グループウェア 2－1
（1993．6．25）

グループウェア開発用フレームワーク

長谷部 浩一, 山口 浩司, 澤島 信介, 友田 一郎, Rajiv Trehan
(株) 東芝 研究開発センター
情報・通信システム研究所

従来のグループウェアシステムではユーザにシステムをどのように見せるかというモデルの問題と、各機能をどのような手段で提供するかというインプリメンテーションの問題とが一体となって議論されていた。しかし、これらは本来別々に議論するべき問題である。そこで、我々はグループウェアシステムのインプリメンテーションで重要な役割を果たすマルチメディアデータ処理をネットワーク上で行なうフレームワークを開発した。本フレームワークはクライアント / サーバモデルに基づきネットワーク透過なメディア処理環境を提供するメディアサーバと、並行オブジェクト指向 C 言語 cooC とから構成される。また、メディアサーバと cooC を利用して簡単なグループウェアシステムを記述できるツールキット Shared Hypermedia Object Oriented Toolkit (SHOOT) を試作した。

Framework for Groupware Systems

K. Hasebe, K. Yamaguchi, S. Sawashima, I. Tomoda, R. Trehan
Communication and Information Systems Research Labs.
TOSHIBA Corporation R&D Center

To develop groupware systems, there are two big subjects, modeling and implementation. Modeling is how to show the system to users. Implementation is how to provide the functions. But in the previous discussion about groupware systems, these two subjects were not distinguished clearly. We are developing the framework for groupware systems which is handling multimedia data in network environment. This framework is consists of Media Server and cooC. Using this framework, we develop an simple toolkit, Shared Hypermedia Object Oriented Toolkit (SHOOT), to program small groupware systems.

1 はじめに

近年マルチメディアについての研究開発が盛んに行われるようになった。グループウェアシステムを初めとした、さまざまなマルチメディアシステムが発表されている。これらのシステムは、ユーザに使いやすいコミュニケーション環境を提供することを目指し、その手段として動画や音声などのマルチメディアデータを扱っている。しかし、システム開発の労力をみると、ユーザ間のコミュニケーションモデルを考えることより、マルチメディアデータを扱う部分の開発に手間がかかっているのが現状である。

グループウェアシステムの構成を考えてみると、共有するべきデータとその表示方法やユーザの指示手段などを中心とするユーザインタフェース(モデル)の部分と、そのモデルにしたがって複数のユーザ間でデータ交換／表示／入力処理を行う部分に大きく分けられることがわかる。さらに、システムの特徴はモデルの部分にあって、メディア処理部は動画を表示する / 音声をネットワーク経由で送信するなど、どのシステムもほとんど同じことを行っていることがわかる。

そこで、我々はさまざまなシステムで利用できるマルチメディアのフレームワークを開発することにした。特に、動画や音声などは従来から計算機で扱ってきた文字などより情報量の多いメディアであるり、人と人との情報交換、つまり、コミュニケーション手段として利用価値があることから、ネットワーク環境下でのマルチメディアフレームワークとして、グループウェアシステムに利用できるようにした。本フレームワークはメディアサーバ [6] [7] と並行オブジェクト指向 C 言語 cooC(Concurrent Object Oriented C) [4] [5] [8] で構成され、ネットワーク透過なマルチメディア環境とオブジェクト指向プログラミング環境を提供する。

2 グループウェアとマルチメディア処理

グループウェアシステムは共同文書作成や電子会議などの、複数人の共同作業を支援する。共同作業では作業を分担して円滑に進めるために参加者間の情報交換、つまりコミュニケーションが大切となる。つまり、グループウェアシステムはコミュニケーション支援ツールの一つということになる。「百聞は一見にしかず」という諺があるが、言葉で何度も説明されるより実際に見た方がわかりやすい場合が多い。これは、話言葉だけでなく文書でも同じであり、文字よりも図形、図形より画像、そして、できれば動画 / 音声の方が分かりやすい場合が多い (実際には資料の作り方に大きく左右されるが)。これは、文字より動画 / 音声では表情やイントネーションのような質的な情報が送れるためである。一度に送れる情報の量が多いということは人と人とのコミュニケーションには重要なことであり、そのために共同作業を支援するシステムではほとんどが動画や音声を利用している。

現在グループウェアシステムとしてさまざまなものが提案されているが、グループウェアの研究課題としては次の 2 つに大別することができる。

モデル
: システムをどの様な形態に見せるか

インプリメンテーション
: どの様にしてその機能を提供するか

まずモデル化はユーザに対してシステムをどのような形態に見せるか、どの様なサービス (機能) をどの様な形態で提供するかということである。例えば共同文書作成では作成する文書をハイパーメディアとして扱うとか、コメント用リンクを張れるか、データのロック機構はどうするかなどで、どの様な機能を付加するか、さらにそれらの機能をどのような場合に、どの様な形でユーザに提供するか研究課題である。ユーザインタフェースの問題と考えることもできる。

一方、マルチメディアシステムのインプリメンテーションでは、システムで扱うデータ (文字、図形、動画、音声など) を実際にどの様にして参加者に提示 / 送信するかという実現手段も課題である。例えば表示機器としては計算機を使うのか通常のテレビを使うのか、データ交換はイーサネットなどの計算機ネットワークを使うのか ISDN、CATV などを利用するのかなどである。

ところが、モデルの面からみるとある程度の品質でデータ交換が可能なら、その実現手段は何でも良いということがわかる。例えば、ユーザ間で動画を交換する場合、実現手段としては CATV の様なアナログケーブルをユーザ間に引いてしまっても、計算機ネットワークを経由させても、ISDN などの交換網を利用しても、動画を共有できるという点ではまったく同じである。高品質な画像 / 音声が必要な場合はそれなりの通信手段を利用しなければならないが、品質があまり問題でない場合は、状態に合わせて安価な低品質回線を利用したり、空いている高品質回線の一部を利用して転送することも考えられる。

しかし、従来型のシステムではモデルと処理手段とがはっきりとは分けられておらず、アプリケーションプログラムが各々のメディア制御装置やネットワークを直接制御 / 管理していたために処理手段を変更するにはプログラムを書換える必要があった。反対に、各プログラムは必要なメディア制御装置の命令をすべて知っていなければならなかった。さらに、ネットワークは別にプログラミングされているため、同じメディア制御装置でもネットワークをはさむかどうかで扱いが異なる場合が多かった。

マルチメディアシステムの特徴は動画や音声など時間的に連続したメディアを扱えることである。従来から計算機が扱ってい文字や図形などの静的メディアの場合、基本的に表示の順番に制限がなく (重ね合わせなどは別の問題)、一度表示してしまえば次に変更の指示があるまでそのままにして置くことができるため、逐次処理型の計算機でも順番に処理をしていけば対応することが可能であった。

しかし、連続メディアの場合、各々のメディアが時間的に連続しているため、逐次的に処理しようとするとどうしてもループ処理をせざるを得ない。しかも、通常動画や音声を同時に複数扱うため、ループ処理が複数必要となる。これに対して、ユーザからの要求は非同期である。このため、ループ処理を行いながら同時に非同期処理を行う必要が生じる。これは現在のような逐次型のプログラミング方式ではかなり複雑なプログラムとなってしまう。

これに加えて、現状のパソコンでは余り問題にならないが、ワークステーションのように複数のプログラムが1台の計算機上で動作可能な場合、複数のプログラムが同時にメディア制御装置をアクセスしようとした場合の排他制御も連続メディアであるために問題が発生する。つまり、文字などは多少表示が遅れても問題がない場合がほとんどのため、Lock/Unlock 程度の排他制御で問題が解決できた。しかし、連続メディアの場合入力 / 出力に一定の時間がかかるため単純に Lock することができない。たとえば、ワークステーション会議システムと文書読み上げシステムが同時に音声を出力しようとした場合を考えてみる。考えられる形態は次の2つである。

1. 一方が音声ポートを排他的に専有し、他方はまったく出力できない。

2. 双方の音声が少しづつ出力される。

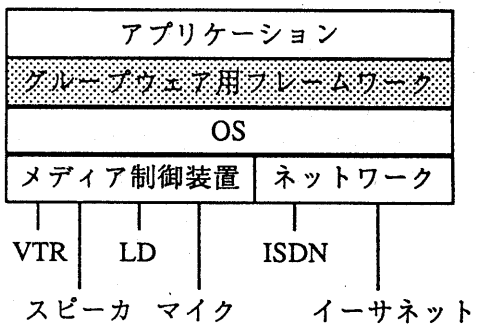

図 1: グループウェア開発用フレームワーク

多くのシステムではメディア制御装置の管理を単純にするため動作中はその制御ポートをオープンしたままの場合が多く、1のようになる場合が多い。一方、2 となるのは、ポートのオープン / クローズを頻繁に行っている場合である。この場合に問題となるのは音声データを出力した順に細切れで音声が出力されることである。さらに、音声の出力には一定の時間が必要なため、2つの音声が出力されるには2倍の時間がかかってしまうことである。つまり、2つの音声を同時に出力するためにはそれらを合成する必要がある。この機能を既存のプログラムに組み込むのは不可能である。特に、ネットワークを経由した別計算機上で合成するというのはかなり複雑となる。

そこで、我々はモデルとメディア処理の間を ISO のプロトコル階層のように階層分けし、メディア処理のフレームワークを開発することにした。特に、グループウェアではコミュニケーション手段としてネットワーク上でマルチメディアデータの交換を行なうため、グループウェアに適している。

3 ネットワークマルチメディアフレームワーク

このフレームワークは大きく2つのパートから構成される。メディアサーバと並行オブジェクト指向言語 cooC (Concurrent Object Oriented C) である。前者はネットワーク透過なメディア処理環境を提供しする。後者は並行動作が可能なオブジェクト指向のプログラミング環境を提供する。

3.1 メディアサーバ

メディアサーバは、クライアント / サーバモデルに基づき、ネットワーク透過なメディア処理環境を提供する。ここではVTRやLaser Disk (LD)などのメディア機器の制御 / 管理をサーバに担当させ、各アプリケーションはクライアントとしてネットワークを経由してサーバにメディア処理を依頼するという形態をとる。同時に動画や音声などのメディアを仮想化し、VTR や LD などという機器の違いをできるだけ隠ぺいする機構をもつ。

メディアサーバの特徴は次の通りである。

1. メディア処理とメディア制御装置の制御をサーバで、ユーザインタフェースをクライアント (アプリケーション) で行なう。(処理の分離)
2. 各メディア制御装置をアクセス (制御) するのは1つのサーバである。(1つのメディア機器を2つ以上のサーバが制御することはない。)
3. メディア制御装置間のデータ交換はすべてサーバ間で行なわれる。
4. メディア処理を行いたいクライアントはサーバに処理を依頼し、自分では処理を行なわない。
5. 動画 / 音声についての操作をできるだけ統一し、機器に依存しないメディア操作環境を提供する。

メディアサーバの概念図を図2に示す。

一般にサーバというとファイルサーバ、計算サーバというようにサーバ用計算機というものを連想しやすい。しかし、メディアサーバでは図2のようにサーバ計算機というものがあるわけではなく、各計算機に接続されているメディア制御装置ごとにサーバが存在する。言い換えると、各メディア制御装置の操作というサービスをネットワーク上で提供するプログラム (プロセス) がメディアサーバである。

VTRやマイク、スピーカなどの各メディア制御装置は必ず一つのメディアサーバに管理される。反対に一つのメディアサーバが複数のメディア制御装置を管理することは可能である。これにより複数プロセスが1つのメディア制御装置を同時にアクセスしようとする競合を避けることが可能となる。

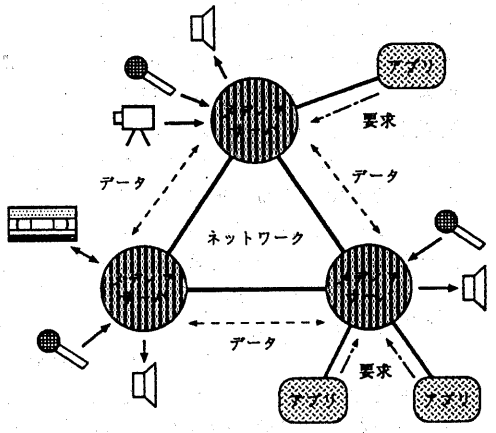

図 2: メディアサーバ

一般に、あるメディア制御装置で入力された動画や音声などは他のメディア制御装置に送られて出力される。通常はこのデータ交換をアプリケーションプログラムが行なっているために、各アプリケーションプログラムがメディア制御装置をアクセスしようとする。しかし、多くの場合入力されたデータはそのまま加工されずに出力されている。つまり、単なる転送を行なっているだけである。そこで、アプリケーションが持っていたメディアの処理部をサーバに移し、データをサーバ間で交換することにした。つまり、あるサーバによってあるメディア制御装置から入力されたデータは、そのまま目的のメディア制御装置を管理しているサーバに渡されて出力される。アプリケーションにメディア処理部が必要なくなったことで、アプリケーションの開発工数を減らすことが可能となった。メディアを加工する場合のみアプリケーションにメディア処理部を付加すれば良い。

また、メディアサーバの環境下ではメディア処理をメディアサーバに依頼する形式はメディアごとにできるだけ統一されている。例えば、VTR と LD、TV Camera からの入力は動画という点では全く同一であり、表示開始、停止などの命令は共通である。これによりメディア制御装置の違いを吸収することが可能となっている。新しいメディア制御装置が追加された場合もその機器に対応するサーバを作成すればそれ以外のサーバを含むアプリケーションは変更する必要がない。

現在のメディアサーバは、ネットワークプロトコルとして TCP/IP を利用している。これは

TCP/IP がワークステーションではほとんど標準といえるほど普及しているためであるが、同時に、TCP/IP が扱えれば計算機の種類は問題とならない。現在は、AS4000 シリーズ (SunOS 4.1) 上で、音声用サーバと動画用サーバが稼働している。必要に応じて順次機種を増やしていく予定である。

3.2 並行オブジェクト指向 C 言語 : cooC

グループウェアでは複数のユーザが複数のメディアを同時に利用してコミュニケーションを行なう。つまり、処理は並行性がある。従来からグループウェアシステムは C++ などのオブジェクト指向言語で記述することが行なわれてきた。ところが、従来のオブジェクト指向言語では 1 プログラム 1 プロセスが原則であり、並行性のあるプログラムを書くには複数のプログラムに分割したり、または複数のスレッドをプログラマがスケジューリングして並行動作オブジェクトを実現していた。しかし、いわゆるマルチスレッドのスケジューリングは非常に複雑である。特に、並行動作しているオブジェクト間の排他制御などは複雑な処理が必要だった。

そこで、メディアサーバと同時に研究を進めているメディア処理のフレームワーク用言語として並行オブジェクト指向 C 言語 cooC (Concurrent Object Oriented C) を利用することにした。

cooC は必要になった時に初めて同期をとる、という通信モデルを採用した並行オブジェクト指向言語で、並行性を実現する仕組みをスレッドとメールキューという形で抽象化したことが特徴である。文法は C、Objective-C [1]の上位互換である。グループウェア専用の言語ではないが、グループウェアシステムを記述するのに適した言語である。

並行処理部の文法は Objective-C に準拠し、さらに排他的に実行されるメソッドの定義が追加されている。並行プログラムではある資源が同時に使用されることが起こり得るが、同時に使用されると問題が生じる可能性のある場合にはこれを防ぐ必要がある。例えば、共有テキストを複数のユーザが変更する可能性のある場合などである。cooC ではこのために排他的なメソッドを宣言できるようにしている。(図 3)

cooC オブジェクトのメッセージ通信は同期型 / 非同期型を融合した形で行なわれる。つまり、要求側は、メッセージ要求を出した後、可能な限りその実行を継続し、メッセージ要求の結果が必要になった時に処理を中断して結果を待つ。この時の並行動作をスレッドとメールキューで制御することが可能である。(スレッドとメールキューについては [4] を参照。)

[1]Objective-C は Stepstone 社の商標

```
#import <cooc/Object.h>

@interface CommonText:Object
char    textString[TextLength];

/* class methods */
+new:(char *) initialText;

/* instance methods */
-(char *) printText;

/* instance exclusive methods */
@exclusive
-(char *) changeText;
@end
@end
```

図 3: cooC による排他制御

4 Shared Hypermedia Object Oriented Toolkit (SHOOT)

グループウェア開発用のフレームワークとしては以上のメディアサーバと cooC を利用することが可能であるが、さらにこれらを利用して簡単なグループウェア用のツールキット Shared Hypermedia Object Oriented Toolkit (SHOOT) [9] を試作した。

SHOOT の主な特徴は次の通りである。

1. 同時型グループウェアに対応する。
2. 動画や音声を文字や図形と同様に扱える。
3. 動画や音声、文字、図形、リンクなどのデータをネットワークで接続された複数の計算機を利用している複数のユーザ間で共有できる。
4. データは動画や音声も含め、キャンバスウィンドウと呼ぶウィンドウ上に張りつけて表現する。
5. データからキャンバスウィンドウに対してリンクを張ることができる。
6. 文字の編集には任意のエディタが利用可能で、しかもユーザごとに指定することができる。

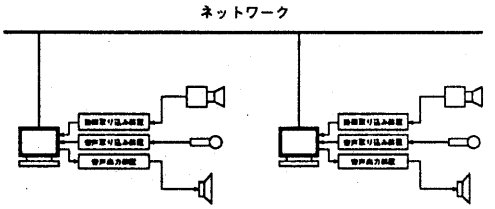

図 4: SHOOT の動作環境

7. エディットモードとブラウズモードを持ち、それぞれをキャンバスウィンドウごとに指定できる。

8. 動画や音声はメディアサーバを利用し、ネットワークに依存せずに動画や音声の交換が可能である。

9. cooC で実装されている。

4.1 SHOOT の動作環境

SHOOT はネットワークで接続された複数のワークステーション (WS) AS4000 シリーズ (SunOS V4.1) 上で動作する。

各 WS には動画取り込み装置、音声取り込み装置、音声出力装置を用意する。現在、音声入出力は AS シリーズ内蔵の音声ポート (電話音質) を利用している。動画取り込み装置としては XVideo, VideoView, VideoPix が利用可能となっている。

ネットワークとしては TCP/IP ネットを利用している。ローカルではイーサネットや光 LAN、離れた事業所間ではデジタルの専用線を経由する。また、動画に関しては当社の LX-7400 という 400Mbps の光 LAN を利用して NTSC をリアルタイムで交換するテストを進めている。

4.2 SHOOT の動作モデル

SHOOT ではユーザ共通のコントローラオブジェクトとデータベースオブジェクトがあり、ユーザの数のユーザインタフェースオブジェクトが生成される。

コントローラは SHOOT の動作を管理するオブジェクトで、データベースオブジェクトはユーザ間で共有するさまざまなデータを管理するオブジェクトである。これらは同一データ空間を共有するユーザすべてに対して 1 つずつ存在する。

ユーザインタフェースオブジェクトはユーザごとに生成され、図 6のようなツールキットをもつ。

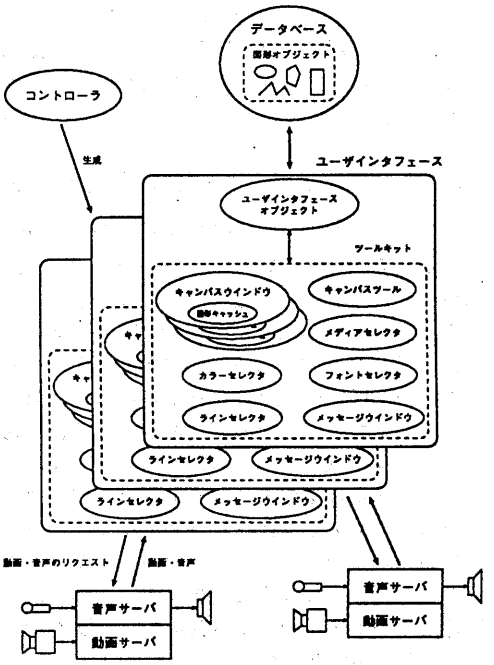

図 5: SHOOT の動作モデル

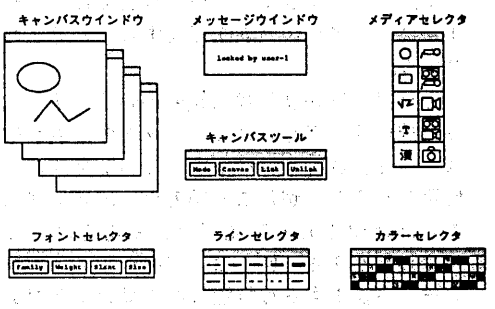

図 6: SHOOT ToolKit

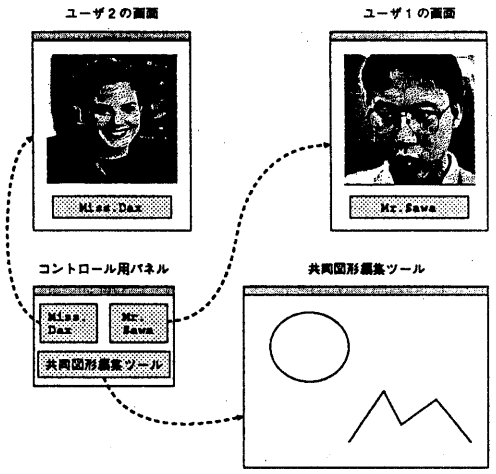

図 7: SHOOT の例

ツールキットは動作メッセージを表示したりデータ編集時の操作に利用するツールと、ユーザ間で共有しているデータベースの内容を表示するキャンバスウインドウで構成される。

キャンバスウィンドウはデータベースの様子を表示する View である。ユーザはキャンバスウィンドウに表示されている図形などを見ながら編集を行ない、編集した結果は必ず共有しているデータベースに write back される。キャンバスウィンドウは自分が所属するユーザインタフェースオブジェクト以外がデータベースを変更することをチェックするために、生成されるとデータベースの監視を開始し、データベースが変更されるとそれを自分に反映させる。

ユーザはキャンバスウィンドウ上のデータから任意のキャンバスウィンドウに対してリンクを張ることができる。これにより、簡単なハイパーメディアシステムとして利用することができる。

現在のバージョンでは、データの編集時は前述した cooC のメソッドの排他制御機能を利用してセレクトしたユーザだけが編集できるようにしている。

SHOOT では動画や音声は図形として扱っている。

図 7 に SHOOT を利用した画面の例を示す。

5 おわりに

グループウェアシステムではユーザ間のコミュニケーションを円滑に行なえるようにするために、相手の表情や声のイントネーションなどの情報を交換するために動画や音声などのマルチメディアデータを利用している。

グループウェアシステムはシステムをユーザにどのような形態に見せるかというモデルの問題と、各機能をどのようにして提供するかというインプリメンテーションの問題に分けることが可能である。マルチメディアデータの処理はさまざまなメディア制御装置やネットワークなどを管理する必要があるが、モデルとしては動画や音声というメディアをどこからどこへどの品質で送るかが重要であり、装置やネットワークに何を利用しているかはインプリメンテーションの問題である。

そこで、我々はメディア処理部分をアプリケーションから独立させ、グループウェアシステムに利用できるネットワーク環境下でのマルチメディアのフレームワークを開発した。

このフレームワークはメディアサーバと並行オブジェクト指向 C 言語 cooC とで構成されている。メディアサーバはクライアント / サーバモデルに基づき、ネットワーク透過なメディア処理環境を提供する。cooC は並行動作するオブジェクトを利用してグループウェアシステムの記述を容易にすることが可能である。

さらに、メディアサーバと cooC を利用し、簡単なグループウェアシステムを記述できるツールキット SHOOT を試作した。

本フレームワークは現在試作の段階であるが、これを利用して実際にグループウェアシステムを記述し、いくつかの環境下で実際に利用する実験を始めている。現在は UNIX を利用しているが、UNIX がリアルタイム性を保証しないためにデータの遅延などが発生することがわかっている。今後、このような問題点について検討し、さまざまなモデルのグループウェアシステムを記述できるフレームワークとしていく予定である。

参考文献

[1] Andy Hopper, *Pandora - An Experimental System for Multimedia Applications*, Operating Systems Review, 12 January 1990

[2] David P. Anderson, et al, *INTEGRATED DIGITAL CONTINUOUS MEDIA: A FRAMEWORK BASED ON MACH, X11, AND TCP/IP*, Report No.UCB/CSD 90/566, CSD, University of California, March 1990

[3] Daniel I. Applebaum, *The Galatea Network Video Device Control System*, Massachusetts Institute of Technology Media Laboratory, 1990

[4] R. Trehan, et al, *Concurrent Object Oriented C (cooC)*, ACM SIGPLAN NOTICES, 1993

[5] R. Trehan, et al, *CSCW Architecture and Shared Drawing Tool Exmple*, 第 8 回ヒューマン・インタフェース・シンポジウム, 計測自動制御学会, pp.341-348, 1992

[6] 長谷部 浩一, 他, *UNIX* ネットワークにおけるオーディオサーバ, 情報処理学会第 44 回全国大会, 3, pp.323-324, 1992

[7] K.Hasebe, K.Yamaguchi, *Continuous Network Media Server*, The International Society for Optical Engineering Symposium on Electronic Imaging Science and Technology, 1662, pp.291-298, 1992

[8] R.Trehan, K.Maeda, *A Distributed Object Oriented Language and Operating System*, IEEE hawaii International Conference on System Sciences, 1993

[9] R.Trehan, et al, *Toolkit for Shared Hypermedia on a Distributed Object Oriented A rchiecture*, The First International Conference on Multimedia, ACM, SIGGRAPH, 1993