組込み Linux[*1]を利用した ホームコンピューティングのためのミドルウェア

Middleware for Next-Generation Home Computing

徳永　英治 [1)]　　石川　広男 [2)]
倉橋　誠 [3)]　　森元　靖庸 [4)]
Eiji TOKUNAGA　　Hiroo ISHIKAWA
Makoto KURAHASHI　　Yasunobu MORIMOTO

1) 早稲田大学理工学部情報学科中島研究室（〒169-8555 東京都新宿区大久保三丁目
４番１号　早稲田大学大久保キャンパス６１号館５０５室　E-mail: eitoku @ dcl.info.waseda.ac.jp ）

2) 早稲田大学理工学部情報学科中島研究室（〒169-8555 東京都新宿区大久保三丁目
４番１号　早稲田大学大久保キャンパス６１号館５０５室　E-mail: ishikawa @ dcl.info.waseda.ac.jp ）

3) 早稲田大学理工学部情報学科中島研究室（〒169-8555 東京都新宿区大久保三丁目
４番１号　早稲田大学大久保キャンパス６１号館５０５室　E-mail: mik @ dcl.info.waseda.ac.jp ）

4) 早稲田大学理工学部情報学科中島研究室（〒169-8555 東京都新宿区大久保三丁目
４番１号　早稲田大学大久保キャンパス６１号館５０５室　E-mail: morimoto @ dcl.info.waseda.ac.jp
）

ABSTRACT. In the future, micro processors will be embedded in various appliances such as home appliances and personal appliances. These appliances will be connected to networks such as Internet and communicate with each other. By the communication, services which are provided in these appliances should be integrated. The computing environments must have the home computing middleware which controls these appliances. Most of recent middleware such as Jini and X10 has no compatibility and adaptability with each other, so that it is difficult to communicate with other appliances which are under any other middleware's control. Although there are some protocol converters which can only convert one to one, we need the framework in which we can integrate any middleware more simply and straightforwardly. In this paper, we propose a framework for connecting home computing middleware framework, which make possible that any appliance under any middleware's control communicates any other appliances. We show also service integration with Internet service and service middleware.

1　背景

近い将来、我々の身の回りのあらゆるものにコンピュータが組み込まれるようになるだろう。家庭内では、AV 機器、冷蔵庫、掃除機はもちろん、衣服や食器などにも組み込まれ、意識しないでコンピュータを使うようになる。

コンピュータが組み込まれた機器、いわゆる組込み機器は単体で機能しても有用性は低い。組込み機器はネットワークに接続されている必要がある。ネットワークに接続された組込み機器が提供する機能をサービスと呼ぶ。また、ここでは組込み機器に限らずネットワーク経由で利用できる機能のことを一般にサービスと呼ぶ。単体のサービスは価値が限られる。一つの機器が提供できるサービスには限界があるからである [7]。サービスは、他のサービスと相互に連携することで価値を増す。複数のサービスを相互に連携し、新たなサービスを構築することをサービス統合という。

例えば、VCR にコンピュータが組み込まれ、ネットワークに接続できるようになったとする。VCR が PC から専用のアプリケーションで遠隔操作できたとしても、有用性はあまりない。しかし、インターネット上にある TV 番組表サービスと連携することによって、容易に録画予約ができるようになる。さらに、PC が提供するユーザプロファイル情報と連携することによって、自動的に録画スケジュールを立てることができる。また、インターネット上の映像、画像サービスと連携することによって、インターネット上の映像を録画することができるようになる。同様に、携帯電話のネットワークと連携することによって、遠隔地から録画予約を変更することができる。

しかし、このような複数のサービスを統合したアプリケーションを開発することは難しい。また、組込みデバイスは、場面に応じた特殊な機能を提供するため、それぞれが固有なプラットフォームで構築され、固有のネットワークで通信していると予想される。異なるプラットフォーム間では、ソフトウェアの移植性が低く、異なるネットワークを通じて協調させることは難しい。

このようなサービス統合の問題を解決するにはサービス同士の協調を助け、ソフトウェア開発を容易にするミドルウェアが必要になる。現状のミドルウェアの一つに Sun Microsystems の Jini[8] がある。Jini は、情報家電や PC 等のあらゆるコンピュータ機器をネットワーク上で連携させるための技術である。Jini ではこの連携のことを Federation と呼ぶ。コンピュータ機器の Federation に

[*1] プロジェクトの縮小で組込み Linux を利用する部分が削られてしまったため、実際には利用していません。

より、各機器が提供するサービスを統合する事ができる。Jini では連携するコンピュータ機器及び、そのコンピュータ機器の機能を総称して、サービス（正確には、Jini サービス）と呼ぶ。Jini により、ネットワーク上の複数のサービスが互いの持つ機能や情報を交換し合うことができるようになる。これが Jini の Federation の考え方である。Jini の Federation の仕組みは、同じく Sun Microsystems のプログラミング言語である Java をベースとし API により規定される。

また、日本とヨーロッパの家電機器メーカー 8 社が策定した情報家電ミドルウェアに HAVi[3] がある。HAVi は情報家電に特化したミドルウェアで、HAVi ネットワークに接続された機器は相互に資源や機能の共有を行うことが可能になる。HAVi は簡単なプラグアンドプレイ機能を持ち、下層では IEEE1394 で通信している。

このようにサービスを統合するミドルウェアは出揃いつつあるが、それぞれに問題点がある。例えば Jini は、Java をベースにしたテクノロジであるため、機器が Java をサポートしていなければならない。また、Jini は汎用的なサービス統合を目指しているので、各機器のプラットフォームに特化したパフォーマンスを引き出しにくい。さらに HAVi は、情報家電に特化したミドルウェアなので、情報家電を制御する API は豊富にそろっているが、PDA や携帯電話などの情報機器の制御には向いていない。

これらの問題点から、現存のいずれかのミドルウェアが全世界すべての組込み機器をサポートすることは考えにくい。また、新たなサービス統合ミドルウェアが開発されたとしても、全世界を統一して恒久的にサポートするのは不可能である。何故なら組込み機器は PC とは異なり、主に単一（もしくは限られた複数）の機能を実現することを最優先に開発されるからである。機能を重視した組込み機器は独自のプラットフォームを採用する。個々のプラットフォームを最大限に生かせるミドルウェアの存在は考えにくい。また、汎用的に考えられた Jini などのミドルウェアでは、情報家電のような特殊なインタフェイスを持った機器を最大限に生かすことができない。

2 目的

この事業の目的は、多様化したミドルウェアを統合するフレームワークを開発することである。具体的には、Jini などの分散ミドルウェア間を、透過的に通信させるためのシステムと API を開発する。また、既存のインターネットサービスとも連携可能にし、実用的なアプリケーションの開発を目指す。

既存のインターネットサービスを含め、ミドルウェアを統合することにより、ホームコンピューティングにおける複雑化したサービスを容易に接続することができる。また、ミドルウェア統合のフレームワークを規定することで、新しいミドルウェアへの対応を容易とする

3 設計

本フレームワーク設計の目標は、以下の通りである。

- レガシミドルウェアを利用したレガシサービスを、特に変更することなしに本フレームワーク上で簡単に利用できる。
- 新しいミドルウェアをへの対応を容易に追加することができる。

図 1 では、本フレームワークの基本的なコンセプトを示している。この図でわかるように、各ミドルウェアネットワークに対して 仮想サービスゲートウェイ (Virtual Service Gateway) とプロトコル変換マネージャ (Protocol Conversion Manager) と、仮想サービスレポジトリ (Virtual Service Repository) が存在する。VSG は、特定のプロトコルを使用して各ミドルウェアと別のミドルウェアを接続する。PCM は、各ミドルウェアのプロトコルを VSG のプロトコルに変換する。VSR は、サービスの機能と位置を記録する。これらについては、後のセクションで述べる。

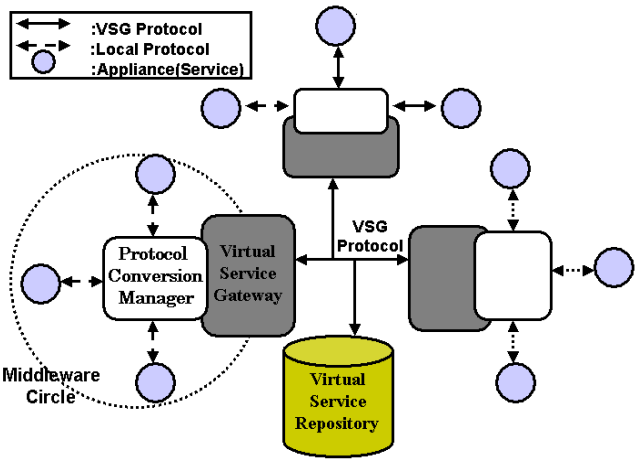

図 1: フレームワークの概略図

3.1 仮想サービスゲートウェイ

仮想サービスゲートウェイ (VSG) は、中間プロトコルを使用して、あるミドルウェアと別のミドルウェアを接続するためのゲートウェイである。中間プロトコルではインタフェースや位置などのサービスに関する情報を扱う。

ここで使用するプロトコルは、サービス統合の用途によって選択する。マルチメディアストリームや音声によるコミュニケーションを扱うサービスを統合するには、ストリームを接続でき、マルチメディアデータを転送できるようなプロトコルでなくてはならない。一方、単純なサービスを統合するには、単純なプロトコルで十分である。

本フレームワークのプロトタイプ実装には、シンプルなプロトコルである SOAP を使用した。

3.2 プロトコル変換マネージャ

プロトコル変換マネージャ (PCM) は、ローカルなミドルウェアのプロトコルを、VSG のプロトコルに変換したり、その逆の操作を行う。VSG には図 2 で示される 2 つのプロキシモジュールがある。それはサーバプロキシ (SP) とクライアントプロキシ (CP) である。

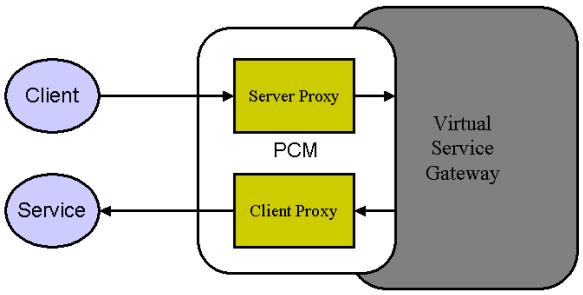

図 2: プロキシモジュール

サーバプロキシ (SP) モジュールは、リモートサービス

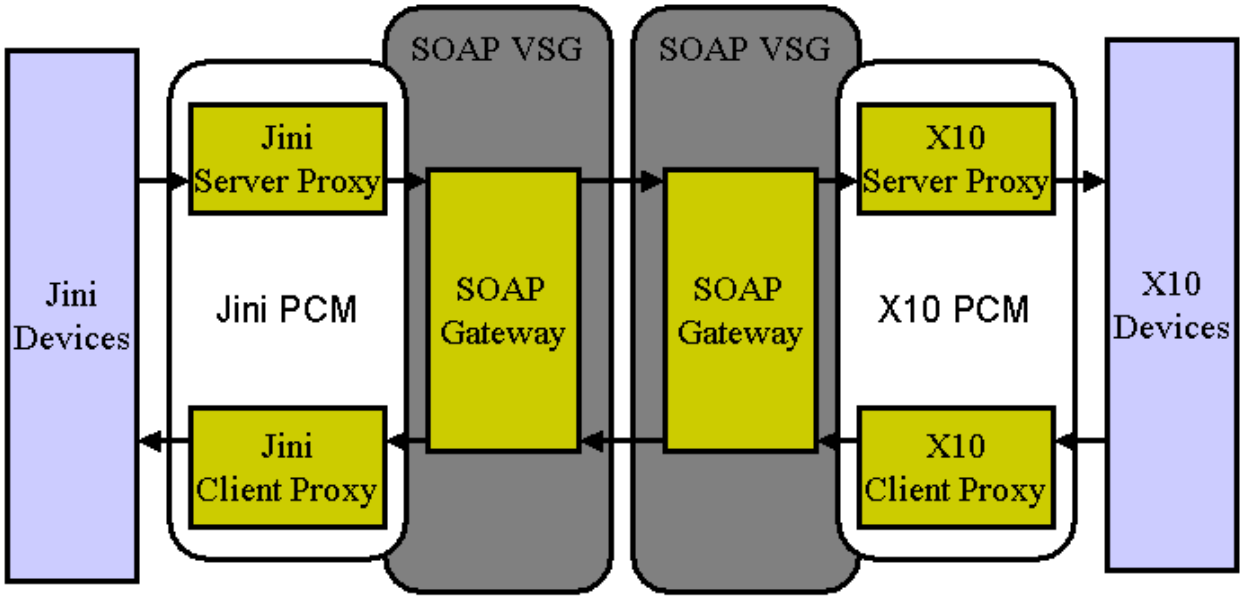

図 3: Jini と X10 のプロトコル変換

のインタフェースを、ローカルサービスに対して提供する。クライアントプロキシ (CP) は、ローカルサービスのインタフェースを、VSG サービスに向けて変換する。リモートクライアントが VSG サービスを要求すると、その要求に応えるローカルサービスをクライアントプロキシが起動する。たとえば、VSG プロトコルが SOAP で、ローカルミドルウェアが Jini の場合、サーバプロキシは SOAP サービスを Jini サービスに変換し、クライアントプロキシは Jini サービスを SOAP サービスに変換する。そして、Jini クライアントがサーバプロキシに要求を送ると、サーバプロキシがリモート SOAP サービスを VSG を経由して呼び出す。また、リモート SOAP クライアントがクライアントプロキシに対して VSG を経由して要求を送ってきた場合、クライアントプロキシが Jini サービスを呼ぶ。

3.3 仮想サービスレポジトリ

仮想サービスレポジトリ (VSR) は、さまざまなサービスの場所や状況などについての情報を格納する仮想的なデータベースである。VSG と PCM は、サービスを探したり状況を把握したりするために、これを利用する。VSR の実装は、VSG プロトコルに依る。たとえば、VSG プロトコルが SOAP の場合、VSG は WSDL と UDDL で実装する。

このフレームワークを使用すると、レガシサービスを統合して使用することが可能となる。レガシクライアントやサービスは、特別な操作を行うことなく、他の種類のミドルウェア上にあるサービスと通信することができる。

4 実装

ここでは、上記のフレームワークに従って、我々が実装したプロトタイプとユーティリティ API について述べる。

VSG プロトコルには XML/HTTP ベースのプロトコルである SOAP を用い、VSG には IBM developer works によって開発された Apache SOAP を利用した。SOAP はシンプルで扱いやすく、HTTP ベースなのでスケーラビリティも高い。また、特定企業に依存しない技術なので、本プロトタイプの実装に採用した。

また、本プロトタイプでは、Jini と X10[10]、及び、SMTP によるメールサービスの PCM を実装した。各 PCM はクライアントプロキシとサーバプロキシを動的に生成する API を提供する。プロキシの動的生成には、Java のバイトコードを動的に生成、変更するための API である Javassist[2] を利用した。

生成されたプロキシを通して、クライアントは他ミドルウェアのサービスへ透過的にアクセスすることができる。例として、図 3 に Jini と X10 のプロトコル変換の図をあげる。また、各ミドルウェアのクライアントは SOAP サービスにもアクセスすることができる。

5 評価

本ミドルウェアを利用して、いくつかのアプリケーションを開発した。アプリケーションの一つは、ユニバーサルリモコンである。

それは、本ミドルウェアからの操作で Jini や Havi の機

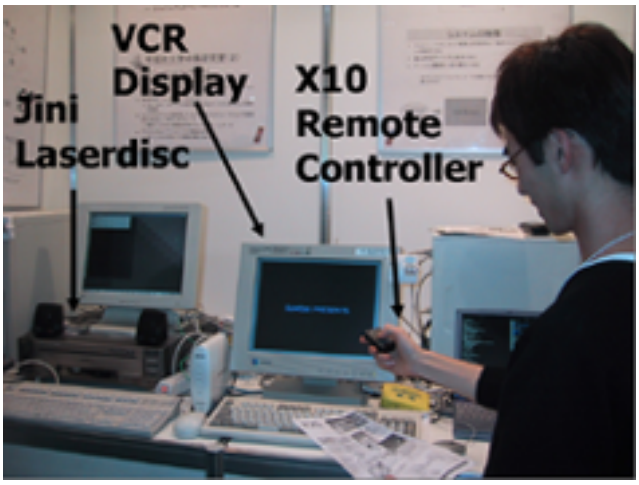

図 4: ユニバーサルリモコン

器を操作できる、X10 のリモコンである。使用時の様子を図５に示す。図では、X10 リモコンを使用して Jini で制御されるレーザーディスクを操作している。また、HAVi の DV カメラも操作することができる。VSG と PCM がこれらのミドルウェア間の差異を隠すため、アプリケーション開発は容易である。

しかし、マルチメディアデータを扱ったり、コンテキスト情報に従った動的なサービス起動を要求するアプリケーションを開発することは難しい。SOAP 通信に利用される HTTP はクライアント/サーバ型のプロトコルであり、非同期通知を使うケー スにはうまく対応できない。また、ストリームデータを扱うことも不得手である。

加えて、現在の HTTP は TCP 上のプロトコルである。TCP スタックは大きくて複雑なものであるから、メモリやプロセッサが貧弱なスモールデバイスでこれを扱うには問題がある。

6 今後の展望

プロトタイプで未実装のフレームワークを実装する。また、他のミドルウェアに対応する PSM を実装し、評価する。

6.1 仮想サービスレポジトリの実装

開発期間の問題で、仮想サービスレポジトリの実装は行わなかった。今後、UDDI と WSDL を利用した SOAP-VSG に対する仮想サービスレポジトリの実装を考えている。

6.2 HAVi-PCM の実装

今後、HAVi に対する PCM の実装を考えている。

7 まとめ

本プロジェクトでは仮想サービスゲートウェイを使った分散ミドルウェア統合フレームワークを提案し、実装として SOAP を中間プロトコルとした統合ソフトウェアを開発した。本ソフトウェアを利用すると、Jini と X10 の二つのミドルウェアが SOAP を介して透過的に通信できる。また、直接 SOAP のサービスを Jini, X10 それぞれのクライアントから呼び出すことが可能であり、SOAP のクライアントから Jini, X10 それぞれのサービスを呼び出すことも可能である。Jini と X10 及び SOAP のサービス統合はほぼ成されたと言える。

Jini、UPnP、HAVi など様々な分散ミドルウェアが提案されながらもそれらを実装したデバイスが産業に出てこないのは、個々の分散ミドルウェアが他のミドルウェアとの互換性に欠けるからであり、また既存のシステムとの協調性に不安が残るからである。

そこで、本プロジェクトの実装により、アプリケーションレイヤで分散ミドルウェア統合を行うことの有用さを示した。アプリケーションレイヤでの分散ミドルウェア統合フレームワークは、実装が容易であること、保守が容易であること、ポータビリティに優れること、等の利点がある。

本プロジェクトのような分散ミドルウェアを統合するシステムは、重要性が指摘されながらも有効な実装及びフレームワークが存在しない。本プロジェクトではアプリケーションレイヤでの統合フレームワークを提案し、実装の一例を示した。

また、SOAP を利用した Web サービス事業は世界規模で急速な普及が予想されている。我々のソフトウェアを利用して Jini 等のミドルウェアを SOAP にマッピングすることで、Web サービスの可能性は飛躍的に広がる。例えば、我々の開発した API を利用すれば、ローカルにある Jini サービスの一部もしくは全てを SOAP にダイナミックにマッピングすることが可能である。さらに UDDI、WSDL を利用することで、Jini サービスを Web サービスであるかのように見せることができる。このような既存のサービスを低コストで Web サービスへ変換することができる機能は、重要な事業性があると考える。

8 参加企業及び機関

なし。

参考文献

[1] H.Aizu, I.Sato, D.Ueno, T.Nakajima, ”A Virtual Overlay Network for Integrating Home Appliances,” Technical Report, Waseda University, 2001.

[2] Michiaki Tatsubori, Toshiyuki Sasaki, Shigeru Chiba, and Kozo Itano. ”A Bytecode Translator for Distributed Execution of Legacy Java Software” ECOOP 2001 Object Object-Oriented Programming,LNCS 2072, Springer VerLag, 2001.

[3] The HAVi organization, ”HAVi Version1.1 Specification,” http://www.havi.org.

[4] T.Nakajima, ”System Software for Audio and Visual Networked Home Appliances using Stateless Thin-Client Architecture,” In Proceedings of the 2nd International Workshop on Ubiquitous Computing and Communication, 2001.

[5] T.Nakajima, H.Ishikawa, E.Tokunaga, Frank Stajano, ”Technology Challenge for Building Internet-Scale Ubiquitous Computing,” 2001.

[6] T.Nakajima, K.Soejima, M.Matsuda, T.Iino, T.Hayashi, ”A Framework for Building Audio and Visual Home Appliances on Commodity Software,” In Proceedings of the IASTED International Conference on Internet, Multimedia Systems, and Applications, 2001.

[7] D.A.Norman, ”The Invisible Computer,” The MIT Press, 1998.

[8] Sun Microsystems, ”JINI Architecture Specification,” http://sun.com/jini.

[9] W3C, ”Simple Object Access Protocol (SOAP) 1.1,” http://www.w3.org/TR/SOAP.

[10] X10 Wireless Technology, Inc., ”CM11A programming protocol,” ftp://ftp.x10.com/pub/manuals/cm11a_protocol.txt.

[11] 磯村 学, 吉原 貴仁, 堀内 浩規, ”情報家電のためのサービス発見プロトコル統合方式の検討,” 情報処理学会情報家電コンピューティング第 1 回研究会, 2001.