システムソフトウェアと
オペレーティング・システム 81－27
（１９９８．５．７）

デジタル家電機器制御のための IEEE1394 UNIX API と Java APIの設計と実装

橋本 幹生[1]、斉藤 健[1]、岡本 利夫[1]、　　徳田 英幸[2]

[1](株) 東芝 研究開発センター　　[2] 慶應義塾大学 環境情報学部

本発表では、Java API を持つ家庭ネットワークのための機器制御プラットホームの試作について報告する。試作ソフトウェアは、家庭内におけるデジタル家電機器の使われ方の検討に基づき、家庭ネットワークにおける動的構成変更や制御手段である IEEE 1394 の特性を考慮したものである。併せて Java API の土台となる BSD UNIX 上の API も紹介する。本実装は家庭ネットワークにおける機器制御アプリケーションの研究プラットホームとなる。

Design and Implementation of IEEE 1394 UNIX API and Java API for Digital Consumer Appliances

Mikio Hashimoto, Takeshi Saito, Toshio Okamoto,　Hideyuki Tokuda

Toshiba Corp. R&D Center　　Keio University

In this paper, we introduce an experimental implementation of UNIX API and Java API for IEEE 1394 suitable for networked home appliances, such as digital VCR. This Java API is designed considering characteristics of such home appliances and their dynamic configuration capability. UNIX API underlying Java implementation is also described. This API provides an experimental platform for home network environment.

1　はじめに

デジタル VCR をはじめとしたデジタル家電機器が家庭環境で使われはじめている。デジタル家電機器には、劣化のない画像編集やパーソナルコンピュータ (PC) から制御可能な周辺機器の拡大といった利点があるのはもちろんだが、それ以上にデジタルインタフェースを持った家電機器が相互接続され、ネットワークを形成して、新しい形の情報処理が可能になる点に注目が集まっている [1, 2, 3, 4]。

2　デジタル家電機器の制御環境

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。我々はこのような家電機器を、Java から制御するための API を試作した。本 API を使って、家電ネットワーク (IEEE 1394) に接続された機器を Java が動作する PC から制御する環境が構築できる。ここで、ターゲット機器は Java VM を持つことなく単純な家電機器制御プロトコル (AV/C) で制御される。Java を使用した理由は、ネットワーク環境が考慮されていることと、マルチプラットホーム対応であることが理由である。家庭環境で、proprietry な制御プロトコルを持つ装置をコントローラの Java API から制御することには次のような意味があると、我々は考えている。

コストの制約 低価格なターゲットが利用可能

多様なターゲット機器への対応 ターゲット自身がバイトコードを持ち、コントローラにアップロードすることで多様な機器でプラグアンドプレイを実現

複数のコントローラ 家庭ネットワークには、PC, PDA, セットトップボックスのように、複数のコントローラが存在し、コントローラの OS プラットホームは統一されていない (図 1)

ただし、Java で IEEE 1394 機器を制御する API はまだ標準化されていない。Write once Run everywhere を理念とする Java にとって、API の標準化は大変重要な問題である。現在、IEEE 1394 上の

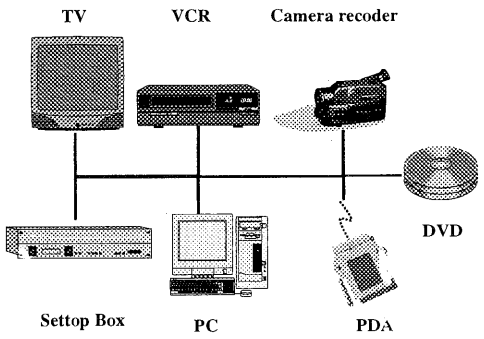

図 1: 家庭ネットワーク

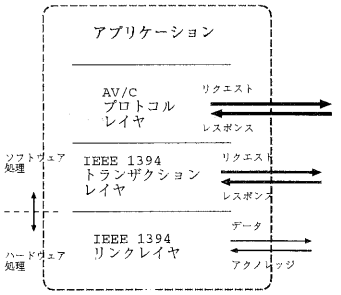

図 2: プロトコルスタック

家電機器制御 API として HAVi, JINI などの標準化候補があるが、本実装はそれらのいずれでもない独自の仕様である。我々は本実装を、標準化に先行して実際に機器を制御できる研究開発プラットホームを提供し，その応用を通じてアプリケーションや API そして OS プラットホーム自身を改良していく環境を構築するものと位置付けている.

以下、IEEE 1394 と家電機器制御プロトコルの概要、IEEE 1394 プロトコルの UNIX API へのマッピングと家電機器制御 Java API についてそれぞれ説明する。

3 家電機器制御プロトコル

本節ではデジタル家電機器の制御プロトコルである AV/C プロトコルと IEEE 1394 インタフェースの特徴を説明する。機器制御の観点から見た時のこれらのプロトコルの特徴は、ソフトウェアの状態数を少なくした上でローカルなネットワーク上で信頼性のある通信機能が提供できることである。

図 2 に AV/C と IEEE 1394 のプロトコルスタックを示す。IEEE 1394 以外のインタフェースの使用も AV/C 規格では考えられてはいるが、現時点でのターゲットは IEEE 1394 インタフェースを持つ装置といってよい。

次に IEEE 1394 と AV/C のプロトコルを下位レイヤから順番に見ていく。リンクレイヤは送信側ノードがデータを送信する DATA とそれを受信したノードがその処理状態を送信側にフィードバックする ACKNOWLEDGE からなる。DATA が受信側のバッファに格納されると、データが受信できたことを示す ack_complete または ack_pending ACKNOWLEDGE パケットが受信側から返される。受信側の ACKNOWLEDGE パケット送信機能は、リンクレイヤをサポートする LSI によって実装され、受信バッファがオーバフローする場合には ack_busy を返す。このため、受信側データバッファのオーバフローによるデータの消失は送信側で検出され、これもまた送信側の LSI レベルで実装されるリトライにより補償される。CRC コードによって検出されるデータのエラーも ack_data_error として送信側に伝えられ、再送処理が行われる。ACKNOWLEDGE によって伝えられる受信状態の一覧を表 1 に示す。ack パケットがデータエラーなどで消失した場合には、送信側でタイムアウトが検出され、再送もしくは異常処理が行われる。タイムアウト時間はハードウェア処理がされることを前提として 3.5μs 前後 (ノード数などで変化する) と極めて短い (表 2)。

IEEE 1394 ではこの ACKNOWLEDGE による

表 1: ACKNOWLEDGE コード一覧

名前	意味
ack_complete	受信成功, トランザクション完了
ack_pending	受信成功, トランザクション継続
ack_busyX ack_busyA ack_busyB	データの受信失敗
ack_data_error	伝送エラーによる受信失敗

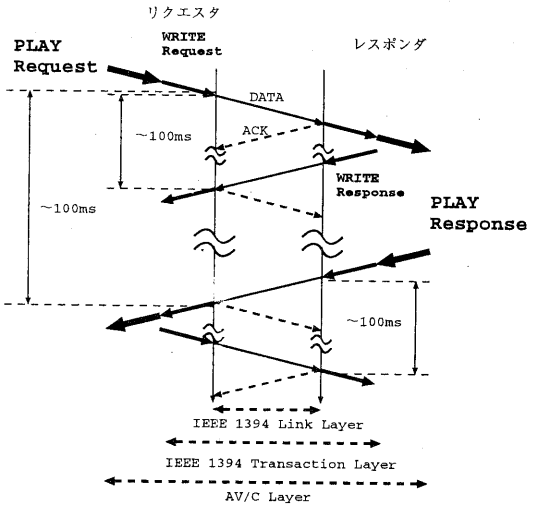

図 3: AV/C プロトコルシーケンス

ハンドシェイク機構によってバッファオーバフローによるデータの消失を回避している。一方、一般的な LAN ではバッファオーバフローを防いでいるのは TCP に代表されるソフトウェアによるフロー制御機構であるが、これはソフトウェアの負荷が大きい。IEEE 1394 ではハンドシェイク機構がリンクの LSI に実装されているため、ソフトウェアの負荷は小さくなる。これは制御用プロセッサに割り当てられるコストが制限される家電機器にとって有効であることはいうまでもなく、とりわけ IEEE 1394 のように極めて高速なデータ転送速度のインタフェースを比較的低速な組み込み用プロセッサで制御する場合に有効である。もちろん、この ack 機構による通信の信頼性の保証はローカルな IEEE 1394 バスに限定されるが、導入期の家庭ネットワークの大きな課題である機器のコスト高を解決する有効な手段である。

リンクレイヤの上のトランザクションレイヤでは、ノード上に仮想的に設けられたオフセットアドレス (もちろん被制御装置の実メモリと一致していてもかまわない) への read/write/lock のオペレーションを要求するパケットが送信側のリクエスタから受信側 (被制御装置) のレスポンダへと送られる。レスポンダのリンク LSI はすぐに ack を返し、受信したリクエストをレスポンダノードのコントローラが処理して対応するオペレーションを行い、結果をレスポンスパケットに格納してリクエスタに送る。レスポンスに対しても送信側のリンク LSI が ACKNOWLEDGE を返す。このトランザクションレイヤも基本的にはハンドシェイクであり、トランザクションレイヤのリクエスト、レスポンスのシーケンスは 2 つの完結したリンクのシーケンスからなっている。トランザクションにもタイムアウトがあり、その値は通常 100ms である (表 2)。

表 2: AV/C と IEEE 1394 のタイムアウト時間

レイヤ	タイムアウト値
AV/C	100ms (∞)
1394 トランザクション	100ms(デフォルト)
1394 リンク	3.5 μ s

AV/C とトランザクションレイヤの関係もトランザクションレイヤとリンクレイヤの関係にほぼ一致する。AV/C プロトコルではコントローラが AV/C リクエストを IEEE 1394 の write リクエストによって送ると、ターゲット (被制御機器) がレスポンスをこれもまた IEEE 1349 の write レスポンスによって返す (図 3)。例えば、被制御機器として VCR を例にとると、AV/C プロトコルで定義された再生を意味する PLAY コマンドをコントローラからターゲットへと送られると、VCR は再生を開始するとともに再生を開始したという現在状態を含んだレスポンスをコントローラへと送り返す。もしこのとき再生が不可能な状態 (テープなしなど) ならば、その旨の情報を含むレスポンスが返される。

4 IEEE 1394 UNIX ソケット API

家電制御のための Java API ライブラリのプラットホームとして、UNIX Socket API とデバイスドライバを実装した。UNIX API レベルでは IEEE 1394 のトランザクションを処理する機能を提供している。

デバイスドライバは、図 4 に示す SCSI インタフェースに似た階層構造をとっている。IP over IEEE 1394 や、家電機器制御に使う IEEE 1394 トランザクションを UNIX API から制御するための raw IEEE 1394 ドライバが、IEEE 1394 コントローラ自体を

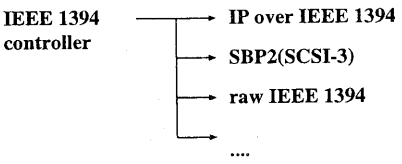

図 4: デバイスドライバ階層

制御する IEEE 1394 コントローラの下に位置している。

API はシステムコールを新たに作ることなく、既存の枠組を拡張することとした。家電機器の画像データを扱うには Isochronous パケットの入出力が必要となるが、機器の制御に限れば Asynchronous トランザクションで十分であり、ここでは Asynchronous トランザクションに限って話を進める。IEEE 1394 トランザクションを UNIX API にマッピングするとき、大きく分けて 2 通りの方法が考えられる。一つの方法は 1 トランザクションを一つのシステムコールとして処理し、システムコールの返り値としてトランザクションのレスポンスをもらう方式である。もう一つはトランザクションのリクエストとレスポンスを別々のシステムコールで送受信する方式である。ソケットに関して言えば、前者は ioctl に対応し、後者は send/recv に対応する。我々は複数のリクエストを並行して扱うことのできる後者を実装の方式として選んだ。理由は応答の遅い家電機器を制御する時の多重度を上げるためである。

一方、read/write には欠点もある。複数のトランザクションを並行して発行するときには、リクエストとレスポンスとの対応づけが必要となる。IEEE 1394 ではトランザクション ID という 8bit の識別子でこの対応づけを行っているが、この ID はデバイスドライバの中で与えられるため、ソケットに send システムコールを発行した時点ではアプリケーションはその値を知ることはできない。また、ソケットレイヤで識別子を付与し、アプリケーションにその値を返す方法も考えられるが、BSD UNIX ではソケットの返り値はユーザ空間からコピーされたデータのサイズを返すようにコーディングされており、汎用性を保ちながら変更することは難しかったため、アプリケーションからソケットにおいて一意となる識別子を与え、対応するレスポンスにその値が付けられることで、アプリケーションがトランザクションのリクエスト—レスポンスを対応づけることを可能にした。

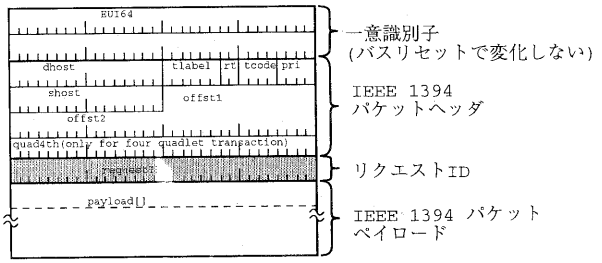

図 5: アウトバウンド側リクエストフォーマット

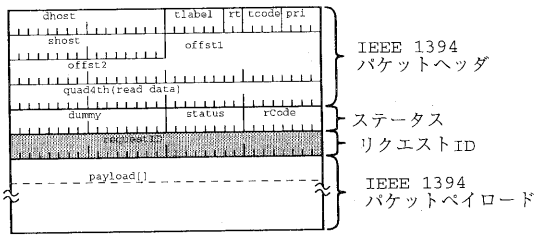

図 6: アウトバウンド側レスポンスフォーマット

トランザクションにはアウトバウンド (リクエスタ側) とインバウンド (レスポンダ側) の 2 種類があるが。この 2 つはソケット API では socket システムコールの protocol パラメータで区別され、それぞれ別のソケットとしてオープンされるものとした。

図 5,6 にアウトバウンド側のリクエスト、レスポンスフォーマットをそれぞれ示す。これらは IEEE 1394 のパケットに EUI64 とリクエスト ID をそれぞれ加えたものである。図 7 に示すプログラム例では、ソケットをオープンし、リクエストパケットにリクエスト ID を書き込んで sendto システムコールでリクエストを発行し、recv でレスポンスを読み込んでリクエスト ID が先に送信したものと一致しているかを確認している。

IEEE 1394 にはバスへのノードの追加、削除を検出して構成変更処理を行うバスリセットがある。バスリセットイベントが発生すると、SIGURG シグナルがソケットをオープンしているプロセスに発行され、バスリセットの発生が通知される。

```
sock = socket(AF_FWO,    SOCK_RAW ,FW_ROUB);
ioctl(sock, FWIOCSIFDSTADDR, "rf0");
……
req.requestID = REQUEST1;
sendto( sock, req,  sizeof(*req) ,0, &addr, 0 );
recv( sock, rcvbuf, size);
if(rcvbuf.reqestID == REQUEST1) {
   ……. }
close(sock);
```

図 7: プログラム例

5 IEEE 1394 Java API

UNIX API で提供されるトランザクションレベルで抽象化されたインタフェースの上に、AV/C などの家電機器制御プロトコルレベルで抽象化されたインタフェースを提供する Java ライブラリを定義する [5]。次の目標をおいてライブラリを設計した。

被制御機器のモデル化 最近の家電機器は 1 つの機器の中に複数の機能を持つものが多い。例えば家庭用ビデオはビデオテープレコーダ (VCR) と TV チューナの機能を持つ。DVD は使い方によってデータストレージ (SBP2) として使われることもあれば、ビデオテープレコーダ相当の装置 (AV/C) として使われることもある。IEEE 1394 や AV/C においてもこのような場合が考えられており、ひとつの IEEE 1394 インタフェースを持つ装置 (node) の中に仮想的に unit と呼ばれる機能が定義され、unit に対してオペレーションを行う。AV/C の API は unit に相当するクラスが提供する。

多重化 Java のクラスライブラリなので、当然複数のスレッドによる入出力に対応していなければならない。アプリケーションにおけるリクエストの扱いは、非同期入出力として扱われるよりも、リクエストに対応するメソッドコールがレスポンスが返るまでブロッキングして、レスポンスが返り値に反映される方が望ましい。アプリケーションが繁雑なリクエスト—レスポンスの待ち合わせ処理をしなくてもすむからである。もし複数のリクエストを並行して処理する必要があれば、複数のスレッドからメソッドコールすればよい。

汎用性 モデル化の項で説明したように機器はさまざまなプロトコルで制御される。一方、IEEE 1394 トランザクションの扱いなど、共通化できる部分も多い。共通部分は共有し、そこにさまざまなプロトコルを実装したライブラリを付加できるようにして実装の簡略化と構成変更に容易に対応できる構造としたい。特に、機器制御に必要なプロトコルを選択してローディングする事で、メモリ容量に制限のある PDA のようなコントローラにもこの制御ソフトが適用できる。

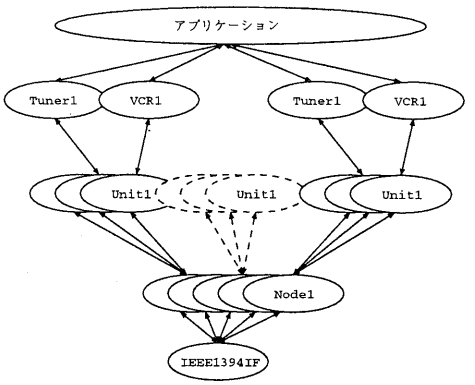

図 8: 階層構造

これらの要求を満たすために、ライブラリは図 8 に示すような階層的な内部オブジェクトの構成をとっている。一番下のレベルに IEEE 1394 インタフェースへのパケットレベルの入出力を制御する機能を提供する IEEE1394IF クラスのインスタンスがある。次に IEEE 1394 バスに接続された装置に対応する Node クラスのインスタンスがあり、IEEE1394IF クラスと相互参照を持つ。各々の Node インスタンスにはいくつかの Unit インスタンスが参照関係を持ち、、これも IEEE1394 または AV/C プロトコルで定義された Unit に対応する。そして、Unit に対してその機能をアプリケーションから利用するためのインタフェースを提供する VCR, Tuner といったクラスのインスタンスが参照関係を持つ。図の例ではバスに 5 つのノードが接続されていて、そのうち Node 1, Node3, Node5 に Unit が定義されている。

Node 1, Node 5 は家庭用ビデオであり、それぞれの Unit 1 と Unit 2 が VCR と Tuner の機能に対応している。アプリケーションは 2 台の家庭用ビデオの機能を使って、例えばビデオ編集操作ができる。

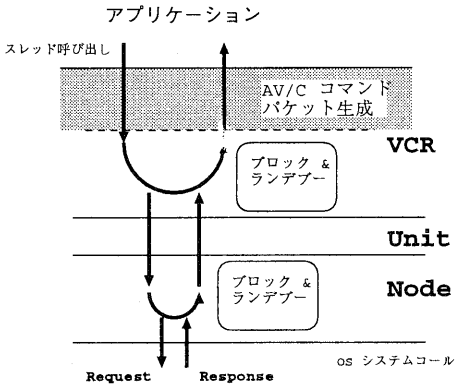

図 9: 動作概要

ライブラリの動作概念図を図 9 に示す。Node クラスは上位に対してリクエストの発行と対応するレスポンスのランデブー制御の機能を提供する。また、必要に応じて再送処理も行う。Unit クラスはノードに存在する仮想的な機能モジュールの抽象化を表現するだけで、特定の機能は担うものではない。Unit に対してリクエストを発行するメソッドを持つが、それは下位の Node クラスのリクエスト発行メソッドを呼び出すだけである。VCR, Tuner といったクラスは上位アプリケーションのメソッド呼びだしに対して、AV/C で定義されたリクエストパケットのフォーマットを生成し、Unit インタフェースを通じてリクエストを発行し、AV/C のレスポンスを待ち合わせて結果をアプリケーションに返す。

図 10 に VCR 制御のアプリケーションプログラム例を示す。コメントにあるように、これは VCR クラスのインスタンスである target に対して、5 秒間順方向再生後、5 秒間逆方向再生して停止するプログラムである。アプリケーションではリクエスト—レスポンスの待ち合わせを意識することなく Play, Wind のメソッドを呼び出すだけでビデオを制御できる。もちろん複数のスレッドを使って制御することが可能である

```
target.Play(PlaybackModeConst.FORWARD);    // 順再生指示
  Thread.currentThread().sleep(5000);       // 約 5 秒継続
state = target.TransportState();            // 状態取得
target.Play(PlaybackModeConst.X1_REVERSE);// 逆再生指示
   Thread.currentThread().sleep(5000);      // 約 5 秒継続
target.Wind(WindSubfuncConst.STOP);         // 停止
```

図 10: プログラム例

6 まとめと今後の課題

Java API を持つ家庭ネットワークのための機器制御プラットホームの試作について報告した。本ライブラリは Java 環境から IEEE 1394 プロトコルによる制御や動的再構成の検証を主眼として設計したため、家電機器 API としてはまだ未完成の部分が多く、機器の多様な機能に対応できるよう API を拡張する必要がある。さらに、今後はターゲット機器の動的構成変更への対応や、上位ミドルウェア、アプリケーションのネットワークを経由したダウンロードの機能拡張の研究を行いたい。

謝辞 本実装は IPA 高度情報化支援事業「ネットワーク指向 R^3 システム [9, 8] のための基盤ソフトウェアの開発」の一部として実施された。

研究について日頃御指導頂いている慶應大学 SFC 研究所の戸辺研究員に感謝いたします。

参考文献

[1] Wickelgren, I.J., "The facts about FireWire" IEEE Spectrum, vol. 34(4), pp17-, 1997

[2] Kitayama, T. et al.: "SMAF: A Framework for Future Embedded Systems", Proceeding of IWNA98(IEEE Workshop on Networked Appliances), 1998

[3] Hashimoto, M. et al: "A home network architecture considering digital home appliances", *ibid.*

[4] 斉藤, 他 "デジタル情報家電の接続を考慮した家庭ネットワークアーキテクチャ", 信学技法 IN97-128

[5] 橋本, 他 "家庭内ネットワークでの機器制御方式", 信学ソサイエティ大会 (1998) B-7-148

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

[7] 1394 Trade Association, AV/C Digital Interface Command Set General Specification Version 2.0.1, 1998

[8] Tokuda, H. et al: "Reliable, Real-Time and Reconfigurable System(R^3-system) Project", Proceeding of IWNA98, 1998

[9] http://www.mkg.sfc.keio.ac.jp/R3