コマンドパイプラインによるマルチメディアストリーム処理

笠松健一 † 古村隆明 * 藤川賢治 † 岡部寿男 † 池田克夫 *
* 京都高度技術研究所 † 京都大学

音声・動画像の送受信システムに必要な機能を，単一の機能ごとに一つのプロセスとして実装し，コマンドパイプラインによって組み合わせて利用するシステムを提案し実装した．
モジュールの組み合わせによって，データの蓄積，マルチキャスト送受信，前方誤り訂正符号の追加などに柔軟に対応できるシステムが実現できた．

1 はじめに

音声や映像のストリーム伝送をインターネットで行うシステムとして，ビデオオンデマンド，実時間放送，インターネット電話等が現在利用されている．

本論文では，映像の取り込み機能や圧縮機能，ネットワーク伝送機能のような機能毎に単体で実行可能なプログラムを実装し，必要な機能を持ったプログラム同士を連携させることによってマルチメディアストリーム処理を行うシステムを提案する．提案するシステムでは，このシステムとは無関係な既存のプログラムや将来実装されるプログラムとも連携して新しい機能をを利用できる．

本論文ではまず既存のアプリケーションの問題点を明らかにし，提案システムの概要を述べる．そして，提案システムがどのような機能を実現するのかを述べた後，各機能を備えるプログラムの設計を行う．その後，提案するシステムの実装と実装したシステムが実現したマルチメディアストリーム処理の一部分を述べる．

2 関連研究と提案するシステム

本章ではインターネットを用いる既存のマルチメディアアプリケーションの問題点について考察する．その後，提案するシステムの概要を述べる．

2.1 既存のアプリケーションの問題点

2.1.1 マイクロソフト社製のマルチメディアアプリケーション

マイクロソフト社製の OS(Windows) 上で動作するマルチメディアアプリケーションは数多く存在する．Windows にはコーデックのための公開された共通のインターフェースが存在し，このインターフェースを用いるアプリケーションは，新たなコーデックが追加された時，そのアプリケーション自体を変更しなくても利用できる．

しかし，ネットワーク伝送などについてはこのようなインターフェースは存在せず，新たなネットワーク伝送方式を利用するアプリケーションは，そのプログラムかそのプログラムが利用するライブラリを変更する必要があり，新たなネットワーク伝送方式の追加は困難である．

次に，マイクロソフト社製の MediaPlayer，MediaEncoder，NetMeeting という三つのアプリケーションについて考える．各アプリケーションはマルチメディアストリーム処理を行う幾つかの機能を備えている．しかし，これらのアプリケーションを複数同時に利用したとしても，各アプリケーションが持つ機能を組み合わせた処理を行うことは不可能である．例えば，MediaEncoder は音声ストリームを放送する機能を，NetMeeting は電話機能をそれぞれ持っているが，これら二つのアプリケーションを利用しても図 1 のようなシステムを実現することは不可能である．

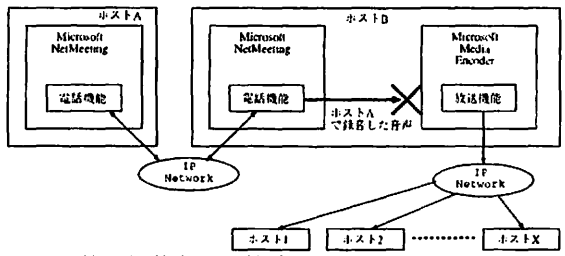

図 1: 放送機能と電話機能を組み合わせたシステム

2.1.2 GStreamer

GStreamer[1] はメディアストリーム処理を行うモジュール (プラグイン) をグラフ構造で繋ぎ，組み合わせて利用することで様々なマルチメディア処理を行う枠組を提供する．そして，コーデックやネットワーク伝送処理など，機能毎に細かく分けられたモジュールも合わせて提供されている．GStreamer に含まれるモジュールのインターフェースは，コーデックだけではなく各機能毎に定められている．

GStreamer [1] と合わせて提供されているモジュールに，カメラなどのから映像を取り込むモジュールと，マイクなどのから音声を取り込むモジュールが存在する．これらのモジュールが生成するメディアストリームには各サンプルが取り込まれた時刻の情報が含まれおらず，ストリームの取り込まれた時刻の情報を，これらのモジュールに他のモジュールが問い合わせる方法も存在しない．そのため，これらのモジュールが生成するストリームの同期をとることは不可能であり，例えば，映像とその映像に同期する音声を取り込み，他のホストへ伝送し，それらを同期再生することは不可能である．

[1] GStreamer バージョン 0.3.1

2.2 提案するシステムの概要

本論文では，映像の取り込み機能や圧縮機能，ネットワーク伝送機能のような機能毎に単体で実行可能なプログラムを実装し，目的の処理に必要な機能を持ったプログラム同士を連携させることによってマルチメディアストリーム処理を行うシステムを提案する．このようなシステムにおいて，プログラムが連携するためには各プログラムのプロセス間での通信が不可欠である．プロセス間通信に用いられる代表的な方法として，

- パイプ，FIFO(名前つきパイプ)
- ソケット
- SystemV の IPC

がある．本研究では，データの伝送効率が悪いが，多くのプログラムが利用しているパイプを用いることとした．これは，処理効率の高いシステムを作ることよりも，既存のプログラムと連携を容易に行ない，提案するシステムのプログラムだけでは実現不可能な処理を実現することを重要視しているからである．また，パイプとソケットあるいはパイプと SystemV の IPC 間で相互に通信方法を変換するプログラムを実装すれば，パイプによる連携が不可能なプログラムとも連携できる．

次にプログラム間のインターフェースについて考える．図 2 のように機能毎にその機能を実現するプログラムを交換しても，他の機能のプログラムを変更することなく利用できるようにする．詳しくは 4 章で述べる．

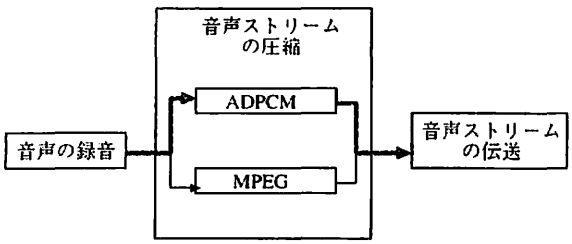

図 2: 機能を実現するプログラムの交換

3 利用するシステムの想定

提案するシステムを実装する前にどのような機能を実装するのか決定する必要がある．そこで，実在するビデオオンデマンド，実時間放送，電話の三つのシステムを，提案システムで実現させることを考える．本章ではこれらのシステムを定義し，これらのシステムが要求する機能を定める．

3.1 ビデオオンデマンド

ビデオオンデマンドシステムは，映画などの過去に記録した映像を利用者が見たい時に見ることができるシステムとする．このシステムでは，過去の映像を伝送するので実時間で伝送することは要求されない．また，同じ映像を同時に多くの利用者が見ることはあまり無いと考え，ユニキャストで伝送することとする．よって，ビデオオンデマンドシステムが要求する機能は以下のとおりである．

- 映像の取り込みと圧縮
- 映像ストリームの保存
- IP ユニキャストでデータの損失が生じない伝送
- 映像ストリームの伸長と再生

3.2 実時間放送

実時間放送システムは，カメラなどを用いて取り込む映像と，マイクなどを用いて取り込む音声を，世界中に散らばる数多くの視聴者が同時に見ることができるシステムとする．このシステムでは，取り込まれた映像と音声を短い遅延時間で再生するようにするため，ネットワーク伝送による遅延が小さくなるようにする．また，同時に同じデータを多くのホストへ伝送する為，マルチキャストを利用することとする．パケットロスに対処するために前方誤り訂正符号を用いる．よって，要求される機能は以下のとおりである．

- 映像の取り込みと圧縮
- 音声の取り込みと圧縮
- 映像ストリームと音声ストリームの伸長と同期再生
- 前方誤り訂正
- IP でマルチキャストを用いて実時間伝送

3.3 電話

電話システムは，利用者が他の利用者に発信し，着信した利用者は通話中でなければ発信した利用者と通話をするシステムとする．このシステムでは，ホスト間で双方向の音声ストリームを低遅延で伝送する必要がある．

通話の開始後は発着信双方のホストが，マイクなどから取り込んだ音声を相手側へ IP で実時間で送信すると同時に，受信した音声ストリームを再生する．よって，電話システムが要求する機能は以下のとおりである．

- 電話の着信
- 電話の発信
- 双方向に IP で実時間伝送
- 音声の取り込みと圧縮
- 音声ストリームの伸長と再生

4 EMON システムの提案と設計

3 章で提案するシステムが要求する機能を定めた．ここで，提案するシステムが前方誤り訂正機能とマルチキャスト伝送機能を備えることから提案するシステムを EMON(Error-correcting Multicast on Open Nodes) システムと名付けた．以降，本論文が提案するシステムを EMON と呼ぶこととする．本章ではまず，想定するシステムが要求する機能を含む範囲で，プログラムが備える機能の分割をおこなう．

音声の取り込み処理と圧縮処理は，一つのプログラムとして実装する方法と，別々のプログラムとして実装する方法が考えられる．複数音声の合成処理を行うことや，新たな圧縮方式への対応を考えると，分けて実装する方が望ましい．また，画像の取り込みと圧縮についても同様である．

次に，映像と音声などの複数ストリーム間の同期再生について考える．これには，同期したい映像と音声などのストリームを単一プログラムで再生する方法と，それぞれの再生プログラムに同期機能を追加する方法が考えられる．実装は単一プログラムとしてしまう方が容易である．しかし，それぞれのストリームごとに同期機能を持つプログラムを実装すれば，複数の映像再生プログラム間での同期を取ったり，複数ストリームを別々のホストで同期しながら再生することも可能となり応用範囲が広がる．以上の考察より，各再生プログラムに同期機能を追加することにする．

以上より，以下の機能を備えるプログラムをそれぞれ設計する．

- 映像の取り込み
- 音声の取り込み
- 映像ストリームの圧縮
- 映像ストリームの伸長
- 音声ストリームの圧縮
- 音声ストリームの伸長
- 映像ストリーム再生機能 (同期機能付き)
- 音声ストリーム再生機能 (同期機能付き)
- ストリームの保存
- 前方誤り訂正
- IP ユニキャストで損失が生じない伝送
- IP マルチキャストで実時間伝送
- 双方向に IP で実時間伝送
- 電話の着信
- 電話の発信

以降本章では，各機能を備えるプログラムを設計し，プログラム間でメディアデータ以外に通信が必要な情報を明らかにする．

4.1 映像，音声の取り込み

映像の取り込みプログラムでは，カメラから映像の取り込みだけを行い，圧縮をせずに映像ストリームを生成する．音声も同様にする．そこで，映像に対して取り込み処理を行うプログラムが備えるべき詳細な機能を考える．

- 取り込む装置の選択
- 生成する映像ストリームの品質の設定
- 各フレームに取り込んだ時刻の情報を付加

映像と音声の同期再生を行う為には，同じ時刻に取り込まれた映像データと音声データの対応を再生を行うプログラムに伝える必要がある．これには，時刻情報を同じ時計を基に付加することが必要である．異なるホストのプログラム間では必要な精度に応じて NTP[2] の技術を利用すればよい．

4.2 他のストリームと同期をとることが可能なメディアストリーム再生

他のストリームと同期をとりながら映像ストリームを再生する機能は，圧縮されていない映像ストリームを再生するプログラムで実現する．音声の再生も同様にする．ここでは，映像の各フレームを取り込んだ間隔で正しく表示するためのバッファ管理 [5] について考える．

- 他のストリームと同期をとらない場合

映像の各フレームを取り込んだ間隔と同じ間隔で表示する為には，ストリームに付加された時刻情報は必須の情報である．ただし，ネットワーク伝送などによって映像ストリームにジッターが生じる場合がある為，プログラムがストリームを入力してから表示を行うまでの時間を小さくする為にはジッターを基にしたバッファ管理を行う必要がある．

- 他のストリームと同期をとる場合

他のストリームと同期をとって再生する為には同じ時刻に取り込まれたフレームを同じ時刻に再生する必要がある．そのためにはプログラム間の時計の共有が必要であり，異なるホストのプログラム間では必要な精度に応じて NTP を利用する．

複数のストリームのうち，バッファ内での遅延などによって再生までに最も時間がかかるストリームを決定し，それに合わせて他のストリームの再生を遅らせることで，全体として最小遅延で同期再生を行うことができる．

4.3 ネットワーク伝送

映像や音声の様々な形式のストリームを IP ネットワークで実時間伝送を行うプロトコルとして RTP[4] が存在する．このプロトコルは UDP を用いることを想定したプロトコルなのでマルチキャスト伝送に利用することができる．ただし，RTP は会議システムの為の機能なども含まれているが，実時間伝送を行うだけならば必要ない機能であるため，実時間伝送機能のみを提供するプロトコルが標準化されることを期待する．そこで，実時間マルチキャスト伝送機能は，RTP に依存しないことが望ましいため，マルチキャスト伝送機能と RTP 形式に変換する機能に分けて実現することとする．また，データの損失が生じないネットワーク伝送機能はユニキャストであれば TCP を用いると容易に実現できるので TCP を用いる．

4.4 前方誤り訂正 (FEC)

前方誤り訂正機能は，データブロックから冗長なデータを生成し，データブロックと冗長なデータの両方をネッ

トワーク伝送し，データブロックのビット誤りや欠損を冗長なデータを用いて復元する機能である．EMON システムでは TCP か UDP を用いてネットワーク伝送を行うが，TCP を利用する時は前方誤り訂正の機能は利用しない．

一方，UDP ではパケットロスが発生する可能性があるため，前方誤り訂正符号を用いてデータ損失を回復する．前方誤り訂正符号としてリードソロモン符合を用いる．この符号をパケット通信に適用すると，同じ長さの K 個のパケットに対し，リードソロモン符号により R 個の冗長なパケットを生成したとき，K+R 個のパケットのうち任意の K 個のパケットから全てのパケットを復元できる．K と R は任意の値[2] を用いる事ができる．

また，誤り訂正を行なうためには各パケットがデータパケットあるいは冗長なパケットのどの部分のデータを含んでいるかという情報が必要である．

電話の音声伝送処理などで，取り込みから再生までの遅延を小さくすることに重点をおく場合には，データブロックが小さくなり 1 パケットに収まる．このような場合は，前方誤り訂正を利用するよりも，同一のパケットを複数送信する方がデータの伝送効率が良いため，実装するプログラムにはこの機能も盛り込む．

4.5 メディアストリームの保存

メディアストリームの保存はパイプを流れるデータをファイルに保存すれば実現できる．また，保存したストリームを読み込む時は保存したファイルをパイプを用いてプログラムに入力すればよい．ただし，ストリームに絶対時刻などの値が付加されている場合，時間の経過によって正しい処理を行うことができなくなるならば，適切な値に書き換える必要がある．

4.6 電話の着信と電話の発信

電話の着信機能は，電話の発信が行われるのを待ち，通話中でなければ通話を開始する処理を行うプログラムによって実現する．通話の開始時に音声ストリームを送信するプログラムと音声ストリームを受信するプログラムを起動すれば良い．

電話の発信機能は，電話の発信を行ない，通話中でなければ通話を開始する処理を行うプログラムによって実現する．また，通話は既に述べた方法で実現できる．

5 提案システムで実現したマルチメディアストリーム処理

本章では 4 章の設計を基に提案する EMON システムの実装について述べる．そして，実装したプログラムと既存のプログラムを用いることによって実現したマルチメディアストリーム処理の一部を述べる．

[2] パケットの長さが n バイト単位の時 $K+R<256^n$ を満たすことが必要

5.1 提案システムの実装

まず，実装したプログラムの一部[3] について，そのプログラムが持つ機能を表 1 に，各プログラムの接続関係を図 3 に示した．ただし，既存のプログラムや tcpconnect との接続関係は含まれていない．実装時に用いたプログラムの実行環境は FreeBSD4.3R Pentium II 400MHz である．ストリームの取り込みと圧縮の処理は一つのプログラムで実装した．これは，実行環境において無圧縮のメディアデータを実時間でパイプを通す処理は負荷が大きいと判断した為である．同じ理由により，伸長の処理と再生の処理も単一のプログラムで実装した．また，電話に関する機能は NOTASIP[3] 方式で実装した．

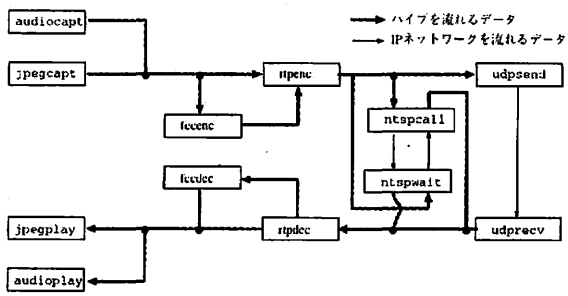

図 3: プログラムの接続関係

表 1: 主要なプログラムの機能

プログラム名	機能
audioplay	音声データを再生
audiocapt	マイクなどから音声を取り込む
jpegplay	映像データを再生
jpegcapt	カメラなどから映像を取り込む
fecdec	FEC(Forward Error Correction :前方誤り訂正) を行なう
fecenc	FEC 用の冗長なデータを追加
rtpdec	RTP ヘッダーを取り去る
rtpenc	RTP ヘッダーを追加
udprecv	マルチキャスト及びユニキャストの UDP パケットを受信
udpsend	マルチキャスト及びユニキャストの UDP パケットを送信
tcpconnect	TCP によるストリームの送受信
ntspcall	双方向 UDP セッションの確立と要求
ntspwait	双方向 UDP セッションの確立と待機

実装を容易にする為に全てのプログラム間で用いるメッセージ形式を一種類に限定した．メッセージが持つべき機能をまとめると以下のようになる．

- 前方誤り訂正を行うパケットの集合の区切り
- 映像や音声のフレームを取り込まれた間隔で再生する為の時刻
- パイプを用いるので，メッセージの区切り

そこで，メッセージの形式を図 4 のように定めた．

マーカービット (図 4 の M のフィールド) は前方誤り訂正を行うパケットの集合の区切りを示す為に用いる．そ

[3] デバッグなどに用いるプログラムも実装した

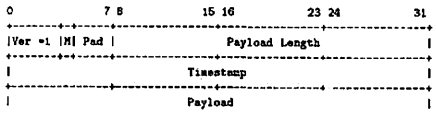

図 4: パイプのメッセージ形式

して，前方誤り訂正を行うパケット毎にこのメッセージを用いることとし，前方誤り訂正を行うパケットの集合毎に最後のパケットに相当するメッセージのみ 1 とする．タイムスタンプは，メッセージに含まれるフレームが取り込まれた時刻の情報で，同期するメディアストリームの間では同じ時刻に取り込まれたフレームに同じ値を設定する．そして，ペイロード長はメッセージの区切りを示す為に用いる．

5.2 想定したシステムの実現

EMON システムを実現する機能を決定する時に想定したシステムはビデオオンデマンドシステム，実時間放送システム，電話システムであった．まず，これらを EMON システムによって実現する方法を述べる．

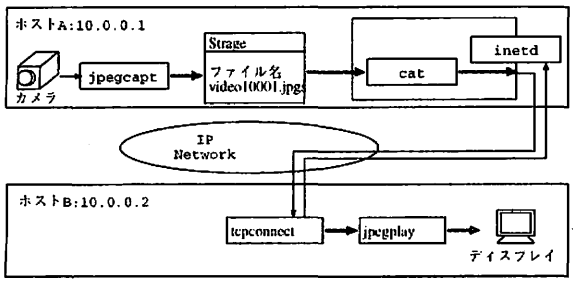

図 5: ビデオオンデマンドシステム

はじめにビデオオンデマンドシステムが要求する処理を行う方法を示す．システムが配信する映像の準備方法であるが，

```
% jpegcapt > video10002.jpgs
```

と実行すると，図5のようにしてカメラなどから jpegcapt が映像を取り込み，配信する映像を video10002.jpgs に保存する．

次に，映像の配信方法を述べる．配信には既存のアプリケーションとして inetd を用いる．inetd を用いると，TCP/IP による接続を待ち，他のホストが接続をしてきた時，ポート番号毎にあらかじめ指定したプログラムを実行し，そのプログラムの標準出力を接続したホストに送信することができる．これを利用して，ホスト A(10.0.0.1) のポート 10002 へ接続をしてきたホストに対し，

```
%cat video10002.jpgs
```

を実行した時の標準出力を送信するように設定し配信を行う．これを受信して再生する方法は，

```
% tcpconnect -A 10.0.0.1 -P 10002 | jpegplay
```

となる．以上のようにしてビデオオンデマンドシステムを実現した．

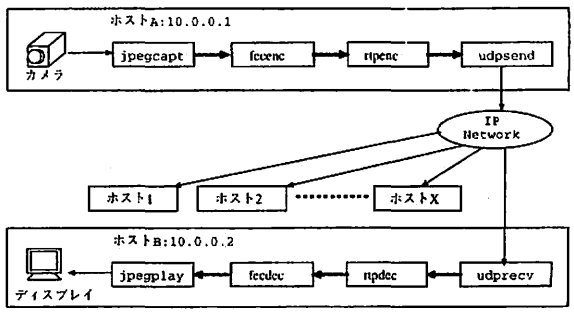

図 6: 実時間放送システム

次に，実時間放送システムが要求する処理を行う方法を示す．ただし，複数ストリームの同期再生機能は実装しなかったので，映像のみ放送する方法を示す．

```
% jpegcapt | fecenc | rtpenc | udpsend -A
225.0.0.1 -P 10002
```

とホスト A で実行すると，図 6 のようにしてカメラなどから jpegcapt が映像を取り込み，圧縮を行なった後，fecenc が各フレームに FEC の為のデータを付加する．rtpenc が RTP パケットへの分割と RTP ヘッダの付加を行い，udpsend がそれをマルチキャスト伝送する．

次にこのようにして放送している映像を受信する方法を述べる．

```
% udprecv -A 225.0.0.1 -P 10002 | rtpdec |
fecdec | jpegplay
```

とホスト B で実行すると，マルチキャストで伝送されている映像を udprecv が受信し，rtpdec が RTP ヘッダーを取り除く．そして，fecdec により前方誤り訂正を行い，jpegplay が映像を表示する．もし，FEC を利用しない場合は，送信側では “fecenc |” を除き，受信側では “fecdec |” を除いて実行する．このようにして実時間放送システムを実現した．

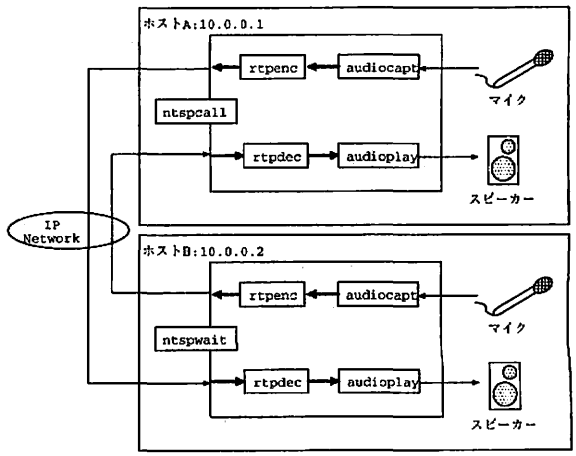

図 7: 電話システム

最後に，電話システムが要求する処理を行う方法を示

す．まず，発信を待ち受ける方法を述べる．

```
% ntspwait 10.0.0.1 10000 "rtpdec |
audioplay" "audiocapt|rtpenc"
```

とホストA(10.0.0.1)で実行すると，ntspwaitがUDP/IPポート10000への発信の待ち受けを開始する．着信があると通話を開始する．通話開始後は図7のようにして“audiocapt|rtpenc”でマイクなどから録音した音声をRTPを用いて相手側へ送信すると同時に，相手が送信する音声ストリームを受信し“rtpdec| audioplay”で再生する．

次に，発信を行う方法を述べる．

```
% ntspcall 10.0.0.1 10000 "rtpdec |
audioplay" "audiocapt|rtpenc"
```

とホストBで実行すると，ホストAへ直ちに発信を行い，通話を開始する．通話開始後の処理は着信側と同じである．このようにして電話システムを実現した．

5.3 想定したシステムを組み合わせたシステム

電話と実時間放送を組み合わせたシステムを実現した．このシステムは，ホストAがホストBへ電話をかけると，ホストAからの音声ストリームをホストBが放送(マルチキャスト)するシステムである．電話をかける方法は既に述べた．そこで，ホストBが行う処理を述べる．

```
% ntspwait -A 10.0.0.2 -P10000 "udpsend -A
225.0.0.1 -P 10002" "udprecv -A225.0.0.1 -P
10002"
```

とホストBで実行すると，ntspwaitにより発信の待ち受け処理を行い，受信したストリームを udpsend が225.0.0.1 ポート10002へマルチキャストで送信する．また，udprecvにより放送されているストリームを受信し，それをホストAへ送信する．

5.4 既存のプログラムとの連携

SSH を利用して，ホストAがマイクから取り込んだ音声ストリームをホストBへ暗号化して伝送し，再生することもできる．

```
% ssh 10.0.0.2 audiocapt | audioplay
```

また，ホストAで

```
% audiocapt | ssh 10.0.0.2 ‘‘audioplay |
audiocapt’’ | audioplay
```

と実行すると，ホストAとホストBの間で双方向音声ストリームの伝送が暗号化して行なわれ，電話のようなシステムとなったが，音声の遅延は大きかった．これは，SSHが実時間伝送を行うプログラムではない為である．

次に，teeとの組み合わせを紹介する．プログラムの出力をプログラムやファイルへ複数同時に出力する処理を行うteeというプログラムが存在する．これを利用すると，下記のように映像をファイルに記録しながらマルチキャスト送信することができる．

```
% jpegcapt | tee save.jpgs | rtpenc | udpsend
-A 225.0.0.1 -P 10002
```

5.5 IPネットワークを利用した無線マイク

無線LANを用いると，無線区間を含むIPネットワークを構築することができる．そして，無線LAN上のホストにマイクを備えると無線マイクとして利用することができる．そこで，小型パソコンを用いてIPネットワークによる無線マイクを実現した．そこで，無線マイクが取り込んだ音声を無線を用いて伝送する方法を述べる．

```
% audiocapt | rtpenc | udpsend -A10.0.0.1
-P10002
```

と実行すると，マイクで取り込んだ音声をRTPを用いて10.0.0.1のポート10002へ伝送する．

6 おわりに

本論文では，映像の取り込み機能や圧縮機能，ネットワーク伝送機能のような機能毎にその機能を持つ単体で実行可能なプログラムを実装し，システムの利用者が行う処理に必要な機能を持った機能を持ったプログラムを連携することによってマルチメディアストリーム処理を行うシステムを提案した．そして，このシステムの設計と実装を行い，コマンドパイプラインによって既存のプログラムと組み合わせることで，IPネットワークを利用した様々なマルチメディアストリーム処理を実現した．

今後の課題は，複数ストリームの同期再生の実装と検証である．また，メディアデータ自身に対する処理を行うプログラムを追加することでより多様な処理を行うシステムにすることである．

参考文献

[1] GStreamer: http://www.gstreamer.net.

[2] Mills, D. L.: Network Time Protocol (Version 3) Specification, Implementation, Request for Comments 1305, Internet Engineering Task Force (1992).

[3] NOTASIP: http://www.notasip.org.

[4] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications, Request for Comments 1889, Internet Engineering Task Force (1996).

[5] 古村隆明: インターネット放送に関する研究 – バッファ管理, 前方誤り訂正, 階層伝送–, 博士論文, 京都大学大学院情報学研究科 (2001).