システムソフトウェアと
オペレーティング・システム 64−16
マルチメディア通信と分散処理 65−16
（1994. 5. 20）

マルチメディア統合環境における QoS 管理機構

吉川耕平† 田中裕之 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マルチメディア処理機能の実現を目指した研究を進めている。我々のモデルでは、データを転送するコネクションでのデータの流れに時間的解像度／空間的解像度といった連続性と、他のコネクションとの同期の程度とを QoS として定義しており、コネクション上の QoS ポイントでその管理を行なう。本稿ではまず、我々の考えるマルチメディア処理の QoS について述べ、続いて QoS ポイント機能を提供するサーバ（CS サーバ）の設計、その基本機能である同期制御、バッファ管理をもとにした QoS の管理手法、高負荷状況での転送レートの変更を容易にするサーバ間プロトコルについて述べる。

QoS management for Integrated Multimedia Environment

Kouhei YOSHIKAWA† Hiroyuki TANAKA 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 of multiple continuous media streams which need to be synchronized with each other. We have already defined QoS, i.e., the degrees of continuity and synchronization of streams. In this paper, we introduce our model of multimedia processing and its implementation of synchronization mechanism called CS Server (Continuity and Synchronization Server). Its basic function is realtime synchronization which is adjustment of transmission rate of streams according to user defined QoS. Then we discuss about protocol among CS Servers. Multimedia must support large size of data, so the protocol must work under overloaded conditions. Our design policy is "No news is bad news".

1 はじめに

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の処理を、従来の非連続メディア処理と統合させたマルチメディア処理機構の研究が盛んに進められている [1][2]。我々もこれを将来の分散環境上で必須の技術と考え、その処理モデル及び実現機構を提案している [3][4]。我々はこのモデルをもとに OS の機能としてマルチメディア処理機構を提供することを目指した研究を、PDE(Parallel & Distributed Environment)-II として進めている。現在 PDE-II では、以下に示す下位層 (L)/上位層 (H) の 2 つの構成に分けて研究を進めている。

- PDE-II/L
  CPU やネットワークバンド幅、メモリといった資源のリザベーション機構 [5]、及び複数の処理間の実時間同期機構 [6][7] の研究。
- PDE-II/H
  特定のアプリケーションを設定し、既存の OS／ファイルシステム／ネットワーク等の上でそれを試作するとともに、後に述べる QoS 制御プロトコルやイベント同期を含む同期機構の有用性の検証、プログラムインターフェイス、エンティティ実装方式とその管理、マルチキャストプロトコルに対応したモデルの拡張等の研究。

本稿では、既に提案しているモデルを簡単に紹介したのち、我々のマルチメディア処理における QoS のとらえ方を述べ、続いて現在 PDE-II/H で進めている QoS 管理のための CS サーバの設計、及び CS サーバ間プロトコルについて述べる。

2 マルチメディア処理モデル

我々の提案したモデルでは、エンティティ（アプリケーションやデバイス）間に張ったコネクション上を、送受信の連続性と、他のコネクションとの同期という時間的制約を持つデータを伝送し、コネクション上に存在する QoS ポイント（図 1.）がその制約を管理する。連続性とは、送信が連続的すなわち周期的に行なわれることを前提として、その時間的解像度（周期時間に相当）と空間的解像度（1 周期あたりのデータ送信量に相当）を守るという制約である。一方同期制約とは、各コネクションに処理周期ごとに論理的な時刻を刻む論理時間を考えた上で、“複数のコネクション間の論理的な時刻の差を一定に保つ” という制約である。

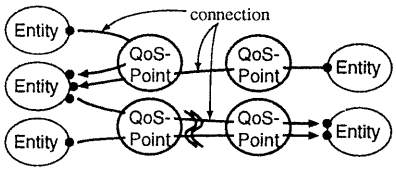

図 1: QoS ポイント

QoS ポイントは、コネクション上でデータ送信エンティティの側からデータを入力として受け、データ受信エンティティの側へそれを出力するポンプの役割を果たす。コネクションは、単一ホスト内だけでなくネットワークをはさんだ複数ホスト間に張ることができ、また同期する必要のある複数のコネクションは QoS ポイントを共有するものとしている。

QoS ポイントにおいて、同期される側のコネクションをマスターコネクション、同期をしていく側のコネクションをスレーブコネクションと呼び、QoS ポイントではスレーブコネクションの論理時間をマスターコネクションのそれに合わせるような同期制御を行なう。なお、実世界をマスターコネクションとみなした場合には自身の連続性制約のみに基づいてあとに述べる実時間同期機構が働き、また自身をマスターコネクションとみなす場合には（連続性制約を含めて）何の時間的制約も受けないことになる。

3 QoS

我々は QoS ポイントにおける各コネクションに以下の制約を設定している。

1. 連続性制約
   - 1 処理あたりの周期時間
   - 1 処理あたりの出力データ量
2. 同期制約
   - 同期すべき論理時間を刻むコネクション

また以上の各々に対して以下の QoS(Quality of Service) を設定可能としている。

1. 連続性の品質
   - 1 処理あたりの出力データ量の最小値
   - 1 周期の遅延時間の最大値

以上が満たされた時、その周期の処理が終ったものとして論理時刻を進めることができる。

2. 同期の品質

   - 同期のずれの許容範囲
     マスターコネクションとの論理時刻間の差の最大値。

   この許容範囲を越えた時、後に述べる同期制御が起動される。

なお送信側／受信側の QoS ポイントに設定する制約及び QoS の値が異なることも許しており、いずれの QoS ポイントも自身の時間的制約と QoS に従ってデータを出力する。

4 CS サーバ

QoS ポイントは、前記制約とその QoS を管理し、またその制御を行なう自律的なプロセスという側面を持つ一方、エンティティに対して通常のポートと同様 send/receive といったインターフェイスを提供する必要がある。我々は QoS ポイントを OS の機能として提供することを目的に、同一ホスト内のすべての QoS ポイントの機能を提供するものを CS サーバ (Continuity & Synchronization Server) と名付けてその設計を行ない、実装を進めている。その構成を図 2に示す。

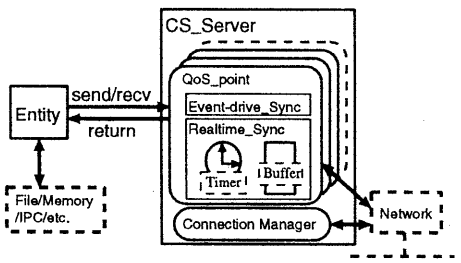

図 2: CS サーバ

CS サーバは、コネクション管理部と、個々の QoS ポイントの実体となる同期制御部を持ち、同期制御部は実時間同期機構及びイベント駆動型同期機構からなる。実時間同期機構は PDE-II/L により提供され、バッファ／タイマ／個々のネットワークポートを持って動作するが、コネクション管理部でも CS サーバ間の制御メッセージを交換できる。CS サーバの基本的な動作を以下に掲げる。

- 個々の QoS ポイントごとに、相手側 CS サーバや同一ホスト内のエンティティとの間でコネクションの管理とデータ送受信を行なう。
- PDE-II/L で提供される実時間同期機構を用いてデータの連続性と同期の監視及び調整を行なう。
- 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。
- 各 QoS ポイントごとに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、周期的にデータ（又は制御）を返す。
- エンティティからの呼び出しの返り値として QoS の状態を報告する。

5 QoS の管理

ネットワークのバンド幅 (bit/sec) は各コネクションに設定された QoS に比べて十分大きいこと、コネクションごとに必要なバッファの量が前記 QoS をもとに PDE-II/L の機能として確保されることを前提として、以下 CS サーバによる QoS の管理について考える。

PDE-II/L の実時間同期機構は、設定された周期ごとにデータを貯めたバッファへのポインタを返すとともに、QoS の状態として 1) 周期ごとの終了時刻に対してどの程度遅延が発生したか、2) 周期あたりに提供する理想的なデータ量のうちどの程度を供給できたか、3) 同期すべきコネクションの論理時刻との間でどの程度ずれが生じているか、といった情報を返す。

また、マスターコネクションに対する同期の QoS を超過した時に以下の同期制御（図 3. 参照）を起動する。

- 遅延挿入
  相対的にデータ入力が多過ぎるコネクションにおいて意図的にデータの出力を遅らせる。
- データスキップ
  遅れぎみのコネクションにおいて、完全に遅れたデータを捨て、次のデータを出力する。

図は、マスターコネクションの論理時刻 (mt とする) と、スレーブコネクションの論理時刻 (st とする) を軸に取っており、各周期の処理が理想的な状態では常に対角線上で、すなわち st = mt の瞬間に始まることを示している。図の遅延挿入の例は、4 回目の

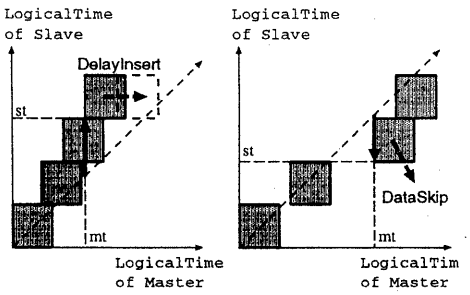

図 3: 遅延挿入とデータスキップ

処理開始の瞬間に mt が st に比べて同期の品質を越えて遅れたため、またデータスキップの例は、3回目の処理開始の瞬間に mt が st に比べて同じく進んだため、それぞれの制御が働くことを示している。

これらの制御は映像と音声の同期といったコネクション間の同期に必須のものであるが、最悪の場合、一時的に QoS を超過した遅延が発生すると、その後通常の速度でデータを受信しても、その多くをデータスキップにより捨ててしまうという問題がある。これを避けるためには、他の CS サーバやエンティティとの間でデータ入出力速度の調整を行なう必要があるが、それを実時間同期機構の中で行なうことは必要以上の負荷となり、本来の同期処理に大きな支障をきたす。また、ネットワークを含むシステム全体の負荷の状況を知らずに個々のコネクションごとに調整を行なっても全体の効率化につながる可能性は低い。

そこでこの調整は、ホスト内の全 QoS ポイントを管理する CS サーバがコネクション管理部により行なうこととした。しかし先に述べた QoS だけでは、その状態変化が同期制御によるものか、あるいはネットワーク等の負荷の変動によるものかを判断できない。そこで、負荷を計るために実時間同期機構から簡単に取り出せる情報として、実時間同期機構におけるバッファの使用率を考えた。

図 4. は、QoS ポイントへの入力データの供給率と、QoS ポイントからの出力データの供給率の相関を示したものである。例えば、データ受信側の QoS ポイントで考えると、入力はネットワークのバンド幅、出力はデータ受信エンティティの処理性能（典型的にはエンティティの周期）と考えられる。データ送信側 QoS ポイントにおいては逆に、入力がデータ送

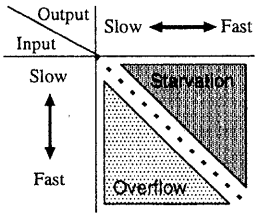

図 4: バッファ使用率と入出力の相関

信エンティティの処理性能、出力がネットワークとなる。入力／出力の率が安定している場合には図の対角線の部分でバッファ使用率は安定するが、相対的に入力が速くなるとバッファの使用率の上がる度合が一定値を越えた「データ溢れ (overflow)」の状態に、また出力が速くなるとバッファの使用率の下がる度合いが一定値を越えた「データ枯渇 (starvation)」の状態に陥ることを表している。同期制御は、入出力の供給率すなわちそれらの速度を論理時間の進行に合わせることを目的としているため、片方にだけ加速度がかかることはこの制御の自由度を失ってしまうことになる。またバッファが完全に溢れてデータが失われたり、あるいは枯渇して供給すべきデータがまったくなくなった場合、いずれも連続メディアの出力が途切れるといった事態に陥る可能性がある。なお、マスターコネクションに遅延が発生すれば、スレーブの方は許される範囲で論理時間を先へ進めるため、出力が速過ぎるという状態も発生し得る。

さらに先の同期制御をバッファ使用率の観点から見ると、遅延挿入にはデータ溢れの場合にさらにバッファの使用率を上げ、データスキップには同じくデータ枯渇の場合にさらにバッファの使用率を下げるという性質がある。これに対し、ある瞬間には QoS の状態に大きな変化がなくともバッファの使用率からその変化を予測し、入出力の速度を調整することが可能になる。予測にもとづいてこれらの調整を行なうことは、状態の変化にすばやく対応でき、その結果システムの挙動を安定させるという効果が期待できる。

次に、調整の方法として入力出力のどちらをどの順序で適用するかが問題となる。

まずエンティティの処理性能の調整の方法としては以下がある。

- QoS の許容範囲内で論理時間を一時的に進めた

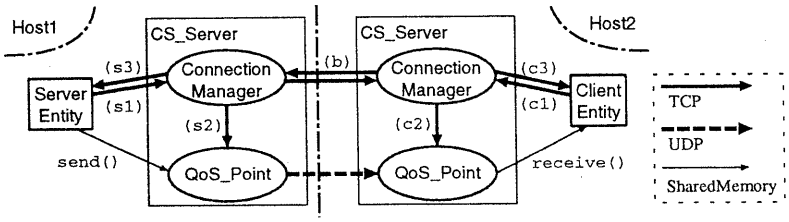

図 5: プロトタイプシステムの構成

り止めたりする。

受信側エンティティへのデータ出力、あるいは送信側エンティティからのデータ入力のタイミングを一時的に動かすことになる。

- QoS の許容範囲内でエンティティの周期時間を変える。

  ずれの度合が一定の割合で大きくなる場合に対応する。これにより以降、定常的に QoS の状態は悪くなる。

なお、将来 CPU スケジューリングに対して要求を出すことが可能になれば、それも使うことになる。

次に、ネットワークの速度調整については、相手側 CS サーバとの間で次節で述べるプロトコルに基づいて行なうことになる。これは、エンティティの周期を変えた場合に必ず行なう必要があるが、追従性はそれに比べてかなり落ちる。

以上のことから、速度調整として、まずコネクションの論理時刻を一時的に変え、続いてネットワークの速度を、最後にエンティティの周期を変えるのが妥当と考えられる。

6 CS サーバ間プロトコル

現在プロトタイプでは下位層に既存のプロトコルを用いているため、バンド幅を予約できないことが大きな問題であるが、将来的にバンド幅の予約が可能になったとしても、異種分散環境での連続性及び同期の管理のための制御プロトコルは必須のものと考えている。

現在試作中のシステム構成を図 5. に示す。コネクション管理は CS サーバ間で TCP を用いた通信を使い、データの送受信についてはエンティティとの間では共有メモリを、他の QoS ポイントとの間では UDP を用いて実装を進めている。

コネクション設定のおおまかな手順を述べる。まず送信側エンティティは、CS サーバに対して送信可能なデータに関する情報（データの型、QoS 等）を送ると (s1)、サーバは QoS ポイント（ポートに相当）を生成して (s2)、エンティティに返す (s3)。この後 CS サーバは自ホストから送信可能なデータに関する情報を他の CS サーバにブロードキャストしておく (b)。次に、受信側エンティティは、受信したいデータに関する情報をサーバに送り (c1)、CS サーバが適切なものを見つければ、QoS ポイントを生成し (c2)、そのコネクションを張って、受信側エンティティに QoS ポイントを返す (c3)。以降、データのやりとりは send/receive を使って QoS ポイントを通して行なう。定常状態では個々のデータに ack を返さず、適当な時間間隔をおいてコネクション管理部の間で受信状態を表す制御メッセージを交換（図中 (b)）する。

データ送受信中の QoS ポイント間のメッセージには以下の種類を用意した。

1. (none): データ枯渇のためバンド幅を上げたい
2. UP: QoS 悪化のためバンド幅を上げたい
3. KEEP: 動作が安定しており、バンド幅を維持したい
4. DOWN: データ溢れのためバンド幅を下げたい

特に問題となるのはバンド幅を上げたいときで、それはネットワークの負荷が上がって混雑しているためにデータの到着が遅れたり消失したりしている場合と、ネットワークは正常であるが相対的にエンティティの処理性能が上がっている場合との 2 つの原因が考えられるためである。それ以外の場合にはネットワークの負荷は正常であると考えてよい。

以上のことから、受信側 QoS ポイント及び送信側 QoS ポイントにおける基本的な戦略を以下のように定めた。

受信側 QoS ポイント：

- 定常状態
  → QoS をもとに UP/KEEP/DOWN を定期的に送信する。
- データ枯渇に陥り始めた場合
  → 何も送らない。
- データ枯渇の状態から回復し始めた場合、あるいは UP を出してその速度が少しでも上がった場合
  → 一旦 KEEP を送信し、以降定常状態に移る。

送信側 QoS ポイント：

- 定常状態
  → KEEP/DOWN にはそのまま答える。小数の UP 要求については自身がデータ枯渇になっていないコネクションであれば、一つずつに順に答え、答えたところから KEEP が来るまで別の UP は無視する。
- 一部のコネクションから何も来ない場合
  → 該当するコネクション以外の自分の管理するコネクションを、制約、QoS、スレーブかマスターかの情報を元に順序付けしてスピードダウン。
- 多くのコネクションから UP パケットが届くか、あるいは何も来ない場合
  → 自分の管理するコネクションを、同様に順序付けして一時切断。

基本的には何か起こればすべてのコネクションの速度を一旦下げ、そこからゆっくりと回復を図るようにし、CS サーバが管理するコネクションのすべてがデータ枯渇に陥らない限り、コネクション間で負荷の分散を図ることとしている。また、データ枯渇に陥った場合など、バンド幅を上げるか、又は他のコネクションのそれを下げてバンド幅を確保するよう受信側 CS サーバが要求を出すが、ネットワークが混んでいる場合にはその要求パケット自体がさらにネットワークに負荷をかけることになりかねないため、送信側 QoS ポイントでは「メッセージが来なければ、ネットワークが混雑してデータ枯渇に陥ったとみなす」という前提を設けた。

7 おわりに

分散マルチメディア環境の実現に向け、処理の連続性及び同期の制御機構やその上での QoS の管理機構を OS が提供することで、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し、またハードウェアの高速化に依存しない連続メディアを扱うシステムの開発が容易になると考えている。

本稿では、マルチメディア統合環境における QoS の扱いについて述べ、その管理を行なう CS サーバの設計と、バッファ使用率をもとにしたサーバ間のプロトコルについて述べた。現在プロトタイプの開発とその評価を進めているところである。特にプロトコルの検証として、コネクション間の負荷分散まで含めた定量的な評価を行ないたい。

また今後は、映像や音声といった個々の連続メディアの性質に依存した QoS の管理やその処理、マルチキャスト上での QoS 管理のためのモデルの拡張などを行なうとともに、実際にマルチメディアを扱うエンティティの実装と、それを使った分散マルチメディア統合環境の構築を進める予定である。

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] 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.

[3] 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.

[4] 吉川, 岡村, 荒木, "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", 情報処理学会マルチメディア通信と分散処理研究会 94-DPS-64-1, Mar. 1994.

[5] 松尾, 岡村, 荒木, 福田, "マルチメディア処理におけるＯＳレベルでのリソース・リザベーション", 情報処理学会システムソフトウェアとオペレーティング・システム研究会 94-OS-63-10, Mar. 1994.

[6] 稲垣, 岡村, 荒木, "PDE-II におけるメディア間同期機構の実現に関する考察", 情報処理学会第４８回全国大会 1H-6, Mar. 1994.

[7] D. P.Anderson and G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism", IEEE Computer, Vol.24, No.10, pp.51-57 Oct. 1991.