第 5 部

IEEE1394 とインターネットの融合技術

第 1 章

はじめに

近年、情報/通信/放送/家電/AV 等の各業界は「デジタル化」、「マルチメディア化」、「ネットワーク化」の大きな動きの中にある。今後これらの各分野は従来の垣根を越えて、「デジタルによる融合」の方向に向かっていくと見られる。このように, 家電や放送、通信などの業界を巻き込んで, デジタル機器が民生機器や家庭内に入ってくると、それらの機器を相互に接続する要求が出てくるのは自然である。この時、従来のアナログケーブルに変わる「デジタルケーブルによる接続」が必要になる。

現在、”IEEE1394”が、この「デジタル AV ケーブル」、あるいは「ホーム AV ネットワーク」の有力候補となっている。IEEE1394 は、数百 Mbps(ビット/秒) の速度を持つ高速バス (ケーブル) である。同期転送の機能をもっているため、遅延や揺らぎがなく、連続的にデータのやり取りを行う必要のある映像や音声の伝送に適している。その他、低コストでの供給が期待できること、プラグアンドプレイの機能があり家庭に受け入れられやすい等の理由から、AV 業界や CATV 業界等が IEEE1394 の採用を決定している。これらと並行して、デジタル映像信号の標準フォーマットである MPEG2 や、DV 等の転送方式も確定され、将来の AV 機器のデジタル接続は IEEE1394 を中心に展開されているといっても良い。また、パソコンのマルチメディア化を目指すマイクロソフト等も、IEEE1394 のサポートを発表している。

このように、IEEE1394 は「家庭内 AV 網」、「パソコンのマルチメディアインタフェース」、「低コストな高速インタフェース」といった色々な側面を持っている。

家庭のインターネット接続を考える場合、インターネットと AV/放送の融合を考える場合、あるいはインターネット機器を低コストで提供することを考える場合、今後 IEEE1394 は非常に重要な位置を占めると考えられる。

このような背景から、WIDE では 97 年 9 月に、「IP over IEEE1394」WG を発足。「IP over IEEE1394」の相互接続実験や、IEEE1394 機器制御 API、IEEE1394 とインターネットとの接続等の検討、実験を行なってきた。

本報告では、本 WG の 98 年度の活動報告を行なう。次節にて、WG 活動の報告として、IEEE1394 とインターネットの相互接続、「IP over 1394 同期チャネル」の検討、IEEE1394 制御 API の実装、1394 over IP トラヒックのジッタ測定、及び合宿や研究会等で開催された BOF の活動のそれぞれについて報告する。最後に、今後の課題と活動予定について説

明する。

第 2 章

WG 活動の報告

本年度の WG 活動としては、IEEE1394 とインターネットの相互接続 (1394/DV on IP)、「IP over 1394 同期チャネル」についての検討、及び IETF へのドラフト投稿、IEEE1394 制御 API の検討/実装、合宿等での BOF の開催、メイリングリストでの議論等を行なってきた。本節では、これらの活動報告を行なう。

2.1 IEEE1394 とインターネットの相互接続 (1394/DV over IP)

2.1.1 富士通研の実装

株式会社富士通研究所では、他に先駆けて 97 年度より通信プロトコル処理をネットワークアダプタにオフロードしてハードウェア処理する Comet システムの上で 1394 over IP の実装を進めて来た。今年度は、実際に運用状態の WAN において 1394 over IP を転送する実験を行なった。Comet システムの詳細は URL: http://www.pds-flab.rwcp.or.jp に詳しく説明されている。

Comet ネットワークアダプタは同時に 2 つのネットワークインターフェースを持つプロトコル処理専用エンジンを搭載した PCI カードである。Comet ネットワークアダプタに IEEE1394 と 100Base-TX を装備し、DV 装置を IEEE1394 で接続する。DV カメラが 125 μ 秒毎に送出する DV ストリームパケットを Comet で受けて IP カプセル化し、100Base-T 側から Ethernet に送出する (図 2.1)。これを受信した Comet では、IP パケットから IEEE1394 同期パケットを取り出して IEEE1394 バスに転送し、DV 機器がそれをリアルタイムで再生する。従来の計算機ネットワークでこうしたプロトコル変換を行う場合、アダプタからホストに割り込みを上げ、プロトコル変換を行い、アダプタにデータを送り返すという処理が必要で、これを 125 μ 秒という短時間内に行うのは難しい。Comet ではネットワークアダプタ内に処理をオフロードすることで 1 パケットあたり 40 μ 秒で処理し安定した画像再生を行える。

1998 年 5 月に Comet を WIDE プロジェクトの広域バックボーンに接続して、1394 over IP を利用し、実運用中の広域網で 125 μ 秒の精度でジッタを抑えられるかの定性的な評価

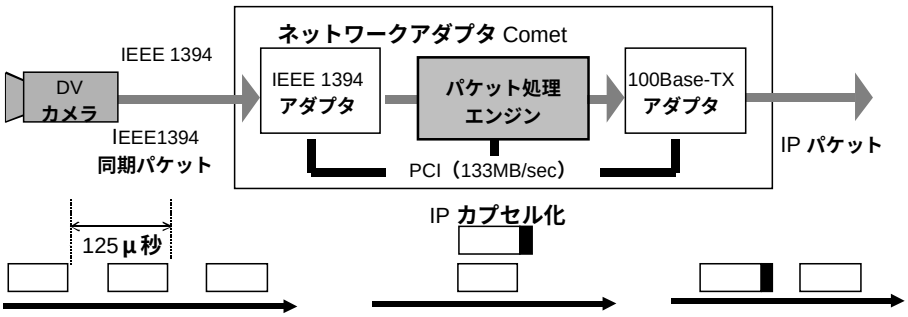

図 2.1: Comet による IEEE1394 同期パケットの IP 化

実験を行った。具体的には、奈良先端大から慶應大学湘南藤沢キャンパス (慶應大学 SFC) と北陸先端大に WIDE 研究会の中継を行いながら、同時に SFC からの画像を奈良先端大に送るという実験を行った (図 2.2)。回線は双方向に 40Mbps 以上ずつを確保し、DV スト

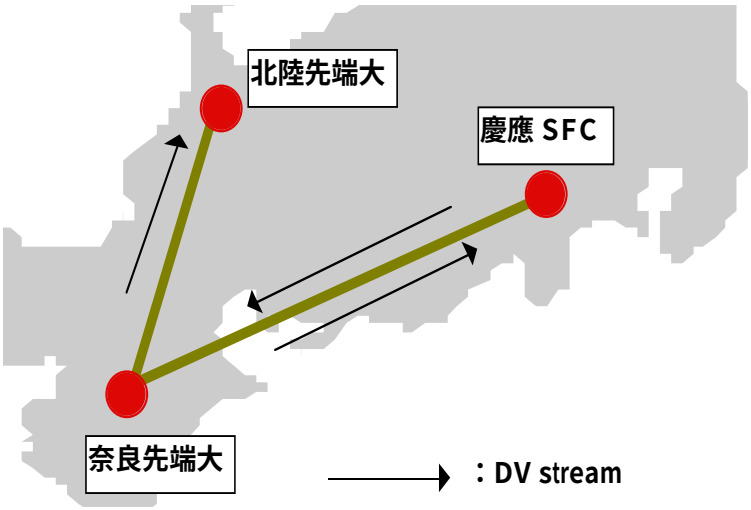

図 2.2: WIDE バックボーンでの DV 通信実験

リームを流しても回線帯域が問題にならないようにした。各経路の途中に存在するルータと ATM スイッチの台数を表 2.1に示す。

奈良先端大から北陸先端大、SFC へ単方向で映像を流している間は、それなりに高品質な画像再生が出来たが、SFC から奈良への映像送信を開始すると何が映っているか判別できないほどに画像が乱れた。この結果から最新専用ルータで接続されたネットワークであっても、双方向のデジタルビデオを 2 ストリーム流すとジッタが完全に制御できなくなることが分かった。本実験後、1998 年 6 月に Interop'98 でも池袋と幕張会場間でビデオストリームを双方向に 2 チャネル流す実験を実施し同様の結果を得た。

一方、Comet では先述したように 1 パケットを 40 μ秒で処理するので、論理的には 3 チャネル分ビデオストリームを処理できる。実際、2 台の Comet を直結して双方向に計 2 チャネルビデオストリームを流しても安定した画像再生ができることを確認した。

表 2.1: WIDE バックボーンでの実験における中継網構成

	IP ルータ	ATM スイッチ
奈良先端〜北陸先端	3 台	3 台
奈良〜慶應 SFC	4 台	2 台

1998 年 11 月にアメリカのオーランドで行なわれた SuperComputing'98 の展示セッションに Comet による 1394 over IP を出展し、日米間の研究用ネットワークである Transpac を経由して日本-アメリカ間でデジタルビデオ映像をリアルタイムに飛ばす世界初の実験を行なった。実験ネットワークの構成を図 2.3に示す。十分な帯域を end-to-end で確保できない、激しいジッタが発生するといった難題があらかじめ予期できたので、以下の機能を Comet ファームウェアに追加し、混雑したネットワーク上でも高品質な DV 再生に成功した。

- 送信側でのフレーム間引き機能
- 受信側でフレーム単位でバッファリングを行なう機能

この実験は地元テレビ局でも取り上げられ、大変好評であった。

来年度は、新しい WIDE のバックボーンネットワークである JB に Comet を接続して 1394 over IP の実験を進めていく予定である。

2.1.2　慶應義塾大学グループの実装

現在、市場にて購入可能な DV カムコーダと VCR(ビデオカセットレコーダ) を AV コーダ/デコーダとして使用する、インターネットベースのリアルタイム AV ツールを開発した。これらの AV デバイスは、IEEE1394 インタフェースを有している。この実験では、DV フォーマットを用いた高品質なビデオ通信と、効率的なデータ圧縮を、以下のような手順で低コストに実現した (図 2.4)。

- 市販の DV カムコーダを、ビデオデータを IEEE1394 パケットとして送出するエンコーダデバイスとして使用
- 市販の DV レコーダデッキ (IEEE1394 端子付) を、映像表示デバイスとして使用
- PC に IEEE1394 とインターネットのプロトコル変換機能を持たせる

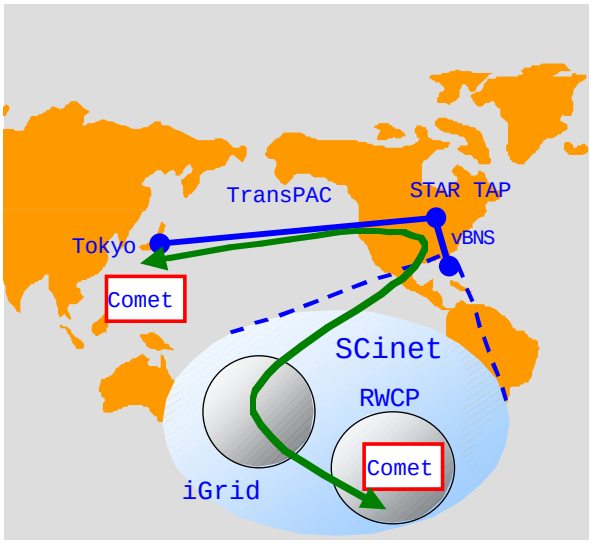

図 2.3: SC98 ネットワーク図

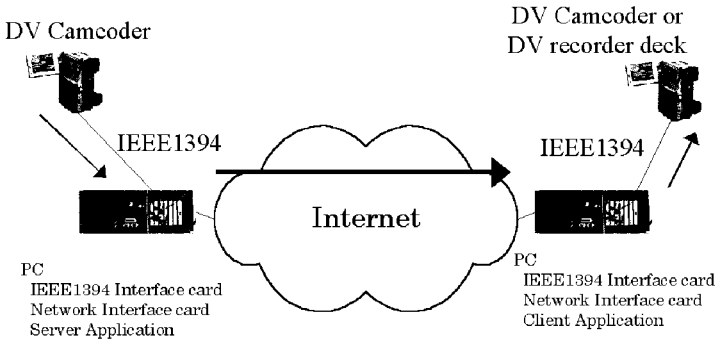

図 2.4: 全体システム

DV カムコーダ、DV デッキは、市販のものをそのまま利用した。また、PC については、128MB メモリ、PCIlynx-1394 カード、100BaseTX カード、MMX ペンティアム 200MHz(ただし、MMX 命令は使わず) のものを使用した。

実験で用いた AV 機器は、DV フォーマットと呼ばれるビデオフォーマットを用いている。DV はフレーム内予測を用いるビデオ圧縮方式であり、映像のピクチャフレームに相当する DV フレームユニットを単位に、データのやり取りがなされる。IEEE1394 上では、DV データは 80 バイトの DIF ブロックと呼ばれる単位にセグメント化される。更に、この DIF ブロックは、いくつかに束ねられて「DIF シーケンス」とよばれる単位にまとめられ、IEEE1394 フレームとして転送される (図 2.5)。いくつの DIF ブロックが、1 つの IEEE1394 フレームに入るかは、IEEE1394 の転送速度に依存する。IEEE1394 上で転送される場合は、CIP ヘッダと呼ばれるヘッダが更に付与される。この CIP ヘッダには、データの発信元、シーケンス番号、符号化方式などのデータが入る。

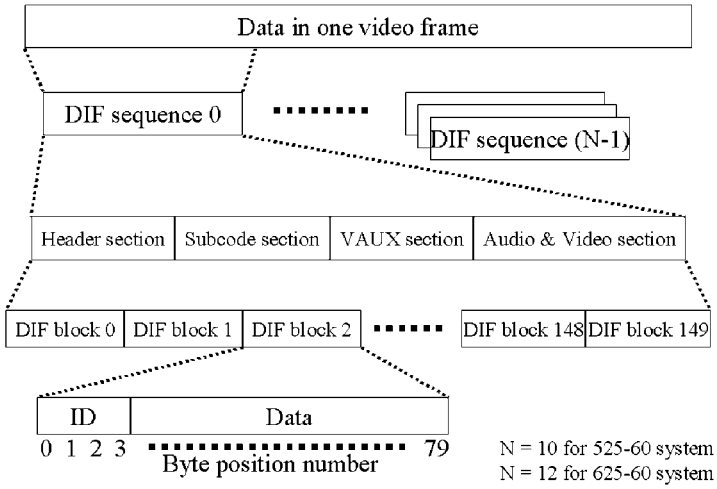

図 2.5: DV データの構造

IEEE1394 とインターネット間のゲートウエイとなる PC は、送信側の PC は IEEE1394 上を流れる DV フォーマットの映像 (音声) を、インターネットパケット (UDP パケットを使用) に乗せ込む。受信側の PC は、この逆の処理を行う。パケットフォーマットを図 2.6 に示す。

ただし、DV ストリームは、NTSC 品質の映像を転送する場合、そのままの形では 35Mbps 以上の帯域を消費してしまう。また、経験的に、音声は映像よりもより重要な情報を含んでいる (TV 会議で映像の一時的中断は許容できるが、音声の中断は人間は許容できない)。よって、映像フレームを間引き、音声フレームはそのまま残すことにより、DV ストリームの効率的な転送を図ることが出来る。図 2.7の様に、送信側の PC が、適当に映像情報を間引くことにより、転送レートを可変に設定できるようにした。

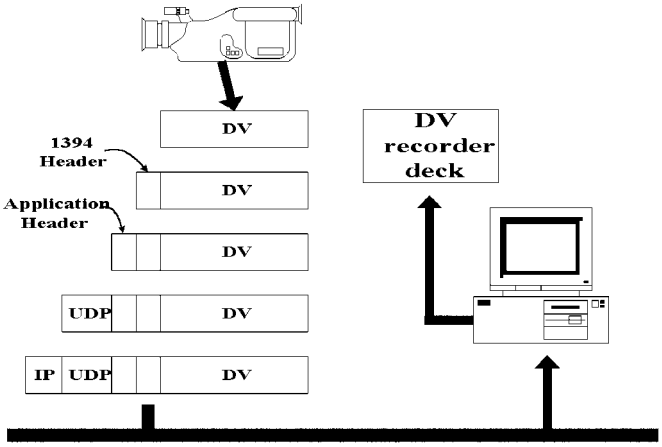

図 2.6: パケットフォーマット

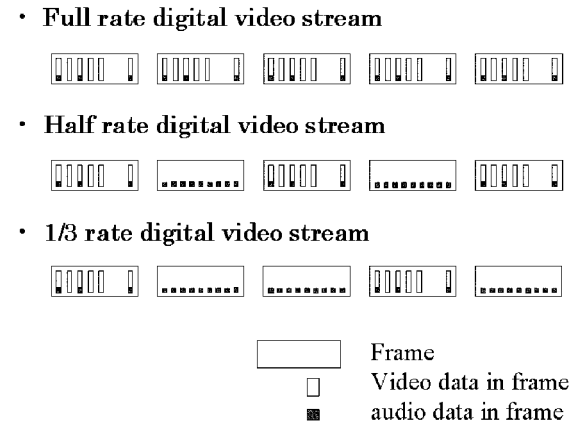

図 2.7: 映像データの間引き

また、パケット廃棄が起こった場合、受信側では、前のフレームの映像を使うこととした。よって、インターネット上の帯域が制限されている場合は、音声のみを送ることで、対処することが可能である。

これらの実験設備を日米にそれぞれ配置し、APAN Trans pacific link を縦断しての DV 通信の実験を 98 年 11 月 10 日から 14 日にかけて行った (図 2.8)。実験は、図 2.9の様に、慶應大学村井教授の講義を、日米間で双方向に行う形で行った。

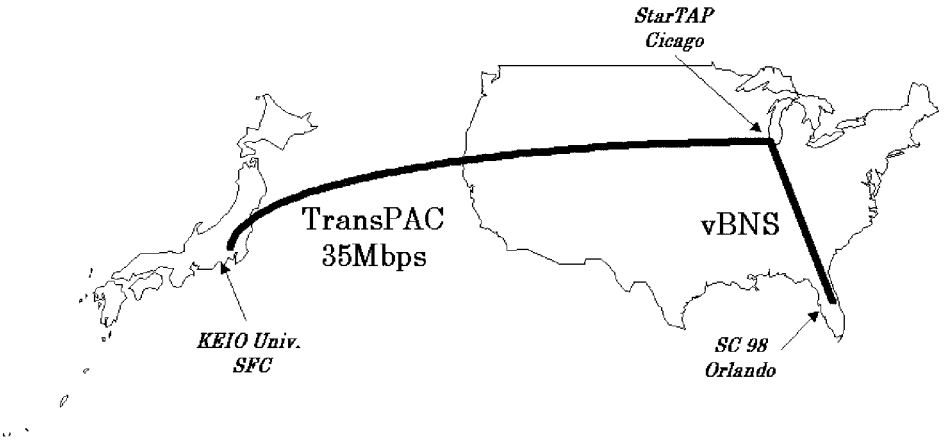

図 2.8: ネットワークトポロジ

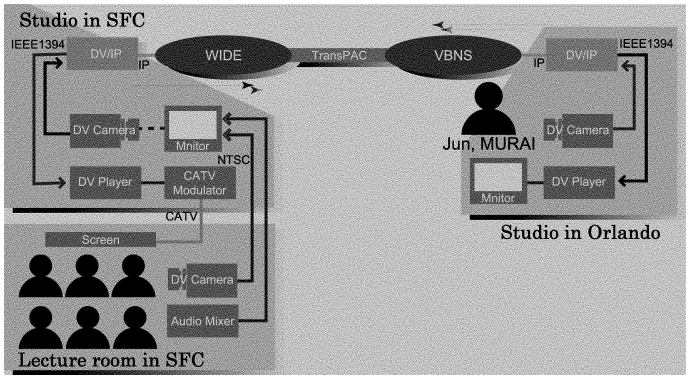

図 2.9: 日米間での遠隔講義

本来、DV データは、転送途中でデータの廃棄等は仮定されていないが、本実験では、DV データをインターネット中を転送するため、ジッタ、パケット廃棄等の問題が発生した。本実験では、エンドホストでの対応のみで、これらの問題に対処したが、多くの問題は、前

述の送信フレームレートの調整で対処が可能であった。しかし、音声パケットが連続的に廃棄された場合、通信が不自然になる現象が見られた。

2.1.3 DV over RTP

先の日米間の DV 映像転送実験では、UDP パケット上に直接 DV フレームをマッピングする方法で行なった。

一方、IETF において、インターネットにおける AV データの転送の検討を行なっているのは AVT(Audio Video Transport)WG である。この WG では、MPEG データや、音声データ等の AV データを、インターネット上で転送するためのトランスポートプロトコル RTP(Realtime Transport Protocol) を定義しており、更に各種の符号化方式を RTP に対応させるためのペイロードフォーマットの検討を行なっている。

本グループを中心としたメンバーは、DV データを RTP でにマッピングさせるためのペイロードフォーマットを検討し、IETF にドラフトの形で提案した [160]。現在、 AVT WG ではこの提案をもとに DV over RTP の標準化作業が進められている。本グループではこのドラフトで提案しているペイロードフォーマットをもとに、システムの実装中である。

2.2 “IP over 1394 同期チャネル” の検討

2.2.1 背景

IETF の IP1394 の WG では、IEEE1394 バス上での IP パケットの転送のためのプロトコルの検討を行なっている。現在のドラフト [67] では、IP パケットの「ベストエフォート転送」のための枠組みとして、以下の方式を規定している。

(1)IP broadcast / IP multicast on 1394 async stream (default channel) (2)IP multicast on 1394 async stream (3)IP unicast on 1394 async write

一方、IEEE1394 は、isochronous transfer mode を持つリンクレイヤネットワークであり、以下のような要求が起こるのは自然である。

(1) 特定の IP フローを、IEEE1394 の同期チャネルを通して転送したい (2) 特定の AV フロー (例えば MPEG2 with CIP Header 等) を、IEEE1394 の同期チャネルを通して転送したい (これを、IP で制御したい)

これらの実現のため、本 WG では、以下の機能を持ったプロトコルを IETF に提案した [162]。

(1) チャネル番号と IP フローの対応関係の通知 (2) 同期チャネルの帯域情報の通知 (3) 転送される IP フローの方向の通知 (4) 転送されるフローの属性の通知

本プロトコルは MCAP(Multicast Channel Allocation Protocol) [67] の拡張として定義される。これは、MCAP に上記の (1)、(2) の機能が含まれることから、MCAP の拡張とするのが最もスムーズと思われるからである。

2.2.2　MCAP の概要

MCAP は、特定の IP マルチキャストを IEEE1394 非同期ストリーム上に転送する場合に、この IP マルチキャストアドレスと、非同期ストリームのチャネル番号の対応関係を通知するプロトコルであり、以下のフォーマットを持つ (通知する IP マルチキャストアドレスが一つの場合)。

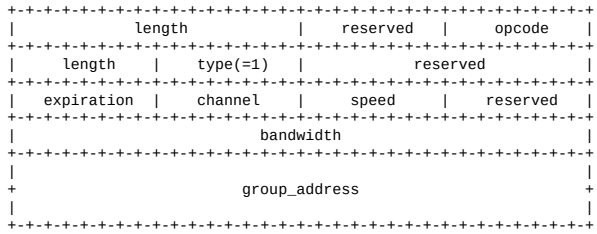

図 2.10: MCAP パケットフォーマット (type=1、IP マルチキャストとチャネルの対応関係の通知)

最初の 16 バイトが MCAP メッセージヘッダ、続くフィールドが MCAP アドレスディスクリプタである。タイプ=1 は、この MCAP パケットが、IP マルチキャストアドレスと、これを送信する IEEE1394 の同期チャネルのチャネル番号の対応関係の通知に使われることを示している。

もし、IP マルチキャストアドレスが ip_M のパケットを、チャネル番号 #y の非同期ストリームに流す場合には、”channel”の領域に #y、”group_address” の領域に ip_M の値を入れる。

その他、MCAP メッセージヘッダの length には MCAP メッセージ全体の長さ、opcode には”Advertise/Solicitation”の意味が入る。また、アドレスディスクリプタの length には、そのアドレスディスクリプタ領域の長さが入る。expiration には、この対応関係が有効な時間が入る。 speed には、この IP マルチキャストフローが転送されるビットスピードが入る。なお、type=1 では、bandwidth フィールドは使わない。

2.2.3　拡張 MCAP のプロトコルオーバービュー (IP フローの同期チャネル上の転送)

特定の IP フローを、特定の IEEE1394 バスの同期チャネルを通して転送する場合について説明する。IP のノード A(IP アドレス ip_A) と、ノード B(IP アドレス ip_B) との間で、特定の IP フローの転送を行なうとする。

特定の IP フローの転送に先立ち、ノード A とノード B の間にて、RTSP[113] 等に代表される上位レイヤのセッション制御により、特定の IP フローのセットアップが行われる。

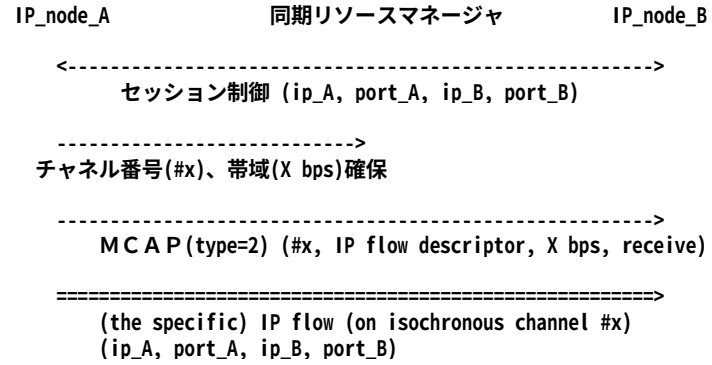

図 2.11: MCAP 手続き (制御の方向とフローの方向が同一の時)

ユーザは、この IP フローを IEEE1394 の同期チャネル上を転送することを考える。この IP フローは、「ノード A からノード B」の方向に転送される場合と、「ノード B からノード A」の方向に転送される場合がある。図 2.11 は「ノード A からノード B」の方向に転送される場合である。この場合の IP フローを、以下の様に表すことにする。

(Sender_IP_address, Sender_port, Receiver_IP_address, Receiver_port) = (ip_A, port_A, ip_B, port_B)

ここで、「ポート」とは、TCP ポート、あるいは UDP ポートであってもよい。IP フロー数は複数本であってもよいが、図 2.10 では 1 本の場合を示している。

セッションを確立したノード A は、必要となる通信資源を、同期リソースマネージャからチャネル番号と帯域を確保することにより、IEEE1394 上に確保する。

つづいてノード A は、(1) 転送する IP フロー、(2) この IP フローを転送する同期チャネルのチャネル番号、(3) 同期チャネルの帯域、(4)IP フローの転送方向、等を受信ノードであるノード B に通知する。この時に、MCAP を用いる。

(注;MCAP タイプ 1 では、上記の (1)、(2)、(3) の通知の機能を既に有する)

ここでは、新たに MCAP のタイプ=2 を定義する。MCAP のタイプフィールドが 2 である場合には、MCAP を、特定の IP フローを IEEE1394 同期チャネル上に転送する場合に、この IP フローの識別子と、同期チャネルのチャネル番号の対応関係を通知するものであるとする。以下にフォーマットを示す。

タイプ=1 との差分は、タイプフィールドが 2 であること、フロー ID タイプが新たに定義されていること、D(方向) ビットが新たに定義されていること、フロー ID が新たに定義されていること、である。

フロー ID タイプは、フロー ID の型を示す。

channel には、この IP フローが転送される同期チャネルのチャネル番号が入る。

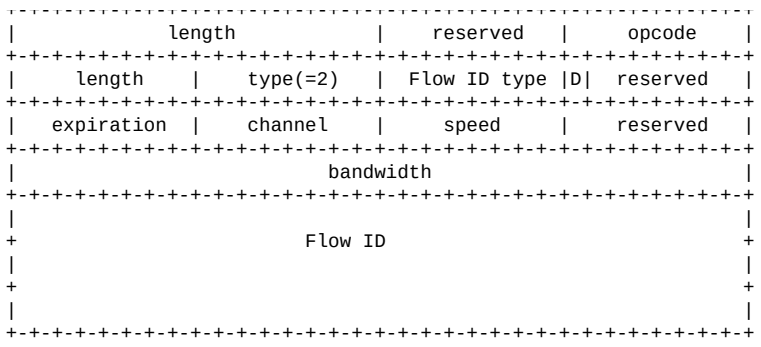

図 2.12: MCAP パケットフォーマット (type=2、フロー ID とチャネルの対応関係の通知)

D ビット (方向ビット) は、以下の意味を持つ。

0; この IP フローを受信せよ、1; この IP フローを送信せよ

bandwidth には、そのチャネルのために確保した帯域情報が入る。フォーマットは、IEEE1394-1995[67] の bandwidth available レジスタと同様であるとする。

フロー ID タイプは、フロー ID タイプ毎にフォーマットが異なるが、本提案では、フロー ID タイプ=2 の場合 (TCP フローの場合) と、 =3 の場合 (UDP フローの場合) を定義する。(この場合、フロー ID は (Sender_IP_address, Receiver_IP_address, Sender_Port, Receiver_port) のフォーマットとなる)

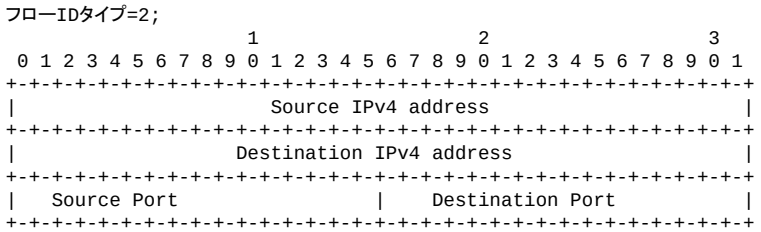

図 2.13: フロー ID フォーマット (type=2)

以上は、同期チャネル上を転送する IP フローがノード A からノード B の方向であった場合であるが、この逆の方向の場合も考えられる。この場合は、上記のシーケンスにおいて、D ビットの値が "1(送信せよ)"となる。

なお、一つの MCAP パケットで、同時に複数の IP フローディスクリプタを転送することも可能である。

なお、ここでは IP フローの同期チャネル上の転送について論じてきたが、同様に特定の AV フロー、例えば IEC61883 標準で規定される MPEG2 や DV フォーマットの AV デー

タの IEEE1394 上のフローを同期チャネル上に転送制御する場合、即ち上記 AV フローの転送制御を IP アプリケーションが行なう様な MCAP を定義すること提案している。詳細は、[162] を参照されたい。

2.2.4 課題

- 本プロトコルは、MCAP(マルチキャストチャネル割当てプロトコル) の拡張として定義したが、別のプロトコルとして定義すべきかもしれない。
- 本提案では、MCAP とおなじく、オペコードとして「Advertise」と「Solicitation」しか用意していない。これらのメッセージのための受信確認信号「Advertise Ack」「Solicitation Ack」等を定義する必要があるかもしれない。
- 本提案では、単一の IEEE1394 上での IP フロー/AV フローの同期チャネル上の転送を扱ったが、RSVP の様な、いくつかのサブネットをまたいで IP フローのために、QoS が確保された「通信パス」を確立する方法とのバインディングは今後の課題である。例えば、RSVP と本プロトコルを組み合わせる方法が考えられる。
- フロー ID フィールドの解析により、フローの方向は導出可能である為、方向ビットは、不要である可能性がある。

2.3 IEEE1394 制御 API の検討と実装

2.3.1 IEEE 1394 プロトコルの UNIX API へのマッピング

IEEE 1394 API への要求

IEEE 1394 に代表されるネットワーク接続を持つデジタル家電機器、例えばビデオカメラが市販されている。このような家電機器を、Unix 環境から制御するための API を検討、試作を行った。

本実装は次に示す 5 項目の条件を満たすように IEEE 1394 API を設計されている。

既存 UNIX API の利用 既存の UNIX API の枠組を大きく変えない。

EUI64 による機器の認識 IEEE 1394 では mac アドレスであるノード ID がバスリセットによるバスの再構成によって変わる。これではアプリケーションからの利用が不便なので、API からは機器固有の不変識別子 EUI64 でアクセスできることが望ましい。

トランザクションの並列処理 機器の間の連携動作を考えた時、同一のホストに同時に複数のアプリケーションが存在してそれぞれ別の機器を制御できることが望ましい。また、複数のトランザクションが並行して発行できたほうがよい。

outbound, inbound PC から機器を制御するためにはリクエスタ (outbound) の機能が必要であり、一方、IEEE 1394 外に存在する機器を IEEE 1394 機器のように見せるエミュレーション動作 [161, 158] をするためにはレスポンダ (inbound) の機能が必要となる。AV 家電機器制御のための AV/C プロトコルでは、制御側、被制御側の双方に outbound, inbound の両方の機能が必要となる。

多重化とアクセス制御 複数の UNIX アプリケーションが、Outbound 側として同一のバス上 (すなわち UNIX ホストから見て同一の 1394 インタフェース) に接続された 1394 機器を、互いに排他的に制御するためには、ソケットレベルでの制御の制御の排他性を保証する機能が要求される場合があるだろう。ホストが Inbound 側で複数の IEEE 1394 機器のエミュレーションをするときに同じことがいえる。他方、UNIX アプリケーションレベルで 1394 機器の構成管理を行う場合は、一つのアプリケーションの一つのソケットから複数の機器の制御をするやりかたが好まれるだろう。

ここでは、ホームゲートウェイ [158] および Java による 1394 機器制御 API と自動構成管理 [159] の実現に向けて、後者の一つのソケットからバス上の任意の機器を制御するインタフェースの検討、実装を紹介する。前者の被制御機器に対応したソケットを使う方法の検討についてはまた別の機会に紹介したい。

これらの機能を UNIX の標準的な API にマッピングする方法を考えた場合、いくつかの方法が考えられる。我々は新たにシステムコールを追加することは避けて、ソケット API にマッピングし、プロトコルドメインとして Asynchronous, Isochronous に対応する AF_FWO, AF_FWI の 2 つを追加した。ドメインを追加したのは INET, CCITT などの既存のプロトコルドメインとはアドレス形式 (EUI64) や操作 (トランザクションベース) が異っていることが理由である。候補には BPF(Berkley Packet Filter) も挙がったが、BPF の実装はイーサネットなど既存のネットワークインタフェースに強く依存してるため、候補から外した。

ここで、ドメインを追加することを前提として、トランザクションの発行を ioctl にマップするか、socket の read/write にマップするかという選択が生じるが、socket の read/write にマップすることを選択した。ここでも socket への read/write の他に ioctl にマップする方法が考えられる。だが、ioctl による実装では排他制御の粒度が大きいため、複数のソケットからリクエストを並行して発行することが困難となる。我々はこの API を通して複数のデバイスを協調動作させることも目標においていたので、ioctl による実装は避けた。

ソケットへのマッピング

次にソケットの枠組の中で、先の条件を実現する方法について検討する。

まず、ソケットを生成する際にはドメイン、タイプ、プロトコルの 3 種類のパラメータを指定する。例えば、インターネットプロトコルの TCP ソケットをオープンする時は、ド

メインは AF_INET, タイプは SOCK_STREAM, プロトコルは null が指定される。この点に留意して以下の議論を進める。

まず EUI64 による機器の認識については、ソケットレイヤの中に EUI64 とノード ID の識別子の対応表を持つことで解決可能である。

2 種類のタイプ 対象とする EUI64 の指定は、ソケットを open してから bind によって指定する方法と、パケット毎に指定する方法がある。ユーザアプリケーションが特定の機器を制御するには前者が適しており、IEEE 1394 バスに接続された機器の情報収集や監視を実現するには後者が適する。このため、ソケットに複数のタイプを設けてどちらの方法でもアクセスを可能とした。前者を SOCK_DGRAM に、後者を SOCK_RAW に割り当てている。実運用では SOCK_RAW はスーパーユーザによる特権アクセスを要求し、SOCK_DGRAM は一般ユーザによる使用が可能とするのが望ましいだろう。

inbound/outbound の 2 種類のプロトコル トランザクションの並列処理については、リクエスト/レスポンスの送受信をそれぞれソケットへの read/write にマッピングする。outbound の場合ソケットへの write としてリクエストを発行し、read としてレスポンスを受けとる。inbound の場合はこの逆となる。リクエストが並行して発行できるならば、受けとったレスポンスがどのリクエストに対応するのかを識別しなければならない。IEEE 1394 ではトランザクションラベルと呼ばれる識別子をリクエスタがパケットに付加することにより、識別している。

単純に考えるとソケットへの出力 (write) の返り値としてトランザクションラベルをアプリケーションが受けとることにより、リクエスト/レスポンスを対応づけることになる。だが、ソケットインタフェースでは、ジェネリックにソケットに書き込まれたバイト数が返されるように実装されている。ここではソケット実装の変更を避けるため、アプリケーションがリクエスト ID と呼ばれる識別子を付加し、対応するレスポンスにその ID を付けることでリクエストとレスポンスの対応づけの問題を解決した。

outbound, inbound の操作はそれぞれ別のソケットから行うこととした。outbound, inbound の区別はソケットの protocol として指定する。outbound, inbound を別のソケットとした理由は、IEEE 1394 では一つのアプリケーションは outbound または inbound どちらか一方しか使わない場合が多いと考えていたためである。

だが、AV/C プロトコルでは必ず両方のタイプのソケットを使うことがわかったため、この点は再度検討する余地がある。

表 1.1にソケットのタイプ及びプロトコルの一覧を示す。

2.3.2 手順

raw モードの IEEE 1394 ソケットの使用手順の概略を説明する。

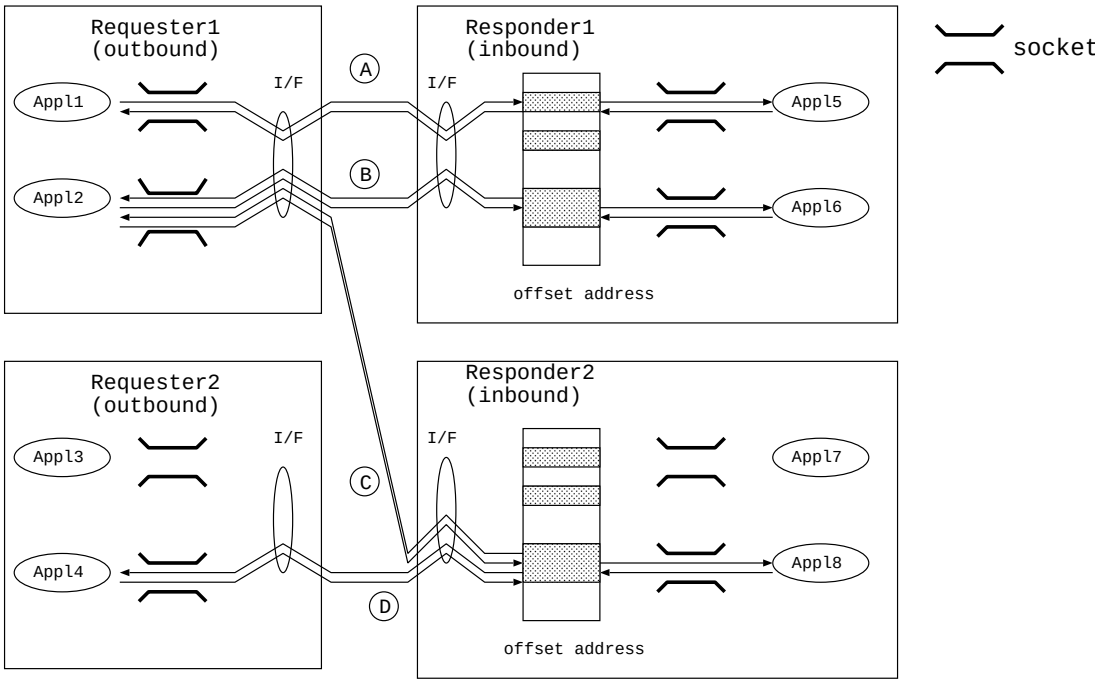

図 2.14: トランザクションの多重化

type	protocol	値	機能
SOCK_RAW	Raw Asynchronous outbound	0	パケット毎に任意のノードにトランザクションを発行
	Raw Asynchronous inbound	1	local offset への bind が必要
SOCK_DGRAM	Cooked Asynchronous outbound	0	target への connect が必要 (リソース管理あり)
SOCK_STREAM	Cooked Isosynchronous	0	リソース獲得手順あり

表 2.2: IEEE 1394 ドメインのタイプおよびプロトコル一覧

raw socket(SOCK_RAW)

Outbound

1. ソケットオープン
2. デバイスのインタフェースへの関連づけ (ioctl)
3. (バスに接続されているノードの EUI64 一覧取得)
4. トランザクション識別子を付加して write システムコールによってトランザクション発行
5. read システムコールでレスポンスの受信。トランザクション識別子によってリクエストと対応づける

6. close

ソケットを作成した後、ソケットと IEEE 1394 インタフェースを関連づける。IEEE 1394 デバイス一覧の取得は ioctl システムコールによって行なう。IEEE 1394 ドメイン共通の ioctl の一覧を表 2.3に示す。次に制御対象機器の探すためにソケットに関連づけたバスに接続されている IEEE 1394 ノードの EUI64 一覧を取得し、リクエストを発行する機器の EUI64 を知る。

INET ドメインではターゲット IP アドレスがアプリケーションによって指定されれば、そのパケットが出力されるネットワークインタフェースはカーネルによって自動的に決定される。当然ここに示したようなソケットとインタフェースの対応づけは必要ない。

残念ながら現在の IEEE 1394 実装では複数の IEEE 1394 インタフェースには対応しているものの、パケットのターゲット EUI64 から動的に出力インタフェースを決定する機構は実装していない。インタフェースをソケットに対応づけ、パケットのターゲットが存在するソケットを選択してパケットを出力するのはアプリケーションの責任となる。 リクエ

名前	入力/出力	意味
FWIOCSIFDSTADDR	出力	ソケットと 1349 インタフェースの関連づけ
FWIOCGIFLOCEUI	入力	1394 インタフェースから自ノードの EUI64 を読み出す。
FWIOCGIFALLEUI	入力	1394 インタフェースに接続されている全てのノードの EUI 64 を取得
FWIOCSIFRCVCHN	出力	Isochronous の受信チャネルを設定
FWIOCSIFRCVCHN	入力	IEEE 1394 ドメインで使用可能なデバイス名の一覧

表 2.3: ioctl 定数一覧

ストを発行する時には、図 2.15に示す形式のデータを使う。これは、IEEE 1394 のリクエストパケット形式に EUI64 と リクエスト ID のフィールドを加えたものである。EUI64 によってターゲットの機器を指定し、リクエスト ID は当該ソケットで一意となる識別子をアプリケーションが与える。

この形式のデータを write システムコールによりソケットに書き込むとリクエストが発行される。メッセージ形式にエラーがあった場合、あるいは指定された EUI64 に対応するノードが存在しない場合はエラーの -1 を返す。リクエストが正常にキューイングされればデータのバイト数を返す。

アプリケーションは、当該ソケットから発行されたリクエストに対するレスポンスを、ソケットの受信データとして、図 2.16に示す形式で読みとることができる。レスポンスに含まれるリクエスト ID から、そのメッセージがどのリクエストに対応するかがわかる。リクエストメッセージにはトランザクションラベル (tlabel) が含まれているが、これはデバイスドライバ中で書き換えられてしまうため、レスポンスメッセージにかかれているトランザクションラベルには反映されない。

リクエストが成功したかどうかはレスポンスメッセージに含まれるステータス (status) とリターンコード (rcode) によってわかる。ステータスおよびリターンコードの一覧を表

2.4,2.5 に示す。ステータスは送信の結果を、リターンコードは受信側から返された結果を表している。

バスリセットが発生した時、ソケットはプロセスに SIGURG を送る。これは、例えばソケットオープン時に取得した接続機器一覧の更新に使われる。

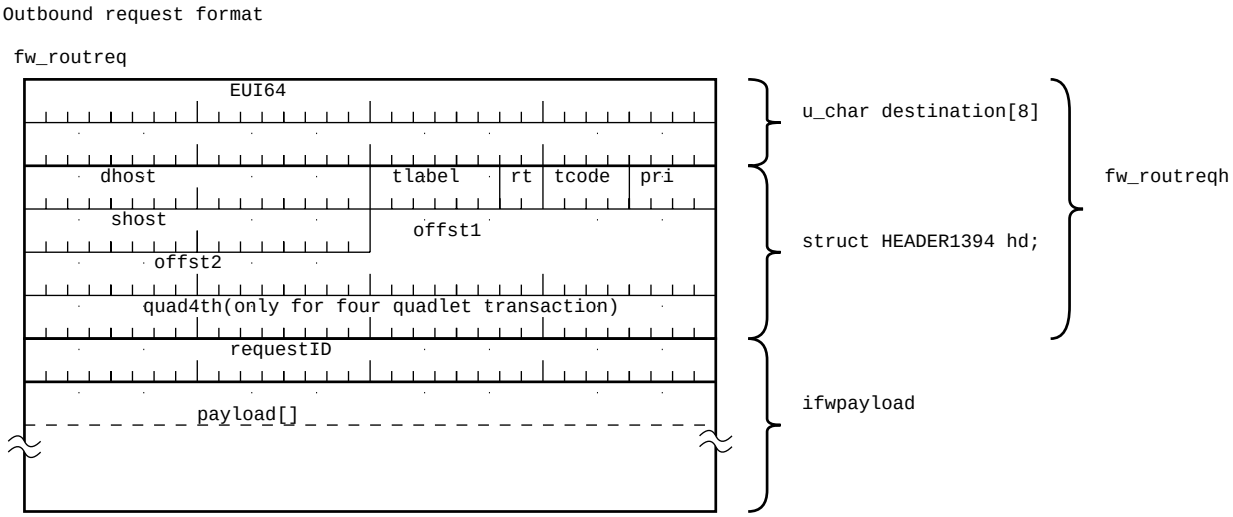

図 2.15: raw outbound request message format

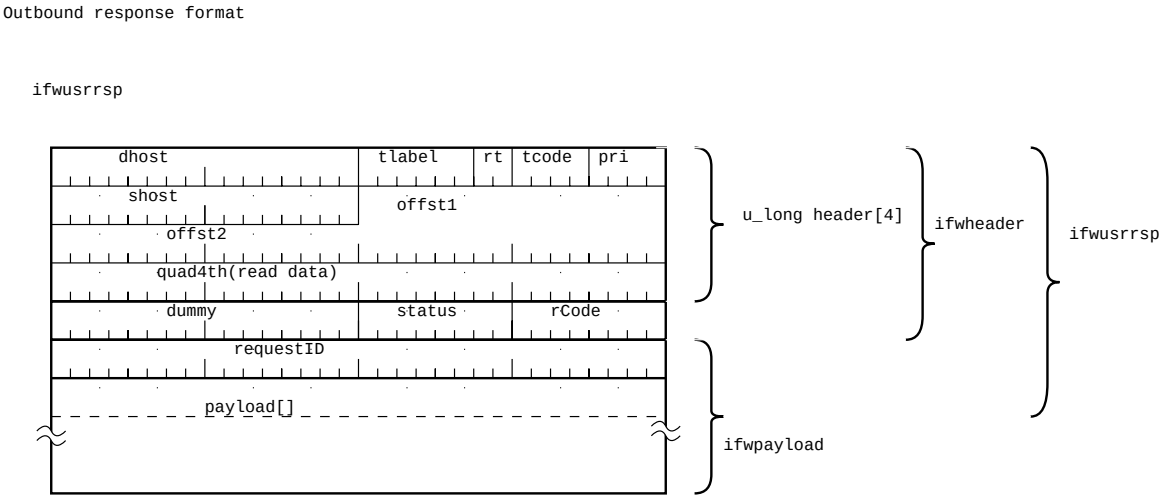

図 2.16: raw outbound response message format

Inbound

名前	値	意味
TSTATUS_COMPLETE	0	トランザクションの正常終了
TSTATUS_TIMEOUT	1	トランザクションレスポンスのタイムアウトエラー
TSTATUS_ACK_MISSING	2	トランザクションリクエストに対する ACK がない
TSTATUS_RETRY_LIMIT	3	busy ACK によるトランザクションの再試行が上限を越えた
TSTATUS_DATA_ERROR	4	データエラー
TSTATUS_RESET	5	発行前またはレスポンス受信前のバスリセットによるトランザクションの廃棄

表 2.4: status 一覧

名前	値	意味
RCODE_COMPLETE	0	トランザクションの正常終了
RCODE_CONFLICT_ERROR	4	発行したトランザクションのタイプとレスポンスが一致しない[1]
RCODE_DATA_ERROR	5	受信側でデータエラーを検出
RCODE_TYPE_ERROR	6	指定されたトランザクションタイプは無効
RCODE_ADDRESS_ERROR	7	指定された offset が無効

表 2.5: rcode 一覧

1. Socket open
2. (使用デバイスの決定)
3. デバイスのインタフェースへの関連づけ (ioctl)
4. bind で受信する offset 範囲とトランザクション種別を指定。
5. read でトランザクション要求の受信。write でトランザクション応答の送信。
6. close

inbound の手順は、デバイスのインタフェースへの関連づけまでは outbound 場合と同様である。次に bind システムコールによってソケットとオフセットアドレスの範囲およびトランザクション種別とが関連づけられる。bind システムコールでは、オフセットアドレスの範囲とトランザクション種別とは sockaddr の形式で与えられる。データ形式を図 2.17 に示す。sockaddr に EUI64 が含まれているので、本来 ioctl によるデバイスへの関連づけ手順は必要ないが、現在の実装では必要となっている。

<table>
<tr><td rowspan="2">sa_len
(1)</td><td rowspan="2">sa_family
(1)</td><td colspan="5">data(20)</td></tr>
<tr><td rowspan="2">eui64(8)</td><td colspan="2">offset(8)</td><td rowspan="2">length
(2)</td><td rowspan="2">flag
(1)</td><td rowspan="2">dummy
(1)</td></tr>
<tr><td>offset_hi</td><td>offset_lo</td></tr>
</table>

図 2.17: sockaddr_fw の構造

bind に成功すると、リクエストパケットの受信を待つ。リクエストパケットの形式を図 2.18 に示す。inbound ではリクエストの発信ノードを EUI64 で表現することはしていない。リクエストに対して応答する時にはリクエストパケットに含まれているリクエストの送信ノード宛に応答を返せばよいからである。EUI64 を判断する必要がある場合はアプリケーションがノード ID から EUI64 への解決を行う。

リクエストに対応する処理が完了すると、アプリケーションはレスポンスのデータを作り、write システムコールを発行してレスポンスパケットを送信する。複数のリクエストがある場合でも、パケットにトランザクションラベル (tlabel) フィールドがあるのでリクエスタではレスポンスに対応するリクエストがわかる。

バスリセットの発生は、outbound の場合と同様シグナル SIGURG によってアプリケーションに通知される。バスリセットが発生した時、リクエスタ側でトランザクションレイヤに保持している発行済リクエストは破棄されると IEEE 1394 の使用では決められている。したがって、バスリセット発生以前に受けとったリクエストに対するレスポンスを送信する必要はない。write システムコールによってキューイング済みのレスポンスパケットは廃棄される。この廃棄はアプリケーションには通知されない。バスリセット発生の時点でレスポンスが未送信のトランザクションに対する処理はアプリケーションに委ねられる (通常破棄される)。

IEEE 1394 ではバスリセットによってノード ID が変わるが、前述のようにリクエスタの側でバスリセット発生以前のトランザクションが廃棄されてしまうため、この問題を細かく考慮する必要はない。リクエスタ側ではバスリセットの前後でトランザクション ID が重複しないようにすることが望ましい。

データの誤りやバッファのオーバフローによるパケットの廃棄が生じても、アプリケーションには通知されない。従って、トランザクションレイヤあるいは、IEEE 1394 コントローラが再送機能を持っていることが望ましい。現在の実装では IEEE 1394 コントローラによる自動再送に依存している。

アドレス sockaddr_fw の形式　BSD TCP/IP 実装における sockaddr_in に相当する構造体が sockaddr_fw である。sockaddr_fw の各フィールドを表 2.6に示す。 TCP/IP の IP アドレスは IEEE 1394 の EUI64 に相当し、ポート番号はオフセットアドレスに相当する。ただし、TCP のポート番号は source/destination ともに存在するが、オフセットアドレスは target(request を受け入れる側) のみ指定されることに注意しなければならない。

*1 ノード ID(16bit) と合わせて IEEE 1212 における 64 bit アドレス空間の位置を指定する

2.3.3　実装

図 2.19に IEEE 1394 ソフトウェア実装の概念図を示す。1394 管理デバイスと呼ぶ IEEE 1394 の中心機能の実装 (ウィンドウズでいうクラスドライバ) の上に native 1394 のデバイスドライバと、native 1394 のプロトコル実装が載る。

raw Inbound request/response format

ifwusrrsp

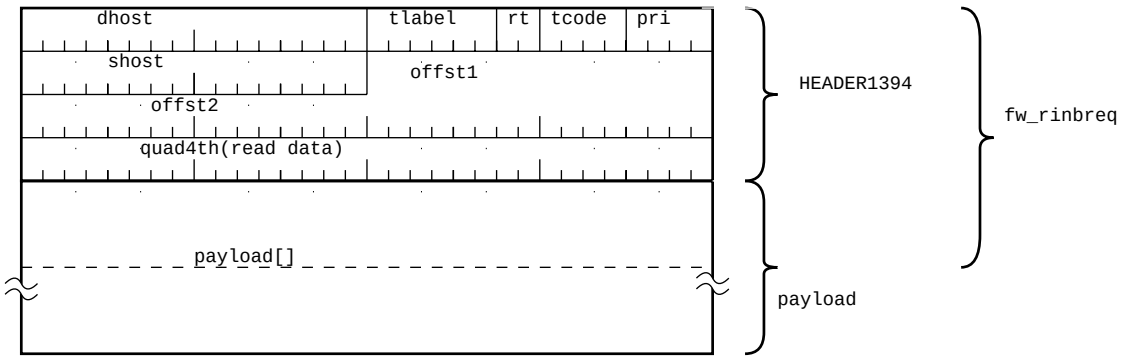

図 2.18: raw inbound request/response message format

名前	サイズ (byte)	意味
eui64	8	EUI64
offset	6	オフセットアドレス (48bit) *1
length	2	1394 パケットのペイロードの長さを指定する。
type	1	発行する、またはされるトランザクションのタイプを表す。

表 2.6: sockaddr_fw

開発環境として OS に BSD/OS2.1 、IEEE 1394 コントローラボードに富士フィルムマイクロデバイス製のチップ評価用 PCI カード (SA2212) を使用した。

native 1394 のデバイスドライバには、機能的にはパケットのキューイングと EUI64 からノード ID への変換機能、それに ioctl の実装が含まれる。native 1394 のプロトコルレイヤは、アプリケーションから発行されたリクエスト/ レスポンスメッセージのフォーマットの検査、ペンディング (リクエストの受信/タイムアウト待ち) リクエストのトランザクションラベル/リクエスト ID の保持などの機能を持つ。

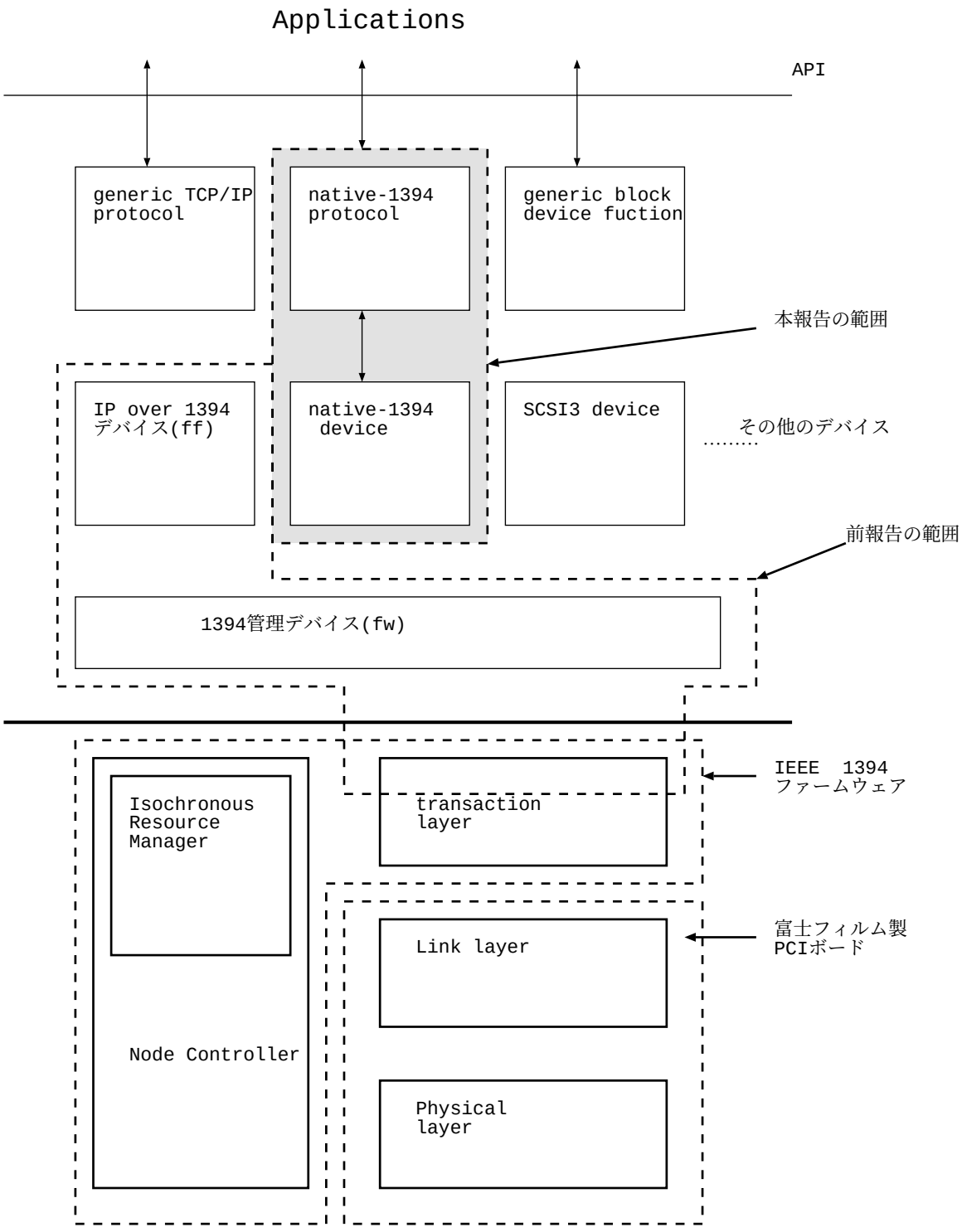

図 2.19: ドライバのアーキテクチャ概念図

第 3 章

その他の活動

3.1 1394 over IP のジッタ特性

2.1.1節で最新のルータで構成されたバックボーンネットワークでもジッタが制御できないという定性的な結果を示した。ここでは現在の高速専用ルータを一段通ることで、どれだけのジッタが発生するのかを定量的に議論する。

受信側 Comet の IEEE1394 ドライバにおけるパケット処理開始時刻を測定して、パケット間隔を算出した。ジッタが制御されていればパケット間隔は 125 μ秒になる (ファームウェア main loop 1 周程度の誤差は出る)。一方、ジッタが一定に保てないと、パケット処理間隔が非常に大きくなったり、逆にファームで 2 個一度にパケットを処理してしまうことが起こる。後者ではパケット処理間隔が 10 μ秒以下としてカウントされる。時刻の計測には Comet ボード上にあるタイマを利用した。DV ストリームに含まれる 12byte の短いパケットについては DMA 転送やネットワーク伝送に要する時間の違いで 125 μ秒単位でパケットが届かないので、その前後ではパケット処理間隔を測定していない。また、パケットロスが発生した場合もその前後では測定を行わないようにした。但し、以下の実験ではパケットロスは発生しなかった。

3.1.1 実験システム

ルータを通さず Comet を直結した場合とルータを通した場合の遅延の揺らぎを計測した。以下ではルータを通る場合の実験システム構成について述べる (図 3.1)。CISCO7500 に 100Base-TX と 100Base-FX のネットワークカードを装着する。それぞれのカードには 2 つずつインターフェースがついており、これで分けられる 4 個のネットワークに 1 台ずつ Comet を配置しルータを越えて DV ストリームを流した時のパケット間隔を測定した。以下の 3 ケースについて計測した。

1. バックプレーンを通らず 1 チャネルビデオストリームを通した場合:
   Comet2 から Comet1 に向けて DV ストリームを 1 チャネル流す。
2. バックプレーンを通って 1 チャネルビデオストリームを通した場合:
   Comet3 から Comet1 に DV ストリームを 1 チャネル流す。

3. バックプレーンを通って 2 チャネルビデオストリームを通した場合:
   Comet2 から Comet4、Comet3 から Comet1 にビデオストリームをそれぞれ 1 チャネルずつ流す。

測定はすべてのケースで Comet1 で行った。3 番目のケースでは Comet2 から Comet4 に向けて流れる DV ストリームは Comet1 が受ける DV ストリームに対してバックプレーンで衝突するノイズになる。

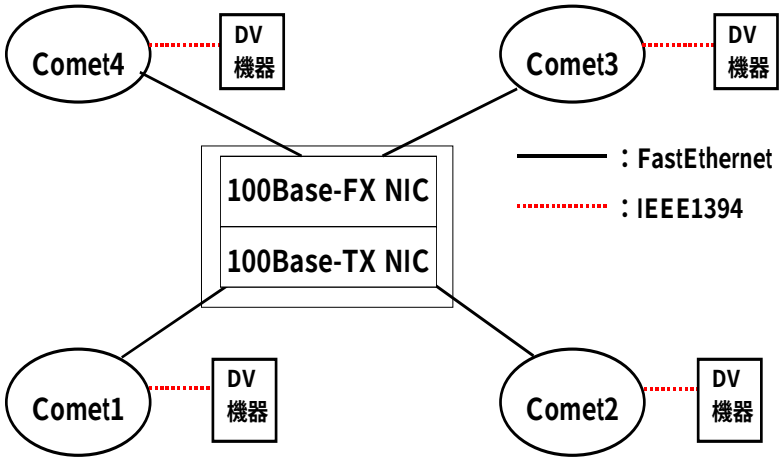

図 3.1: ルータを用いたジッタ測定実験システム構成

3.1.2　測定結果

測定結果を図 3.2に示す。20 万回パケット処理間隔を測定し、16tick(約 15 μ秒) のきざみで分布を求めて表示したものである。Comet ファームウェアで 2 個のパケットを 1 度に処理した場合のパケット処理時間間隔は 10 μ秒未満なので、グラフでは 0 μ秒として表示されている。

- Comet 直結の場合には 125 μ秒を中心とした正規分布に従う。また、0 μ秒の所が 0 度に処理することもなく、ジッタが制御できている。
- ルータを 1 段通るとパケット処理間隔が 300 μ秒弱まで膨らむ。つまり、ルータを 5 段も通れば、1.5m 秒にも迫るジッタになる。また、パケットを 2 個同時に処理することが起きた。
- 一方で バックプレーンを通る、通らないやバックプレーンに 30Mbps のノイズが有るか無いかの影響ははっきりと見えなかった。

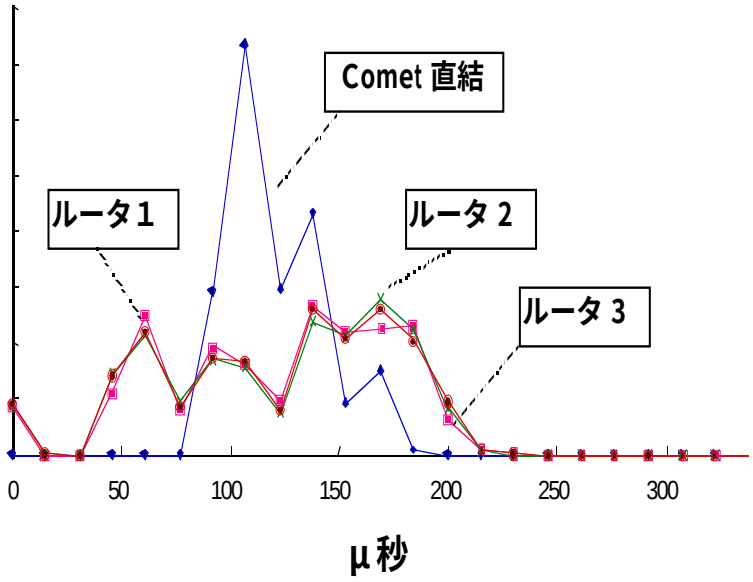

図 3.2: パケット間隔測定結果

WIDE バックボーンでの実験では画像が激しく乱れたが、今回の測定中は DV ストリームを 2 チャネル流しても時々ブロックノイズがのる程度でかなり高品質の再生ができ、現在のバックボーンネットワークの振る舞いが完全に再現できたわけではない。ルータを多段接続した運用状態のネットワークでジッタを定量的に評価するのが今後の課題である。

第 4 章

合宿や研究会での BOF の開催

4.1 98 年 5 月研究会でのホームネットワーク BOF の開催

98 年 5 月に奈良先端科学技術大学院大学で開かれた研究会において、ホームネットワーク BOF を開催した。参加者は 13 名。

ここでは、話題を IEEE1394 に限定せず、広く家庭内及び家庭間ネットワークに関する議論を行った。具体的には、以下の点について話し合い、メーリングリストを作って議論を継続することが合意された。

- 家庭内及び家庭間における高速インフラストラクチャーの可能性
- 家庭間に高速回線が引かれた場合のアプリケーションモデル
- 地域に密着したアプリケーションの可能性
- 家庭間や地域内において手軽にリアルタイム画像を送ることができるようになった場合の可能性と問題点

4.2 98 年 9 月合宿での BOF の開催

98 年 9 月に熱海で開かれた合宿において、BOF を開催した。内容は以下の通りである。

- IETF での IP1394 の動向報告
- IP1394 規格の IP マルチキャストの実現方法の説明
- IPv6 over IEEE1394 の議論
- 1394 機器制御用 API の議論
- 1394 over IP 実験における DV トラヒックの IP バックボーンルータ通過の際のジッタ値測定実験

- ホームゲートウエイの議論

その後、今後の活動予定を討議し、1394 機器制御用 API の議論を WG のチャータとして加えることとした。

4.3 99 年 3 月合宿での BOF の開催

99 年 3 月に伊東で開かれた合宿において、BOF を開催した。内容は以下の通りである。

- IETF での IP1394 の動向報告
- DV over IP 実験の報告
- 1394 機器制御用 API の実装報告
- IP over 1394 同期チャネルの議論

その後、本 WG の趣意書により、本 WG の活動期限が 99/3 までとなっていたため、本 WG の今後の活動についての議論を行ない、来年度も WG の活動を継続することとし、WIDE に趣意書を提出、更新された。

第 5 章

今後の予定

これまで、本 WG では「IPv4 over 1394、及び IPv6 over 1394 の IETF 規格の相互接続試験」を主な目的として、チャータを組んできたが、「WG 活動の範囲を制限せず、1394 に関してジェネリックに研究できるテーマを WG の目的として、活動をしくべき」とのコンセンサスを得、来年度は、以下のようなチャータで活動を行なっていくこととした。

「IEEE1394 は、低コスト高速バス (〜 400Mbps)、同期データ転送 (QOS 転送機能) のサポート、デジタル家電業界によるデジタル AV インタフェースとしての採用等、多くの注目すべき点を持っているネットワーク/バス技術である。ネットワークと家電/AV の融合、あるいは低コストネットワークを考えた場合、IEEE1394 は、今後非常に重要になると考えられる技術である。本 WG では、この IEEE1394 とインターネットの融合技術の研究開発、及び IEEE1394 開発環境の整備を目的として取り組んでいく。」

99 年度は、1394API 策定、1394/DV over IP、IP over 1394 同期チャネル、1394 ライブラリ作成等をテーマに、WG 活動を具体的に行なっていく予定である。