JAIST Repository
https://dspace.jaist.ac.jp/

Title	異種ビデオネットワーク間接続に関する研究
Author(s)	中田，潤也
Citation	
Issue Date	2000-09
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1424
Rights	
Description	Supervisor:丹 康雄，情報科学研究科，修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修 士 論 文

異種ビデオネットワーク間接続に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

中田 潤也

2000 年 9 月

修 士 論 文

異種ビデオネットワーク間接続に関する研究

指導教官　丹 康雄 助教授

審査委員主査　丹 康雄 助教授

審査委員　松澤 照男 教授

審査委員　篠田 陽一 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

810201　中田 潤也

2000 年 8 月

Copyright © 2000 by NAKATA, Junya

目 次

1 はじめに **1**

2 ビデオネットワーク **2**

2.1 計算機系 2

2.2 電話系 3

2.3 家電系 5

3 相互接続のために必要となる機能 **7**

3.1 フォーマット変換 7

3.2 制御プロトコル変換 7

4 相互接続の形態 **9**

4.1 個別に接続を行う方法 9

4.2 ハブとなるネットワークを介した接続を行う方法 10

5 ビデオネットワーク統合アーキテクチャ **11**

5.1 VIA の階層構造 12

5.1.1 フロントエンドネットワーク 12

5.1.2 コアネットワーク 13

5.1.3 ゲートウェイ 14

5.2 VIA における識別子 15

6 VIA における機器の取り扱い **16**

6.1 入力のみを扱う機器 16

6.2 出力のみを扱う機器 17

6.3 一入力一出力を扱う機器 17

6.4 一入力多出力を扱う機器 18

6.5 多入力一出力を扱う機器 19

7 実際の動作 **21**

7.1 ゲートウェイのアタッチ 22
7.2 ゲートウェイのデタッチ 23
7.3 エンドターミナルノードのアタッチ 23
7.4 エンドターミナルノードのデタッチ 23
7.5 接続の確立 24
7.6 接続の開放 24

8 JAIST VideoLAN をベースとした実装 **26**

8.1 JAIST VideoLAN 26
8.2 追加が必要となるコンポーネントの検討 27
8.2.1 資源管理 28
8.2.2 制御プロトコル変換 28
8.2.3 フォーマット変換 29
8.3 実装の形態 31

9 各コンポーネントの動作 **33**

9.1 リソースマネージャ 33
9.1.1 リソースインフォメーションエージェントとの対話を行うモジュール 33
9.1.2 オペレーションターミナルとの対話を行うモジュール 38
9.1.3 データベースエンジン 41
9.2 リソースインフォメーションエージェント 41
9.2.1 エンドターミナルノードの管理を行うモジュール 41
9.2.2 シグナリングを行うモジュール 42
9.3 コントロールプロトコルトランスレータ 42

10 まとめ **44**

11 今後の課題 **45**

11.1 付加的なサービスへの対応 45
11.2 資源の分散管理 45

12 おわりに **47**

第 1章

はじめに

コンピュータ機器の高性能化やネットワークの高速化、家電機器のインテリジェント化などを背景に多くのビデオネットワークが実用化されている。これらの技術の発展によって、より高品質の映像を高速に送受信することが可能となりつつある。また、その利用範囲もビデオオンデマンドやテレコンファレンスなどの従来の利用法に加え、遠隔教育や医療用途などの新たな分野への利用も始まっている。

これらのビデオネットワークはそれぞれ規模や用途に応じて、また利用するネットワークを最大限に活用できるように特化された形態を持っており、そのために各ネットワーク間で互換性を確保することは難しく、相互接続性を有するものは少数である。

しかし、これらのビデオネットワークをそれぞれの特徴を活かして接続し、相互に運用することが可能となれば、より多くの利用者に対して高品質な映像の柔軟な送受信を行うことが可能となる。

本研究では、各種ビデオネットワークを抽象化し、相互運用を可能とするフレームワークの提案を行う。

第 2章

ビデオネットワーク

ビデオネットワークとはリアルタイムに映像の送受信を行うことを主な目的とするネットワークである。

ビデオネットワークにはさまざまな種類が存在する。これらはそれぞれが利用される規模や目的、送受信に用いる媒体などに応じさまざまな形態を持っている。

相互接続を行う場合にはこれらのさまざまな面で生じる相違を吸収する必要がある。この際に、何らかの妥協を行うことが必要となる場合があるが、ネットワークの性能をなるべく損なわないような接続を行うことが望ましい。そこで、まず始めに各種ビデオネットワークの分類を行う。

ビデオネットワークを分類するための項目として利用目的や利用する媒体などがある。

利用目的としてはテレコンファレンス、ビデオオンデマンド、ストリーム配信などがあげられる。

ここでは、それらのうち利用する媒体による分類を行う。

2.1 計算機系

計算機ネットワーク上で映像の送受信を行うシステムとしてはマルチメディア通信アプリケーションの標準規格である DAVIC、RealSystem などのストーリミングシステムがあげられる。

これらのシステムでは RTP を用いることが多い。RTP はネットワーク層以下が QoS 保証のない場合でもリアルタイム性が要求されるデータを伝送することができるプロトコルで、対話性が重視されるアプリケーションで利用される。RTP は下位層を規定しておらず、またそれ自身に QoS を保証する機構を備えていないので、RTP に付随している

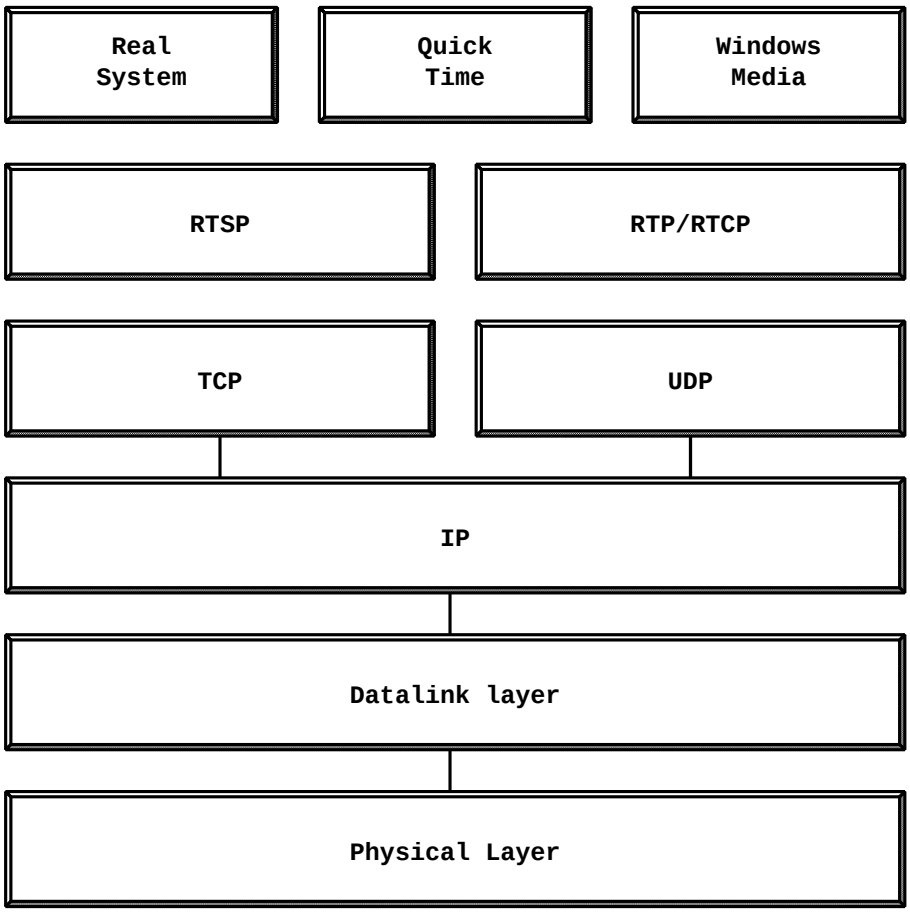

図 2.1: protocol structure of streaming protocol

RTCP などを用いて伝送の確認を行う。

アプリケーションレベルでのストリームの制御にはRTSP が用いられる。RTSP は下位層を規定せず、リアルタイムデータ伝送の制御を行う。

実際のストリーミングを行うプロトコルとして、RealSystem、QuickTime、Windows-Media などがある。

2.2 電話系

電話回線上で映像の送受信を行うシステムとしてはH.320 系テレコンファレンスシステムがある。

H.320 系のオーディオビジュアル通信システムはITU-T で標準化が行われたシステム

で、音声、動画、静止画と共有ホワイトボード、ファクシミリ情報、ビットマップ情報などのテレマティック情報を多重化したものをストリームとして伝送を行う。

これらのシステムでは使用するネットワークや帯域幅に応じたプロトコルが用意されている。

H.320 シリーズには、B-ISDN を用いる H.310、H.321、N-ISDN を用いる H.320、GSTN を用いる H.324 などがある。

H.320 シリーズは図 2.2 に示すような構成を持っており、映像、音声、テレマティックデータ、システム制御情報などを論理的な伝送路を経由して伝送する。これらの論理チャネルは多重化され、一つのストリームとして送信される。

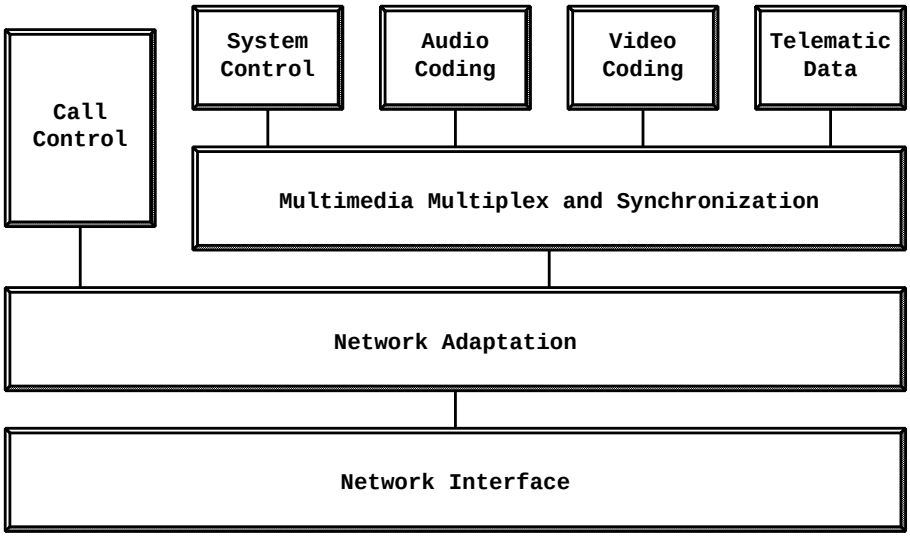

図 2.2: protocol structure of Audio-visual Communication

映像符号化方式としては H.260 シリーズ [10, 31, 25] が、音声符号化方式としては G.700 シリーズ [6, 7, 14, 8, 15] が、テレマティックデータの符号化としては T.80、T.120 シリーズ [9, 13, 19, 12] が用いられる。

これらを多重化する多重化方式としては H.220 シリーズ [26, 29, 16] が用いられる。

これらのシステムは基本的にピアツーピアで接続を行うが、H.231[18]、H.247[20]、H.331[11] に規定される MCU を用いることによって複数の端末同士の接続に対応することができる。

この場合の構成を図 2.3 に示す。このような接続を行った場合には対地の映像の内一つか、もしくは複数の映像が一つの画面に合成されたものを受信することになる。この場合でも接続は一対一で、合成はストリームレベルではなく、アプリケーションレベルで行われる。

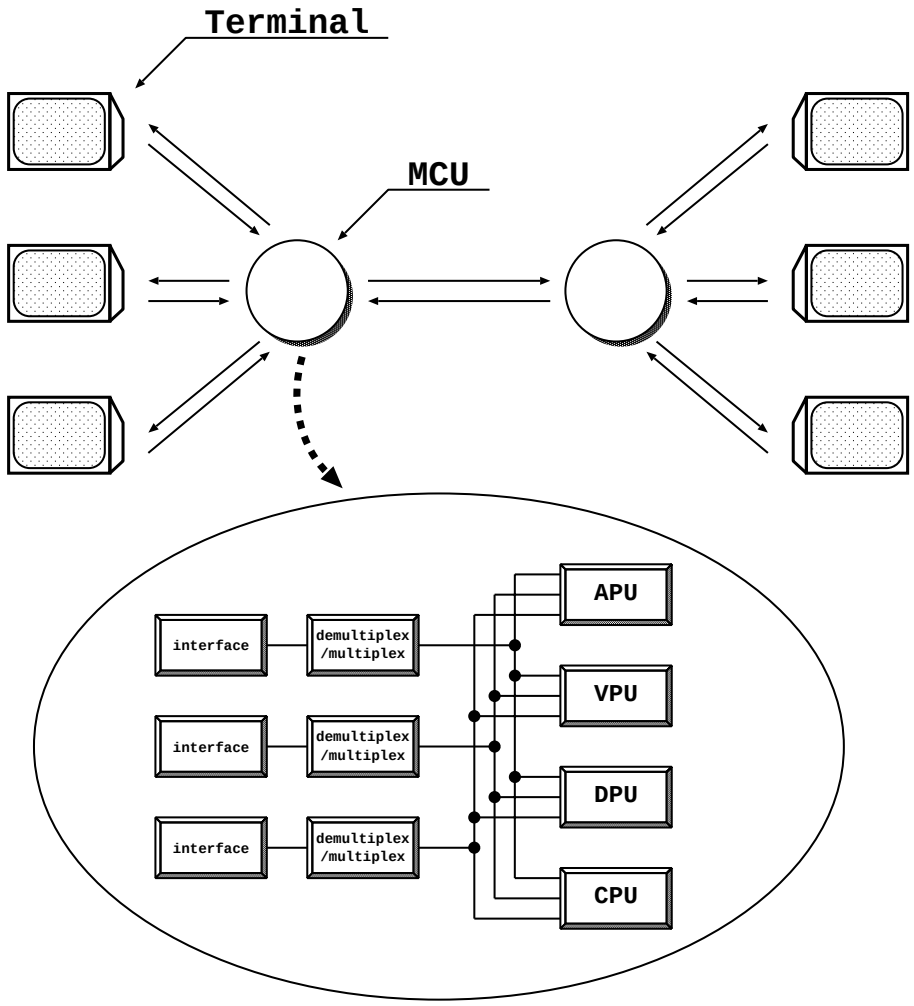

図 2.3: multipoint extension

2.3 家電系

家電機器で構成されるビデオネットワークでは、家電同士を接続しネットワーク化するための規格として提案された IEEE1394[5] が用いられる。

IEEE1394 は最大 400Mbps の伝送速度を持った高速シリアルバスである。63 台までの機器を一つのバス上に接続でき、機器を論理的なツリー状に接続されたノードとして管理を行う。

映像の伝送は帯域が保証されたアイソクロナス転送を用いて行われる。接続の形態としては、一対一の伝送を行うポイントツーポイントコネクション、一つの論理チャネルに対

して出力を行うブロードキャストアウトコネクション、一つの論理チャネルから入力を行うブロードキャストインコネクションがある。

これらの接続を確立した上で、$125\mu sec$ 毎のアイソクロナスサイクル中にデータを送信する。

バス上で扱うことができるフォーマットとしてはDVやMPEGがある。また、バスに接続された機器を制御するためのAV/C[1]と呼ばれるプロトコルも規定されている。AV/Cを用いることによってバス上の機器の制御を行うことができる。

このIEEE1394をベースとしてより高度なAV機器のネットワーク化を実現するための規格がHAVi[4]である。

HAViはAV/Cより高度な機器の操作を行うことができる。また、資源管理やストリーム制御を行う専門の管理ノードを置くことによって、ネットワーク内のリソースを集中的に管理することが可能となっている。

第 3章

相互接続のために必要となる機能

さまざまなビデオネットワークを相互に接続するためには各ネットワーク間で生じる相違を吸収する必要がある。

各ネットワーク間で異なる部分として、物理的なネットワークの違いの他に映像フォーマット、制御プロトコルがある。それぞれの部分において変換を行い、場合によっては不足している機能を補うゲートウェイを実装することによってビデオネットワークの相互接続が可能となる。

3.1 フォーマット変換

ビデオネットワークの映像フォーマットとして、計算機系のネットワークではMPEG、RealVideoなどが、電話系のネットワークでは図3.1に示すようなITU-T勧告で規定される符号化方式が、家電系ネットワークではDVやMPEGが用いられる。

ビデオネットワークの相互接続を行う際にはこれらのフォーマット間で変換を行う必要が生じる。

これらのフォーマット間で相互変換を行う際の問題点として、変換の処理に要する時間が大きくなるとリアルタイム性が損なわれることがあげられる。また、広帯域を利用するフォーマットほど変換処理を高速に行う必要がある。

3.2 制御プロトコル変換

ビデオネットワークの中にはストリームに対する制御を行うための制御プロトコルを持っているものがある。

system	based network	video codec	audio codec
H.310	B-ISDN	H.262	G.711
H.320	N-ISDN	H.261	G.711 G.722 G.728
H.321	B-ISDN	H.261	G.711 G.722 G.728
H.324	GSTN	H.261 H.263	G.722 G.723.1

図 3.1: codec of teleconferencing systems

制御プロトコルには大きく分けてセッションの制御を行うものとアプリケーションの制御を行うものとがある。セッションの制御では使用帯域の制御や使用する符号化方式の選択などが行われる。アプリケーションの制御ではストリームの再生、停止などの操作が行われる。

セッションの制御プロトコルとして、計算機系のネットワークでは主に RTP に含まれる RTCP などが、電話系のネットワークでは H.242 や H.245 が用いられる。家電系のネットワークでは帯域に余裕があることや、複数の符号化方式を利用することができる機器がないことなどからセッションの制御は行われない。

アプリケーションの制御プロトコルとして、計算機系のネットワークでは RTSP や DAVIC U-U プリミティブが、家電系のネットワークでは AV/C が用いられる。電話系のネットワークではストリームの再生や停止といった制御は行われず、接続した後は映像を流したままにするのが一般的であるため、このようなアプリケーションの制御は用いられない。

ビデオネットワークを接続する際にはこれらの制御プロトコル間の変換を行う必要があるが、各ネットワークの概念の相違のために必ずしもコマンドは一対一には対応しない。このような場合にはいくつかのコマンドを組み合わせて相当する動作を行うような変換を行うことや、またはその動作が不可能であることを指示元に伝える必要がある。

第 4章

相互接続の形態

ビデオネットワークを相互接続するためには各ネットワーク間の相違を吸収し、接続の仲立ちを行うゲートウェイを介して接続を行う。

接続の際の形態としてまず考えられるのは図 4.1 に示すようにネットワークの境界毎にゲートウェイを設け、個別に接続を行う方法である。この方法では各ネットワークは対等な立場で接続される。

もう一つの方法としては、図 4.2 に示すように中心となるネットワークを設け、そのネットワークをハブとして各ネットワークを接続する方法がある。

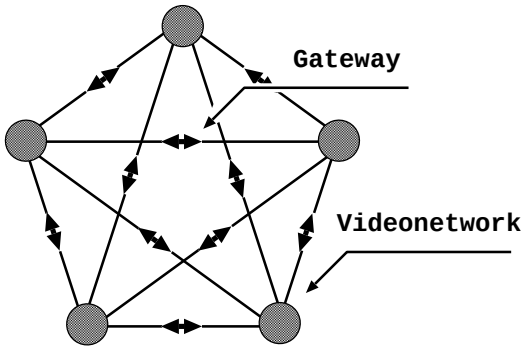

図 4.1: flat structure

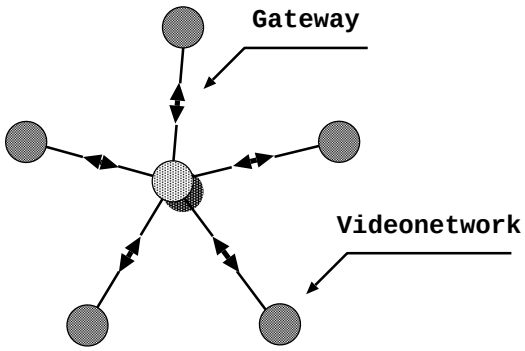

図 4.2: layered structure

4.1 個別に接続を行う方法

図 4.1 に示すように個別に接続を行う場合、比較的少数のビデオネットワークの接続を行う場合には構造が単純となり、コストを抑えることができる。

しかし、n 個のネットワークに対して $\frac{n(n-1)}{2}$ 個のゲートウェイが必要であり、ネットワークの増設を行う場合に追加しなければならないゲートウェイの数が二次関数的に増加していくなどの理由から、多数のビデオネットワークを接続する場合には適さない。

従って、この形態の接続は少数のネットワークを接続する場合にのみ有効である。

4.2 ハブとなるネットワークを介した接続を行う方法

図 4.2 に示すようにハブとなるネットワークを介して接続を行う場合には、必要となるゲートウェイの数が n 個のネットワークに対して n 個であり、一つのネットワークを増設する際には常に一つのゲートウェイを追加すれば良いため、拡張性に優れている。

また、ハブとなるネットワークでは周辺のネットワークの資源を抽象化することによって統一された概念で扱うことが可能となり、ネットワークの種類に依存しない集中管理を行うことが容易となる。これにより、利用者に対してネットワークの種類を意識させない操作環境を提供することが可能となる。

第 5章

ビデオネットワーク統合アーキテクチャ

本研究では、異種ビデオネットワークを相互に接続し、利用者がネットワークの違いを意識せずに直観的に操作することを可能とするシステムの構築を行う。

まずはじめに、接続されたネットワーク上の資源を統一した手法で扱う資源管理手法を検討する。

ビデオネットワーク上の機器は全て映像の入出力を行うことを目的としている。つまり、ビデオネットワーク上の機器の本質的な機能は映像の入出力であると考えることができる。

そこで、資源管理の単位を、ある時点において入力か出力の一方のストリームを一本だけ扱うことが可能であるノードとし、ネットワーク上の機器をノードの集合として扱うことにする。これにより、ある機器がどのネットワーク上に存在するかに依存せずに扱うことが可能となる。

このような資源管理の単位を用いてシステムを管理する場合には、ノードを単位として資源を管理するネットワークを中心に配置し、そのネットワークを介してビデオネットワークを相互に接続することで異種ビデオネットワーク間接続が可能となる。

機器の抽象化を行う機構は中心となるネットワークとの接続を行うゲートウェイ内に実装する。

このゲートウェイではビデオネットワークの相互接続に必要な機能のうち、制御プロトコルの変換も併せて行う。しかし、相互接続のために必要となるもう一方の機能である映像フォーマットの変換はここでは行わない。

この理由として、制御プロトコルは各ネットワーク間でそれほど大きな概念の差はなく、変換を行った際の損失はそれほど大きいとは考えられないが、映像フォーマットは変換することによる時間的、品質的な損失が大きいと考えられるためである。また、システ

ムの内部で映像フォーマットの変換を行う場合には新たなフォーマットを扱うためにはシステムの変更が必要となることも不利な要素となる。

そこで、映像フォーマット変換の機能は外部で実現することとする。このような変換を行う機構も他の機器と同様にノードの単位に分割することによって、通常の機器と何ら変わりなく扱うことが可能である。

本研究では以上のような構成で各種ビデオネットワークの接続を行い、利用者がネットワークの違いを意識せずに直観的に利用することを可能とするビデオネットワーク統合アーキテクチャVideo-network Integration Architecture(VIA) の提案を行う。

5.1 VIA の階層構造

ハブとなるネットワークを介した接続は中心のネットワーク、ゲートウェイ、周辺のネットワークの三つの要素から構成されている。

VIA ではこれらをそれぞれコアネットワーク、ゲートウェイ、フロントエンドネットワークと呼ぶ。

VIA の概略図を図 5.1 に示す。この例では三つのノードを持つフロントエンドネットワークと二つのノードを持つフロントエンドネットワーク、それにループバックパスのみを持つフロントエンドネットワークをコアネットワークを介して接続した場合を示している。

5.1.1 フロントエンドネットワーク

フロントエンドネットワークは接続される各種ビデオネットワークそのものであり、とくに機器の追加は必要ない。

フロントエンドネットワークの構成要素としてはエンドターミナルノードやループバックパスがある。

エンドターミナルノードは各種ビデオネットワーク上の機器を抽象化し、入力か出力いずれかのストリームを一本のみ扱うことが可能である仮想のノードとしたものである。VIA では全ての機器をこの論理的なノードの集合として扱う。

ループバックパスとはフォーマット変換を目的とするフロントエンドネットワークなどの場合に用いられる概念で、出力ノードから入力ノードまでのストリームの流れを示す仮想の経路である。

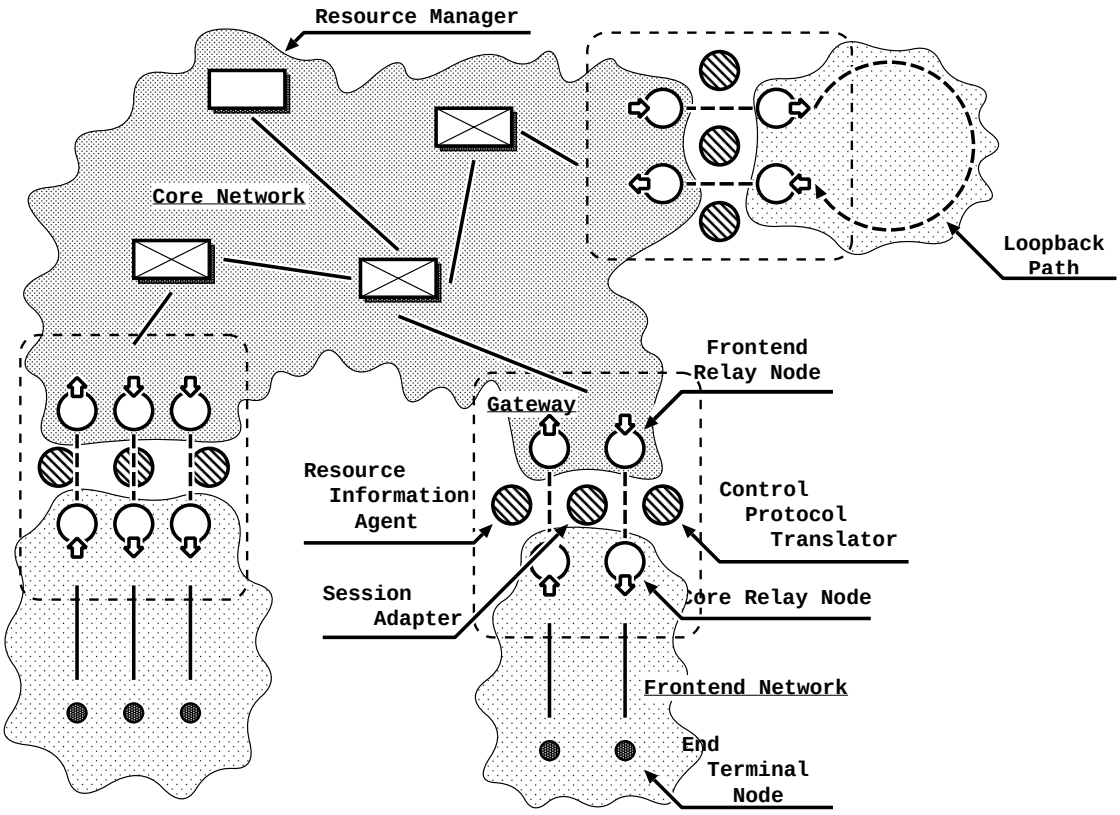

図 5.1: outline of VIA

フロントエンドネットワークの役割はこれらのエンドターミナルノードやループバックパスとゲートウェイ内のフロントエンドリレーノードとの間の接続を行うことである。

5.1.2 コアネットワーク

コアネットワークは接続された全てのネットワーク上の資源を管理し、接続の確立や開放を行う。

コアネットワークにはリソースマネージャが存在しており、フロントエンドネットワーク上の資源の管理を行っている。そのためにリソースマネージャ内にはノードの管理を行うデータベース、ゲートウェイの管理を行うデータベース、セッションの管理を行うデータベースがあり、ゲートウェイのリソースインフォメーションエージェントからの要求を受け、エントリの追加や削除が行われる。

5.1.3 ゲートウェイ

ゲートウェイはさまざまな形態を持つビデオネットワークそのものであるフロントエンドネットワークをコアネットワークに接続するための仲介を行う。

ゲートウェイはフロントエンドネットワーク側の接続の端点となるフロントエンドリレーノード、コアネットワーク側の接続の端点となるコアリレーノード、フロントエンドネットワーク内の資源情報をコアネットワークのリソースマネージャに通知するリソースインフォメーションエージェント、フロントエンドネットワーク内の制御コマンドを VIA の中間形式に変換するコントロールプロトコルトランスレータ、物理的なメディアやネットワークなどの相違を吸収するセッションアダプタなどで構成されている。

これらの要素はそれぞれがフロントエンドネットワークを抽象化し、コアネットワークに接続するための機能を担っている。

フロントエンドリレーノードはフロントエンドネットワーク側の接続の端点となり、エンドターミナルノードやループバックパスと接続される。

コアリレーノードはコアネットワーク側の接続の端点となり、他のコアリレーノードと接続される。これらのリレーノードは一対一に対応しており、一方のノードに入力されたストリームは透過的に他方のノードから出力される。ストリームはリレーノードからリレーノードに至る間にセッションアダプタを通過し、フロントエンドネットワークとコアネットワークの相違から生じる変換などの処理を受ける。

リソースインフォメーションエージェントはフロントエンドネットワーク内の資源情報を収集し、コアネットワークのリソースマネージャに通知する。この際に、フロントエンドネットワーク上の機器からエンドターミナルノードの集合へのマッピングが行われる。

コントロールプロトコルトランスレータはフロントエンドネットワーク内の制御プロトコルによる制御コマンドを受け、コアネットワーク内の制御プロトコルに変換し、コアネットワークに送り出す。ここでは同時にフロントエンドネットワーク側の制御コマンドに対して応答の代理も行う。

セッションアダプタはフロントエンドネットワークのストリームをコアネットワークに送り出す際に必要となる各種の変換を行う。セッションアダプタではネットワーク間の物理的な違いの吸収をはじめ、フロントエンドネットワーク内のストリームをコアネットワークのストリームとして送り出すためのさまざまな処理を行う。しかし、セッションアダプタにおいてストリームのフォーマットの変換など、ストリームに含まれるデータそのものを変換することはない。

5.2 VIA における識別子

VIA ではゲートウェイに対してコアネットワーク内で一意となるゲートウェイ ID を付与する。また、エンドターミナルノードに対してフロントエンドネットワーク内で一意となるエンドターミナルノード ID を付与する。これにより、ゲートウェイ ID とエンドターミナル ID の対を用いることによってエンドターミナルノードはシステム内において一意に決まる識別子を持つことになる。

ゲートウェイ ID はゲートウェイがコアネットワークに接続された際にコアネットワークのリソースマネージャによって付与される。エンドターミナルノード ID はフロントエンドネットワークに機器が接続された際に個々のエンドターミナルノードに対してゲートウェイのリソースインフォメーションエージェントが付与する。

VIA ではこのゲートウェイ ID とエンドターミナル ID の対を用いてノードの管理を行っている。

第 6 章

VIA における機器の取り扱い

VIA ではフロントエンドネットワーク上の機器を抽象化し、エンドターミナルノードの単位に分割し、管理を行う。

この章では実際の機器がどのように分割され、エンドターミナルノードとなるかについて述べる。

まずはじめにビデオネットワーク上の機器を入力ストリーム数と出力ストリーム数によって分類する。

ビデオネットワーク上の機器は表 6.1 に示すように分類される。これらの機器がどのようにエンドターミナルノードの単位に分割されるかを以下に示す。

表 6.1: devices of video-networks

入力	出力	例
1	0	カメラ、ビデオプレイヤー
0	1	モニタ、ビデオレコーダ
1	1	テレコンファレンス端末、フォーマット変換装置
1	1 or more	ストリーム分配装置
1 or more	1	マルチプレクサ

6.1 入力のみを扱う機器

入力のみを扱う機器にはカメラや再生時の VCR などが含まれる。

これらの機器が接続されると図 6.1 に示すように一つの入力エンドターミナルノードがアタッチされる。

6.2 出力のみを扱う機器

出力をのみ扱う機器にはモニタや録画時の VCR などが含まれる。

これらの機器が接続されると図 6.2 に示すように一つの出力エンドターミナルノードがアタッチされる。

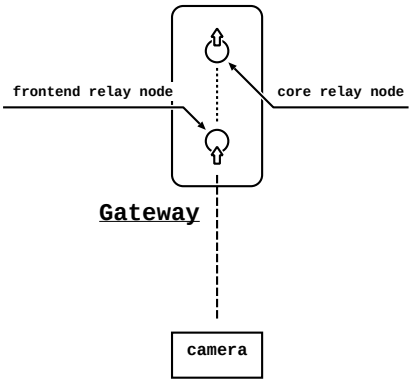

図 6.1: logical model of camera

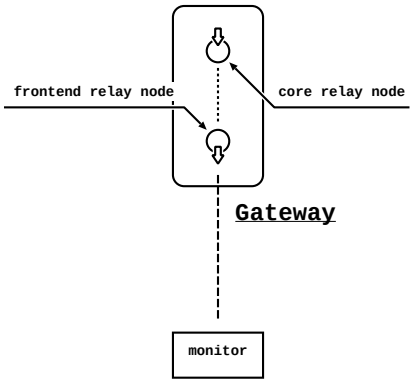

図 6.2: logical model of monitor

6.3 一入力一出力を扱う機器

一入力一出力を扱う機器にはテレコンファレンス端末、フォーマット変換装置などが含まれる。

これらの機器が接続されると一つの入力エンドターミナルノードと一つの出力エンドターミナルノードがアタッチされる。

テレコンファレンス端末の場合には図 6.3 に示すようにエンドターミナルノード間に因果関係はなく、独立したエンドターミナルノードとして動作する。

フォーマット変換装置の場合には図 6.4 に示すように 2 つのエンドターミナルノードの間にはループバックパスが存在する。これは 2 つのエンドターミナルノードが実際の機器の入力の端子と出力の端子に相当し、一方のエンドターミナルノードから入力したストリームが暗黙のうちに他方のエンドターミナルノードから出力されることを示している。

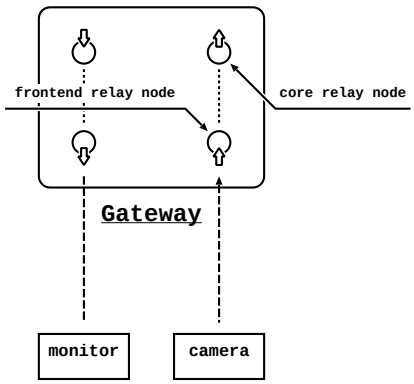

図 6.3: logical model of teleconference terminal

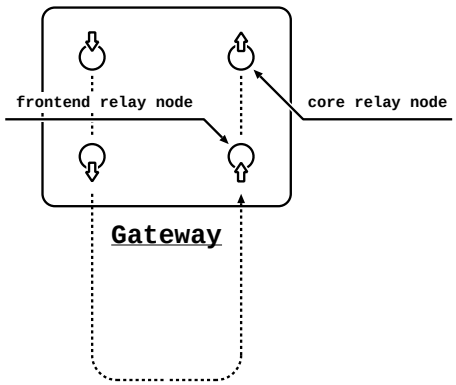

図 6.4: logical model of transcoder

6.4 一入力多出力を扱う機器

一入力多出力を扱う機器にはストリーム分配装置が含まれる。これによって疑似的なマルチキャスト配信を実現することが可能となる。

このような機器の場合には図 6.5 に示すように一つの入力エンドターミナルノードと複数の出力エンドターミナルノードとして動作する。

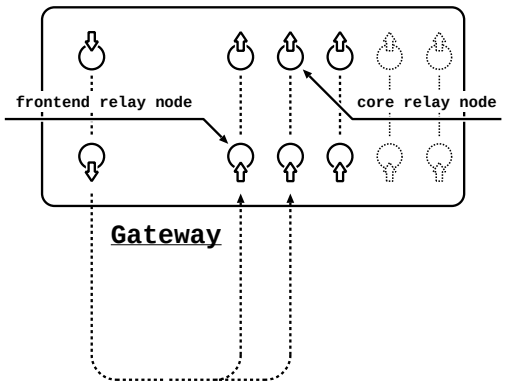

図 6.5: logical model of distributor

出力ストリーム数が m 個で固定の場合にははじめから一つの入力エンドターミナルノードと m 個の出力エンドターミナルノードがアタッチされる。

出力ストリーム数が上限 n 個の可変である場合には、はじめに一つの入力エンドター

ミナルノードと一つの出力エンドターミナルノードがアタッチされる。そして一つの出力エンドターミナルノードに対して接続が確立され、ビジー状態になるとさらに一つの出力エンドターミナルノードがアタッチされる。このように、出力エンドターミナルノードに対して接続が確立される毎に一つの出力エンドターミナルノードがアタッチされ、常に一つの出力エンドターミナルノードが待ち状態となる。この動作が n 個の出力エンドターミナルノードがビジー状態になるまで繰り返される。

反対に出力エンドターミナルノードに対する接続が開放されるときには待ち状態の出力エンドターミナルノードを一つ残し、デタッチされる。これを繰り返し、最終的には元の一つの入力エンドターミナルノードと一つの出力エンドターミナルノードがアタッチされている状態に戻る。

6.5 多入力一出力を扱う機器

多入力一出力を扱う機器にはマルチプレクサが含まれる。

このような機器の場合には図 6.6 に示すように複数の入力エンドターミナルノードと一つの出力エンドターミナルノードとして動作する。

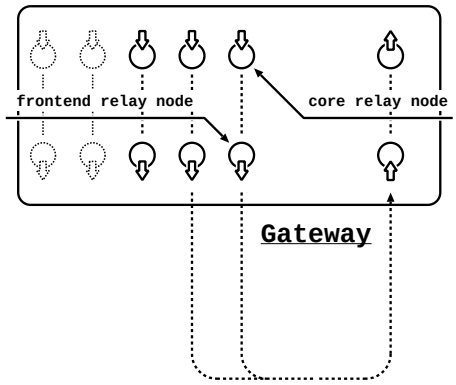

図 6.6: logical model of multiplexer

入力ストリーム数が m 個で固定の場合にははじめから m 個の入力エンドターミナルノードと一つの出力エンドターミナルノードがアタッチされる。

入力ストリーム数が上限 n 個の可変である場合には、はじめに一つの入力エンドターミナルノードと一つの出力エンドターミナルノードがアタッチされる。そして一つの入力エンドターミナルノードに対して接続が確立され、ビジー状態になるとさらに一つの入力

エンドターミナルノードがアタッチされる。このように、入力エンドターミナルノードに対して接続が確立される毎に一つの入力エンドターミナルノードがアタッチされ、常に一つの入力エンドターミナルノードが待ち状態となる。この動作が n 個の入力エンドターミナルノードがビジー状態になるまで繰り返される。

反対に入力エンドターミナルノードに対する接続が開放されるときには待ち状態の入力エンドターミナルノードを一つ残し、デタッチされる。これを繰り返し、最終的には元の一つの入力エンドターミナルノードと一つの出力エンドターミナルノードがアタッチされている状態に戻る。

第 7章

実際の動作

ゲートウェイやフロントエンドネットワークの機器が VIA の一部として稼働しはじめるためにはコアネットワークやフロントエンドネットワークにアタッチされる必要がある。アタッチされているエンドターミナルノードに対して接続が要求されると接続が確立され、機器間で映像の送受信が開始される。

エンドターミナルノードがアタッチされてから実際に映像の送受信が行われるまでのメッセージの流れを図 7.1、図 7.2 に示す。

これらの図の動作では、まずリソースインフォメーションエージェントがエンドターミナルノードのアタッチをリソースマネージャに通知①する。これに対して ack②が返される。これにより、エンドターミナルノードがリソースマネージャに登録される。

次に、オペレーションターミナルがシステム内のエンドターミナル情報をリソースマネージャに問い合わせる③。これに対し、システム内のエンドターミナルノードの情報④が返され、オペレーションターミナルはこの情報を利用者に提示する。

利用者が接続を指示すると、オペレーションターミナルはリソースマネージャに接続の要求⑤を行う。

接続の要求に対し、リソースマネージャは送信側のセッションアダプタに送信の要求⑥を行う。送信側のセッションアダプタはフロントエンドネットワーク内の接続を行い、次にコアネットワーク内のシグナリングを行う。

ack⑦が返ると次に受信側のセッションアダプタに受信の要求⑧を行う。

受信側のセッションアダプタはフロントエンドネットワークの接続を行う。

ack⑨が返ってくるとセッションがデータベースに登録される。これにより接続は確立され、セッションアダプタ間でストリームが送受信され、コントロールプロトコルトランスレータ間で制御コマンドの送受信が行われる。

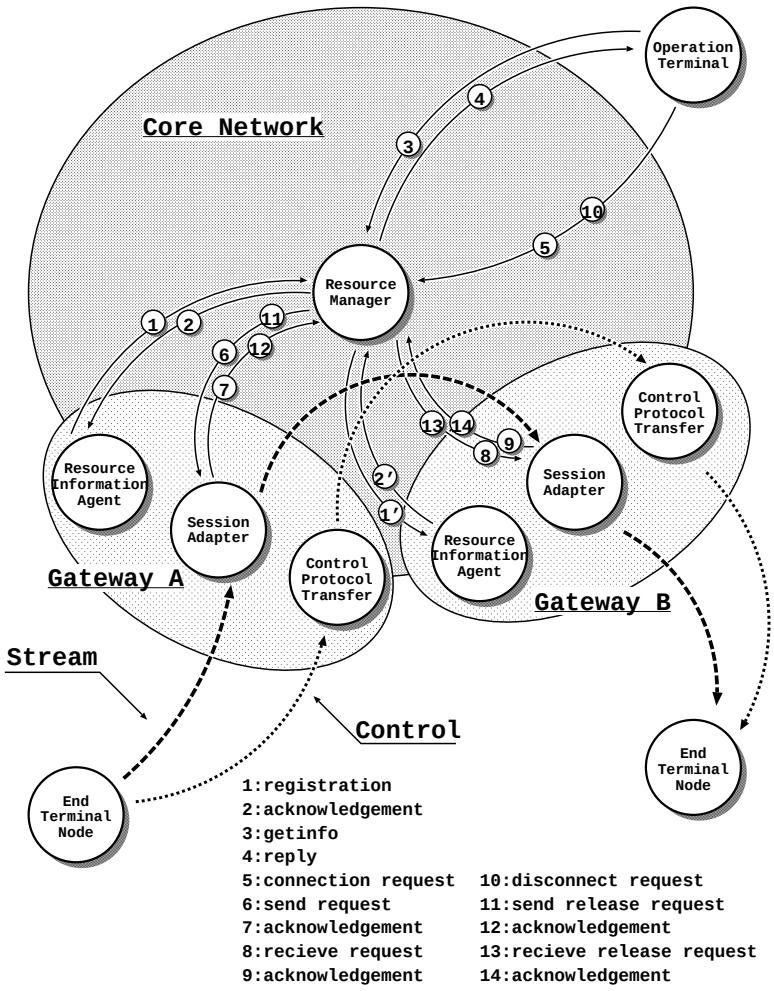

図 7.1: message flow

リソースマネージャはオペレーションターミナルから接続の開放⑩を要求されるとセッションアダプタに対して接続の開放を要求⑪、⑬する。セッションアダプタから ack⑫、⑭が返ってくると該当するセッションをデータベースから削除する。

以下では、ゲートウェイのアタッチ、ゲートウェイのデタッチ、エンドターミナルノードのアタッチ、エンドターミナルノードのデタッチ、接続の確立、接続の開放の個々の動作について述べる。

7.1 ゲートウェイのアタッチ

ゲートウェイがコアネットワークに接続されるとアタッチの処理が行われる。

ゲートウェイはコアネットワークに接続されるとリソースマネージャに対してアタッチの要求を行う。リソースマネージャはコアネットワーク内で一意となるようなゲートウェイ ID を決定し、ゲートウェイに通知するとともにゲートウェイを管理しているデータベースへの登録を行う。この処理によってゲートウェイは VIA の一部として稼働することができるようになる。

7.2 ゲートウェイのデタッチ

ゲートウェイがコアネットワークから離脱する場合にはデタッチの処理が行われる。

ゲートウェイはリソースマネージャに対してデタッチの要求を行う。リソースマネージャは該当するゲートウェイを介した接続を全て開放した後、ゲートウェイをデータベースから削除し、肯定応答をゲートウェイに返す。この処理によってゲートウェイは VIA から切り離される。

7.3 エンドターミナルノードのアタッチ

フロントエンドネットワークに機器が接続されるとアタッチの処理が行われる。

機器の接続を感知したゲートウェイ内のリソースインフォメーションエージェントは、機器が論理的に何個のエンドターミナルノードとみなすことができるかを判断する。そして、それぞれにフロントエンドネットワーク内で一意となるエンドターミナルノード ID を付与し、図 7.1 の①に示すようにコアネットワーク内のリソースマネージャに対して通知を行う。これを受けたリソースインフォメーションマネージャではノードの管理を行うデータベースへの登録を行い、図 7.1 の②で示すように ack を返す。

この処理により、エンドターミナルノードは接続の端点となることができるようになる。

7.4 エンドターミナルノードのデタッチ

フロントエンドネットワークから機器が取り外されるとデタッチの処理が行われる。

機器が取り外されたことを感知したリソースインフォメーションエージェントはコアネットワークのリソースマネージャに対してデタッチの要求を行う。リソースマネージャはこの要求を受け、該当するエンドターミナルノードを接続の端点とする接続を全て開放し、エンドターミナルノードをデータベースから削除する。

この処理によってゲートウェイは VIA から切り離される。

7.5　接続の確立

図 7.1 の⑤に示すように操作端末からリソースマネージャに対して接続が要求されると、接続の確立処理が行われる。

リソースマネージャは接続の端点となるエンドターミナルノードに対して接続が確立していないことを確認し、図 7.1 の⑥、⑧に示すように接続の端点となるエンドターミナルノードが属しているフロントエンドネットワークが属する双方のゲートウェイのセッションアダプタに対して接続のセットアップの指示を行う。

接続のセットアップが成功すると、セッションを管理するデータベースに登録を行い、図 7.1 の⑦、⑨に示すように ack を返す。

7.6　接続の開放

操作端末からリソースマネージャに対して接続の開放⑩が要求されると、接続の開放処理が行われる。

リソースマネージャはゲートウェイのセッションアダプタに対して接続の開放⑪、⑬を要求する。ack⑫、⑭が返ると、セッションの管理を行うデータベースからセッションの削除を行い、操作端末に肯定応答を返す。

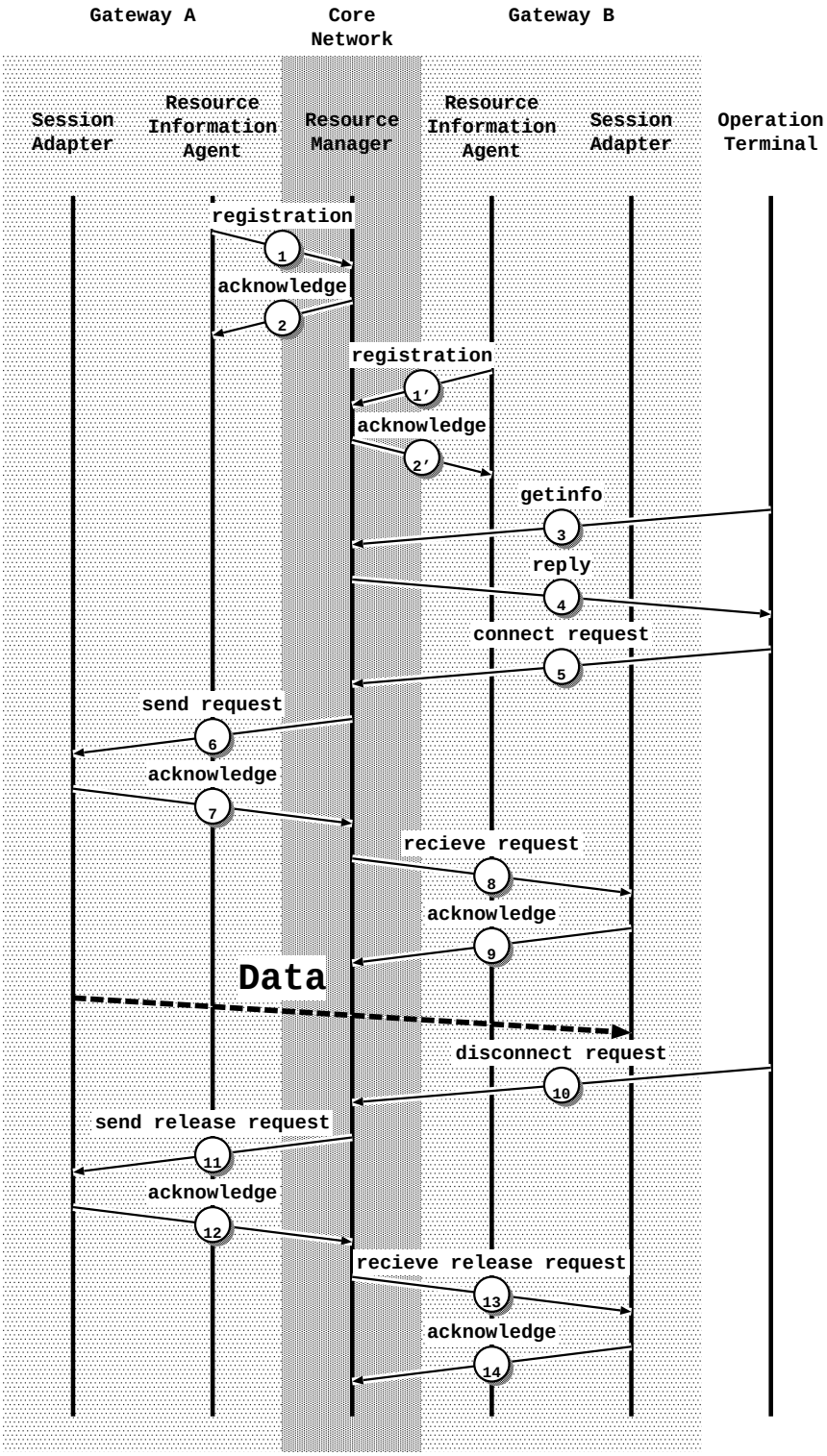

図 7.2: time flow

第 8章

JAIST VideoLANをベースとした実装

本章ではJAIST VideoLANをベースとしたVIAの構築を行う。これによって得られた知見は他の既存のビデオネットワークをベースとした実装の際にも役立つものと思われる。

8.1 JAIST VideoLAN

JAIST VideoLANシステムはATMとIEEE1394を用い、リアルタイムでDVの高品質な映像を送受信するシステムである。

JAIST VideoLANシステムは図8.1に示すように以下のコンポーネントで構成されている。

ターミナルシステム ATMとIEEE1394間のブリッジ

IEEE1394上のDVフレームを分割しATMセルのペイロードとして送り出す。また、機器の接続を感知しリソースマネージメントエージェントに通知する。

リソースマネージメントエージェント システム内の資源管理主体

ATM網の内部に位置し、システム内の全ての機器の管理、コネクションの確立、開放を行う。

内部にいくつかのデータベースのテーブルを持ち、システム内の全てのノードやコネクションなどのリソースや機器情報を管理している。また、接続の確立や開放の要求を受け、ターミナルシステムに対して指示を行う。

オペレーションターミナル 利用者がシステムを操作するための端末

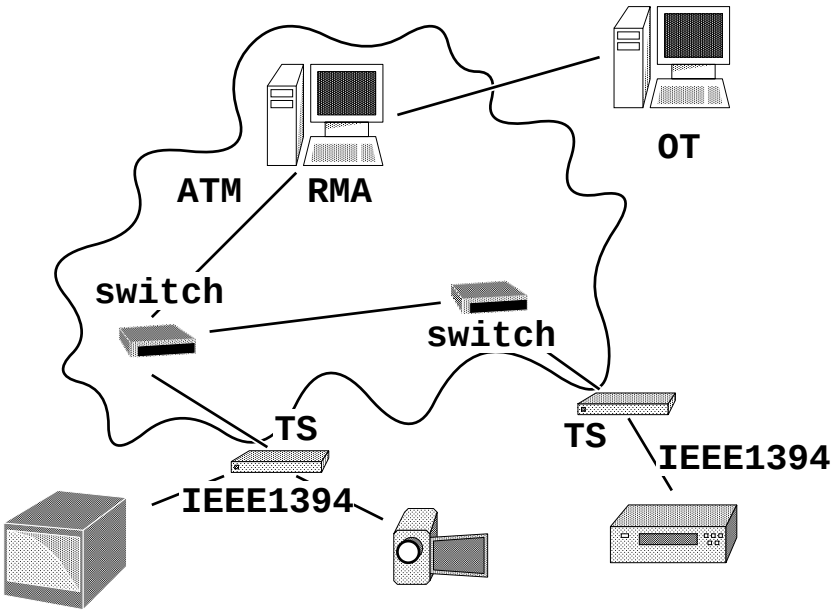

図 8.1: outline of VideoLAN

利用者からの指示を受け、リソースマネージメントエージェントに対して要求を行う。

8.2 追加が必要となるコンポーネントの検討

このような構成を持つ JAIST VideoLAN をベースに VIA を構築する。できるだけ既存のコンポーネントを流用したまま構築を行うために VIA と JAIST VideoLAN の両システムを検討し、相違が生じる点についてのみコンポーネントを追加する。

第 3 章でも検討したとおり、ビデオネットワーク間において概念の相違が生じる部分としては、制御プロトコル、映像フォーマットがある。また、今回用いる JAIST VideoLAN では独自の手法による資源管理を行っているため、資源管理についても相違が生じる。

8.2.1 資源管理

JAIST VideoLAN では ATM 内のリソースマネージメントエージェントにおいて全てのノードやコネクションなどの管理を行っている。また、機器とそれに対応する機器の名称や設置箇所などもデータベースを用いて管理を行っている。それらの機器の管理には IEEE1394 の規格で規定されている機器の固有 ID を利用している。

したがって、JAIST VideoLAN では末端のネットワークは IEEE1394 であると仮定されており、また中央のリソースマネージメントエージェントにおいて末端のネットワークの機器の固有情報を管理していることになる。

VIA ではフロントエンドネットワークの機器を抽象化し、個々の機器の特性を隠蔽した運用を行うことを目的の一つとする枠組である。その観点において、JAIST VideoLAN のリソースマネージメントエージェントは末端のネットワークの種類に依存しており、また、末端の機器の固有情報を保持しており、VIA の構成に馴染まない。

従って、あらたな資源管理機構としてのリソースマネージャをコアネットワーク内に新たに構築する必要がある。その際に、従来のリソースマネージメントエージェントが持っていたデータベースのテーブルのうち、コネクションを管理するテーブルは機器を識別する ID の見直しを行った上でリソースマネージャに配置する。また、接続された機器を管理していたテーブルは管理の単位をエンドターミナルノードとした上で、リソースマネージャにおいて管理を行う。リソースマネージメントエージェントが持っていたもう一つのテーブルである機器の固有情報のテーブルはゲートウェイのリソースインフォメーションエージェントに移管し、情報の利用はフロントエンドネットワーク内で閉じたものとする。ただし、利用者の便宜を図るため、機器の固有情報をゲートウェイ内のリソースインフォメーションエージェントからコアネットワークのリソースマネージャに通知する。しかし、そのような機器の固有情報に依存した動作は一切行われない。

これによって、IEEE1394 に限らず、さまざまなビデオネットワーク上の機器のネットワークに依存しない管理が可能となる。

8.2.2 制御プロトコル変換

現状の VideoLAN は内部にストリーム制御のための機能を持っていない。また、IEEE1394 上の制御プロトコルである AV/C の検出を行う機能も持っていない。

しかし、さまざまなビデオネットワークを相互に接続した場合にあるネットワーク上から他のネットワーク上の機器を制御することが可能であることが望ましい。

そこで、VideoLAN をベースとして VIA を構築する際には、VideoLAN のシステムに

対しIEEE1394上のAV/Cを検出する機構を追加する必要がある。また、検出したAV/Cを用いてさまざまな他のフロントエンドネットワーク上の機器を制御し、また他のネットワークからIEEE1394上の機器を制御するために各種制御プロトコルを抽象化した新たなプロトコルを設計し、プロトコル間の相互変換を行うコンポーネントの追加を行う。

新たに設計するプロトコルは各種ビデオネットワークとの相互変換を考え、各種ビデオネットワークの制御体系を包括するものになることができることが望ましい。しかし、実際には全てのプロトコルを内包することは難しいため、各プロトコルの共通する概念を抽出し、設計することになる。

実際にストリームの制御のためにビデオネットワークに要求される機能はそれほど多くはないため、そのように設計したプロトコルは十分な機能を持つものとなるはずである。

ビデオネットワーク制御プロトコル

ストリームの制御のために要求される機能として、大まかなものとして再生、録画、一時停止、停止、早送り、巻戻しなどが挙げられる。

各種ビデオネットワークのストリーム制御プロトコルをもとにこれらの機能を有するビデオネットワーク制御プロトコル (VNCP) を設計する。

代表的なビデオネットワーク制御プロトコルとしてAV/C、RTSP、DSM-CC UUプリミティブを考える。

VNCPのコマンドを表8.1に示す。図中の※の部分は一対一に対応するコマンドはないものの、複数のコマンドの組合せによって実現できることを示す。ただし、これらのコマンドを実現するためにはゲートウェイの内部にネットワークの状態やタイムコードを保持する状態変数を設ける必要がある。

また、これらの制御体系では巻戻しや早送りのために時間が必要であることは考慮されていないが、家電機器からAV/Cを用いて制御を行う事を考えると人間の感覚に一致させるためにウェイトが必要になるものと思われる。

8.2.3 フォーマット変換

各種ビデオネットワーク間で異なる映像のフォーマットに対応するためにはフォーマットの変換の機構を持つ必要がある。

システムに変換の機構を持たせるためには網の内部に変換の機構を持つ必要がある。しかし、このような構成をとる場合にはシステムが変換を行う機構の特性を知っており、それに応じた動作を行わせることになる。

表 8.1: VNCP

AV/C		VNCP	RTSP	DSM-CC
forward play	c3 31	PLAY(FORWARD, SLOWEST)	PLAY Scale: +SLOWEST	—
	≀			
	c3 38	PLAY(FORWARD, NORMAL)	PLAY Scale: +NORMAL	resume()
	≀			
	c3 3f	PLAY(FORWARD, FASTEST)	PLAY Scale: +FASTEST	—
reverse play	c3 41	PLAY(REVERSE, SLOWEST)	PLAY Scale: -SLOWEST	—
	≀			
	c3 48	PLAY(REVERSE, NORMAL)	PLAY Scale: -NORMAL	—
	≀			
	c3 4f	PLAY(REVERSE, FASTEST)	PLAY Scale: -FASTEST	—
record	c2 75	RECORD()	RECORD	—
forward pause	c3 7d	FWD_PAUSE()	PAUSE	pause()
reverse pause	c3 6d	REV_PAUSE()	PAUSE	pause()
record pause	c2 7d	REC_PAUSE()	PAUSE	pause()
stop	c4 60	STOP()	PAUSE	pause()
forward	c4 75	FORWARD()	※	※
rewind	c4 65	REWIND()	※	※
time code	51 71 ff ff ff ff	TIMECODE()	GET_PARAMETER npt	status

VIA ではコアネットワークは末端の機器の固有情報に依存しないような設計になっている。そこで、フォーマットの変換の機構も網内で実現するのではなく、フォーマット変換を行うフロントエンドネットワークを実装することで実現する。これにより、フォーマット変換を行う機構の詳細は隠蔽され、他のフロントエンドネットワークと同様に扱うことが可能となる。

また、フォーマット変換の機構の詳細に依存せずに動作するため、変換の機構の実態がどのようなものであっても感知しない。つまり、フォーマットの変換がソフトウェア処理で実現されているかハードウェア処理で実現されているかなどを気にせずに動作させることができることになる。このため、既存のフォーマット変換装置を利用してフォーマット変換を行うフロントエンドネットワークを実装することが容易となる。

8.3 実装の形態

全節までで述べたとおり、VideoLAN をベースとして VIA を構築するためには、資源管理、制御プロトコルの検出、制御プロトコルの変換を行う各コンポーネントの追加が必要となる。

コアネットワーク内で資源管理を行うリソースマネージャは従来の VideoLAN のリソースマネージメントエージェントを代替するものであるため、コアネットワーク内に位置する必要がある。

フロントエンドネットワーク内の資源管理を行うリソースインフォメーションエージェントと制御プロトコルの検出を行い、VNCP との間で相互変換を行うコントロールプロトコルトランスレータは VIA のゲートウェイを構成するコンポーネントであり、フロントエンドネットワークとコアネットワークの両方に接する必要があるためにターミナルシステムと同様にネットワークの境界に配置する。これらのコンポーネントをプロキシエージェントと呼ぶ。プロキシエージェントはセッションアダプタとして機能する VideoLAN のターミナルシステムと協調し、VIA のゲートウェイとして動作する。

VideoLAN をベースとしてこれらのコンポーネントを追加し、VIA を構築したときの全体の構成を図 8.2 に示す。

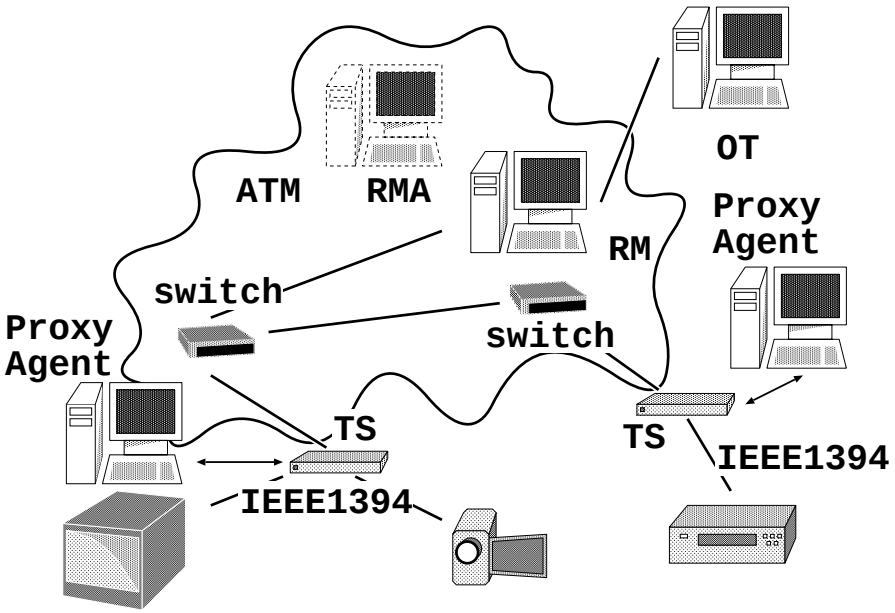

図 8.2: outline of VIA

第 9章

各コンポーネントの動作

前章で述べた通り、いくつかのコンポーネントを追加することによってJAIST VideoLANシステムをVIAとして動作させることができる。

ここでは各コンポーネントを実装する際の仕様について述べる。

9.1 リソースマネージャ

リソースマネージャは大きく分けると3つの部分から構成されている。

9.1.1 リソースインフォメーションエージェントとの対話を行うモジュール

一つはゲートウェイ内のリソースインフォメーションエージェントとの対話を行い、エンドターミナルノードのアタッチやデタッチの処理を行うモジュールである。

ゲートウェイのアタッチの通知を受けるとゲートウェイに対してコアネットワーク内において一意となるゲートウェイIDを割り当て、データベースに登録し、肯定通知を通知する。

アタッチの際に受け取るゲートウェイやエンドターミナルノードに関しての付加情報はASN.1でシリアライズして送られる。シリアライズされた情報のデコードを行うモジュールの生成には`snacc(ftp://ftp.cs.ubc.ca/pub/local/src/snacc/)`を用いた。

ゲートウェイのアタッチに用いられる情報は図9.2に示す通りである。

図9.1に示す情報の各フィールドの内容は以下の通りである。

ip-addr ゲートウェイのIPアドレス

```
Gateway
DEFINITIONS ::=
BEGIN
Gateway ::= SET {
    ip-addr     [0] IpV4Addr,
    atm-addr    [1] AtmAddr,
    name        [2] VisibleString,

AtmAddr ::= IA5String (
    FROM (
        "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
        "8" | "9" | "a" | "b" | "c" | "d" | "e" | "f" ) |
    SIZE ( 20 ) )

IpV4Addr ::= IA5String (
    FROM (
        "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
        "8" | "9" | "a" | "b" | "c" | "d" | "e" | "f" ) |
    SIZE ( 8 ) )
END
```

図 9.1: definition of module gateway

ゲートウェイがリソースマネージャに登録された後のリソースマネージャとの通信はこのアドレスを用いて行われる。

atm-addr ゲートウェイの ATM アドレス

ゲートウェイの ATM アドレスを示す。

このフィールドはシグナリングを行う際に用いるものであり、コアネットワークが ATM ではないときには他の値を表すか、もしくはこのフィールドが必要ではなくなる。

name ゲートウェイに付与された名前

ゲートウェイの名前を示す文字列であり、オペレーションターミナルのユーザインタフェースで利用される。

エンドターミナルノードのアタッチに用いられる情報は図 9.2 に示す通りである。
図 9.2 に示す情報の各フィールドの内容は以下の通りである。

gateway-id ゲートウェイの ID

エンドターミナルノードが所属するゲートウェイの ID を示す。

```
Terminal
DEFINITIONS ::=
BEGIN
Terminal ::= SET {
    gateway-id  [0] IdNumber,
    node-id     [1] IdNumber,
    format-type INTEGER,
    is-output   BOOLEAN,
    name        [3] VisibleString,
    loopback-id [4] IdNumber,
option-id   [5] VisibleString,
    icon-image  OCTET STRING }

IdNumber ::= IA5String (
    FROM (
        "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
        "8" | "9" | "a" | "b" | "c" | "d" | "e" | "f" ) |
    SIZE ( 16 ) )
END
```

図 9.2: definition of module terminal

ゲートウェイのアタッチ時にリソースマネージャから付与されたIDで、コアネットワーク内のゲートウェイを一意に識別することができる。

node-id エンドターミナルノードのID

ゲートウェイ以下のフロントエンドネットワーク内のエンドターミナルノードに付与されたIDで、ゲートウェイIDと共に用いることでシステム内のエンドターミナルノードを一意に識別することができる。

format-type エンドターミナルノードが扱うことができるストリームのフォーマット

各エンドターミナルノードが扱うことのできるストリームのフォーマットを示す。このフォーマットのフィールドが一致するエンドターミナルノード間でのみ接続が行われる。

is-output エンドターミナルノードのストリームの方向

エンドターミナルノードが出力を行うものであるかどうかを示す。このフィールドはブール値をとり、trueで出力を、falseで入力を表す。この値が異なるエンドターミナルノード間でのみ接続が行われる。

name エンドターミナルノードに付与された名前

エンドターミナルノードの名前を示す文字列であり、オペレーションターミナルのユーザインタフェースで利用される。オペレーションターミナルのユーザインタフェースで利用されるだけで、システムの動作には影響を与えない。したがって、このフィールドはオプションである。

loopback-id ループバック ID

フォーマット変換を行うフロントエンドノードの場合などに対になるノードの ID を示す。この値はオペレーションターミナルのユーザインタフェースで利用される。オペレーションターミナルのユーザインタフェースで利用されるだけで、システムの動作には影響を与えない。したがって、このフィールドはオプションである。

option-id 個々の機器を認識するための ID

フロントエンドネットワークの種類によっては、機器を個別に認識することが可能な ID を機器に付与することがある。例えば、IEEE1394 のインタフェースを持つ機器などはノードユニーク ID と呼ばれる 64 ビット長の ID を持っている。このような値をこのフィールドに格納しておくことで別のゲートウェイに接続した場合でもここの機器を識別することが可能となる。この値はオペレーションターミナルのユーザインタフェースで利用される。オペレーションターミナルのユーザインタフェースで利用されるだけで、システムの動作には影響を与えない。したがって、このフィールドはオプションである。

icon-image ユーザインタフェースで提示されるアイコンイメージ

このフィールドにはユーザインタフェースを通して利用者に提示されるアイコンのイメージが格納されている。このイメージを用いることによって機器に応じたアイコンを利用したユーザインタフェースを利用者に提示することができ、より直観的なインタフェースの実現が可能となる。この値はオペレーションターミナルのユーザインタフェースで利用される。オペレーションターミナルのユーザインタフェースで利用されるだけで、システムの動作には影響を与えない。したがって、このフィールドはオプションである。

リソースインフォメーションエージェントとリソースマネージャの間の情報の通信は以下のプロトコルを用いて行われる。

このモジュールが受け付けるコマンドには`attach_gateway`、`detach_gateway`、`attach_node`、`detach_node` がある。

attach_gateway ゲートウェイのアタッチ

書式

```
attach_gateway serialized_data_block
```

詳細

図 9.1 に示したデータを ASN.1 を用いてシリアライズしたものを引数としてとり、データベースへの登録を行う。

serialized_data_block 図 9.1 に示したデータを ASN.1 を用いてシリアライズしたもの

detach_gateway ゲートウェイのデタッチ

書式

```
detach_gateway gid
```

詳細

デタッチするゲートウェイを表すgid を引数としてとり、データベースからの抹消を行う。

gid デタッチするゲートウェイの ID

attach_node エンドターミナルノードのアタッチ

書式

```
attach_node serialized_data_block
```

詳細

図 9.2 に示したデータを ASN.1 を用いてシリアライズしたものを引数としてとり、データベースへの登録を行う。

serialized_data_block 図 9.2 に示したデータを ASN.1 を用いてシリアライズしたもの

detach_node エンドターミナルノードのデタッチ

書式

`detach_node gid tid`

詳細

デタッチするエンドターミナルノードを表す`tid`と`gid`の対を引数としてとり、データベースからの抹消を行う。

gid デタッチするエンドターミナルノードが所属するフロントエンドネットワークのゲートウェイの ID

tid デタッチするエンドターミナルノードの ID

これらのコマンドによって登録された情報はデータベース内のゲートウェイインフォメーションテーブルとエンドターミナルノードインフォメーションテーブルで管理される。

これらのテーブルの内容を表 9.1、表 9.2 に示す。

表 9.1: gateway information table

Gateway ID	IP Address	ATM Address	Name
ゲートウェイ ID	ゲートウェイの IP アドレス	ATM アドレス	ゲートウェイの名称

表 9.2: end terminal node information table

Gateway ID	End Terminal Node ID	Format Type	Id Output
ゲートウェイ ID	エンドターミナルノードの ID	エンドターミナルノードが扱うことができるフォーマット	ストリームの方向

Name	Loopback ID	Option ID	Icon Image
エンドターミナルノードの名称	ループバックパスの接続先の ID	機器固有の ID	アイコンイメージ

9.1.2 オペレーションターミナルとの対話を行うモジュール

次にオペレーションターミナルとの対話を行い、エンドターミナルノードやセッションの情報の提供や接続の確立、開放を行うモジュールである。

オペレーションターミナルから情報の提供の要求を受けるとデータベース上の該当する情報を返す。

接続の確立の要求を受けると接続の両端のエンドターミナルノードが待ち状態であることを確認した後、両端のゲートウェイに対して接続を指示する。

接続の開放の要求を受けると接続の両端のゲートウェイに対して接続の開放を指示する。

リソースマネージャとオペレーションターミナルの間の通信は以下のプロトコルを用いて行われる。

このモジュールが受け付けるコマンドには、get_info、connect、disconnect がある。

get_info エンドターミナルの情報の取得

書式

```
get_info [[key value] ...]
```

詳細

エンドターミナルノードの情報を取得する。

以下のキーに対して値を指定することで選択的な取得が可能となる。

gid ゲートウェイの ID

tid エンドターミナルノードの ID

gname ゲートウェイの名称

tname エンドターミナルノードの名称

format フォーマット

lid ループバックパスの接続先の ID

oid オプション ID

load_image エンドターミナルのアイコンイメージの取得

書式

```
load_image gid tid
```

詳細

ゲートウェイ ID とエンドターミナルノード ID の対で指定されたエンドターミナルノードについて登録されたアイコンのイメージを取得する。

gid ゲートウェイの ID

tid エンドターミナルノードの ID

connect 接続の確立

書式

```
connect src_gid src_tid dst_gid dst_tid
```

詳細

接続元と接続先のゲートウェイ ID、エンドターミナルノード ID を受け取り、接続を確立するためにゲートウェイのセッションアダプタに要求を行った後、データベースへの登録を行う。

src_gid 接続元のゲートウェイ ID

src_tid 接続元のエンドターミナルノード ID

dst_gid 接続先のゲートウェイ ID

dst_tid 接続先のエンドターミナルノード ID

disconnect 接続の開放

書式

```
disconnect src_gid src_tid [dst_gid dst_tid]
```

詳細

接続元と接続先のゲートウェイ ID、エンドターミナルノード ID を受け取り、接続を開放するためにゲートウェイのセッションアダプタに要求を行った後、データベースからの抹消を行う。

src_gid 接続元のゲートウェイ ID

src_tid 接続元のエンドターミナルノード ID

dst_gid 接続先のゲートウェイ ID

dst_tid 接続先のエンドターミナルノード ID

これらのコマンドによって確立されたコネクションはデータベースのコネクション情報テーブルで管理される。

このテーブルの内容を表 9.3 に示す。

表 9.3: connection information table

Source Gateway ID	Source End Terminal Node ID	Destination Gateway ID	
送信元ゲートウェイ ID	送信元エンドターミナルノード ID	送信先ゲートウェイ ID	

Destination End Terminal ID	Format
送信先エンドターミナルノード ID	ストリームフォーマット

9.1.3 データベースエンジン

最後に以上の二つのモジュールの下位に位置し、データベースへの排他制御を伴うアクセスを行うデータベースエンジンである。

データベースエンジンは上位のモジュールからのトランザクション要求に応じ、データベースの管理を行う。

今回のデータベース部分の実装はPostgreSQL を用い、C 言語からデータベースの操作を行うライブラリである libpq を利用した。

9.2 リソースインフォメーションエージェント

リソースインフォメーションエージェントに限らず、ゲートウェイの実装はそのフロントエンドネットワークの種類に依存する。ここでは、フロントエンドネットワークとしてIEEE1394 を用いた場合について述べる。

リソースインフォメーションエージェント内のモジュールとしてはエンドターミナルノードのアタッチやデタッチの処理を行うモジュールや接続の際にシグナリングを行うモジュールがある。

9.2.1 エンドターミナルノードの管理を行うモジュール

このモジュールでは JAIST VideoLAN のターミナルシステムの機能である、IEEE1394 機器が接続されたことを感知しリソースマネージメントエージェントに通知する機構を利用し、エンドターミナルノードのアタッチやデタッチを感知し、リソースマネージャへの通知を行う。

エンドターミナルノードのアタッチを感知するとフロントエンドネットワーク内で一意となるエンドターミナルノード ID を割り当て、データベースへの登録を行い、リソース

マネージャに通知する。

この段階で機器の抽象化が行われ、論理的なエンドターミナルノードの集合の形でリソースマネージャへの通知が行われる。

エンドターミナルノードのデタッチを感知するとデータベースからの抹消を行い、リソースマネージャに通知する。

これらの情報はエンドターミナルノードインフォメーションテーブルで管理される。

これらのテーブルの内容を表9.4に示す。

表 9.4: end terminal node information table

End Terminal Node ID	Format Type	Id Output	
エンドターミナルノードの ID	エンドターミナルノードが扱うことができるフォーマット	ストリームの方向	

Name	Loopback ID	Option ID	Icon Image
エンドターミナルノードの名称	ループバックパスの接続先の ID	機器固有の ID	アイコンイメージ

9.2.2 シグナリングを行うモジュール

このモジュールでもシグナリング自体はJAIST VideoLANのターミナルシステムの機能を利用する。

つねにリソースマネージャからの指示を待ち、リソースマネージャから接続の確立や開放の要求を受けるとターミナルシステムに対してシグナリングの指示を行う。

9.3 コントロールプロトコルトランスレータ

IEEE1394では機器の制御のためのプロトコルとしてAV/Cという規格が採用されている。コントロールプロトコルトランスレータではIEEE1394バス上のAV/Cコマンドフレームを拾いあげ、VNCPに変換し、コアネットワークに送り出す。また、VNCPを受け、AV/Cに変換し、IEEE1394バス上に送り出す。

これらの機能の実現のためにはIEEE1394上のアシンクロナスパケットに含まれるAV/Cコマンドフレームを検出する機構が必要となる。

そのためにIEEE1394インタフェースを持つPC上でWIDEプロジェクトによるIEEE1394デバイスドライバ(ftp://new-tremaine.cc.uec.ac.jp/pub/firewire/)を用い、さらにAV/Cを取り出すための仮想デバイスである/dev/avc0を実装した。

コントロールプロトコルトランスレータではこの仮想デバイスからIEEE1394のバス上のAV/Cコマンドフレームを検出し、前章で述べたVNCPに変換し、コアネットワークに送り出す処理を行う。また、逆にVNCPを受け、AV/Cに変換しIEEE1394のバス上に送り出す処理も行う。

第 10章

まとめ

本研究では研究題目でもある異種ビデオネットワーク間接続を実現するビデオネットワーク統合アーキテクチャの提案を行った。

各種ビデオネットワーク間で異なる方式が用いられる制御プロトコルや資源管理体系に関して変換を行う機構を実装したゲートウェイを用いることでビデオネットワーク相互間の接続が可能となる。

また、複数のビデオネットワーク上の機器を一つの概念で扱うための資源管理手法を用いることによって、機器が接続されるビデオネットワークの種類に関わらず統一した手法で管理を行うことが可能となる。また、このような資源管理手法を採用することによって、利用者に対してネットワークの違いを意識させずに家電感覚で操作することができる操作環境を提供することが用意となる。

このような資源管理を行うためにネットワーク上の機器を論理的なノードという単位に分割して管理し、これらの論理的なノード間の接続を行うコアネットワークを介して接続を行う。

また、このように抽象化したノードのみを対象とするコアネットワークを用いることによって、新たなビデオネットワークの接続の際にもゲートウェイを実装するだけでコアネットワークには一切変更を加える必要がなくなり、拡張性に優れたシステムとすることができる。

将来どのような新たな規格が誕生するか予想することが難しいビデオネットワークに対して、このように拡張性に優れ、家電感覚で利用できるような相互接続の枠組の必要性は今後益々高くなっていくと考えられる。

第 11 章

今後の課題

11.1　付加的なサービスへの対応

今後は異種ビデオネットワーク間の単純な接続だけではなく、さまざまな付加的なサービスへの対応が必要となる。

ビデオネットワークに必要となる付加的なサービスとして、帯域の予約や課金、認証などが考えられる。

これらのサービスを VIA の枠組に採り入れるためには、コアネットワークにこれらのサービスを含めたセッション管理を行うセッションコントロールマネージャの存在が必要になると考えられる。また、ゲートウェイ内のセッションアダプタにもセッションコントロールマネージャと協調して動作する機構を設ける必要がある。

これらの機構を設けることによって現在の CATV などの放送型のサービスと VoD などのオンデマンド型のサービスを統合する新たなサービスが実現できる。

11.2　資源の分散管理

相互接続を行うネットワークの数の増加に伴い、管理すべき機器の数も増加する。このような資源の増加に対応するためには、管理を行う主体を増やす必要がある。現在の枠組では、唯一のリソースマネージャで全ての機器を管理することになっているが、このような方法で管理できる機器の数には限界がある。

そこで、システム内に複数のリソースマネージャを配置することを可能とする必要がある。

現在の枠組の範囲内で複数のリソースマネージャを配置する方法として、コアネット

ワーク以下の全てのネットワークを一つのフロントエンドネットワークとしてより上位のコアネットワークに接続する方法、つまり、階層化することで拡張性を確保する方法が考えられる。また、あたかも現在の枠組におけるフォーマット変換のためのフロントエンドネットワークのように見え、コアネットワーク間のブリッジとなるフロントエンドネットワークを設け、他のコアネットワークと対等な立場で接続する方法が考えられる。しかし、いずれの方法もあるコアネットワークが他のコアネットワーク内の構成情報を取得することができず、オペレーションターミナルが両者の情報を併合し、利用者に提供することが必要となる。

このように現在の枠組の中では複数のリソースマネージャを用いた運用は可能であるものの、一つのインタフェースから全てのネットワークの情報を取得することはできない。また、前節で述べたような帯域予約や認証や課金を行うことが困難となる。

そこで、本質的な解決のためには VIA を複数のリソースマネージャの存在に対応したシステムとし、リソースマネージャ間で連絡を取り合い、協調した動作を可能とする必要がある。このようなシステムの実現に際しては情報のコンシステンシをいかに確保するかが重要となると考えられる。

第 12章

おわりに

本稿では異種ビデオネットワーク間接続を行う際に必要となる機能について述べ、それらを実装したゲートウェイによって相互接続が可能となることを述べた。また、システムの制御に用いるためのコマンドを抽象化する制御プロトコルである VNCP を提案した。さらに、相互に接続されたネットワーク上の機器を統一して取り扱うための資源管理の手法について述べた。

これらの手法を採用し、異種ビデオネットワーク間の接続を可能とする枠組である VIA を提案し、既存のビデオネットワークである JAIST VideoLAN をベースとして VIA を構築する方法について述べた。

また、JAIST VideoLAN をベースとして VIA を構築するためのいくつかのコンポーネントについての実装を行った。

最後に VIA が今後対応するべきサービスについての検討を行った。

参考文献

[1] 1394 Trade Association. *AV/C Digital Interface Command Set General Specification, Version 3.0*, 1998.

[2] *RTP: A Transport Protocol for Real-Time Applications*, 1996.

[3] *RTSP: Real Time Streaming Protocol*, 1998.

[4] *HAVi SPECIFICATION 1.0*, 2000.

[5] IEEE. *IEEE Std 1394-1995, Standard for a High Performance Serial Bus*, 1995.

[6] ITU-T. *G.711, Pulse code modulation (PCM) of voice frequencies*, 1988.

[7] ITU-T. *G.722, 7 kHz audio-coding within 64 kbit/s*, 1988.

[8] ITU-T. *G.728, Coding of speech at 16 kbit/s using low-delay code excited linear prediction*, 1992.

[9] ITU-T. *T.81, Digital compression and coding of continuous*, 1992.

[10] ITU-T. *H.261, Video codec for audiovisual services at p x 64 kbit/s*, 1993.

[11] ITU-T. *H.331, Broadcasting type audiocisual multipoint systems and terminal equipment*, 1993.

[12] ITU-T. *T.127, Multipoint binary file transfer protocol*, 1995.

[13] ITU-T. *T.82, Coded representation of picture and audio information*, 1995.

[14] ITU-T. *G.723, Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s*, 1996.

[15] ITU-T. *G.729, 8 kbit/s CS-ACELP speech coder*, 1996.

[16] ITU-T. *H.223, Multiplexing protocol for low bit rate multimedia communication*, 1996.

[17] ITU-T. *H.322, Visual telephone systems and terminal equipment for local area networks which provide a guaranteed quality of service*, 1996.

[18] ITU-T. *H.231, Multipoint control units for audiovisual systems using digital channels up to 1920 kbit/s*, 1997.

[19] ITU-T. *T.126, Multipoint still image and annotation protocol*, 1997.

[20] ITU-T. *H.247, Multipoint extension for broadband audiovisual communication systems and terminals*, 1998.

[21] ITU-T. *H.321, Adaptation of H.320 visual telephone terminals to B-ISDN*, 1998.

[22] ITU-T. *H.323, Packet-based multimedia communications systems*, 1998.

[23] ITU-T. *H.324, Terminal for low bit-rate multimedia communication*, 1998.

[24] ITU-T. *H.332, H.323 extended for loosely coupled conferences*, 1998.

[25] ITU-T. *Video coding for low bit rate communication*, 1998.

[26] ITU-T. *H.221, Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices*, 1999.

[27] ITU-T. *H.310, Broadband audiovisual communication systems and terminals*, 1999.

[28] ITU-T. *H.320, Narrow-band visual telephone systems and terminal equipment*, 1999.

[29] ITU-T. *H.222.0, Information technology - Generic coding of moving pictures and associated audio information: Systems*, 2000.

[30] ITU-T. *H.245, Control protocol for multimedia communication*, 2000.

[31] ITU-T. *H.262, Information technology - Generic coding of moving pictures associated audio information*, 2000.

[32] Katsushi Kobayashi. Design and Implementation of Firewire Device Driver on FreeBSD. In *1999 USENIX Annual Technical Conference*, 1999.

[33] Yasuo Tan. Scaling up IEEE 1394 DV network to an enterprise video LAN with ATM technology. In *Digetst of technical papers of IEEE International Conference on Consumer Electronics 1998*, Vol. 1, pp. 112–113. IEEE, 1998.

[34] Yasuo Tan. Scalable digital video network with IEEE1394 and ATM. In *International Distributed Conference 1999*, 1999.

[35] Yasuo Tan, Takashi Nomura, Hirofumi Tamori, and Koji Koshiba. Plug and play campus digital video network with IEEE1394 and ATM. In *Proceedings of the International Conference on Computer Communication 1999*, 1999.

[36] Texas Instruments. *PCILynx Functional Specification Revision 1.2*, 1997.

[37] 谷口雅幸, 丹康雄. マルチメディアネットワークサービスの家電機器による利用法に関する一手法. 情報処理学会 第 59 回 (平成 11 年後期) 全国大会, pp. 4C–02, 1999.

[38] 倉岡貴志, 丹康雄. AV 系ネットワークシステムにおける資源管理に関する一手法. 情報処理学会 第 59 回 (平成 11 年後期) 全国大会, pp. 4C–08, 1999.

[39] 丹康雄. JAIST Video LAN - 実世界指向マルチメディアネットワーク. 人工知能学会研究会資料 SIG-FAI-9802, 人工知能学会, 東京, 9 1998.

[40] 丹康雄. 実世界指向マルチメディアネットワークの実現に関する一考察. 情報処理学会 第 59 回 (平成 11 年後期) 全国大会, pp. 5C–08, 1999.

[41] 丹康雄. 統合ビデオネットワークアーキテクチャによる異種ビデオネットワーク間接続. 情報処理学会 第 61 回 (平成 12 年後期) 全国大会, pp. DE–07, 2000.

[42] 丹康雄, 野村隆, 田守寛文. 家電的ユーザーインタフェースを有する大規模マルチメディア LAN システム. In *Interaction '99*, 1999.

[43] 中田潤也, 丹康雄. 異種ビデオネットワーク間接続に関する研究. 情報処理学会 第 61 回 (平成 12 年後期) 全国大会, pp. 1J–02, 2000.

[44] 木村範彦, 丹康雄. マルチキャスト通信システムにおける中間ノードによる資源の有効利用法. 情報処理学会 第 59 回 (平成 11 年後期) 全国大会, pp. 4C–09, 1999.

[45] 小峯隆宏, 町澤朗彦, 丹康雄, 野村隆, 濱田元, 中川晋一, 久保田文人. ATM 通信網における DV 画像通信の検討. 信学技報 CQ99-65, 電子情報通信学会, 東京, 2 2000.

謝辞

本研究をまとめるにあたり、常に研究の方向性についての指針を与えて下さり、また研究者としての物事の捉え方の心得を授けて下さった丹康雄助教授に深く感謝致します。

また、研究に関して度々貴重な意見を与えて下さった丹研究室の木村範彦君、倉岡貴志君、谷口雅幸君、大林隆之君、田中徹君、登弘聡君、牧野義樹君、吉岡正巳君に感謝致します。

時には研究以外のことに目を向けさせてくれることで研究生活をより有意義なものとして下さったR-8石川の皆さん、川端康代さんに感謝致します。

最後に研究生活をさまざまな面で援助して下さった家族に感謝致します。

付録 A

データベース内の各テーブル

リソースマネージャ

ゲートウェイインフォメーションテーブル

Gateway ID	IP Address	ATM Address	Name
ゲートウェイ ID	ゲートウェイの IP アドレス	ATM アドレス	ゲートウェイの名称

エンドターミナルノードインフォメーションテーブル

Gateway ID	End Terminal Node ID	Format Type	Id Output	
ゲートウェイ ID	エンドターミナルノードの ID	エンドターミナルノードが扱うことができるフォーマット	ストリームの方向	

Name	Loopback ID	Option ID	Icon Image
エンドターミナルノードの名称	ループバックパスの接続先の ID	機器固有の ID	アイコンイメージ

コネクションインフォメーションテーブル

Source Gateway ID	Source End Terminal Node ID	Destination Gateway ID	
送信元ゲートウェイ ID	送信元エンドターミナルノード ID	送信先ゲートウェイ ID	

Destination End Terminal ID	Format
送信先エンドターミナルノード ID	ストリームフォーマット

リソースインフォメーションエージェント

エンドターミナルノードインフォメーションテーブル

End Terminal Node ID	Format Type	Id Output	
エンドターミナルノードの ID	エンドターミナルノードが 扱うことができるフォーマット	ストリームの 方向	

Name	Loopback ID	Option ID	Icon Image
エンドターミナルノードの 名称	ループバックパスの 接続先の ID	機器固有の ID	アイコンイメージ

付録B

各モジュール間の通信を行うプロトコル

リソースマネージャとリソースインフォメーションエージェント間の通信

attach_gateway ゲートウェイのアタッチ

書式

```
attach_gateway serialized_data_block
```

詳細

図 12.1 に示したデータを ASN.1 を用いてシリアライズしたものを引数としてとり、データベースへの登録を行う。

serialized_data_block 図 12.1 に示したデータを ASN.1 を用いてシリアライズしたもの

```
Gateway
DEFINITIONS ::=
BEGIN
Gateway ::= SET {
    ip-addr     [0] IpV4Addr,
    atm-addr    [1] AtmAddr,
    name        [2] VisibleString,

AtmAddr ::= IA5String (
    FROM (
        "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
        "8" | "9" | "a" | "b" | "c" | "d" | "e" | "f" ) |
    SIZE ( 20 ) )

IpV4Addr ::= IA5String (
    FROM (
        "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
        "8" | "9" | "a" | "b" | "c" | "d" | "e" | "f" ) |
    SIZE ( 8 ) )
END
```

図 12.1: definition of module gateway

detach_gateway ゲートウェイのデタッチ

書式

```
detach_gateway gid
```

詳細

デタッチするゲートウェイを表すgid を引数としてとり、データベースからの抹消を行う。

gid デタッチするゲートウェイの ID

attach_node エンドターミナルノードのアタッチ

書式

```
attach_node serialized_data_block
```

詳細

図 12.2 に示したデータを ASN.1 を用いてシリアライズしたものを引数としてとり、データベースへの登録を行う。

serialized_data_block 図 12.2 に示したデータを ASN.1 を用いてシリアライズしたもの

```
Terminal
DEFINITIONS ::=
BEGIN
Terminal ::= SET {
    gateway-id  [0] IdNumber,
    node-id     [1] IdNumber,
    format-type INTEGER,
    is-output   BOOLEAN,
    name        [3] VisibleString,
    loopback-id [4] IdNumber,
option-id   [5] VisibleString,
    icon-image  OCTET STRING }

IdNumber ::= IA5String (
    FROM (
        "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
        "8" | "9" | "a" | "b" | "c" | "d" | "e" | "f" ) |
    SIZE ( 16 ) )
END
```

図 12.2: definition of module terminal

detach_node エンドターミナルノードのデタッチ

書式

```
detach_node gid tid
```

詳細

デタッチするエンドターミナルノードを表すtid とgid の対を引数としてとり、データベースからの抹消を行う。

gid デタッチするエンドターミナルノードが所属するフロントエンドネットワークのゲートウェイの ID

tid デタッチするエンドターミナルノードの ID

リソースマネージャとオペレーションターミナル間の通信

get_info エンドターミナルの情報の取得

書式

```
get_info [[key value] ...]
```

詳細

エンドターミナルノードの情報を取得する。

以下のキーに対して値を指定することで選択的な取得が可能となる。

gid ゲートウェイの ID

tid エンドターミナルノードの ID

gname ゲートウェイの名称

tname エンドターミナルノードの名称

format フォーマット

lid ループバックパスの接続先の ID

oid オプション ID

load_image エンドターミナルのアイコンイメージの取得

書式

```
load_image gid tid
```

詳細

ゲートウェイ ID とエンドターミナルノード ID の対で指定されたエンドターミナルノードについて登録されたアイコンのイメージを取得する。

gid ゲートウェイの ID

tid エンドターミナルノードの ID

connect 接続の確立

書式

```
connect src_gid src_tid dst_gid dst_tid
```

詳細

接続元と接続先のゲートウェイ ID、エンドターミナルノード ID を受け取り、接続を確立するためにゲートウェイのセッションアダプタに要求を行った後、データベースへの登録を行う。

src_gid 接続元のゲートウェイ ID

src_tid 接続元のエンドターミナルノード ID

dst_gid 接続先のゲートウェイ ID

dst_tid 接続先のエンドターミナルノード ID

disconnect 接続の開放

書式

```
disconnect src_gid src_tid [dst_gid dst_tid]
```

詳細

接続元と接続先のゲートウェイ ID、エンドターミナルノード ID を受け取り、接続を開放するためにゲートウェイのセッションアダプタに要求を行った後、データベースからの抹消を行う。

src_gid 接続元のゲートウェイ ID

src_tid 接続元のエンドターミナルノード ID

dst_gid 接続先のゲートウェイ ID

dst_tid 接続先のエンドターミナルノード ID