QoSに基づいた連続メディアデータの淘汰機構

稲垣英太郎　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

マルチメディアのデータをマルチキャストして放送などを行なう場合、ある受信側の計算機の処理能力が不足すると、連続メディアを円滑に再生できなくなる。これは、連続メディアの品質を変更することで回避できるが、従来のシステムは、このような機能を提供していない。このため我々は、データの取捨選択をして連続メディアの品質を変更する"データ淘汰機構"の開発を行なった。本稿ではデータ淘汰機構の概要について述べ、またそのプロトタイプの評価、検討を行なう。

1　はじめに

近年、分散環境でマルチメディアを利用するアプリケーションの中には、マルチキャストやブロードキャストを利用して、音声や動画の放送を行なえるものが現れ始めている。このようなアプリケーションを利用する場合、受信側のある計算機が過負荷であるなどして、送信されてくるデータを処理し切れなくなると、連続メディアを円滑に再生できなくなる、などの弊害が発生する。

このような状況を回避するためには、連続メディアの再生に要する負荷を軽減することが必要であるが、これは送信されてくる連続メディアの品質を受信側で変更することにより行なうことができる。

ところが従来のシステムでは、連続メディアの品質を変更する機能を提供していないため、アプリケーションレベルでその操作を行なっているのが現状である。しかし、これではデータを取捨選択する機能をアプリケーションに備える必要があるため、プログラマに負担がかかるという問題がある。また、たとえアプリケーションでそのような操作を行なっても、連続メディアの品質を変更する段階で不要になったデータをアプリケーションに渡すオーバーヘッド、即ちカーネル空間からユーザ空間に不要なデータをコピーするオーバヘッドがかかるため、あまり負荷を軽減できないという問題が残る。

我々は、不要なデータをユーザ空間にコピーしないことによって、この問題を解決する機構を、現在開発を行なっている分散並列／マルチメディアOS PDE-IIの通信プロトコルの処理レイヤに備えることにした。ユーザ空間にコピーするデータとしないデータの振り分けは、この機構の内部で取捨選択することにより行なうが、アプリケーションからは不要なデータがいつのまにか消えてしまう、即ち淘汰されてしまうように見えることから、我々はこの機構を"データ淘汰機構"

"A Data Dismissal Mechanism based on QoS",
Eitaro INAGAKI, Koji OKAMURA and Keijiro ARAKI
Graduate School of Information Science,
Nara Institute of Science and Technology

と命名した。我々はデータ淘汰機構を開発するに当たり、まずデータを取捨選択するための指標である「データ淘汰機構におけるサービスの品質（QoS）」の定義を行ない、データ淘汰機構がアプリケーションの提示するQoSに基づいてデータの取捨選択を行なえるようにした。

本稿の構成は次の通りである。2章では、データ淘汰機構について述べる前段階として、PDE-IIにおけるマルチメディアの処理について説明する。次に3章でQoSの定義およびデータ淘汰機構について述べる。4章で試験的に実装したデータ淘汰機構のプロトタイプを用いて、データの取捨選択を行なうことで計算機の負荷がどれほど軽減できるかを検討する。最後に、5章で本稿のまとめを行なう。

2 PDE-IIにおけるマルチメディアの処理

人間が知覚できる音声や映像などの知覚メディアを計算機で処理できる表現メディアに符合化した場合、その表現メディアの処理には知覚メディアの時間的制約が必要になる[1]。このような時間的制約を必要とするメディアを連続メディアと呼ぶ。また、多数の連続メディアを統合した一群のメディアをマルチメディアと呼ぶ。

我々は 分散並列／マルチメディア OS PDE-II を構築する上で、マルチメディアのデータ通信とその処理のモデル化を行なった。本章では、データ淘汰機構について説明する前段階として、我々の提案するマルチメディア処理モデルについて述べた後、QoSを定義する上で重要になる連続メディアの品質とその変更方法について説明する。

2.1 マルチメディア処理モデル

図1に、我々の提案するマルチメディアの処理およびそのデータ通信を統合したモデルを示す[2]。図1は、多数のエンティティと呼ばれるデータの処理や通信を行なう実体が、コネクションと呼ばれる仮想伝送路を通じてマルチメディアのデータ通信を行なっている様子を示している。

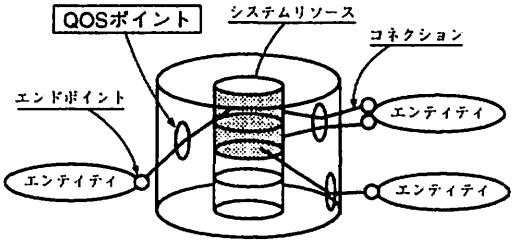


図1: マルチメディア処理モデル

エンティティとは、プロセスやデバイスなど、マルチメディアのデータ処理およびそのデータ通信を行なう主体である。

またコネクションとは、エンティティ間に張られた仮想伝送路であり、この上をマルチメディアのデータが流れる。コネクションは単一の連続メディアのみを流す伝送路である。このため、例えば音声と動画をエンティティ間で通信し合うためには、2本のコネクションが必要になる。

図1中、システムリソース円筒の芯に相当する部分は、エンティティ間で通信を行なうために必要なシステムのリソース、CPUタイムやメモリ、ネットワークバンド幅などを抽象化したものであり、システムリソースと呼ばれる。システムリソースの高さはシステムの全処理能力を表しており、網がかかった部分は、データの通信やその処理のために利用されているリソースであることを示している。

エンドポイントとは、データの入出力口

であり、エンティティはこれを介してデータの送受信を行なう。エンティティはエンドポイントを任意個持つことが可能である。

コネクションと円筒の表面との交わりに位置するのがQoSポイントである。ここで、連続メディアの品質および連続メディア間の同期をとる。QoSポイントは、一本のコネクションに対し、受信側および送信側の2箇所に存在する。データ淘汰機構はこのQoSポイントを構成する要素であるが、これに関しては次章で述べる。

マルチメディア処理モデルにおいて、エンティティは任意のエンティティに対してコネクションを張ることが可能であり、また任意の方向にデータ通信を行なうことができる。このためマルチメディア処理モデルでは、システムのリソースを円筒表現し、その周囲の任意の場所にエンティティを配置することができる。

2.2 連続メディアの品質とその変更

連続メディアの品質は、時間的解像度と空間的解像度で表すことができる[3]。このため、連続メディアの品質を変更することは、時間的解像度と空間的解像度の両者あるいはどちらか一方を変更することである。

空間的解像度とは、ある単位時間内に処理するデータのサイズのことであり、動画の一画面あたりのサイズや、音声の量子化ビット数などに相当する。本稿では、この単位時間内に処理する一群のデータを、データフレームと呼ぶ。一方時間的解像度とは、一秒間に処理するデータフレーム数のことであり、例えば一秒間に表示する動画の画面数や音声のサンプリング周波数に相当する。

次に時間的解像度と空間的解像度の変更方法を、動画を例にとって具体的に説明する(図2参照)。図2において、(a)は4枚の画面から成る動画の例を、(b)は空間的解像度を変更した例を、(c)は時間的解像度を変更した例を示しており、画像一枚のピクセル数は横x ピクセル、縦y ピクセルであるとする。

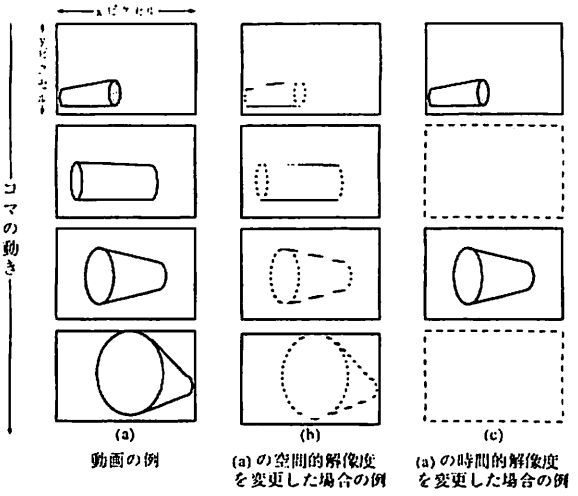


図2: 動画像における時間的解像度と空間的解像度の管理

図2(b)は、図2(a)を1ラインおきに再生してデータフレームのサイズを変えることで、空間的解像度を変更した動画の例を示している。一方図2(c)は、図2(a)を一画面おきに再生して1秒間に処理するデータフレームの数を変えることで、時間的解像度を変更した動画の例を示している。つまり、空間的解像度の変更はデータフレーム内部のデータを取捨選択することで、また時間的解像度の変更はデータフレーム単位でデータを取捨選択することで行なうことができる。

3 データ淘汰機構

図3に、図1におけるQoSポイントを拡大した図を示す。QoSポイントは、図3に示すように、同期機構およびデータ淘汰機構から構成されている。同期機構は、複数のコネクションを伝送する連続メディアデータの流れを調節して、連続メディア間の同期の管理

を行なうものである [4]。一方データ淘汰機構は、コネクションを流通する連続メディアのデータを取捨選択して連続メディアの品質を変更するものである。以下で、データ淘汰機構について説明する。

3.1 QoS

図 2(a) の空間的解像度を変えて、図 2(b) のような再生を行なうには、x ピクセル受理した後、x ピクセル破棄すればよい。また、図 2(a) の時間的解像度を変えて、図 2(c) のような再生を行なうには、x × y ピクセル受理した後、x × y ピクセル破棄すればよい。つまり時間的解像度と空間的解像度両者の変更は、どちらも「i バイト受理／破棄した後、j バイト破棄／受理する」という操作で行なうことができる。言い方を換えると、破棄するデータのサイズと受理するデータのサイズを変えることで、時間的解像度と空間的解像度どちらも自在に変更することができる。このため、データの取捨選択を行なう場合、連続メディアの品質の変更は次のパラメータで指定することができる。

- 破棄するデータのサイズ
- 受理するデータのサイズ
- 破棄から先に行なうか、受理から先に行なうか

データ淘汰機構におけるサービスの品質である QoS は、上記の 3 項目で表される。データ淘汰機構はアプリケーションの提示する QoS に基づいてデータの取捨選択を行なう。

3.2 通信プロトコルの処理レイヤとデータの取捨選択

図 4 に、データ淘汰機構のプロトタイプの開発ベースである BSD 系 UNIX、NetBSD の通信プロトコルの処理レイヤの構成を示す [5]。

データ淘汰機構では、データの取捨選択をトランスポート層かソケット層のどちらかで行なう。トランスポート層で取捨選択を行なう方式を粗粒度方式、ソケット層で取捨選択する方式を細粒度方式と呼ぶ。どちらの方式でデータの取捨選択を行なうかは、データ淘汰機構が判断する。以下で、両方式について説明する。

- **細粒度方式**

  データフレーム内部のデータを取捨選択する方式。read システムコールの発行時に、カーネル空間であるソケット層からユーザ空間のバッファへ、破棄するデータをコピーしないことでデータの取捨選択を行なう。QoS の「破棄するデータのサイズ」あるいは「受理するデータのサイズ」のどちらかがデータフレームのサイズより小さい場合、データ淘汰機構はこの方式でデータの取捨選択を行なう。

  この方式は、空間的解像度および時間的解像度の変更に用いることができる。

- **粗粒度方式**

  データの取捨選択をデータフレーム単位で行なう方式。破棄されるデータフレームを構成するパケットを、トランスポート層で解放することによりデータの取捨選択を行なう。細粒度方式に比べて、不要なデータをソケット層で処理するオーバーヘッドがない分、計算機の負荷を軽減する上でより有効である。QoS の「破棄するデータのサイズ」および「受理するデータのサイズ」両者がデータフレームのサイズより大きな場合、データ淘汰機構はこの方式でデータの取捨選択を行なう。

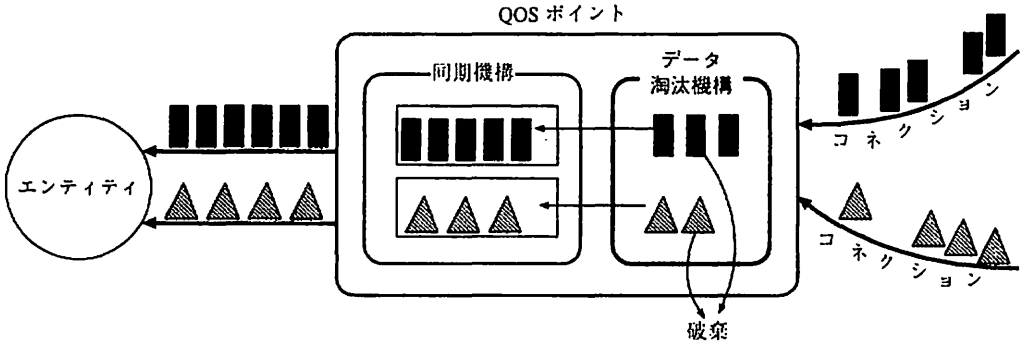


図 3: QoS ポイントとデータ淘汰機構

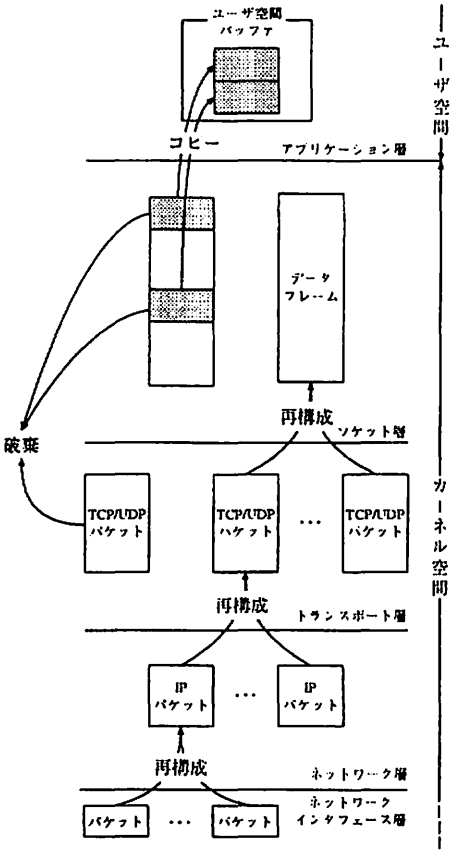


図 4: 通信プロトコルの処理レイヤとデータの取捨選択

この方式は、主に時間的解像度の変更に用いることができる。

3.3 データ淘汰機構インタフェース

通信プロトコルに TCP/IP または UDP/IP を用いている多くの通信プログラムは、一般にソケット API (Application Program Interface) を用いて記述されている。我々は PDE-II においても、これらの通信プロトコルを用いたエンティティ間の通信には、ソケット API を用いることにした。このためソケット API は従来のままとして、データ淘汰機構の機能を利用するため、以下のシステムコールを新たに用意した。

- int dismiss(int fd, struct dsmqos *qos);

  QoS を指定するためのシステムコール。QoS の各パラメータは、次に示す dsmqos 構造体に格納してデータ淘汰機構に渡す。

```
struct dsmqos {
  int brk; /* 破棄するデータサイズ */
  int rcv; /* 受理するデータサイズ */
  short flag; /* 受理が先か破棄が先かを示す*/
};
```

fd はソケットのファイルディスクリプタである。返り値はとしてシステムコールの成否を返す。

このシステムコールは fd が有効ならいつでも用いることができる。このため、データの受信を行なっている最中でも QoS を変更することができる。また、破棄するデータのサイズに 0 を指定すると、データの取捨選択は行なわれなくなる。

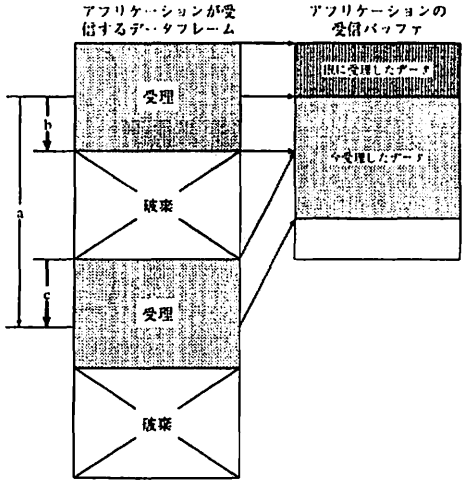
- int readp(int fd, char *buf, int size, int *rcvsize);

取捨選択されたデータの受信は、read システムコールを発行することで行なうことができる。しかし、細粒度方式でデータの取捨選択を行なう場合、バッファのポインタ管理が繁雑になってしまう。そこで、我々は取捨選択されたデータを受信する上でより便利なシステムコール readp を用意した。

readp システムコールは、バッファへ書き込んだデータのサイズを第 4 引数に返す点のみが、read システムコールと異なる（図 5参照）。readp システムコールを用いることで、取捨選択されたデータの受信の記述を次のように、簡潔に記述することができる。

```
for (ds = off = 0; ds < size;
    ds += readp(fd, buf + off, size, &rcvsize))
    off += rcvsize;
```

取捨選択されたデータを受信する場合、read システムコールおよび readp システムコールともに、返り値として受理したデータのサイズと破棄したデータのサイズの和を返す（図 5参照）。



a ··· 一度の read/readp で受理／破棄したデータ総のサイズ。両システムコールともに返り値としてこの値を返す。

b, c ··· 一度の read/readp で受理した即ちユーザ空間のバッファに書き込まれたデータのサイズ。readp は b と c の和を第 4 引数 (rcvsize) に返す。

図 5: readp の返り値と第 4 引数

4 データ淘汰機構のプロトタイプを用いた実験とその評価

今回行なった実験では、データの取捨選択を行なうことにより、1) 計算機が単位時間内に処理できるデータ量とデータフレーム数がどのように変化するか、2) 計算機が一定の転送率でデータを通信している時、データの取捨選択を行なうことにより、受信側の計算機において必要となる CPU の処理能力がどのように変化するか、の 2 項目について調べることを目的とした。このため 1) と 2) の実験では、データの送信を行なうプロセスと、送信側が送るデータの受信とその取捨選択を行なうプロセスをそれぞれ用意し、測定は受信側で行なった。1) の実験では、送信プロセスに間断なくデータを送り続けさせた上で、受信プロセスが 1 秒間に処理できるバイト数

とデータフレーム数を測定した。本稿では、前者のバイト数をバイトスループット、後者のデータフレーム数をフレームスループットと呼ぶ。両スループットともに、高い値を示すほど良い結果であることを示す。一方、2)の実験では、送信プロセスに一定の転送率でデータを送信させて、受信プロセスのシステムタイムを測定した。システムタイムは短いほど良い結果であることを示す。

評価実験に用いた計算機は、i486DX2/66MHz CPU 搭載の Gateway 2000 4DX-2-66V と、同 CPU 搭載の DELL 466/L である。両計算機とも、OS には PC/AT 互換機上で動作する UNIX を搭載しており、前者の計算機には、データ淘汰機構のプロトタイプ（以下、単にプロトタイプと呼ぶ）をソケット層に実装した NetBSD を、後者の計算機には Linux を使用している。両計算機の間は Ethernet でつながれており、ネットワークを使用した実験では、プロトタイプを実装した NetBSD を受信側に用いた。

以下に、我々が行なった 3 通りの実験について説明する。

実験 1　スループットの測定その 1

本実験は、ネットワークを介したプロセス間通信において、時間的解像度と空間的解像度を変更するそれぞれの場合で、取捨選択するデータのサイズを変更することにより、バイトスループットとフレームスループットがどのように変化するかを測定した。このため、$i \times i (i = 1 \sim 512)$ バイトのデータフレームを間断なく 2048 フレーム転送して、次のようなデータの取捨選択を行なった。

(a) データを破棄しない

(b) データを 1 バイトおきに破棄して、空間的解像度を変更する。

この手法は、量子化ビット数 16 ビットの音声を 8 ビットに変更する時や、映像の色数 16 ビット = 65536 色を 8 ビット = 256 色に変更して、空間的解像度を変更する場合などに有効である。

(c) データフレームを 1 ラインおきに破棄して、空間的解像度を変更する。

この手法は、映像を 1 ラインおきに再生して、空間的解像度を変更する場合などに有効である。

(d) データフレームを 1 つおきに破棄して、時間的解像度を変更する。

この手法は、動画を 1 画面おきに再生して、時間的解像度を変更する場合などに有効である。

実験 2　スループットの測定その 2

本実験は、1 ノード内でプロセス間通信を行なった以外 1) と全く同じである。

実験 3　システムタイムの測定

本実験では、実際に動画をプロセス間で送受信した状況を想定して、一定の転送率でデータ通信を行ない、動画の時間的解像度と空間的解像度を変更するそれぞれの場合で、システムタイムがどのように変化するかを測定した。動画の品質は、1 画面あたりのピクセル数を 640 × 480 、1 ピクセルあたり 8 ビット = 256 色、1 秒間に表示する画面枚数を 15 枚として、次の 2 通りの実験を行なった。

(A) 動画の空間的解像度を変更するために、各画面をライン単位で取捨選択して、一画面あたりのライン数を $480, 240, \cdots 30$ と減少させる。

(B) 動画の時間的解像度を変更するために、画面単位で取捨選択を行ない、1 秒間に表示する画面枚数を $15, 7.5, \cdots 1.875$ と減少させる。

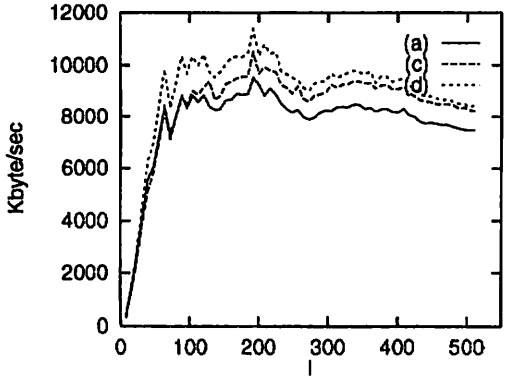


図 6: バイトスループットの比較

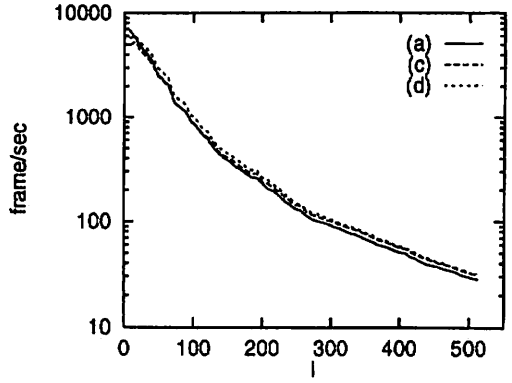


図 7: フレームスループットの比較

本実験で想定した品質の動画を送受信するためには、約 35 Mbps の転送速度が必要になり、最大 10 Mbps の転送速度しか持たない Ethernet を用いたのでは、秒間 15 枚の再生ができない。このため、本実験では 1 ノード内でプロセス間通信を行なった。

4.1 実験 1–スループットの測定その 1

実験の結果、(a), (c), (d) それぞれのバイトスループットおよびフレームスループットどちらもほとんど同じであり、データの取捨選択を行なうことによる変化は見られなかった。一方、(b) の両スループットは、他のそれらの 30% 以下であり、バイト単位でデータの取捨選択を行なう場合、取捨選択を行なわない場合よりもスループットが低下することが分かった。

(a), (c), (d) のスループットが同じであったのは、ネットワークのスループットが受信側の計算機のそれよりもかなり低いため、計測結果にネットワークのスループットが現れてしまったからであると考えられる。一方、(b) のスループットが他に比べて低いのは、取捨選択を行なう粒度があまりにも細か過ぎて、CPU のメモリアクセスがネックになったからであると考えられる。これは、CPU は一般に、メモリアドレスが連続したデータのメモリへの書き込みは高速に行なえるよう設計されているが、アドレスの隣接していないデータを、しかもバイト単位でメモリに書き込むような場合には、メモリへの書き込み速度が著しく低下するためである。このため、破棄するデータのサイズおよび受理するデータのサイズとしてあまり小さな値を指定すると、逆に負荷の高騰を招く恐れがある。

以上のことから、データ淘汰機構は、音声など個々のデータフレームのサイズが小さい連続メディアを扱うにはあまり適していないということが分かる。

4.2 実験 2–スループットの測定その 2

本項では、バイト／フレームスループットの向上に効果のない (b) は除外し、(a), (c), (d) の結果について検討する。図 6 に (a), (c), (d) それぞれについて測定したバイトスループットのグラフを、図 7 にフレームスループットのグラフを示す。図 6 および図 7 ともに、取捨選択するデータのサイズが最も大きい (d) が最も良い結果を、次いで (c) が良い結果を示している。ここで、空間的解像度を変更する (c) の結果に注目してみると、i の値が 1 ～ 100 の間即ち 100 バイト以下のデータを

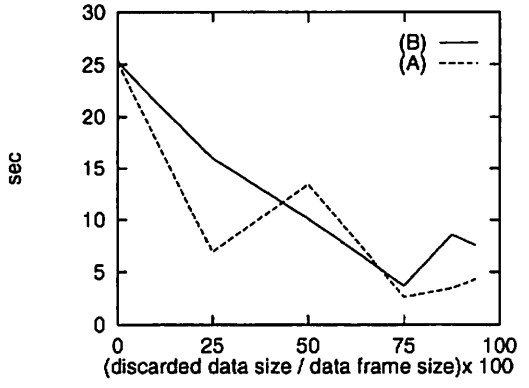


図 8: システムタイムの測定

取捨選択した場合では、データを破棄しない (a) と結果はほとんど変わっていないが、そこから徐々に (a) より良い結果を示し始め、i が 500 を越える付近即ちデータの取捨選択を 500 バイトで行なう付近から、時間的解像度を変更する (d) とほぼ同等の結果を示し始めている。このことからプロトタイプでは、100 バイト以上のデータの取捨選択を行なうことで効果が現れることが分かる。

4.3 実験 3–システムタイムの測定

図 8に (A), (B) それぞれについて測定したシステムタイムのグラフを示す。図 8の横軸は、データフレームのサイズに対する破棄するデータサイズの百分率である。図 8では、破棄するデータのサイズが大きくなるに従って、時間的解像度および空間的解像度のどちらを変更する場合でも、システムタイムは減少していく傾向にある。このことから、破棄するデータのサイズはできるだけ大きな方が、よりシステムタイムの短縮につながることが分かる。図 8からは、時間的解像度を変更する場合と空間的解像度を変更する場合の違いは特に認められなかった。

以上の実験 1、実験 2、実験 3 の結果から、データ淘汰機構は、動画のように個々のデータフレームのサイズが比較的大きな連続メディアを扱うのに適しているといえる。

5 まとめ

本稿では、我々が定義した通信プロトコルレイヤにおける QoS およびデータ淘汰機構について述べ、データ淘汰機構のプロトタイプの評価を行なった。

データ淘汰機構は、受信側でデータを取捨選択して受信側の計算機の負荷を軽減することを目的とする機構である。実験の結果、データの取捨選択はなるべく大きな単位で行なった方が効率が良く、逆にデータの取捨選択を細かい単位で行なうと、極端な性能低下を招くことが分かった。このため、音声のように個々のデータフレームが小さなメディアに対するデータ淘汰機構の親和性は低く、実際音声に対してデータの取捨選択を行なっても、意味をなさないばかりか、かえって負荷の高騰を招く恐れがある。一方動画のように、個々のデータフレームがかなり大きなメディアに対しては、粗粒度方式でデータを取捨選択することにより、計算機の負荷を軽減できることが分かった。また細粒度方式でデータの取捨選択をする場合でも、ライン単位で取捨選択を行なえば、粗粒度方式ほどではないが、ある程度負荷を軽減することができることが分かった。

今後の課題としては、粗粒度方式の効率向上のため、トランスポート層へのデータの取捨選択機能の実装が挙げられる。また現状のデータ淘汰機構では、連続メディアの非圧縮データを扱うことはできても、圧縮されたデータを扱うための配慮がなされていない。今後は、階層化符合化の技術を採り入れて、圧縮されたデータも取り扱えるようにする予定である。

謝辞

本研究行なうに当たって、多くの貴重な御助言、御指導を頂いた奈良先端科学技術大学院大学 荒木研、福田研の皆様に感謝致します。

参考文献

[1] 岡村, 吉川, 稲垣, 荒木 : "PDE-II の概要 ～ QOS に基づいたマルチメディア処理モデル ～ ", 情報処理学会第 48 回, 全国大会, 1H-5, March, 1994.

[2] 岡村, 吉川, 稲垣, 荒木 : "QoS 指定可能なマルチメディア処理モデルの提案", マルチメディア通信と分散処理ワークショップ, November, 1993.

[3] 船渡, 徳田 : "Real-Time Mach 3.0 における連続メディアサーバの実験—QOS 制御を取り入れた QuickTime Player の評価—", 情報処理学会研究報告, マルチメディア通信と分散処理, DPS-61-11, July, 1993.

[4] 岡村, 稲垣, 松尾, 荒木, 福田 : "マルチメディア同期機構の試作と評価", 情報処理学会研究報告, マルチメディアと分散処理, DPS-65-14, March, 1994.

[5] S.J.Leffler, M.K.Mckusick, M.J.Karels, J.S.Quarterman : 中村 明, 相田 仁, 計 宇生, 小池 汎平 共訳 : "UNIX 4.3BSD の設計と実装", 丸善株式会社, 1991.