

機器連携におけるネットワークミドルウェア統合システムの提案

高山 洋史[†] 小坂 隆浩^{††} 佐藤 健哉[†]

ユビキタスネットワークにおける重要な課題のひとつにネットワークを介して機器の機能を提供し合う機器連携がある。機器連携においては、ネットワークに横断的に配置することができるネットワークミドルウェアが必須となる。既存のネットワークミドルウェアに Jini, UPnP, HAVi などがありそれぞれ独自のネットワークドメインを構成している。ここでのネットワークドメインとは、機器連携が可能なネットワーク上の領域を指す。しかし、既存のミドルウェアはミドルウェア同士の相互接続性を確保していない。そのため、同一ネットワークに接続されている機器でもミドルウェアが違えば機器連携は実現できない。機器連携を実現するためにはネットワークミドルウェアの相互接続性を確保することが不可欠である。そこで、各ミドルウェアの抽象化の違いを統一するためのルールを規定し、プロトコル変換を行うトランスレータを用いたシステムの提案を行う。提案システムによりミドルウェア同士の相互接続性が確保され、理論上全ての機器で連携が可能となる。

Proposal of an Integrate System for Network Middlewares in Device Interaction

YOUJI TAKAYAMA,[†] TAKAHIRO KOITA^{††} and KENYA SATO[†]

In ubiquitous networks, device interaction is important and it provides functions each other via networks. For the device interaction, network middlewares that can connect networks are needed. Existing network middlewares, for example Jini, UPnP and HAVi, establish original domains. However, existing network middleware does not ensure network middleware - network middleware interconnection. To realize device interaction, network middleware - network middleware interconnection is required. Therefore, we propose a system integrating network middlewares. And, we propose a mapping rule to integrate difference of abstraction among network middlewares. The proposal system provides network middlewares interconnection and can realize device interaction for all devices.

1. はじめに

近年、ユビキタスに向けた取り組みのひとつであるホームネットワークにおける機器連携が注目されている。これまで単体で機能を提供してきた機器がネットワークに接続され、複数機器の機能を相互に利用することが可能になった。このように機能を提供しあうことを機器連携と呼ぶ。2008年3月に行われた総務省主催の次世代ホームネットワーク実証実験¹⁾においても機器連携に関する実証実験が多数行われた。実証実験では、特にホームゲートウェイ²⁾を用いた Ethernet, PLC, IEEE1394, IEEE802.11 など、有線、無線に関らない様々な物理的な回線の相互接続環境を実現した。

一方、機器連携においてネットワーク内の機器を相

互に利用することが可能なネットワークミドルウェアが重要である。各家庭内 LAN に接続された様々な接続形態の機器がネットワークミドルウェアを搭載することで、柔軟に機器連携が実現可能となる。ネットワークミドルウェアは、ネットワークを介したアプリケーションに共通な機能を API として提供し、アプリケーションから下位のシステムやネットワークの詳細を隠蔽している。現在、機器連携のためのミドルウェアに Jini³⁾, UPnP⁴⁾, HAVi⁵⁾ などがあり、各ミドルウェアは独自の方法を用いて機器連携を行っている。各ミドルウェアは独自のネットワークドメインを構築し、ミドルウェアを搭載した機器はこのドメインに参加する。ここでのネットワークドメインとは、機器連携が可能となるネットワーク上の領域を指し、ドメインに参加とは、機器またはサービスがドメイン内で利用可能なアドレスを持つことである。

しかし、既存のネットワークミドルウェアはミドルウェア同士の相互接続性を確保しておらず、物理的に接続されていても機器連携を行うことはできない。ユ

[†] 同志社大学大学院工学研究科
Graduate School of Sciences and Engineering, Doshisha University

^{††} 同志社大学理工学部
Faculty of Engineering, Doshisha University

ビキタスネットワークに向けた機器連携を行うためにはミドルウェア同士の相互接続性を確保することが不可欠である。本稿では各ミドルウェアを統合し、相互接続性を確保するシステムの提案を行う。提案システムを用いることにより、全ての機器で連携が可能となる。

2. 関連研究

異なるネットワークミドルウェア間の機器で連携を行うための研究がいくつか行われている。Web プロトコルを用いたホームコンピューティングミドルウェアの統合⁶⁾では、インターネットを介して機器の連携を行っている。インターネットと家庭内 LAN のゲートウェイを用いて複数の家庭内 LAN を接続し、機器連携のメッセージを HTTP 上で転送する。また、An Architecture for Jini/UPnP Interoperability⁷⁾がある。この研究では Jini と UPnP の機器を連携させることが可能となる。この研究では、UPnP - Jini の相互変換ゲートウェイを実現している。ゲートウェイ上でお互いのプロキシサービスを生成し、プロキシサービスを介してサービスを呼び出す。また、Universal Middleware Bridge⁸⁾は、複数のネットワークミドルウェアのメッセージのブリッジを実現している。各ネットワークミドルウェアごとにアダプタを作成し、アダプタ上で抽象機器を生成し、マッピングを行い機器連携を実現している。アダプタ同士がメッセージのやり取りを直接行う。メッセージを送信する際のルーティング表の生成は、UMB Core と呼ばれるモジュールが責任を持つ。

3. 相互接続に向けた問題点

3.1 機器の抽象化の違い

既存のネットワークミドルウェアは、上位アプリケーションから、下位システムやネットワークの詳細を隠蔽することで機器連携の開発を容易にする抽象化の概念を用いている。しかし、UPnP, Jini, HAVi などの既存のミドルウェアにおいては抽象化の方法が異なっている。抽象化方法は、大きく分けて機器とサービスを区別する方法と区別しない方法がある。例えば、UPnP では、機器をデバイス、機器が持つ機能をサービスとして抽象化を行っている。デバイスが複数のサービスを内包する形をとっている。また、UPnP と同様に HAVi でも、デバイスとサービスを分けて抽象化を行っている。一方、機器とサービスを区別しない Jini では、機器と機器が持つ機能であるサービスを区別しておらず、機器のソフトウェア、データベース

などの情報、そしてシステムを使うユーザまで全てをサービスとして抽象化している。抽象化方法が異なるとシステムが大規模・複雑化してしまう。したがって、異なる抽象化の違いを吸収する仕組みが必要となる。

3.2 サービスを示すアドレス体系の違い

ネットワークミドルウェアでは、ネットワーク上にあるサービスを識別するための識別子を用いてサービスへアクセスしている。サービスの識別子は、アプリケーションレイヤのアドレスにあたり、機器の識別を行うトランスポートレイヤのアドレスとは異なる。UPnP では、サービスの識別子を UPnP Forum⁴⁾で規定している。一方、Jini では開発者が自由に識別子を設定することができ、あらかじめアプリケーションが利用したいサービスを既知でなければならない。このように各ネットワークミドルウェアでサービスの識別子が違うため、新たにアドレス空間を用意し、各サービスの識別子とマッピングを行う必要がある。

3.3 プロトコル変換

ネットワークミドルウェアは、機器連携において複数のプロトコルを同時に用いることが多い。機器連携の代表的なミドルウェアである UPnP と Jini のプロトコルスタックを図 1 に示す。

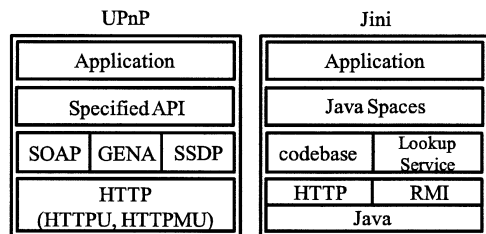


図 1 プロトコルスタック

UPnP ではイベント通知に General Event Notification Architecture (GENA) と Simple Service Discovery Protocol (SSDP) を同時に用いている。一方、Jini ではサービスの登録にサーバを、サービス呼び出しの際に Java RMI を用いている。このように単純なプロトコルの 1 対 1 変換でネットワークミドルウェアの統合を実現するのは非常に難しい。それぞれのプロトコルを扱える仕組みが必要となる。

4. 提案システムの構成と動作

4.1 機器連携ステップ

既存のネットワークミドルウェアの多くが、Plug & Play 方式を採用している。Plug & Play 方式とは、ネットワークに接続されている機器が、ユーザによる

手動設定なしで他の機器のサービスを利用できる方式である。本研究では、一般の Plug & Play 方式を採用する。Plug & Play は以下のステップを経て実現することと定義する。

(1) ドメインへの参加

各ネットワークミドルウェアは独自のドメインを用いており、機器はまずドメインに参加する。トランスポート層のアドレスやアプリケーション層のアドレスを用いて他の機器と通信を行う。

(2) 他のリソース（機器やサービス）の検索・発見

ドメインに参加した後は、ネットワークのどこにリソースがあるのかを検索・発見しなければならない。既存手法としては、サービス呼び出すための Java オブジェクトをサーバに登録したり、サービスの詳細を示した XML をマルチキャストすることで実現している。

(3) サービスの実行

リソースの発見後は、ネットワークを介してサービス呼び出す。ネットワークを介したサービスの呼び出しにも様々な方法がある。トランスポート層を隠ぺいする透過性を実現した Java RMI や、HTTP などの上で XML に基づいたメッセージを交換することでサービス呼び出す SOAP などである。

4.2 抽象化の統一

3.1 節で各ネットワークミドルウェアで抽象化の方法が異なり、機器とサービスを区別する方法と区別しない方法の2通りあることを述べた。本研究では、機器を Device、Service、Action の集合として扱うこととする。Device とは、トランスポート層における機器の抽象化である。Service とは、アプリケーション層のサービスの抽象化である。Action とは、サービスのプログラムレベルにおける関数名や引数の抽象化である。2通りの抽象化方法の統一ルールを図2に示す。

機器とサービスを区別する抽象化	機器とサービスを区別しない抽象化
Device → Device	(仮想的に生成) Device
Service → Service	Service → Service
Action → Action	Action → Action

図2 抽象化の統一

機器とサービスを区別しない抽象化では、本研究で扱う方法と1対1とのマッピングを行うことが可能である。一方、機器とサービスを区別しない方法では、Device の抽象化を必要とする。したがって、提案システムが仮想的に Device を生成することでマッピングを可能とする。

4.3 アドレス空間の設計

ネットワークミドルウェアを用いてサービスを実現した場合、サービスを識別するためのアドレスが必要となる。このサービスを示すアドレス体系は各ネットワークミドルウェアで異なる。本研究では独自のアドレス空間を用意し、各サービスのアドレスとの変換を行う。以降、独自のアドレス空間を Global ID(GID)と表現する。GID と各サービス名との変換表を図3に示す。

Middleware UPnP	GID - Middleware	Middleware Jini
サービス名 GID	GID Middleware	GID サービス名
Printer 1	1 Jini	1 print out
Audio 2	2 UPnP	2 Audio Player
Content Directry 3	3 UPnP	3 Contents

図3 アドレス変換

UPnP のクライアント機器が Printer というサービスを呼び出したとする。次に、Printer を GID に変換し、GID を元にどのドメインにサービスがあるかを認識する。

4.4 モジュール化による多様なプロトコルへの対応

ネットワークミドルウェアでは、複数のプロトコルを同時に用いている。したがって、単純なプロトコルの1対1変換とはいかず、別の仕組みが必要となる。そこで、各ネットワークミドルウェアごとにモジュールを用意し、多様なプロトコルを同時に扱える仕組みを採用する。

4.5 提案システム概要と構成

提案システムは、Inter-Platform と Adaptive module で構成され、仮想機器の生成とメッセージ変換を行うことを目的としている。仮想機器とは、ソフトウェアで機器を仮想的に実現し物理機器として見せかける仕組みである。メッセージの変換とは、機器から送られてくる発見要求メッセージや、サービス呼び出し要求・応答メッセージなどを異なるミドルウェア用に変換する機能である。図4にシステムの構成を示す。

Adaptive module は、ネットワークミドルウェア毎に用意し、提案システムとドメインに参加している機器と通信することを可能にする。Adaptive module 間

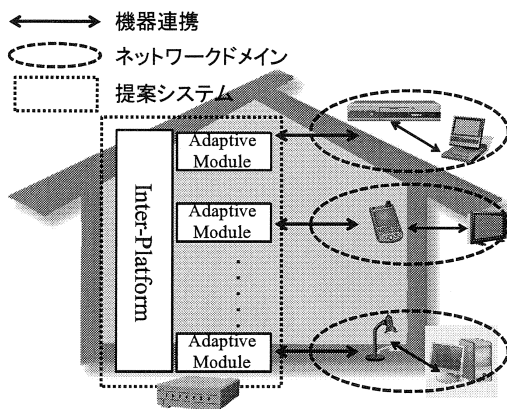


図 4 システム構成

でメッセージのやり取りを行う際は、Inter-Platform を介して行う。提案システムのモジュール構成を図 5 に示す。

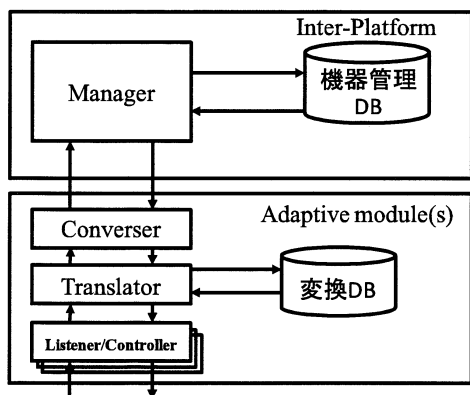


図 5 モジュール構成

Inter-Platform は、全ての機器を管理する Manager と機器管理 DB を持つ。Adaptive module は、Listener/Controller, Translator, Converter, 変換 DB で構成されている。

4.5.1 Inter-Platform

Inter-Platform は、Manager と機器管理 DB で構成される。Manager とは機器管理 DB と Adaptive module とのインタフェースとなる。Inter-Platform の役割は、全ての機器の情報の管理、Adaptive Module から送られてきたメッセージの転送、Adaptive Module からの問い合わせに対する返答である。機器の情報の管理には、GID をキーとし、Resource Description を参照できる機器管理 DB を用いる。機器管

理 DB は、メッセージの転送や問い合わせに対する返答の際のルーティング表の役割も併せ持つ。Resource Description の詳細については 4.6.1 節で述べる。

4.5.2 Adaptive module

Adaptive module は、Converter, Translator, Listener/Controller, 変換 DB で構成されている。Converter は、Inter-Platform とメッセージの送受信を行う。ここでのメッセージとは、サービスの検索・発見、サービス呼び出しのためのメッセージのことである。Translator は、各ネットワークミドルウェアのメッセージと提案システム用のメッセージとの変換を行う。Listener/Controller は、仮想機器となり、他のドメイン上にある機器の数だけ存在する。Listener/Controller は、同じドメイン上にある他の機器からメッセージの送受信を行う。

4.6 機器連携ステップにおける提案システムの動作

4.6.1 機器のドメイン参加

機器連携の第 1 ステップとしてドメイン参加がある。ドメイン参加時の流れを図 6 に示す。

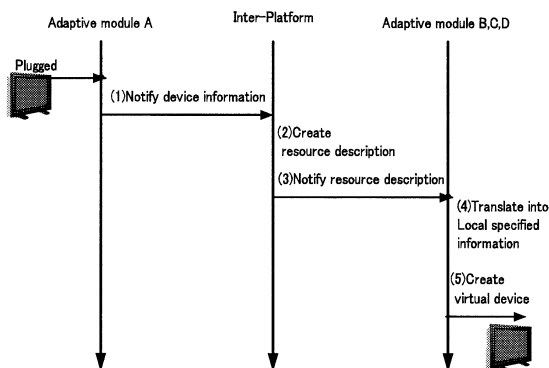


図 6 ドメイン参加の流れ

機器がネットワークに接続され自身のドメインに参加したときに、Adaptive module は機器の参加を検知する。検知後は新たに接続された機器の情報を Inter-Platform に通知する。Inter-Platform は、通知された情報から Resource Description を生成する。Resource Description は、Device, Service, Action の 3 つの情報を階層構造として持っている。書式は、UPnP Device Architecture⁹⁾ で規定された論理プロパティとサービスアクションを用いる。Resource Description を図 7 に示す。

Resource Description を生成した後、他の全ての Adaptive module に渡す。他の Adaptive module は、Resource Description を元に仮想機器を生成し、自身

```

<?xml version="1.0"?>
<device>
  <deviceType>Device Type</deviceType>
  <serviceList>
    <service>
      <serviceID>Service ID</serviceID>
      <serviceType>Service Type</serviceType>
      <actionList>
        <action>
          <actionID>Action Name</actionID>
          <argumentList>
            <argument>
              <name>formalParameterName</name>
              <dataType>Data Type</dataType>
              <direction>in xor out</direction>
            </argument>
          </argumentList>
        </action>
        <!--more actions-->
      </actionList>
    </service>
    <!--more services-->
  </serviceList>
</device>

```

図 7 Resource Description

のドメインに参加させる。この仮想機器がプロキシの役割となり、他のドメインにある機器を自身のドメインにあるかのように見せることが可能である。

4.6.2 リソース（機器、サービス）の検索・発見

提案システムは、リソース発見のために Advertise-ment 方式と Search 方式をサポートする。Advertisement 方式とは、サービスを提供する機器が利用する側の機器に、機器やサービスの存在を通知する方法である。Advertisement 方式が実行されるタイミングは、ドメイン参加時、オンライン状態であることを通知するとき、機器やサービスの状態が変わったとき、機器がドメインから離脱するときに行われる。例えば、ドメイン参加時の場合、Resource Description に加えて alive メッセージや ByeBye メッセージなどを送ることで Advertisement 方式を実現する。一方、Search 方式とは、サービスを利用する側の機器が、機器連携実現のために必要とするサービスを持つ機器を検索する方式である。Search 方式の流れを図 8 に示す。

Adaptive module A と同じドメイン内にある機器をクライアント、Adaptive module B と同じドメインにある機器をサーバとする。そして、サーバは機器とサービスの情報をシステムに登録しているものとする。クライアントがミドルウェアの仕様で決められた方式で Discovery Message を送信する。Discovery Message を受け取った Adaptive module A は、検索されたサービスがあるか Inter-Platform に問い合わせ管理している Resource description とマッピングを行う。

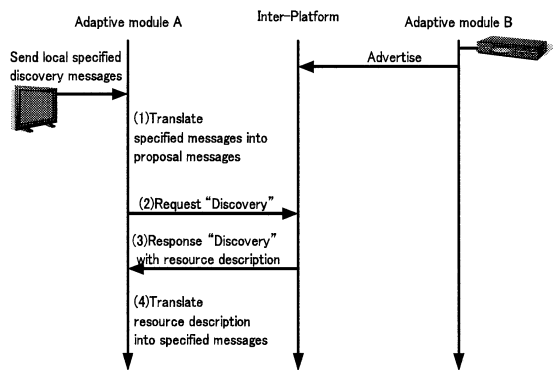


図 8 Search 方式

4.6.3 サービスの実行

リソースの発見後、サービスの実行が必要となる。サービス実行までの流れを図 9 に示す。

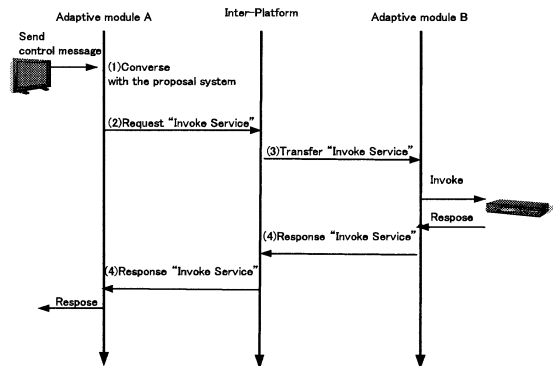


図 9 サービス実行方式

リソース検索時と同様に、Adaptive module A と同じドメイン内にある機器をクライアント、Adaptive module B と同じドメインにある機器をサーバとする。サーバの発見後、クライアントはミドルウェアの仕様で決められたインタフェースを介してサービスにアクセスする。Adaptive module A は、サービス呼び出し要求を受け取るとメッセージを Inter-Platform に転送する。Inter-Platform はサーバが接続されている Adaptive module B にメッセージを渡し、サービスを呼び出す。呼び出し後は、サービスの呼び出しに対するレスポンスを逆の順序で転送する。

4.7 個別対応機能の検討

4.7.1 ストリームデータの変換

AV 系機器では、機器の操作に加えて、機器が持つコンテンツを視聴するアプリケーションが重要となる。例えば、音楽や動画などのコンテンツをネットワークを介して視聴可能とするストリームデータを用いたアプリケーションなどである。UPnP をベースとした機器連携の標準である DLNA¹⁰⁾ では、ストリームデータ配信の際、標準で HTTP を、オプションで RTP を用いる取り決めをしている。このように TCP/IP 上でストリームデータを配信するためには、HTTP、RTP を利用するのが一般的である。したがって、HTTP、RTP の相互変換ゲートウェイを実現することが有効である。

4.7.2 応答時間による処理

ネットワークミドルウェアで用いられるプロトコルにはタイムアウトなどの時間が存在する。例えば、Java RMI を用いる Jini ではサービスの呼び出しに対する応答が 15 秒以上なければタイムアウトとみなされる。一方、SOAP を用いる UPnP では自由にタイムアウトを設定することが可能である。提案システムを介してサービスを呼び出すと 2 つ以上の基準となる時間が存在してしまう。中間 ACK などを返すことでこの問題に対応することが可能である。

4.7.3 イベント処理

機器連携の重要な機構にイベント通知がある。UPnP では、eventing プロセスとして、機器が稼動状態の変化などのイベントを、クライアント側の機器に通知する仕組みを実現している。イベント通知には、Subscription と Event Message が用いられる。Subscription とは、クライアント機器が通知して欲しいイベントをあらかじめサービス提供機器に登録することである。Event Message とは、登録のあったイベントの発生をクライアント機器に通知することである。提案システムでイベント通知を実現するためには新たな仕組みが必要となる。各機器のイベントに対応するようなイベントハンドラを提案システムに組み込むことでこの仕組みを実現する。

5. 考 察

本章では関連研究との比較を行い、提案システムのメリットについて述べる。Web プロトコルを用いたホームコンピューティングミドルウェアの統合は、インターネットを介しての機器連携を目的としており、本研究とは、本質的に目的が異なる。An Architecture for Jini/UPnP Interoperability では、1 対 1 変換の

ゲートウェイを実現し、ネットワークミドルウェア間の機器連携を可能にしている。1 対 1 変換を用いる場合、ミドルウェアの数が N 個あると変換機は $N * (N-1) / 2$ 個必要となる。一方、本研究では、 N 個の Adaptive module しか必要としておらず、1 対 1 変換よりも開発の際のコストはかからない。また、Universal Middleware Bridge は本研究と同じ相互連携を実現している。この研究では、各ネットワークミドルウェアに対応したモジュール同士が直接通信しており、他のモジュールの状態を常に把握しておかなければならずオーバーヘッドとなっている。各モジュール部分がメッセージのやり取りの際に必要なチャンネルは N 個である。一方、本研究で提案したシステムでは、Adaptive module と Inter-Platform 間の一チャンネルのみ考慮すればよい。したがって、モジュール部分にかかる負荷が、関連研究よりも小さい。

6. まとめと今後の課題

ユビキタスネットワークにおける重要な課題の一つに機器連携がある。機器連携を行うためにはネットワークミドルウェアが重要であり、現在、様々なネットワークミドルウェアが実現されている。しかし、ネットワークミドルウェア間の互換性は確保されておらず、物理的に機器同士が接続されている状況でも機器連携は実現できない。本研究では、ミドルウェアの相互接続性を確保し、全ての機器で連携が可能になるシステムの提案を行った。提案システムは、Inter-Platform と Adaptive module で構成され、仮想機器の生成とメッセージ変換を可能にする。また、機器連携をドメイン参加、リソースの検索・発見、サービスの実行と定義し、提案システムの動作を示した。その結果、 N 個の Adaptive module の実現でネットワークミドルウェア間の相互接続性を確保することができた。今後は、メッセージの不一致に関する問題に取り組む。サービスを実行する際、サービスを呼び出すためのインタフェース (実行するメソッド名やパラメータ、戻り値、メソッドの属するクラス名など) について既知である必要がある。クライアントの機器とサービス提供機器のインタフェースの引数の数や型などが一致していなければ、機器連携は不可能である。セマンティック技術を用いる方法とインタフェースの変換表を作成する 2 つの方法を検討している。

参 考 文 献

- 1) 次世代ネットワーク実証実験. http://www.soumu.go.jp/snews/2008/pdf/080212_2.pdf.

- 2) 石原 智宏, 助川 聖, 島田 裕一, ネットワークサービスの発展を支える ホームゲートウェイ, 雑誌 FUJITSU 特集:「次世代ネットワーク」, Vol. 57, No. 4, FEBRUARY 2005, pp.366-372, 2006.
- 3) Jini Network Technology, Sun Microsystems.
<http://www.sun.com/software/jini/>.
- 4) UPnP Forum, <http://www.upnp.org/>.
- 5) HAVI Home Pages, <http://www.havi.org>.
- 6) 上野乃毅, 中島達夫, 佐藤一郎, 副島康太, Webプロトコルを用いたホームコンピューティングミドルウェアの統合, 情報処理学会論文誌, Vol.44 No.SIG10, pp.177-186, 2003.
- 7) J. Allard, V. Chinta, S. Gundala, G. G. Richard III:Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability, Applications and the Internet 2003 Symposium, pp.268- 275, 2003.
- 8) Design of a Universal Middleware Bridge for Device Interoperability in Heterogeneous Home Network Middleware, IEEE Transactions on Consumer Electronics, Vol.51, No.1, 2005, pp.314-318, 2005
- 9) Universal Plug and Play device Architecture Version 1.0, 2000.
<http://www.upnp.org/resources/documents/CleanUPnPDA101.20031202s.pdf>.
- 10) Degital Living Network Alliance,
<http://www.dlna.org/home>.