや、受信者の有無に関わらず放送型サービスの提供を行なうもの等について、その存在と、送信可能なデータの型、送信データの時間的／空間的解像度などの仕様を、大域的なアドレスを用いて知らせておくことができる。アプリケーション（エンティティ）は、立ち上げ時には相手となるQoSポイントを指定してOSに自身の動作に必要なQoSポイント（周期スレッド）の生成を要求し、それ以降は既に生成したQoSポイントにコネクションの管理を要求することになる。

6 おわりに

マルチメディア統合環境の実現に向けた同期機構の考察を行なった。また、それを実現する手法として、データ転送の連続性及びその同期の管理を行なう周期スレッドを提案した。実時間同期機構の提供により、ハードウェアの高速化に依存せずに連続メディアを扱うプログラムを記述でき、また時間的な制約を記述するというプログラマの負担を大きく軽減することができる。さらに、イベント駆動型同期の提供により、より柔軟なシステムの構築が可能となる。

なお、QoSポイント生成の要求に対して、周期的な動作を可能な限り保証するため、相応のシステムリソース（CPU時間、ネットワークバンド幅、バッファ領域等）を確保する必要があり、筆者らのグループでもその方式を検討中である。

また周期スレッドの中で行なう通信以外の処理（ダミー／補間データの作成等）について、処理時間の予測をいかに行ない、またいかにそれらをライブラリ化して提供するかを検討していく。

参考文献

[1] D. P.Anderson, "Metascheduling for Continuous Media ", ACM Trans. on Computer Systems, Vol. 11, No. 3, pp.226-252, Aug. 1993.

[2] 岡村, 吉川, 稲垣, 荒木, "QOS 指定可能なマルチメディアモデルの提案", 情報処理学会マルチメディア通信と分散処理ワークショップ-(34), Nov. 1993.

[3] D. P.Anderson and G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism", IEEE Computer, pp.51-57, Oct. 1991.

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

[5] W. Gropp and E. Lusk, A test implementation of the MPI draft message-passing standard", TR ANL-92/47, Argonne National Laboratory, Dec. 1992.