「マルチメディア通信と分散処理ワークショップ」 平成16年12月

パケットフローコントロールによるDV転送品質向上の研究

入野 仁志[1] 杉浦 一徳[1] 中村 修[2] 村井 純[2]

慶應義塾大学大学院政策・メディア研究科[1] 慶應義塾大学環境情報学部[2]

本研究では既存のDV転送システムであるDVTSにおいて，プロセスの休止を用いた送信間隔の制御によるフローコントロール機能の設計，実装，評価を行った．LinuxのDVTS実装は，設計上，バーストトラフィックを生成する．現在，様々なハードウェア版DVTSの開発が進んでいるが，このようなハードウェア版DVTS機器は多くの場合資源が少なく，バースト的な通信に対応できない．本研究ではこれらの機器への対応を視野にいれ，トラフィックの平滑化を行い，パケットを等間隔で送信することによりバースト的な通信を発生させないようにした．本研究により，ソフトウェア版DVTSと現在開発が進んでいる各種のハードウェア版DVTS機器との相互接続性を高め，DV転送の品質を向上させた．

Improvement of DV Transmission using Packet Flow Control

Hitoshi Irino[1],Kazunori Sugiura[1], Osamu Nakamura[2], Jun Murai[2]

[1]Graduate School of Media and Governance, Keio University
[2]Faculty of Environmental Information, Keio University

In this research, we design and implement a flow control mechanism by controlling the transmitting interval in the existing DV transmission system: DVTS. Performance in flow control mechanism for new implementation is evaluated. DVTS on Linux generates burst network traffic due to its OS design. While various hardware version of DVTS is under development, many of these equipments often have limited resources, thus cannot respond to such burst network traffic. To support the limitation of these equipments, we introduce the flow control mechanism to avoid sending burst traffic by transmitting each packet at equal intervals. From this research, the software version of DVTS has achieved interoperability with other hardware version of DVTS equipments, and improved the overall quality of DV transmission.

1 はじめに

1.1 背景

広帯域データリンクが広く普及したことにより，これまで利用が困難であった単位時間あたりのデータ量が大きい，低圧縮で低遅延，高解像度，高フレーム数の高品質映像メディアを用いたコミュニケーションが可能となった．

高品質映像メディアを用いたコミュニケーションツールの代表例としてDVTS(Digital Video Transport System)[1]があげられる．DVTSは安価に構築可能な高品質映像メディア転送システムとして数年前から開発が行われ，現在では広く利用されている．

1.2 本研究の目的

現在DVTSは産学協同のDVTSコンソーシアム[2]において開発が行われている．DVTSは実用化及び普及の段階に入っており，AV(映像音声)専用機器などとしてハードウェア上での実装が行われるようになった．それらの専用機器は価格や筐体の大きさを抑制する必要もあり，汎用計算機に比べて少ない資源で動作させる必要があることが多い．本論文では，映像転送において重要となる映像の送受を行うためのバッファの量をDVTSで用いられる資源と定義する．

本研究では，資源の少ない専用機器等のDV転送端末とのDV転送の品質及び相互接続性の向上を目的とする．バッファの量が少ない端末との大容量の映像データ転送における必要事項を述べ，それに基づいた実装と評価を行う．

[1],[2]Keio University Shonan Fujisawa Campus
5322, Endo, Fujisawa, Kanagawa 252, Japan
E-Mail: irino@sfc.wide.ad.jp

2 DVTS

DVTSはDV機器からIEEE1394[3]インタフェースを用いてDVフォーマットのデータを読みだし，そのデータに対しIPパケット化を行い，転送を行うリアルタイム伝送を目的としたシステムである．DVTSはDVフォーマットを用いることでNTSC映像方式の場合で片方向約30Mbpsの帯域を利用して現行TV相当の720×480の解像度と29.97fpsのフレームレートの映像の送受が可能である．通信方式としてはUDPを用いる．その理由はDVフォーマットはフレーム内圧縮方式であり一部のデータ欠損が複数フレームに渡って連続的に影響をあたえることはないのに対して，輻輳制御による実効帯域幅の変動は連続的な映像の乱れになるためである．パケット到達順序を保証しないUDPでパケット到達順序を把握する方法としてRTP(RealtimeTransport Protocol)[4]を用いる．以上からDVTSが送受信するパケットを一般にDV/RTP[5]パケットと呼ぶ．DVTSの動作概要を図1に示す

2.1 ソフトウェア版DVTSと専用ハードウェアの差異

開発が行われているハードウェア実装されたDVTSの一つとしてDV機器に直接IPスタックを乗せDV/RTPパケットの送信または受信が可能な機器がある．

DV機器に直接IPスタックを乗せた専用機器は

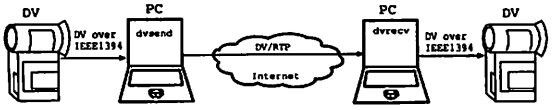

図 1: DVTS の動作概要図

バッファの量が制限される上に，DV/RTP パケットを受け，直接映像を描画するため，DV/RTP パケットのフローパターンが映像に直接影響する．単位時間あたりの消費量を下回る転送を行えばバッファアンダーランが起き，また上回る転送を行えばバッファオーバランが起こり、どちらの場合も映像に悪影響を与える．即ち，機器のバッファ量の制限が多いほどバースト的な通信に対応できない．バッファの量が制限されかつ固定される傾向にある専用機器に対してバッファアンダーラン及びバッファオーバランを起こさないためには一定量のデータを等間隔で送る平滑化されたデータ転送を行う必要がある．

3 既存の DVTS の実装と環境による IEEE1394 デバイスドライバの差異

3.1 IEEE1394 ネットワーク上を流れる DV データのパケット構造

DV データは DIF ブロックと呼ばれる 80 バイトのデータを基本単位として構成される．映像方式が NTSC の場合，フレームレート 29.97fps であり，その 1 枚のフレームは 1500 個の DIF ブロックで構成される．IEEE1394 ネットワーク上でも同様にこの DIF ブロックを単位に DV データが送信される．

IEEE1394 の転送方式は映像などの実時間性のデータ転送に適したアイソクロナス (同期) 転送とそれ以外のデータ転送に用いるアシンクロナス (非同期) 転送の 2 種類存在する．DV データの転送はアイソクロナス転送が用いられる．アイソクロナス転送は 1 秒間に 8000 回等間隔で行われる．NTSC 映像方式の DV が IEEE1394 アイソクロナス転送を用いて転送する 1 回あたりの平均転送データ量を式 3.1に示す．

$$\frac{FR \times FS}{C} = \frac{29.97 \times 120000}{8000} = 449.55$$

FR: DV Frame Rate
FS: DV Frame Size
C : IEEE1394 Isochronous Cycle

式 3.1:NTSC 映像方式 DV データの IEEE1394 アイソクロナス転送 1 回あたりの平均転送データ量

式 3.1より 1 回の IEEE1394 アイソクロナス転送で転送しなければいけないデータ量は 449.55Byte と算出できる．DV データのアイソクロナス転送は DIF ブロックを単位として行うため 1 つの IEEE1394 アイソクロナスパケットに内包される平均 DIF ブロック数は $449.55/80 = 5.62$ と算出される．実際には転送する DIF ブロックは整数値でなければならないため，IEEE1394 上での DV データ転送では DV データを内包する IEEE1394 アイソクロナスヘッダ，CIF ヘッダ，6 個の DIF ブロックで構成されるパケットとデータ量の調整のために，DV データを一切含まず，IEEE1394 アイソクロナスヘッダと CIF ヘッダのみで構成される IEEE1394 アイソクロナスパケットの 2 種類のパケットで構成されている．一般に後者のパケットを Empty CIP(Common Isochronous Packet) と呼ぶ．Empyt CIP の発生頻度は NTSC 映像方式の DV の場合

$$\frac{29.97 \times 120000}{80 \times 6 \times 8000 - 29.97 \times 120000} = 15.76$$

となりおよそ 15 回または 16 回に 1 回の割合で発生する．

3.2 環境による IEEE1394 デバイスドライバの差異

DV データを取り扱うことの出来る IEEE1394 のデバイスドライバは各 OS ごとに実装及びアクセス方法が異なるが大きく分けると 2 種類にわけられる．

- IEEE1394 アイソクロナス転送で受け渡されるデータをそのままユーザランドに渡す
- IEEE1394 アイソクロナス転送で受け渡されるデータをバッファリングし DV フレームを構成するデータ量単位でユーザランドに渡す

前者に含まれるのが FreeBSD, NetBSD の IEEE1394 デバイスドライバや Linux の raw1394 デバイスドライバである．一方後者に含まれるのが Linux の dv1394 デバイスドライバである．Linux の 2 種類のデバイスドライバはそれアクセス方法が異なる．raw1394 が raw1394_hadle という独自のデータ構造を前提にして IEEE1394 機器のデータの読み書きをするのに対して，dv1394 はファイルディスクリプタを介して DV 機器のデータの読み書きをする．一方 FreeBSD や NetBSD のデバイスドライバもファイルディスクリプタを介して DV 機器とのデータの読み書きを行う

現在の DVTS はプラットフォームの OS として，FreeBSD, NetBSD, Linux に対応している．2 種類の実装方法が存在する Linux では dv1394 デバイスドライバを利用している．この理由として複数の OS の対応をするにあたり，DV 機器のアクセスにファイルディスクリプタを利用している他の OS との親和性を考え，raw1394 ではなく dv1394 を採用している．

3.3 dv1394 デバイスドライバを用いた DV 送信側の実装における問題点

3.3.1 dvsend の DIF ブロック処理ルーチン

DVTS の送信側プログラムである dvsend では以下の処理ルーチンを繰り返す．以下，本論文ではこの処理ルーチンを dvsend の DIF ブロック処理ルーチンと呼ぶ

1. *read()* または *recv()* システムコールを利用したデバイスドライバからの DV データの読み込み
2. DIF ブロックの正当性の検査とフレーム終端の検知
   DIF ヘッダの SCT フィールド及び DBN フィールドの値の検査を行う．

3. *memcpy()* による送信 IP パケット用のバッファへのコピー
4. *sendto()* システムコールによる IP パケット送信
利用者が指定した数の DIF ブロック数がバッファへコピーされた場合か，またはフレーム終端に当たるデータがバッファへコピーされた場合に IP パケットの送信が行われる．

3.3.2 dv1394 デバイスドライバを用いた DV 送信側の実装における問題点

IEEE1394 アイソクロナス転送で受け渡されるデータをそのままユーザランドに渡すデバイスドライバを利用した場合は *read()* または *recv()* システムコールを呼ぶことで処理がブロックされるため，125μ 秒毎に発生する IEEE1394 アイソクロナス転送のタイミングにほぼ即したタイミングで処理が行われる．dvsend では Empty CIP をのぞいて DIF ブロック処理が行われるため DV/RTP パケット送信間隔に若干の揺らぎがあるが，理論上ほぼ等間隔の送信が行われる．

一方 dv1394 はフレーム単位でデータをユーザーランドに転送するため，*read()* システムコールは 1 フレームあたりに 1 度しか呼ばれない．またそれ以外の処理においては処理をブロックするシステムコールや関数を利用していないため，結果として IEEE1394 アイソクロナス転送のタイミングとはかけ離れて DIF ブロック処理ルーチンが実行される．計算機の処理能力によってこれらの DIF ブロック処理に要する時間は変わるが，多くの場合 1 フレーム分の送信処理が先に終り，次の 1 フレーム分の DV データの読み込まで長い時間処理がブロックされるという現象が起こる．ある短時間に送信処理が連続して行われ一定時間休止するため現在の dv1394 を利用した dvsend の実装はバースト的な通信を発生させる．特に計算機の処理能力が高いほど送信処理が早く終わるために，よりバースト的な通信を発生させる．

4 dvsend のレートコントロール機構の設計

第 3.3.2項において現在の DVTS 送信側プログラムには設計上の理由から dv1394 を用いて実装されているが，その実装は理論上バースト的な通信を発生させることを述べた．一方第 2.1節で現在開発が進んでいるハードウェア実装された DVTS 専用機器では，バッファの量が少なく，かつ IP パケットのフローパターンが直接映像に影響を与えるためそれらの機器を受信機として用いる場合は平滑化されたデータ転送を行う必要があることを述べた．従って dv1394 を用いた現在の DVTS 送信側プログラムの実装はハードウェア実装の DVTS 専用機器を受信機とした場合に，受信側が正しく映像を描画できなくなる．つまりこの二者間での映像転送は品質が悪く，相互接続性が低いと言える．本研究では現在の設計を大幅に変えず，dv1394 を利用した上で，DV/RTP パケットの送信間隔を一定にさせるためのレートコントロール機構を設計及び実装する．

4.1 送信間隔の算出と休止による等間隔のパケット送信手法

dvsend の DV/RTP パケット送信間隔を一定にするためには，1 フレーム分のデータを送るためにかかる時間を 1 フレーム分のデータを送信するために必要な IP パケット送信回数で割ることで算出できる送信間隔に基づいて DV/RTP パケットの送信処理を行うことで実現できる．DV/RTP パケット送信間隔を算出する式を式 4.1に示す．

$$(1 \div FR) \div ((FS \div DS) \div N) = \frac{80 \times N}{FR \times FS}$$

FR: DV Frame Rate
FS: DV Frame Size
DS: DV DIF Size (80Byte)
N : DIF Packet Number in a IP Packet

式 4.1:DV/RTP パケット送信間隔 (単位:秒) の理論値の算出式

式 4.1により求められた値を，以下，本論文では送信間隔理論値と呼ぶ．IP パケット送信後，送信間隔理論値から，DIF ブロック処理にかかった時間を差し引いた時間の休止を行うことで，理論上一定間隔の送信が可能となる．

4.2 Linux の時間精度に応じた実現方法の検討

MTU を Ethernet の 1500Byte とした場合，1 つの IP パケットに内包することのできる DIF ブロックの数は最大 18 となる．この条件で NTSC 映像方式の DV の送信間隔理論値は式 4.1より

$$\frac{80 \times 18}{29.97 \times 120000} = 40 \times 10^{-5}$$

と求められ，約 400μ 秒となる．一方 i386 アーキテクチャにおいて Linux のタイマ割り込み周期は 2.4 系カーネルで 10m 秒，2.6 系カーネルで 1m 秒であるため，先に求めた送信間隔理論値の精度で休止することが難しい．該当するプロセスの優先度を上げ *nanosleep()* を用いると，*nanosleep()* は 2m 秒の精度以下にはビジーループを用いるため，期待する時間精度を実現できる可能性が高い．しかし優先度をあげるためには管理者権限が必要となる上に期待した精度が確実に得られるとは限らない．全ての DV/RTP パケットが等間隔で送信されるためには，送信処理をするごとに休止を行うことが理想的であるが，上述のとおり期待した精度で休止を行えない可能性がある．そのため本機構では送信処理を一定回数まとまって行った後に，休止を行う．このために *nanosleep()* を呼び出した時に実際に CPU の割当てがプロセスに戻ってくるまでの時間を事前に計測し，その実測値を送信間隔理論値で割ることで *nanosleep()* を呼ぶ割合を決定する．

5 dvsendのレートコントロール機構の実装

5.1 実装環境

実装に用いたソフトウェア環境を表1に示す.

表 1: 実装に用いたソフトウェア環境

OS	Linux Kernel 2.4.26 (Debian Gnu/Linux)
プログラミング言語	C言語 (gcc 3.34)

5.2 優先度の向上

nanosleep() で 2ms 以下の精度を得るためには優先度を上げる必要があるため, *sched_setscheduler()* を利用して優先度をあげる. この処理方法を図 2に示す.

```
pid = getpid();
sched_getparam(pid, &sp))
sp.sched_priority+=1;
sched_setscheduler(pid, SCHED_RR, &sp);
```

図 2: 優先度の向上

5.3 nanosleep() 呼び出しによる実際のプロセス休止時間の計測

nanosleep() を呼び出してから CPU 割り当てが戻るまでの実際にかかる時間を計測するための処理方法を図 3に示す.

```
tsreq.tv_nsec=dvsend_param->dvrtpInterval;
for(i=0; i < TIMEAVE_MEASURE_COUNTER; i++){
  gettimeofday(&tmp_prev, NULL);
  nanosleep(&tsreq, NULL);
  gettimeofday(&tmp_cur, NULL);
  timersub(&tmp_cur, &tmp_prev, &tmp_sub);
  subTotal += tmp_sub.tv_usec;
}
dvsend_param->doSleepCount =
(subTotal /TIMEAVE_MEASURE_COUNTER * 1000) /
dvsend_param->dvrtpInterval;
if( ((subTotal / TIMEAVE_MEASURE_COUNTER *
1000) % dvsend_param->dvrtpInterval) != 0){
  dvsend_param->doSleepCount+=1;
}
```

図 3: 実際のプロセス休止時間の計測処理

nanosleep() を呼ぶ直前と直後に *gettimeofday()* を呼び出しその差分により, *nanosleep()* を呼び出してから CPU 割り当てが戻ってくるまでの実際にかかる時間を算出する. この時間を以下, 本論文では実休止時間と呼ぶ. さらにこの処理ルーチンを一定回数行い実休止時間の平均値を算出する. 実休止時間の平均値を送信間隔理論値でわり, 整数値に切り上げる. この算出された整数値の回数だけ DV/RTP パケットの送信処理を行った後に, プロセスを休止させる. 以下, 本論文ではこの算出された整数値を休止待機回数と呼ぶ. 本実装では休止待機回数を複数の関数で参照するために, dvsend 特有の情報を持つ構造体 dvsend_param に格納するために新たに構造体 dvsend_param 内に整数型の変数を宣言した.

5.4 プロセス休止処理

プロセスの休止処理には *nanosleep()* を用いる. 休止時間をできる限り正確にするために以下の 2 項目に関して時間を計測する.

- 実休止時間
- DIF ブロック処理ルーチンを休止待機回数実行したときにかかった実行時間

```
if(tsreq.tv_nsec > 0){
  gettimeofday(&exec_prev, NULL);
  gettimeofday(&sleep_prev, NULL);
  nanosleep(&tsreq, NULL);
  gettimeofday(&sleep_cur, NULL);
  timersub(&exec_prev, &exec_cur, &exec_sub)
  gettimeofday(&exec_cur, NULL);
  timersub(&sleep_cur, &sleep_prev,
           &sleep_sub);
}
```

図 4: プロセス休止処理と実休止時間及び実行時間の計測処理

nanosleep() の引数に設定する休止時間は以下のように算出する

送信間隔理論値×休止待機回数−実休止時間−実行時間

他のプロセスの動作状況によっては実行時間が極端に大きくなる可能性がある. そのため休止時間が計算上, 負の値になる可能性がある. そのため, 図 4 ではこの一連の処理を休止時間が正の値のときのみ行うようにしている. 一方負の値の場合は休止処理と計測処理は行わないが, 処理がされた物と見なして実休止時間の情報を保持している変数から 送信間隔理論値 × 休止待機回数 の値を差し引く. この処理を図 5に示す.

```
else{
  sleep_sub.tv_usec -=
    dvsend_param.dvrtpInterval * dvsend_p
    aram.doSleepCount /1000;
}
```

図 5: 休止時間が負の値になった場合の処理

6 評価

本章では, 実装したレートコントロール機構の有効性を評価する. まず LAN 環境で本機構の性能を定量的に計測し、評価する. さらに実際のインターネット環境での動作を評価する.

6.1 LAN 環境での動作

3 種類の実装を以下の計 4 種類の条件で実行し計測を行った.

1. Kernel 2.4 系上で管理者権限を用いて dv1394 とレートコントロールを使った本実装を実行した場合
2. タイマ割り込み周期が Kernel 2.4 系に比べて短い Kernel 2.6 系上で一般ユーザ権限で dv1394 とレートコントロールを使った本実装を実行した場合
3. dv1394 を利用し，レートコントロールを行なわない dvts1.0e 実装を実行した場合
4. raw1394 を利用する dvts0.9a 実装を実行した場合

Kernel 2.6 系上での管理者権限を用いた本実装の実行，及び Kernel 2.4 系上での一般ユーザ権限を用いた本実装の実行はそれぞれ上述 1 及び上述 3 とほぼ条件が変わらないため評価を行っていない．

6.1.1 評価手法

dvsend を実行する計算機と DV/RTP パケットを受信する計算機を 100Mbps スイッチハブを介して接続し，受信側で tcpdump[6] を用いてパケットキャプチャを行ないその情報を解析した．送信側でパケットキャプチャ行なわなかった理由は本機構がプロセス優先度をあげて実行するため，tcpdump のパケットキャプチャの性能に影響を与えてしまい，他の 2 実装と同一条件において，正確なパケットキャプチャが行えないためである．非常に単純なトポロジのため，ネットワーク遅延はほぼないと考えられ，受信側におけるパケット到達間隔がほぼそのまま送信側の送信間隔であると考えられることから，このような評価手法をとった

6.1.2 評価環境

表 2に評価に用いたソフトウェア環境，表 3に評価に用いたハードウェア環境を示す．

表 2: 評価に用いたソフトウェア環境

OS	Linux Kernel 2.4.26 Linux Kernel 2.6.6
使用ライブラリ	libraw1394 0.10.1

表 3: 評価に用いたハードウェア環境

CPU	Intel(R) Pentium(R) III 1266MHz
メモリ	512MB

6.1.3 送信間隔の評価

Kernel 2.4 上で管理者権限を用いて dv1394 とレートコントロールを用いる本実装のパケットフローを図 6，Kernel 2.6 上で一般ユーザ権限を用いて dv1394 とレートコントロールを用いる本実装のパケットフローを図 7，dv1394 を用いるがレートコントロールは一切行わない dvts1.0e のパケットフローを図 8，raw1394 を用いる dvts0.9a22 のパケットフローを図 9に示す．図 6，図 7，図 8，図 9はそれぞれパケットキャプチャをはじめて最初の 0.4 秒間のパケットフローを図示しており，縦軸が受信バイト数，横軸が到達時間 (単位:秒) を示している．従って線の集中している部分はパケットが連続して到達したことを示す．

各実装のパケット到達間隔の情報としてパケットキャプチャした約 25000 パケットの最小値，最大値，

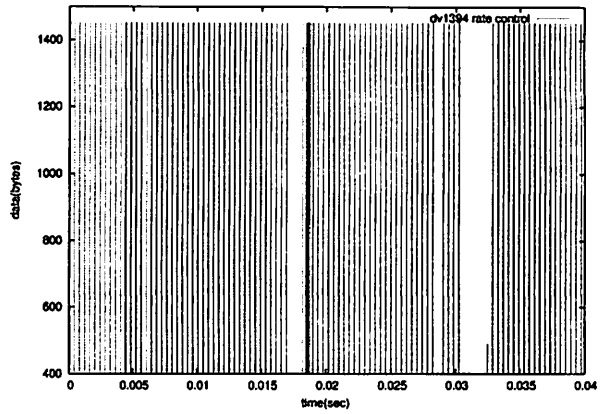

図 6: Kernel 2.4 上で管理者権限を用いて dv1394 とレートコントロールを行った場合 (本実装) の DV/RTP パケット到達間隔

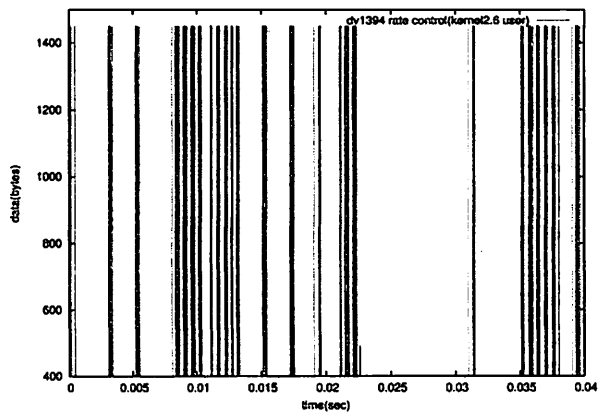

図 7: kernel 2.6 上で一般ユーザ権限を用いて dv1394 とレートコントロールを行った場合 (本実装) の DV/RTP パケット到達間隔

平均及び標準偏差を表 4に示す．平均及び標準偏差は小数点第 7 位で四捨五入した．

表 4: 各実装のパケット到達間隔

実装	最小値	最大値	平均	標準偏差
dv1394+レートコントロール (Linux 2.4, 管理者権限)	0.000039	0.011838	0.000397	0.000310
dv1394+レートコントロール (Linux 2.6, 一般ユーザ権限)	0.000037	0.036211	0.000397	0.001202
dv1394	0.000037	0.052942	0.000396	0.002662
raw1394	0.000037	0.001132	0.000397	0.000373

6.1.4 LAN 環境における評価に対する考察

図 8は第 3.3.2で述べた dv1394 デバイスドライバの問題を示しており，本評価に用いた計算機では 1 フレーム分の映像の送信処理が約 0.01 秒で終了しその後の約 0.23 秒は全く送信が行われない結果となっている．図 9では raw1394 でバイスドライバを用いてほぼ等間隔でパケットが到達していることがわかる．

図 6に示した本実装は一部分実休止時間が大きくなり等間隔になっていない部分があるが，それ以外は他の実装より細かい粒度で等間隔に平滑化された

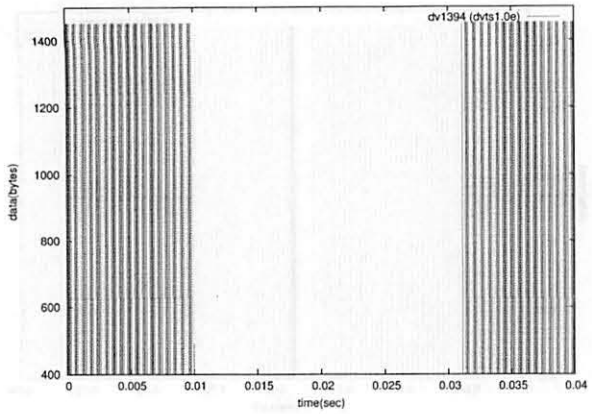

図 8: dv1394 を利用しレートコントロールを行なわなかった場合 (dvts 1.0e 実装) の DV/RTP パケット到達間隔

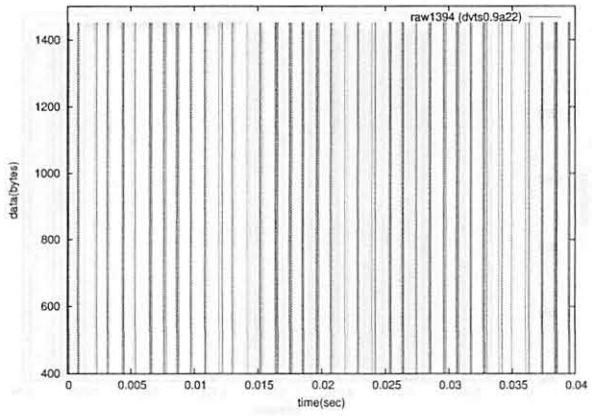

図 9: raw1394 を用いる dvts0.9a 実装の DV/RTP パケット到達間隔

パケットが到着している．等間隔の DV/RTP パケットを行う送信として機能しており，dv1394 デバイスドライバの問題を解決している．

また表 4より本実装が最も標準偏差が小さく，送信間隔にばらつきがないことがわかる．また標準偏差が raw1394 を利用した場合よりも小さいことから，Linux における，等間隔の DV/RTP パケット送信手法として最も効果がある方法であると言える．タイマ割り込み周期が Kernel 2.4 系に比べて短い Kernel 2.6 系では，優先度を上げなくてもある程度の効果がみられることがわかる．ただし優先度を上げていないため，他のプロセスの動作状態により影響されやすいと考えられる．

6.2 インターネット環境での動作

WIDE プロジェクト [7] のネットワークに属するホストから 8 ホップはなれた商用プロバイダの提供する FTTH サービスのネットワークに属するホストへ約 10 秒間 DV/RTP の転送を行った．計測環境としてこの 2 ホスト間の ping による 1000 パケットの RTT 計測と netperf による 10 秒間のスループットを表 5に示す．到着間隔を図 10に、その統計結果を表 6に示す．

表 5: 実験に用いた 2 ホスト間の性能

RTT 最小	RTT 平均	RTT 最大	RTT 標準偏差	スループット
7.08ms	8.37ms	17.205ms	0.649 ms	78.68Mbps

表 6: 各実装のパケット到達間隔

最小値	最大値	平均	標準偏差
0.000009	0.034178	0.000396	0.001817

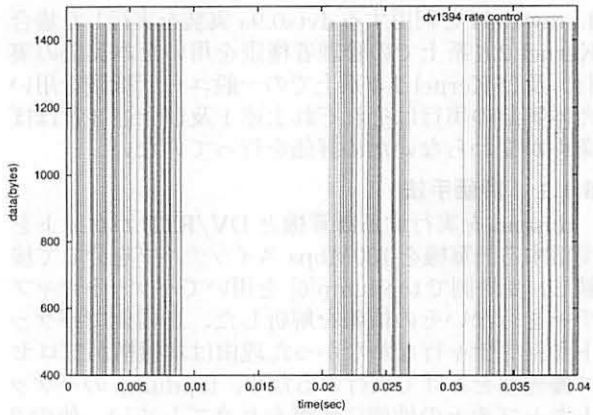

図 10: インターネット上で本実装の DV/RTP パケット到達間隔

レートコントロールしない図 8と比較して，若干の改善が見られインターネット環境においても効果があることがわかる．

7 まとめ

本研究では現在の Linux における dv1394 デバイスドライバを用いた DVTS 送信側プログラムの実装が，バースト的な通信を発生させることを述べ，この問題が現在開発されているハードウェア実装の DVTS 専用機器との通信において問題になることを述べた．この問題の解決方法としてプロセスの休止によるパケットフローの平滑化機構の設計，実装及び評価を行った．等間隔で送信するために必要なパケット送信間隔を求め，さらに処理にかかる時間や実際に休止を行う時間を求めることで本機構は OS の時間精度に応じて休止時間を調整することを可能とした．この機構はプロセスの優先度を上げた状態で動作させると，raw1394 デバイスドライバを用いた実装よりも細かい粒度で IP パケットの送信制御ができることが評価によりわかり，現在の Linux における DV/RTP パケット送信のフローパターンの平滑化において最も効果がある方法と言えた．本研究により資源の少ない専用機器等の DV 転送端末との DV 転送の品質及び相互接続性を向上させ，転送される DV 映像の品質向上を可能とした．

参考文献

[1] Akmichi Ogawa. DVTS(Digital Video Transport Sysytem). *http://www.sfc.wide.ad.jp/DVTS/*.
[2] http://www.dvts.jp/.
[3] Institute of Electrical and Electronics Engineeers, Inc. IEEE Std 1394-1995 High Performance Serial Bus. Aug 1996.
[4] Audio-Video Transport Working Group. RFC1889 RTP: A Transport Protocol for Real-Time Applications. *http://www.ietf.org/rfc/rfc1889.txt*, pages 1–75, January 1996.
[5] Akimichi Ogawa, Katsushi Kobayashi, Kazunori Sugiura, Osamu Nakamura, Jun Murai. Design and implementation of dv based video over rtp. http://www.sfc.wide.ad.jp/DVTS/pv2000/index.html.
[6] http://www.tcpdump.org.
[7] http://www.wide.ad.jp/.