特別研究報告書

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

指導教官　　岡部 寿男 助教授

京都大学工学部情報学科

笠松 健一

平成14年2月8日

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

内容梗概

近年、データ通信技術とマルチメディアデータ圧縮技術の進歩により、インターネットで文字や画像だけではなく、音声や映像の伝送も行なわれるようになった。インターネットに常時接続できる場所は、学校や会社だけではなく、家庭や街角にも広がった。したがって、IP ネットワークを用いて情報通信を行なうことができる機会は増加している。

インターネットで電話や実時間放送システムを実現するアプリケーションプログラムはすでに存在する。それらのプログラムは、映像の取り込みや圧縮、ネットワーク伝送などの機能を多数実装しているが、その大半は、他のプログラムと連携はできない。つまり、それらのプログラムに他のプログラムが持つ機能を追加することや、それらのプログラムが持つ機能の一部を他のプログラムが利用することができない。そのため、既存のアプリケーションシステムでは行なわない処理の実現の為には、C や JAVA のようなプログラミング言語の知識が要求される。これらの問題は、各機能毎にモジュールを実装し、そのモジュールを簡単に組み合わせて利用できるようにすれば解決する。

本論文が提案するシステムは、機能毎に実装した実行可能なプログラム同士を主に対話シェルのインターフェースを用いて連携させる。連携したプログラム群によって様々な処理を行なう。モジュールを実行可能プログラムとした理由は、既存のプログラムだけではなく将来実装されるプログラムを変更すること無くモジュールとして利用する為である。これは、ライブラリでは実現が困難である。プログラムのプロセス間の通信を行なう方法としてパイプとソケットと SystemV の IPC を比較する。比較の結果、既存のプログラムとの連携をコマンドパイプラインを用いて容易に実現するために、パイプを用いる。

このような構造をもつシステムを実装する前に、どのような機能をプログラムに実装するのかを考える。そこで、実際の利用を想定したシステムとしてビデオオンデマンド、実時間放送、電話の三つのシステムについて考察し、これらのシステムが要求する機能を実装することにする。これらシステムが持つ機能を備えたシステムを本論文で EMON システムと名付ける。そして、EMON システムが持つ機能毎にプログラムを設計する。音声とその音声に同期する映

像を再生する機能の設計では、それぞれ異なるホストで再生を行なうことができるようにした。前方誤り訂正の機能の設計では、前方誤り訂正の利用方法を述べた後、小さなデータのパケットを頻繁に伝送する処理では前方誤り訂正が有効ではないことを述べる。このような処理は、例えば、電話システムで音声ストリームを低遅延で伝送する際に行なわれる処理である。前方誤り訂正が有効ではない原因は、トランスポート層以下で必要なデータの伝送が増大する為である。このような場合は、前方誤り訂正よりも同一のパケットを複数個送信する方がよい。

EMON システムの実装したプログラムの機能とプログラム間の接続関係について述べる。それから、EMON システムを用いて実現したシステムの実現方法を述べる。まずはじめに、利用を想定した三つシステムと、電話システムと実時間放送システムを組み合わせたシステムの実現方法を述べる。次に、既存のプログラムとして SSH と tee をとり上げ、それぞれと EMON のプログラムを連携して行なった処理の実現方法を述べる。最後に、無線 LAN をもちいて実現した IP ネットワーク上の無線マイクについて述べる。

Multimedia Stream Processing by Command Pipeline

Ken-ichi KASAMATSU

Abstract

Progress of data communication technology and multimedia data compression technology make it possible to transmit not only characters or pictures but also audio and video on the Internet. Everyone can get access to the Internet even at homes and street corners as well as in schools and companies. Therefore, the opportunity of communicating on IP networks is on the increase.

Application programs which realize a telephone system or a real-time broadcast system on the Internet have already been developed. Those programs implement various functions, e.g., video capturing, video compression, network transmission, and so on. However, most of them cannot cooperate with other programs. In other words, a function of a program can hardly be added to another program. When a data processing system that is not realized by any existent application systems is constructed, knowledge of a programming language such as C or JAVA is required. A system that easily integrates modules each of which is implemented for one function may solve these problems.

In this paper, we propose a system which consists of executable stand-alone programs, each of which is implemented for one function. These programs are connected by the command pipeline. The reason why we let a module to be executable is that not only existent programs but also future programs can be applied as modules without any changes. We compare three inter-process communication methods used on UNIX; pipe, socket and IPC of systemV. As a result of comparing them, we chose pipe, since it easily implements connection with existent programs.

We clarify which functions should be implemented in the programs, before structuring the system. We consider required functions in actual systems, video on demand, real-time broadcast and telephony systems, and implement the required functions in our system called "EMON" We design each function as one of programs in the EMON system. In designing functions of synchronized playback audio and video, we let them to be replayed on different hosts. In designing functions of forward error correction, we adopt the Reed-Solomon

code in order to handle packet loss. However, the Reed-Solomon code is not effective, when only small size of packets are transmitted, in e.g., a telephone system, where low latency is important. This is because the Reed-Solomon code requires buffering for error correction. In such a case, a method that transmits the same packets several times is effective.

We mention the functions of the programs in the EMON system and relation between the programs, and then describe applications realized by the EMON system; the three systems mentioned above and a combined system of telephone system and real-time broadcast system. Then, we show processing examples of the EMON system that is combined with SSH and tee, which are existent UNIX programs. Last, we describe a wireless mike on an IP network by using a wireless LAN.

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

目次

第1章　はじめに

近年、映像や音声のマルチメディアストリームを圧縮する技術が進歩し、MPEGを中心に幅広い分野で応用されている。また、データ通信技術の進歩によってインターネットの広帯域化が進んでいる。従来インターネットではEメールやウェブなどによって文字や静止画を主に伝送されていたが、このような環境変化にしたがって、音声や映像のストリームの伝送も行われるようになった。そして、音声や映像のストリームの伝送を応用した様々なマルチメディアアプリケーションが利用されるようになった。また、学校や会社をはじめ、家庭や街角でもインターネットに常時接続する環境が整いつつあり、IPネットワークを用いて映像や音声を伝送可能な環境は広がっている。

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。それでも、インターネットで音声や映像のストリーム伝送を、多くの人々が行えるようになってから間もないため、現在存在しない全く新しいシステムが次々に登場すると考えられる。

現在利用されているシステムを実現するマルチメディアアプリケーションの大半は、システム毎に別々のアプリケーションプログラムが存在している。そして、これらのアプリケーションは多数の機能を備えているのだが、そのアプリケーションが目的とするシステムとしてしか利用できない。また、他のアプリケーションとの連携を考慮してい場合、既存のプログラムを活用できないだけではなく、将来実現される機能をそのアプリケーション自身を変更すること無く利用することは不可能である。このような柔軟性の低いアプリケーションでは、さらに、IPネットワークを利用できる環境が広がっている今日、IPネットワークを十分に活用することは期待できない。そこで、既存の特定のシステムには捕らわれない、柔軟なマルチメディアストリーム処理システムが求められる。

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、システムの利用者が行う処理に必要な機能を持ったプログラム同士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、

本来このシステムには無い機能をを利用できる。つまり、機能毎に別々のプログラムとしたことによって、新たな機能の追加をその機能を持つ単体で実行可能なプログラムの追加によって行えるシステムを提案する。

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、提案するシステムの概要を述べる。そして、提案するシステムがどのような機能を実装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。それから、提案するシステムの実際について述べた後、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。

第2章　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として、マイクロソフト社製のマルチメデイアアプリケーションとGStreamer[1]というアプリケーションを取り上げ、これら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

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。図1のように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、ある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何も変更しなくても利用できる。ところが、ネットワーク伝送などに

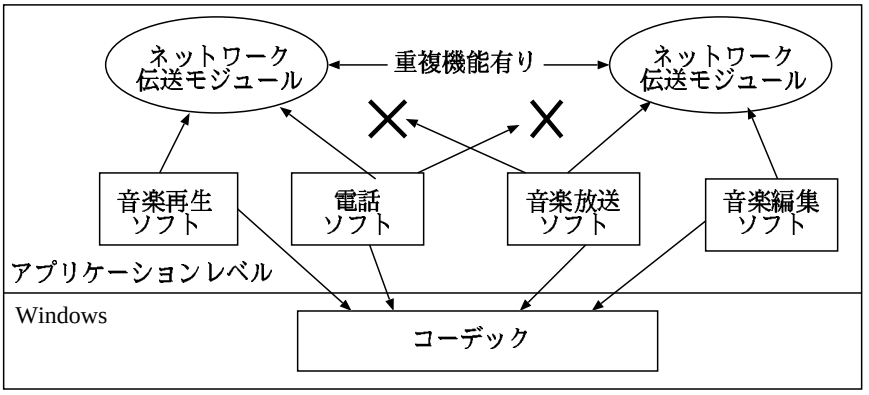

図1: コーデックとネットワーク伝送の位置づけの違い

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。もし、Windows がネットワーク伝送方式についてもコーデックのように共通のインターフェースを備えれば、この問題は解決されると考えられる。

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。まず、これらのアプリケーションが持つ機能の一部を述べる。

- MediaPlayer
  映像や音声のマルチメディアストリームを IP ネットワークから受信し、伸長した後、映像の表示および音声の再生が可能なアプリケーションである。そのため、マルチメディアストリームをインターネットで受信する機能、映像や音声のストリームを伸長する機能と、映像や音声のストリームを再生する機能は備えている。
- MediaEncoder
  カメラやマイクなどから音声や映像を取り込み、圧縮した後、記憶装置に保存したり IP ネットワークで放送することが可能なアプリケーションである。そのため、カメラやマイクから映像や音声を取り込む機能、映像や音声のストリームを圧縮する機能と、マルチメディアストリームを IP ネットワークで放送する機能は備えている。
- NetMeeting
  H.323[2] 会議システムのクライアントで、電話システムとして利用可能なアプリケーションである。そのため、電話の発信、着信の制御や双方向に音声ストリームを IP ネットワークで伝送する機能と、コーデックにより圧縮された音声ストリームを生成する機能は備えている。

以上のように、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 2 のようなシステムを実現することは不可能である。それから、スピーカやマイクなどの装置と計算機間の音声通信は、現在は主にア

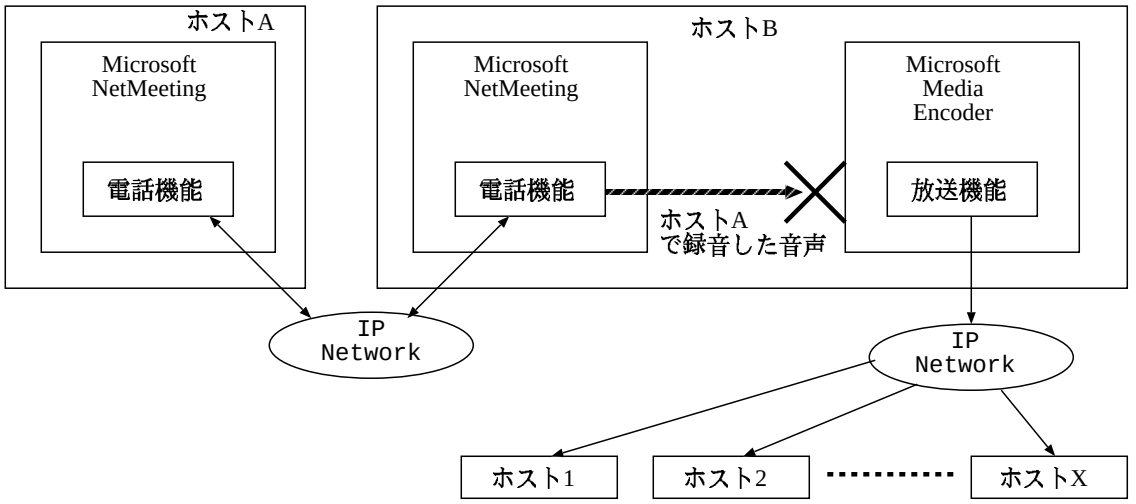

図 2: 放送機能と電話機能を組み合わせたシステム

ナログデータによって行なわれているが、IP ネットワークを利用できる環境が広がっているので、これらの装置と IP を用いてデータ通信をすることも考えられる。しかし、MediaPlayer を利用して IP ネットワークで音声を受信しながら、IP を用いてデータ通信を行なうスピーカに MediaEncoder を IP ネットワークで伝送することは不可能である。ただし、MediaPlayer と MediaEncoder に関連するソフトウェア開発者向けのツールがマイクロソフト社から提供されているので、それによって図 2 のようなシステムを実現することは可能かもしれない。だが、これらのツールを利用する為にはプログラミング言語などの知識が必要であるため、アプリケーションの利用者がこれらのツールを利用することは困難である。つまり、プログラミング言語などの知識が無ければ、多様なストリーム処理を行うことができないという問題がある。もし、各機能毎に様々な用途に利用できるモジュールを実装し、適切なインターフェースを備えておき、これらのモジュールの組み合わせを容易に記述する方法を提供すれば、各モジュールが持つ機能を組み合わせた多様な処理を行うことは容易になると考えられる。この場合、取り上げた三つのアプリケーションは、必要な機能を備えたモジュールとモジュールのある組み合わせに特化したユーザインターフェース加えたアプリケーションと考えることができる。

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。また、一般に公開されているため、新たな機能を持ったモジュールの追加は容易である。さらに、利用するモジュールの組み合わせを GUI や対話シェル型のインターフェースを用いて容易に記述することができる。それから、パイプを用いて既存のプログラムをモジュールとして利用することも可能である。

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。もし、映像や音声の取り込みをを行うモジュールが他のメディアとの同期を考慮して、取り込まれた時刻の情報をストリーム付加しておけば、複数のストリームの再生や多重化を行う時に同期をとることが可能になる。ただし、他のメディアと同期するメディアを取り込みむモジュールは、時刻の情報としてどのような情報を付加するのかを決定する必要がある。また、再生を行うモジュールについても同様の問題を持っている。

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、システムの利用者が行う処理に必要な機能を持ったプログラム同士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。そこで、UNIX ホスト上で利用可能な代表的なプロセス間通信の方法に

[1)] GStreamer バージョン 0.3.1

ついて考察する。

- パイプ、 FIFO(名前つきパイプ)
  プログラムの標準入力と他のプログラムの標準出力を繋ぐときに用いられることが多く、数多くのUNIXのプログラムがこのようなパイプの利用方法を想定している。また、プログラムの接続関係をコマンドパイプラインを用いて容易に記述することができる。また、FIFOを用いるとより複雑なプログラムの接続関係を記述することができる。しかし、データの伝送の性能が良くないことや、プログラムの実行前に接続先のプログラムを決定していない場合、パイプの利用は簡単ではない。
- ソケット
  UNIX DOMAINソケットやIPソケットを用いると、プログラム間の接続構成を動的に変更したり、他のホストのプロセスと通信することも簡単に行なうことができる。しかし、パイプに比べると同一ホストのプロセス間通信に用いられることは少ない。
- SystemVのIPC
  メッセージキューや共有メモリーはパイプに比べてデータ伝送効率が高く、様々な機能を備えたプロセス間通信機能を提供する。しかし、パイプに比べると利用しているプログラムは少ない。

以上の考察から、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。なぜなら、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。プログラム間のインターフェースを機能毎に定めなければ他の機能をもつプログラムとの連帯ができず、機能毎に様々なプログラムを実装することもできない。そこで、プログラム間のデータ通信はパイプを用いることとしたので、パイプのストリームをメッセージのストリームとして用い、このメッセージの形式を定めることによって機能の違うプログラム間のインターフェースを定める。そして、図3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

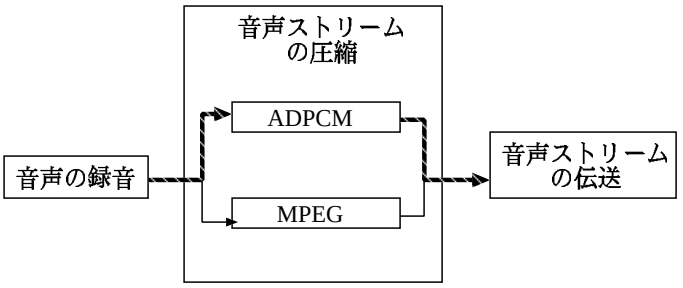

図 3: 機能を実現するプログラムの交換

ラムを変更することなく利用できるようにする。ここで、どのような機能の対してインターフェスを定めるのかを考えなければならない。もし、機能を細かく分けると、その機能に特化したインターフェースを定めることが可能になる為、性能が高いプログラムを実装できるが、インターフェースの増加によって、プログラムが連携するために対応すべきインターフェースの数が増加し、実装が困難になる。その一方、機能を細かく分けなければ、特定の機能に依存する要素をインターフェースに含める必要があるため、インターフェースが複雑になり各機能を実現するプログラムの実装が困難になる。よって、適切な機能の分類を行うことが必要である。

第3章　利用するシステムの想定

本論文が提案するシステムは、機能毎にそれぞれプログラムを実装するシステムであることをすでに述べた。そのため、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、実在する実際のシステムを想定し、それらのシステムが要求する機能を提案するシステムが要求する機能とすることとする。想定するシステムはビデオオンデマンド、実時間放送、電話の三つのシステムとした。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たい映像を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と考える。また、利用者は見たい時に見たい映像を選択して見る為、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。そこで、映像を送信するホストは、複数の映像をあらかじめ保存しておき、利用者のホストが要求する映像を利用者のホストへIPでデータの損失が無いようにユニキャストを用いて送信する処理を行うこととする。そして、利用者のホストは映像を受信し、その映像を再生する処理を行うこととする。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でユニキャストを用いて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

本論文で述べる実装では考慮しないが、映像ストリームを途切れること無く表示する為には他にも必要な機能がある。まず、映像ストリームが必要とする帯域に比べて、ネットワーク伝送経路の帯域が不足している場合、不足している帯域と映像ストリームの長さに比例して受信ホストに大きな記憶装置が必要となる。それから、伝送時に発生するジッターの大きさよって受信ホストに必要なバッファーが大きくなるため、十分なバッファーを確保できない受信ホストへの伝送は不可能になる。

これら二つの問題は、用いる伝送経路に対し帯域やジッターなどのQoS保証を行うと解決することができる。それから、映像の品質の低下が許されるならば、利用可能な帯域の変化に応じて伝送する映像ストリームの品質を変化させ、映像ストリームが必要とする帯域よりも常にネットワーク伝送経路の帯域が広くなるようにすれば良い。また、単一の映像に対して複数の受信ホストが存在する場合に限られるが、マルチキャストを用いると伝送帯域の不足する問題を解決できる可能性がある。

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。そこで、放送するホストは、カメラやマイクなどを

用いて映像と音声を取り込み、前方誤り訂正の為のデータを付加した後、IP でマルチキャストを用いて実時間で送信する処理を行うこととする。そして、視聴者のホストは映像と音声のストリームを受信し、前方誤り訂正を用いてデータの損失を回復し、映像と音声を同期再生する処理を行うこととする。よって、実時間放送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
- 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

本論文で述べる実装では考慮しないが、映像ストリームを途切れること無く表示する為には他にも必要な機能がある。ビデオオンデマンドシステムと同様にネットワーク伝送経路の帯域不足や伝送時に発生するジッターによって受信ホストに必要なバッファーが大きくなる。帯域やジッターなどの QoS 保証を行うとビデオオンデマンドシステムと同様に解決することができる。しかし、映像の品質の低下が許される場合は、ビデオオンデマンドシステムと同様に伝送する映像ストリームの品質を変化させる方法は有効ではない。これは、単一のストリームに対して複数の受信者が存在する為、利用可能な帯域が十分にある受信者が受信するストリームの品質も低下させてしまう為である。また、受信者の数に応じて複数の品質のストリームを伝送するとマルチキャストの利点である受信者数のスケーラビリティを低下させてしまう。そこで、受信者が利用できる帯域幅の範囲を幾つかにわけて階層伝送を行うと、受信者の数に関わらず各受信者が利用できる帯域に応じた品質のストリームを受信することができる。

3.3 電話

電話システムは、全ての利用者が他の利用者に発信することが可能で、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。

そこで各ホストは、通話中でない時は着信待ちの処理を行い、着信すると通話を開始し、発信を行う時は、利用者が指定したホストへ発信処理を行い、通話中でなければ通話を開始する。通話の開始後は、発信側、着信側の双方のホ

ストが、マイクなどから音声を取り込み、相手側へIPで実時間で送信すると同時に、相手側の音声ストリームを受信し、再生処理を行うこととする。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

第4章　EMONシステムの提案と設計

第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。想定するシステムが要求する機能を全て挙げると次のようになる。

- 映像の取り込みと圧縮
- 映像ストリームの伸長と再生
- 音声の取り込みと圧縮
- 音声ストリームの伸長と再生
- 映像ストリームと音声ストリームの伸長と同期再生
- 映像ストリームの保存
- 前方誤り訂正
- IPでユニキャストを用いてデータの損失が生じない伝送
- IPでマルチキャストを用いて実時間伝送
- 双方向にIPで実時間伝送
- 電話の着信
- 電話の発信

まず、音声の取り込みと圧縮の機能について考えると、取り込み機能と圧縮機能に分けることが考えられる。想定したシステムでは求められてい処理である

が、例えば、複数の音声ストリームを一つの音声ストリームに混ぜ合わせる処理は、圧縮前のストリームの方が処理が簡単な場合がある。また、音声ストリームの圧縮方法は既に多くの方法が存在するが、将来新たな圧縮方法が登場することも考えられる。そこで、音声の取り込みの機能と音声ストリームの圧縮機能に分けることとする。ただし、音声の圧縮機能を映像の圧縮機能として用いることは不可能であると考えられる。同様の理由により映像の取り込みと圧縮機能、映像ストリームの伸長と再生機能、音声ストリームの伸長と再生機能についても、取り込みと圧縮あるいは伸長と再生を別々の機能とする。

次に、映像ストリームと音声ストリームの同期再生機能について考えると、映像ストリームの再生機能と音声ストリームの再生機能の両方を含んでいる。そこで、これらの三つの機能を実現する他の機能の集合を考えることにする。各機能が重複しないような機能の集合は二通り考えられる。

- 再生機能を統一する場合
  映像ストリームと音声ストリームの同期再生機能のみとし、これによって映像ストリームの再生と音声ストリームの再生も行う。
- 同期機能をそれぞれ追加する場合
  他のストリームと同期をとることが可能な映像ストリーム再生機能と他のストリームと同期をとることが可能な音声ストリーム再生機能の二つの機能によって映像と音声の同期再生を行う。

再生機能を統一する場合、映像ストリームと音声ストリームの同期再生機能が一つのプログラムとなる為、同期機能をそれぞれ追加する場合よりも実装が容易であると考えられる。これは、同期再生の為には映像再生処理と音声再生処理を連携して行う必要がある為である。つまり、再生機能を統一する場合は単体のプログラムの中で連携するが、同期機能をそれぞれ追加する場合はプログラム間で連携する必要がある。一方、機能をそれぞれ追加する場合は映像の再生と音声の再生のプログラムが分かれている為、これらを別々のホストで実行することが可能である。これは例えば、実時間放送システムが放送する映像ストリームとそれに同期する音声ストリームをそれぞれ別のホストで再生することが可能である。ただし、別々のホストで実行する場合はネットワーク伝送遅延などの影響により、メディア間の再生時刻のズレが増大すると考えられる為、大きなズレが許されない場合は同期再生を行なうことは不可能であると考える。これは例えば、ステレオの音楽を左右別々のホストで再生する場合が考えられ

る。以上の考察より、同期機能をそれぞれ追加することとする。

それから、映像ストリームの圧縮や表示のように、メディアデータの中身を処理する機能は、メディアの種類毎に別々の機能が必要であるが、ネットワーク伝送やメディアストリームの保存などの機能は必ずしもメディアデータの中身を処理する必要が無いと考える。そこで、映像ストリームの保存機能と音声ストリームを双方向に実時間でネットワーク伝送機能の代わりにメディアストリームの保存機能とメディアストリームを双方向に実時間でネットワーク伝送機能を EMON が備えることにする。

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
- 他のストリームと同期をとることが可能な映像ストリーム再生機能
- 他のストリームと同期をとることが可能な音声ストリーム再生機能
- 映像ストリームの保存
- 前方誤り訂正
- IP でユニキャストを用いてデータの損失が生じない伝送
- IP でマルチキャストを用いて実時間伝送
- 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

映像の取り込み機能を, 映像をカメラから取り込み、圧縮を行っていない映像ストリームを生成する処理を行うプログラムによって実現する。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
  カメラが複数存在する場合、映像を取り込みを行うカメラの選択が必要である。もし、カメラによって取り込み処理が異なっているならばカメラ毎に別のプログラムを実装する。
- 生成する映像ストリームの品質を設定
  映像を実時間放送する場合、伝送に利用可能な帯域を越えない映像を利用者が選択する必要がある。また、用いるカメラの性能を越える高い品質の映像を取り込むことは不可能である。
- 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。また、映像の取り込みを行うプログラムと、その映像に同期する音声の取り込みを行うプログラムが生成するストリームを入力するプログラムによって直ちに単一のストリームに多重化する場合であっても同期をとることができない場合がある。それは、映像がカメラに取り込まれてから多重化されるまでの処理時間と音声がマイクに取り込まれてから多重化されるまでの処理時間の差が時間変化する場合である。もし、各ストリームに時刻情報を付加すれば、ストリームを多重化するプログラムや同期再生を行なうプログラムに、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伝達できる。しかし、時刻情報を同じ時計を基に付加することが必要である。もし、別々の時計を基に時刻情報を付加すると、時計のズレの時間変化が同期のズレに繋がる。時計の共有は、単一ホストのプログラム間ではホストが持つ同じ時計を用いればよい。また、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[3] 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他のストリームと同期をとりながら再生する処理を行うプログラムによって実現する。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。もし、ジッターをもとにしたバッファ管理行わず、ストリームを入力後、直ちに再生処理を行うと、遅延の大きなフレームは再生が遅れ、遅延の小さなフレームは再生が早くなり、取り込んだ間隔と同じ間隔で再生することができない。
- 他のストリームと同期をとる場合
  他のストリームと再生同期をと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。そのためにはまず、プログラム間の時計の共有が必要であるが、単一ホスト内ならばホストの時計を、複数のホスト間なら NTP[3] の技術を利用すれば良い。次に、ストリームの時刻情報と再生する時のホストの時刻との対応が必要である。まず、各ストリームを取り込んだ間隔で再生するが、できる限り再生までの遅延時間を小さくする処理は、同期をとらないバッファ管理でも行うことができる。そして、この処理の中で取り込みから再生までにかかる時間を求めることは可能である。したがって、この時間を同期するストリームを再生するプログラムの間で共有し、最も再生までに時間がかかるストリームに合わせて再生することができれば、取り込みから再生までの時間が最小の同期再生を行うことができる。しかし、再生までの遅延時間の小さなストリームに対して大きなバッファーが必要になるが、各プログラムが利用できるバッファーには限界がある。そのため、再生までの遅延時間がストリーム間で大きく異なる時、同期再生するストリームの中でさらに同期再生するストリームの集合に分ける必要がある。ただし、再生処理の代わりに伸長処理を遅らせるようにすれば必要なバッファーを小さくすことは可能である。

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に RTP[4] というプロトコルが存在する。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である。したがって、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機能である。だから、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、実時間伝送を行う機能を RTP に依存しない機能にするため、UDP を用いてマルチキャスト伝送する機能とストリームを RTP で伝送する形式に変換する機能に分け、それぞれの処理を行なうプログラムを組み合わせて実現することとする。

次に、双方向に実時間でネットワーク伝送する機能について考えると、ストリームを RTP で伝送する形式に変換するプログラムを実装することとしたので、これに UDP を用いてユニキャスト伝送する機能を加えることで実現する。それから、データの損失が生じないネットワーク伝送機能は TCP を用いると容易に実現できるので、TCP を用いて実現する。

4.4 前方誤り訂正

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と考える。TCP 層はデータの損失とビット誤りが生じないようにする為、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 層ではデータのビット誤りは生じないようにするが、データの損失が生じる。そこで EMON では、前方誤り訂正機能を UDP を用いる時にデータの損失を回復する処理を行うプログラムによって実現する。

このような前方誤り訂正を行う技術として、リードソロモン符合を用いて前方誤り訂正を行う技術が存在する。この技術を用いると、同じ長さの K 個のデータに対し、リードソロモン符号により R 個の冗長なデータを生成したとき、K+R 個のデータのうち任意の K 個のデータにより K+R 個全てのデータを復元できる。K と R は任意の値[1)]を用いる事ができる。そこで、メディアストリームのネットワーク伝送に用いることを考える。まず、図 4 のようにデータブロックを同じ長さの K 個のデータパケットに分割して格納し、これに対する R 個の冗長なパケットを生成する。そこで、K 個データパケットのみを送信した場合

[1)] データの長さが n バイト単位の時 $K + R < 256^n$ を満たすことが必要

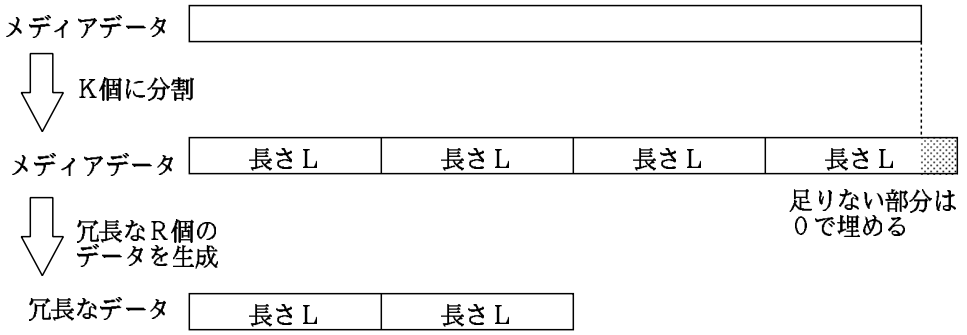

図 4: リードソロモン符合による冗長なパケットの生成

を考えると、K 個全てのパケットが受信できなければデータブロックの受信はできない。一方、K 個のデータパケットに加えて R 個の冗長なパケットも送信した場合を考えると、K+R 個のうち任意の K 以上のパケットを受信するだけでデータブロックの受信ができるため、データブロックの損失割合を低下できる可能性がある。ただし、送信するパケットの増大によりパケットの損失率を増加する可能性がため、データブロックの損失割合を必ずしも低下させるわけではない。それから、誤り訂正を行な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。

ここで、電話の音声伝送処理を UDP/IP による RTP を用いて取り込みから再生までの時間が短くなるように行なうことを考える。この時、短い音声のデータブロックを短い時間間隔で伝送するため、データブロックが十分に小さいと、一つのデータブロックが一つの RTP パケットのペイロードに収まる。つまり、前方誤り訂正を利用すると、一つの RTP パケットで伝送可能なデータブロックを複数のパケットで伝送することになる。そのため、前方誤り訂正を利用すると、RTP 層以下で必要なパケットヘッダを余分に送信する必要がある為、データブロックが小さいほどメディアデータの伝送効率が低下する。したがって、前方誤り訂正を利用するよりも、同一のデータ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可能性がある。ただし、同一のデータのパケットを複数受信することが可能な伝送プロトコルを用いる必要がある。そこで、RTP はこれが可能であるので、メディアストリームを RTP を用いて伝送する形式に変換するプログラムはこの機能を備えることにする。

4.5　メディアストリームの圧縮、伸長

メディアストリームの圧縮機能は、圧縮されていないメディアストリームから圧縮したメディアストリームを生成する処理を行うプログラムによって圧縮機能を実現する。また、伸長はこの逆の処理を行なえば良い。ここで、映像ストリームの圧縮について考えると、H.263 や MPEG をはじめ多くの圧縮伸長技術が存在する。そのため、各圧縮技術ごとにプログラムを実装する場合と、各プログラムが複数の圧縮技術を実装する場合が考えられる。また、複数の圧縮技術を組み合わせた圧縮技術が存在すれば、その圧縮技術を複数のプログラムにより実装することも考えられる。これらの問題は、圧縮技術間の類似度などに依存し、個々の圧縮技術について考察する必要がある為、実装する圧縮技術によって決定する必要がある。これは、音声ストリームの圧縮についても同様である。

4.6　メディアストリームの保存

メディアストリームの保存機能は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。これは、プログラムが入出力する全てのメディアストリームはパイプを流れるからである。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、そのような値を適切な値に置き換える必要がある。

4.7　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。通話は音声ストリームの送信と受信を同時に行なうことによって行なえるので、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る処理を行なえば実現できる。

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。

第5章　提案するシステムが実現したマルチメディアストリーム処理

本章では第4章の設計を基に行なった提案するシステム (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。

5.1　提案するシステムの実装

まず、実装したプログラムの一部[1)]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5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。

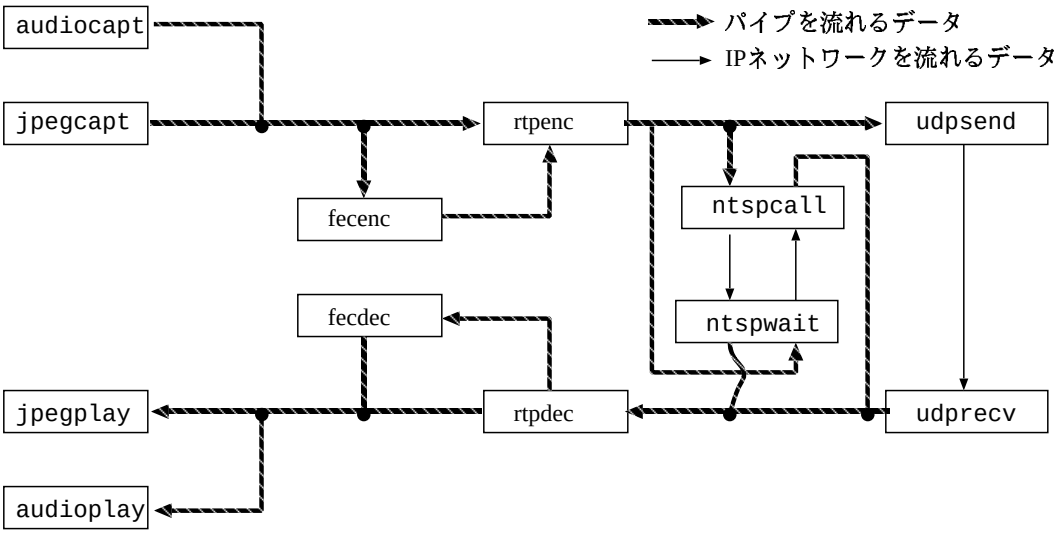

図5: プログラムの接続関係

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。この環境と実装に利用できた時間の関係により、設計と実装では異なる部分がある。まず、ストリームの取り込みと圧縮の処理を別々のプログラムではなく、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。実装に先だって、

[1)] デバッグなどに用いるプログラムも実装した

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

%dd if=/dev/zero bs=‘echo ’720∗480∗2’|bc‘ count=30 |cat >/dev/null
を実行すると、約 0.42 秒かかった。これは、映像取り込み装置から NTSC 相当の映像一秒分を主記憶に転送し圧縮を行わずに一つパイプを通す処理の参考時間と考えられる。また、実行環境において圧縮などの処理も同時に行う必要がある。そこで、ストリームの取り込みと圧縮の処理を単一のプログラムに実装した。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に実装した。それから、電話に関する機能は NOTASIP[5] 方式で実装した。NOTASIP では発信と着信のための処理が存在せず、音声ストリームの伝送によって発信と着信の処理が行われる。そこで、電話の着信待ちとメディアストリームを双方向に伝送をする機能と電話の発信とメディアストリームを双方向に伝送をする機能をそれぞれ実装した。

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。そのため、通信するプログラムの対応によって不要な情報も含まれている場合がある。ただし、必要な情報は全て含まれるようにした。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 6 のように定めた。マーカービット (図 6 の M の

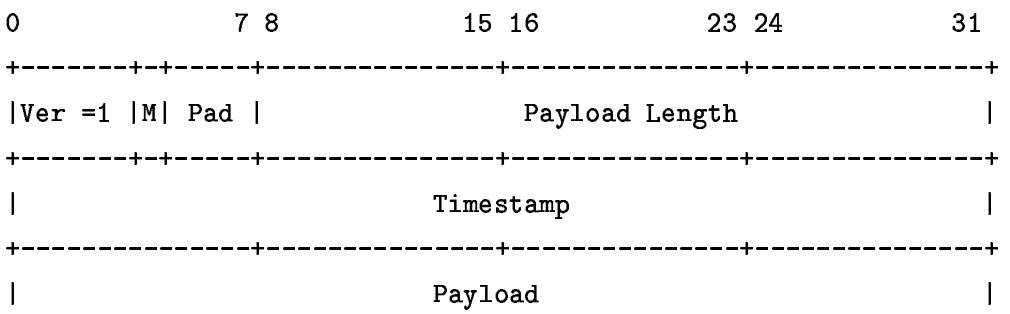

図 6: パイプのメッセージ形式

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。それから、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 システムが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、これらのシステムを EMON システムによって実現した方法を述べる。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。まず、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 7 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 sample.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

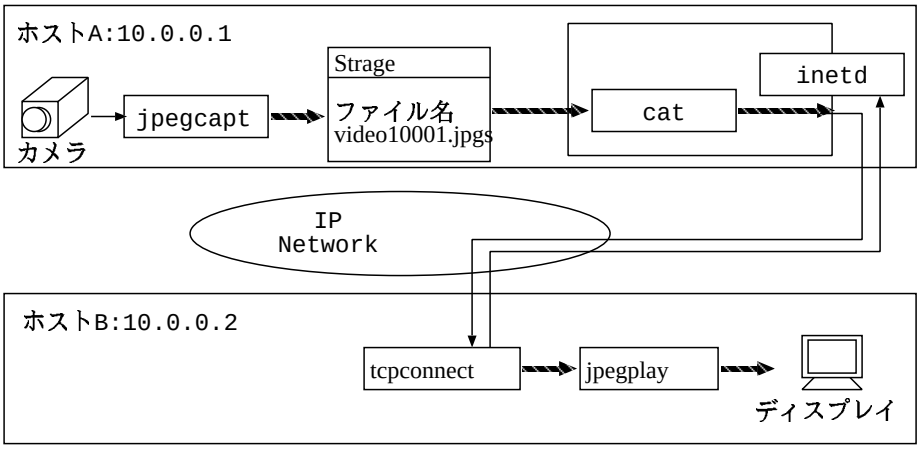

図 7: ビデオオンデマンドシステム

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の実行を開始し、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。このようにして配信さる映像を再生する方法を述べる。

% tcpconnect -A 10.0.0.1 -P 10002 | jpegplay

とホスト B で実行すると、まず、ホスト A のポート 10002 へ TCP/IP で接続し、ホスト A が video10002.jpgs の送信を開始する。そして、その映像ストリームを jpegplay で再生する。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。

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。では、映像を放送する方法を述べる。

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

とホスト A で実行すると、図 8 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。そして、RTP を用いて伝送をする為に fecenc の出力を rtpenc がパケット毎に分解し、RTP のヘッダーを付加した後、udpsend がそれをマルチキャスト伝送する。もし、FEC を利用しない場合は場合は “fecenc |” を除いて実行する。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。

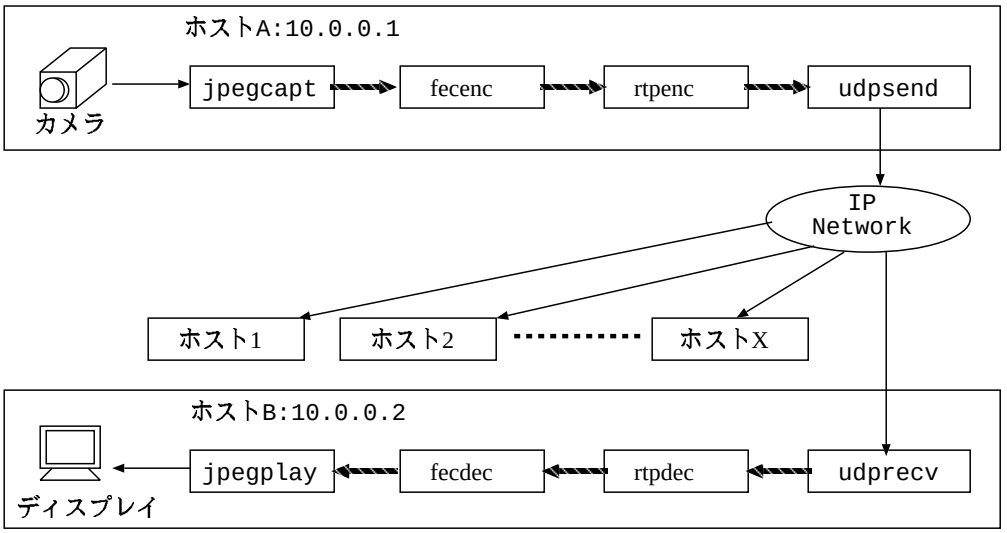

図 8: 実時間放送システム

% 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 を利用しないように放送していたならば、“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。

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 への発信のまち受けを開始する。まち受け中にホスト A の UDP/IP ポート 10000 へ発信があると、通話を開始する。通話開始後は図 9 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。

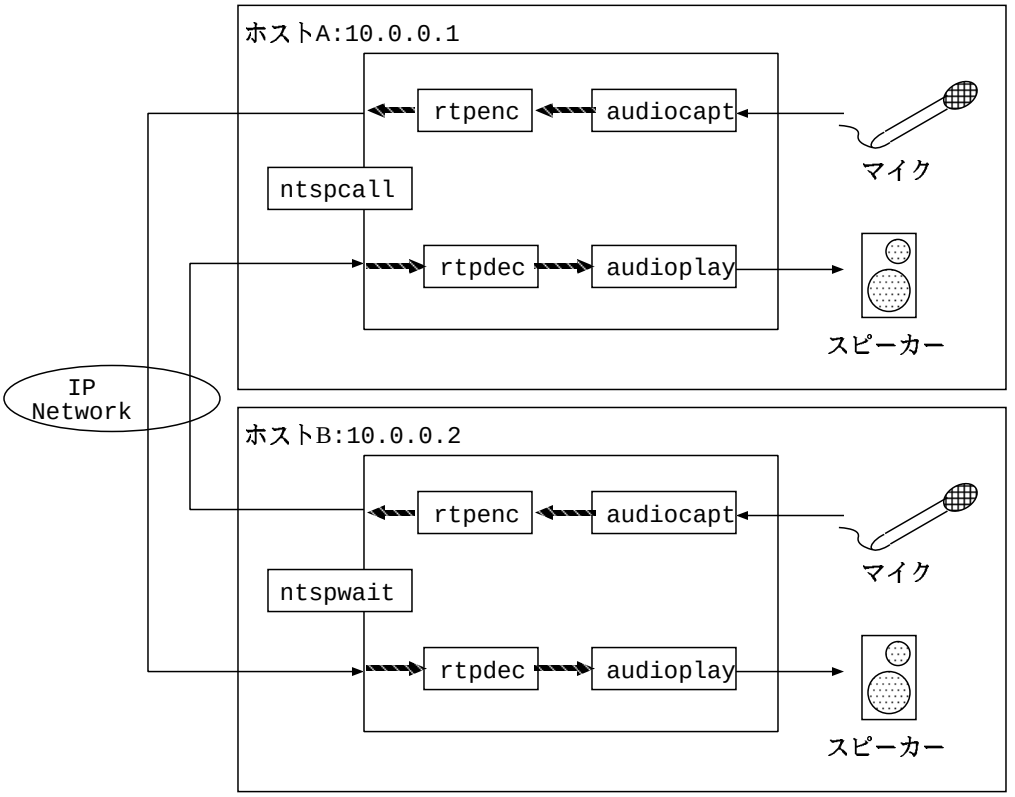

図 9: 電話システム

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図 10 のような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 へ電話をかけた時、ホスト A が送信する音声ストリームをホスト B が放送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う。通話開始後は、ホスト B が受信するストリームを udpsend により 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。

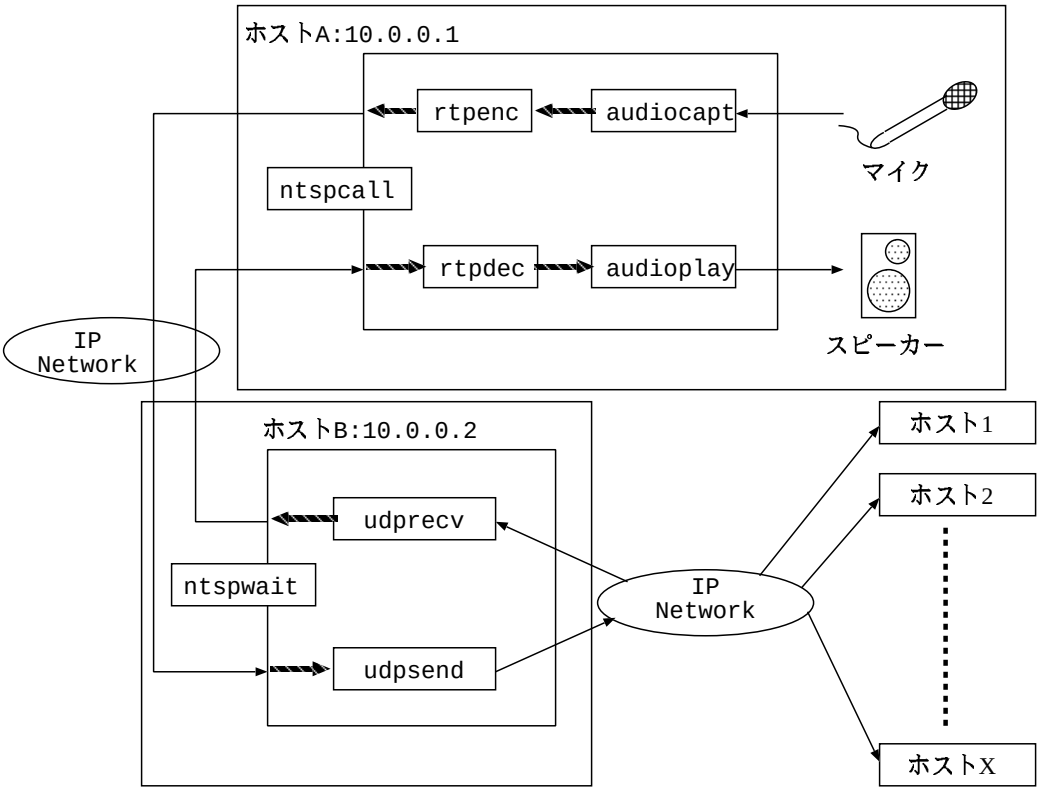

図 10: 電話と実時間放送を組み合わせたシステム

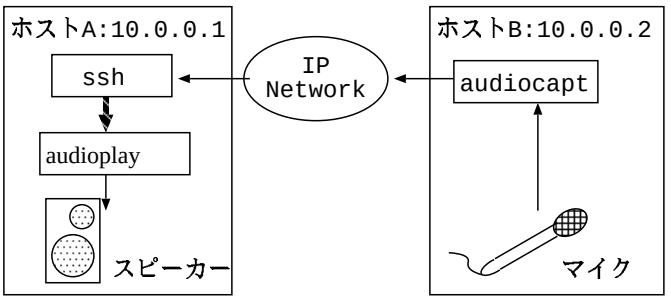

図 11: SSH を用いた処理

5.4 既存のプログラムとの連携

ホストの認証とデータの暗号化により、二つのホスト間に安全な通信経路を提供する SSH というプログラムが存在する。これを利用して、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、再生するシステムを構築した。では、このシステムの実現方法を述べる。

% ssh 10.0.0.2 audiocapt | audioplay

とホスト A で実行すると、図 11 のようにホスト A とホスト B の間で SSH よっ

て暗号化された伝送が行なえるようにした後、ホストBでaudiocaptが録音、圧縮を行ない、それをSSHがホストAへ伝送し、ホストAでaudioplayがそれを再生する。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。

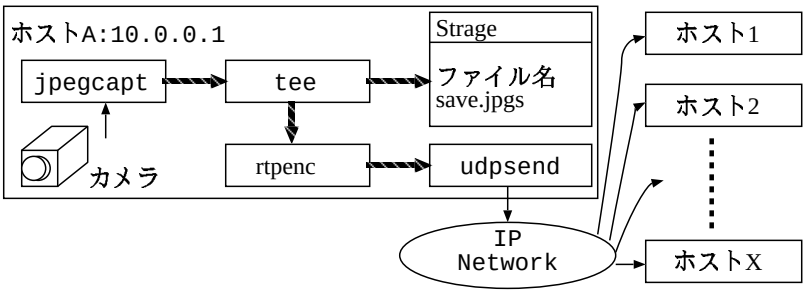

図12: teeを用いた処理

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。そこでteeを利用し、カメラなどから取り込んだ映像をファイルに保存しながらマルチキャスト伝送する方法を述べる 。

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

とホストAで実行すると、図12のようにホストAにおいてjpegcaptがカメラなどから取り込んだ映像を、teeがsave.jpgsに保存しながらrtpencに出力する。そして、udpsendが映像ストリームをマルチキャスト伝送する。

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、図13マイクで取り込んだ音声をRTPを用いて10.0.0.1のポー

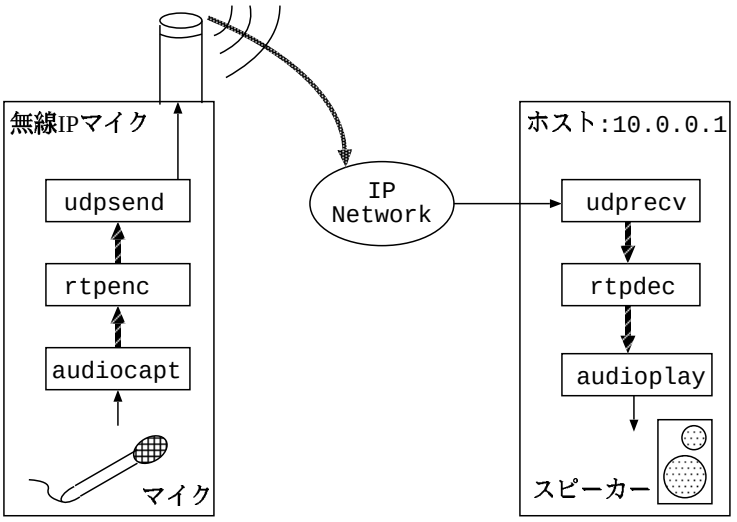

図 13: 無線 LAN を用いた無線マイク

ト 10002 へ伝送する。

第 6 章　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設計を行ったが、実装をしなかったため、これを実装し、同期再生が行えるかどうかを検証することである。それから、ネットワーク伝送に関わる機能を中心に設計と実装を行ったが、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。

謝辞

本研究の機会を与えて下さり、数々の貴重な御助言を賜わりました京都大学の岡部寿男助教授に心より感謝致します。本研究を進めるにあたり熱心な御指導を下さった藤川賢治博士と、本研究を始めるきっかけを与えて下さった古村隆明博士に深く感謝致します。また、研究全般において支えて下さった旧池田研究室の皆様と、研究会で活発な議論をして頂いた美濃研究室の皆様に感謝致します。

参考文献

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

[2] the International Telecommunication Union: H.323, *http://www.itu.int*.

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

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

[6] 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).

[7] W. リチャード・スティーヴンス: UNIX ネットワークプログラミング, トッパン, chapter 3, pp. 105–202 (1992).

[8] Mills, D.: Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, Request for Comments 2030, Internet Engineering Task Force (1996).