モーバイルコンピューティング 12−10
（2000．2．4）

ユビキタス環境において様々な機器を柔軟にコントロールするためのフレームワーク

会津宏幸† 中島達夫††

本論文では，ユビキタス環境において様々な機器を柔軟にコントロールするためのフレームワークを提案する．本フレームワークの目標は，ユーザーの身近に存在するあらゆるデバイスを用いて様々なデバイスをコントロールすることにある．従来のシステムと比べて次の３つの特徴を持つ．１つ目の特徴は，各ユーザの身近に存在するPDAなどを情報家電をはじめとしたデバイスのコントローラとして用いることが可能となることである．２つ目の特徴は，複数のデバイスを１つのデバイスであるかのように扱うことが可能となることである．３つ目の特徴は，コントロールの方式をユーザの好みや状況に応じてパーソナライゼーションすることが可能となることである．本アーキテクチャは汎用的なコントロールアーキテクチャを提供するので，情報家電だけではなくコンピュータ上で実行されるアプリケーションの制御にも用いることが可能である．

A flexible framework for controlling devices flexibly in ubiquitous computing environments

HIROYUKI AIZU † and TATSUO NAKAJIMA††

This paper proposes a new software architecture for controlling devices in ubiquitous computing. Our architecture has the following three characteristics that are not provided by existing ubiquitous computing architecture. The first charactersitics is that a user's PDA can be used as a remote controller for controlling respective A/V devices. The second charactersitics is that several devices can be composed, and it is controlled as a single device. The third charactersitics is that a user can customize a way to control devices according to the user' location and situation. Since our architecture is general purpose, it also can be used to control applications executed on computers.

1. はじめに

コンピュータの小型化およびネットワークの発達により，家電をはじめとしたあらゆる機器に計算機が組み込まれ，ネットワークに接続されようとしている．また，計算機の小型化，省電力化，携帯電話などの無線通信デバイスの発達により，ユーザーが服を着たりアクセサリを身につける感覚で常に計算機を身につけ利用できるウェアラブルコンピュータも実用化しつつある．よってユーザにとって，いつでもどこでも計算機にアクセスできるユビキタスコンピューティング環境が実現されつつある．

その一方で従来の計算機のユーザインターフェースは，ケーブルで接続され機器に組み込まれていたり，や専用の無線デバイス等で接続され，接続されている計算機個別の専用デバイスであった．つまりユーザーインターフェースを流用して，別のデバイスを操作することができない．例えば，コンピュータが２台存在したとする．片方のコンピュータに接続されたキーボードやマウスを操作すれば，接続されている方の計算機は操作できるが，もう１台は接続を切替えないと操作できない．

これに対して，Bluetooth 等の近距離無線デバイスの登場により有線という制約がなくなり，通信プロトコル等も規格化されていることから入出力デバイスの共用化が可能となる．ウェアラブルコンピュータや組み込み用計算機の入出力デバイスは，そのデザインや大きさなどの制約から最低限の入出力インターフェースしか装備していないものがある．そこで，身近に存在する，より使いやすいデバイスを流用して操作することで利便性が向上すると考えられる．

† 北陸先端科学技術大学院大学 情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology.
†† 早稲田大学 理工学部 情報学科
Department of Information and Computer Science, School of Science and Engineering, Waseda University.

また近年, 音声認識や画像処理技術が発達し, 音センサーや画像センサーを使って直接デバイスに触れることなく, 音声や身振りで機器に命令を下すことが可能となりつつある. これによりキーボードをはじめとした, ハードウェアスイッチにこだわる必要がなくなる. また, 他の様々なセンサーを使い, 組み合わせ使用することで新しいユーザーインターフェースとなる可能性も存在する.

本研究では, ユビキタスコンピューティング環境において, 身近に存在するあらゆるデバイスを利用し, 様々なデバイスを柔軟にコントロールできるフレームワークを目指す. ユーザーが複雑な設定をしなくても使えることが望ましい.

本論文では特に A/V デバイスを対象に提案するシステムの特徴を説明するが, 5 節に示すように様々なアプリケーションの制御に利用することが可能である.

2. 汎用デバイスコントロールフレームワーク

本節では, 我々が提案する柔軟にコントロールするためのフレームワークについて説明する.

2.1 本システムの目標

本フレームワークの目標は以下にあげる 3 つである.

- あらゆるデバイスを利用して, あらゆるデバイスの制御を可能とする.
- 複数デバイスのコンポジションを可能にする.
- 制御の方式をカスタマイズ可能にする.

第 1 の目標は A/V デバイスのみではなく様々なデバイスの制御を 1 つのリモコンから可能とすることである. つまり, 制御対象として, コンピュータ上のアプリケーションや A/V デバイス以外の家電も制御可能にすることである.

第 2 の目標は, 複数のデバイスを 1 つのデバイスであるかのように扱うことを可能とすることである. この機能により, 複数のカメラとディスプレイを接続することにより作られたビデオコンファレンスアプリケーションを全体を 1 つのビデオコンファレンスデバイスとして用いることを可能とする.

第 3 の目標は, デバイスの制御を柔軟にカスタマイズすることを可能とすることである. つまり, GUI の表示を各ユーザの好みを反映させるとか, デバイスのコンポジションをユーザの好みや近くにあるデバイスに応じてアドホックに接続することなどを可能とする.

2.2 基本アーキテクチャ

本システムは図 1 に表されている, 以下の 7 つのコンポーネントから構成される.

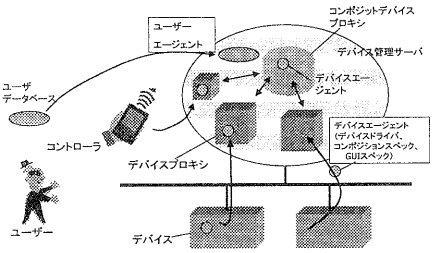

図 1 基本アーキテクチャ

- デバイス (Device)
  家電などの実際に動作して機能を果たす. 本システムのデバイスは HAVi のの FCM(Functional Component Module) のように, 各機器を単機能のコンポーネント単位まで分解して取り扱う. 例えば, TV は, チューナー, アンプ, ディスプレイ, コンバーターといった単機能のデバイスがコンポジットしたものとして構築される.
- デバイスプロキシ (Device Proxy)
  デバイスをコントロールためのインターフェースとなる. リモートに存在するデバイスのコントロールは全てデバイスプロキシを通して行われる. 実際にデバイスをコントロールするための通信手段, 手順はデバイスプロキシが把握しており, さまざまな通信メディアを混在させることができる. デフォルトでは汎用デバイスプロキシが用いられるが, デバイスエージェントの情報によりインターフェースが追加され各デバイス固有のプロキシとして対応可能.
- デバイス管理サーバ (Server)
  デバイスプロキシを生成, エージェントが転送され実行される. Lookup サービスも行う.
- デバイスエージェント (Device Agent)
  各デバイスの持っている機能や性能情報を保有している. 複数のデバイスを接続しコンポジションする際の設定を持つことも可能.
- パーソナルエージェント (Personal Agent)
  ユーザーの位置や使用するデバイスのコンフィグレーションをカスタマイズに必要な情報を持つ.
- 環境エージェント (Environment Agent)
  デバイスがネットワークへ接続されたり, ネットワークから切り離されたという, 環境の変化を監視, 状態を把握するエージェント.
- コントローラ (Controller)

ユーザーインターフェースを司り，ユーザーに対してボタンなどを表示したり，ユーザーからの入力を受けつけることができる．

2.3 システムの基本動作

本節では，実際にデバイスをネットワークに接続したときどのように制御されるかについて述べる．

まずはじめに，デバイスはプラグアンドプレイによりネットワークに接続される．ネットワークにデバイスが接続されるとデバイスはデバイス管理サーバーを探しサーバーにプラグイン要求を出す．プラグインされると，デバイス管理サーバー内に接続されたデバイスに対応するデバイスプロキシが生成される．この時，プロキシデバイスはデバイスのネットワーク上のアドレスを保持し，今後のサーバとデバイス間の通信にはプロキシデバイスに登録されたアドレスが利用される現在の我々の実装でプラグアンドプレイをサポートするためのディレクトリサービスとして Jini を利用する．また，マルチメディアデバイスとしてカリフォルニア大学バークレイ校の Mash ツールキット [1] を用いる．

この後，デバイスはデバイスプロキシにデバイスエージェントを転送する．デバイスエージェントはデバイスプロキシがデバイスと通信するために必要なコードと GUI を生成するための GUI スペック，デバイスのコンポジションを決定するコンポジションスペックを含んでいる．コンポジションスペックはデバイス間の接続を制御することにより，複数のデバイスを 1 つのデバイスであるかのように見せることを可能とする．デバイス管理サーバがコンポジションスペックを受け取るとコンポジットデバイスプロキシを生成する．コンポジットデバイスプロキシはコンポジションスペックに従いデバイスプロキシの接続を行う．ここで言う接続とは，コントローラにより生成されたイベントをどのプロキシデバイスのイベントに配送するかを決定することである．現在の我々の実装では移動エージェントしてお茶の水大で開発された MobileSpace[2] を用いている．また，イベント配送のメカニズムとしては JavaBeans を用いている．

リモートコントローラもデバイスとして同様の手順でプラグインされる．はじめに，リモートコントローラ上にはコントロール可能なデバイスのリストが生成される．このリストはユーザエージェントから送られてきたプリファレンスや現在のユーザの位置などの環境情報を元にカスタマイズすることが可能である．例えば，ユーザの好みに応じて GUI を配置したり，ユーザが不必要な GUI 部品を表示しないことでユーザの好みに応じた制御が可能となる．また，ユーザの現在位置を用いて，ユーザが現在いる部屋のデバイスのリストのみを生成して表示したり，ユーザの向いている方向に存在するデバイスのリストのみを表示することが可能となる．GUI スペックをカスタマイズするための情報はコントローラを利用するユーザの情報を管理するデータベースからパーソナルエージェントを転送することにより可能となる．パーソナルエージェントはユーザの過去の履歴や現在位置などのユーザの現在の状況をモニタリングする手段を提供するのみではなく，ユーザのプリファレンスなども含んでいる．

また，デバイスを制御するための GUI はデバイスエージェントにより GUI スペックとしてデバイス管理サーバに送られている．GUI スペックはパーソナルエージェントが持つ情報を用いてカスタマイズされコントローラに転送される．ユーザが GUI を操作することにより生成されたイベントはプロキシデバイスに転送される．このイベントは，デバイス管理サーバが持つコントローラプロキシにより受け取られ JavaBeans のイベントに変換される．このイベントは JavaBeans のイベント配送機能を用いてデバイスプロキシに配送される．最終的に，デバイスプロキシはデバイスエージェント内のコードを用いてデバイスにコマンドを転送する．我々のフレームワークではデバイス管理サーバは 1 つのアプリケーションとして実現される．そのため，イベント配送のメカニズムとして JavaBeans を用いることが可能であり，複雑な分散イベント管理が不要なため実装が大変容易となる．

3. リモートコントローラ

本節では，我々のフレームワークがどのようにしてリモートコントローラを扱うかに関して述べる．我々のフレームワークでは PDA などのタッチパネルを持ったデバイスを用いることを前提としているが，携帯電話などのそれ以外のデバイスを用いることも可能である．また，表示デバイスを持たないものをコントローラとして用いることも可能である．

3.1 GUI スペック

デバイスエージェントは各デバイスの制御方法を記述したドキュメントである GUI スペックをデバイス管理サーバに転送する．GUI スペックは，ディスプレイに表示すべき GUI パーツと，パーツを操作した場合に発生するイベントに関して記述されている．PDA などのデバイスは GUI スペックを受けることでコントローラとして用いることが可能となる．GUI パーツの操作により発生したイベントはパケットにパックさ

れデバイス管理サーバが持つコントローラプロキシに転送される．コントローラプロキシは受け取ったイベントを JavaBeans のイベントに変換してデバイスプロキシに配送する．また，デバイスがコンポジションされる場合は，複数のデバイスの GUI スペックが合成される．この場合は，これらのデバイスを統合するデバイスが持つ GUI スペックに従い合成するか，すべての GUI スペックをマージしたものを新しい GUI スペックとするかの戦略により新しい GUI スペックを生成する．現在の実装では GUI スペックを記述するための言語として XML の拡張言語である BML(Beans Markup Language)[3] を用いている．

3.2 GUI のパーソナライゼーション

はじめにコントローラが接続されたとき，現在コントロール可能なデバイスのリストが表示される．ユーザはこれらのリストから利用したいデバイスを選択すると，ターゲットデバイスを制御するための GUI が表示される．制御可能なデバイスのリストはデバイス管理サーバが自動的に GUI スペックを生成し，コントローラに転送する．ここで生成されるリストは，パーソナルエージェントから得られたユーザの位置や状況によりフィルタリングされ，行動履歴を用いてリストの順番が並べ替えられる．これを実現するためには，ユーザの位置情報だけではなく各デバイスが置かれている位置に関する情報も必要である．

また，各デバイスを制御するための GUI スペックを各ユーザの好みに応じて変更することも可能である．このため，パーソナルエージェントが管理するユーザの履歴や好みを用いる．GUI スペックの各パーツは属性を持ち，その属性とユーザの好みから表示する GUI スペックを決定する．例えば，GUI パーツの属性がオプショナルである場合は通常は表示しない．また，ユーザの履歴を用いて全く利用されていない GUI パーツを表示しないようにすることができる．

3.3 明示的コントロール

我々のフレームワークでは，様々なデバイスをコントローラとして用いることが可能である．例えば，カメラに接続されたジェスチャ認識ソフトウエアをコントローラとして用いることも可能である．この場合は，ユーザが明示的にコントローラプロキシを生成する必要がある．この場合，コントローラプロキシは，ジェスチャ認識アプリケーションの結果をイベントに変換し，デバイスプロキシに配送する．現状では，コントローラプロキシの生成はユーザの責任であるが，将来的には GUI スペックとして記述可能とする予定である．これにより，ユーザはプログラムを記述しなくても様々なデバイスを用いたコントロールを容易に実現することが可能である．

3.4 デバイスの選択

4.2 節で述べたように，コントローラがデバイス管理サーバに認識されたとき，デバイス管理サーバは利用可能なデバイスのリストを GUI スペックとして生成し，コントローラに転送する．この場合は，コンポジットされたデバイスのリストのみが表示される．これらのリストの表示法はデバイス管理サーバのマネジメントとして大変重要なものである．例えば，はじめに，デバイスのカテゴリや部屋のリストを表示することによりユーザが操作したいデバイスを容易に見つけることが可能となる．

現状では，ユーザがデバイスをコンポジションする場合はコンポジションスペックをユーザが記述する必要がある．将来的には，コントローラ上に利用可能なデバイスをビジュアルに表示し，それらの接続を指定するだけでコンポジションスペックが生成されるようにする予定である．

4. デバイスのコンポジション

本節ではデバイスのコンポジションについて説明する．我々のフレームワークでは，各デバイスは単機能をを提供し，テレビやビデオなどのデバイスはコンポジットデバイスとして定義される．つまり，テレビはチューナ，アンプ，スピーカ，ディスプレイデバイスを接続したコンポジットデバイスを 1 つのデバイスとして扱うことを可能としたものと考えることができる．我々のフレームワークではこれだけでなく，様々なデバイスを接続することが可能である．ネットワークにつながっている複数の情報家電のデバイスを組み合わせることにより機能拡張を行ったり，まったく新しい機能を持った家電として使用できる可能性がある．

4.1 コンポジションスペック

コンポジションスペックはコントローラプロキシが生成したイベントを複数のデバイスプロキシがエキスポートするイベントにマッピングするための仕様を記述したドキュメントである．我々のフレームワークでは JavaBeans を用いてイベントの配送をおこなうことができるので，コンポジションスペックとして BML を用いる．コンポジションスペック内には，Beans の接続方法が BML 言語を用いて記述されている．

デバイス管理サーバはコンポジションスペックを受け取るとコンポジットデバイスプロキシを生成する．コンポジットデバイスプロキシは結合されている各デバイスの GUI スペックを集めてマージすることによりコンポジットデバイスの GUI スペックを生成し，コ

ントローラに転送する．

4.2 イベント

イベントの配送はデバイスプロキシを実現するBeansとコントローラプロキシを接続することにより実現される．実際にBeansを接続する際問題となるのは，Beansの名前付けをどのようにおこなうかである．名前の抽象度を高くすることにより，利用可能なデバイスを用いたデバイスを生成することが可能となる．例えば，PCに接続されたチューナボードとネットワーク上に接続されたディスプレイとスピーカを接続することによりアドホックなテレビデバイスを生成することが可能である．この場合は，各デバイスの名前を抽象的に記述し，それらの名前を用いてコンポポジションスペックを記述することにより実現される．また，特定のデバイスを指定する抽象度の低い名前を用いることにより通常のテレビデバイスを実現することができる．このようなメカニズムを実現するためには，現在の実装で用いているJava Beansの明示的な結合法をJiniのLookupサービスのようなサービスの内容で結合できるものに変更する必要がある．

1つのイベントが複数のデバイスを同時に制御する場合も存在する．例えば，複数のアンプのボリュームを一括操作する場合や装置のON/OFF，メディアの再生のスタート・ストップなどを行う場合，GUIにより生成されたイベントを同時に複数のデバイスに配送する必要がある．この場合，イベントの配送順序が重要となる場合もある．現在の実装では，JavaBeansを用いているためイベントの配送がどのような順番で行われるかに関して議論することはできない．イベントをどのような順序でデバイスプロキシに配送するかなどのイベント配送のポリシを今後検討する必要がある．

4.3 コンポジションの方式

我々のフレームワークでは，現在，3つのコンポジションの方式を提供している．1番目の方式は，コンポジションスペックを現在利用な可能なデバイスのリストから自動生成する方法である．2番目の方法は，デバイスがコンポジションスペックをデバイス管理サーバに提供する方法である．3番目の方法は，ユーザがデバイス管理サーバにコンポジションスペックを提供する方法である．

4.3.1 自動コンポジション

あらかじめユーザーやデバイスがコンポジションについてプリファレンスを持っていなくとも，各デバイスの持つ機能やデフォルトのポリシに従って自動的にコンポジションを行う．例えば，CDデバイスのようなオーディオ出力を持つデバイスとアンプデバイスのようなオーディオ入力を持つデバイスをプラグインすることで自動的に音楽を流す準備が整う．当然，ユーザーやデバイスが最初から持つコンフィグレーションにより抑制することも可能である．

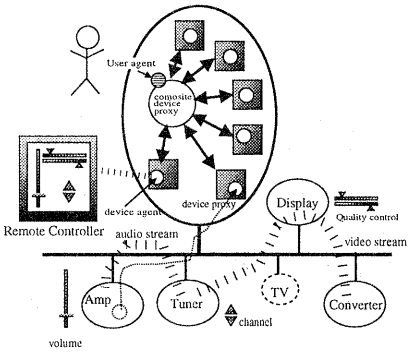

図2 テレビの構成

4.3.2 デバイスによるコンポジション

デバイスがプラグインの時点からデフォルトのコンポジションの設定を保持していて，その設定に従い自動的にコンポジションして一つの機器として振舞うことを可能とする．例えば，TVのようにアンプ，チューナー，ディスプレイといった複数のデバイスが1つのケースに入っているとする．この時，TVとしてコンポジションをおこなうためのコンフィグレーションがデバイスエージェントに含まれていればプラグイン時に自動的にTVとしてコンポジションが行われる．

4.3.3 明示的コンポジション

ユーザが既存のデバイスから新しい家電機器を生成したり，新しい機能を持ったデバイスを既存のデバイスに追加することにより拡張したり，ネットワークに接続されたデバイスのアドホックなコンポジションを行う場合は，ユーザが明示的にデバイスの結合法を指定する必要がある．その場合，ユーザが明示的にコンポジションスペックを記述して，デバイス管理サーバに渡す．

5. システムの動作例

5.1 テレビの例

ここでは，図2で示すように複数デバイスが暗黙的コンポジションの手続きにより各デバイスがコンポジットされテレビが構築される例を説明する．最初に各デバイスはネットワークに接続されるとデバイス管理サーバーを探しサーバーにプラグイン要求を出す．プラグインされると，デバイス管理サーバー内では各デバイスのデバイスプロキシが設置される．デバイスエー

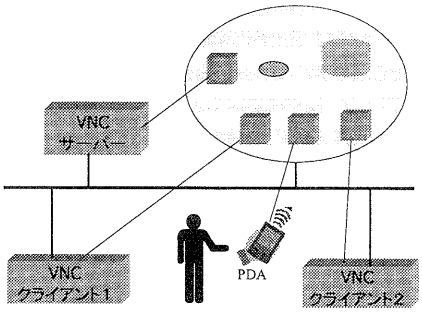

図 3　VNC への適用例

ジェントを持っているデバイスはデバイスプロキシにエージェントを送り出す. エージェントの持つ情報に従いデバイスプロキシが機能拡張されインタフェースが追加される. この例では, 仮想的なテレビデバイスが存在し, テレビデバイスエージェントに含まれるテレビとしてのコンポジションスペックをもとにコンポジションが行われ, その結果, テレビコンポジットデバイスプロキシが作成される.

リモートコントローラもデバイスとしてプラグインされる. リモートコントローラ上にはコントロール可能なデバイスのリストが表示される. ユーザーがコンポジット済のテレビデバイスを利用することを選択することで, テレビ仮想デバイスのデバイスエージェントが持つ GUI スペックをコントローラに送ることにより, テレビコントロール用の GUI がリモートコントローラーの画面に表示される.

ユーザーがリモートコントローラに表示されている GUI を操作するとリモートコントローラのデバイスプロキシに伝達され, イベントとしてテレビコンポジットデバイスプロキシに伝えられる. イベントを受け取ったコンポジットデバイスプロキシはコンポジション情報に従い, イベントを各デバイスプロキシにイベントを配送する.

コンポジットデバイスプロキシ内でイベント配送を設定, 更に各デバイス間の実データ配送経路の設定が必要であれば各デバイスプロキシを通して設定を行う.

5.2　アプリケーションへの適応例

ここでは AT&T ケンブリッジ研究所が開発した VNC(Virtual Network Computing)[4] に本フレームワークを応用する例をあげる. VNC はサーバ上で生成したデスクトップの GUI を, 軽量なクライアント上に表示するシステムである. 資源の乏しい計算機上でも表示可能であることから, ネットワークに接続されたあらゆる端末から, デスクトップを操作可能とし, ユビキタス環境を実現するシステムである. 例えば最初にユーザが VNC サーバとなる計算機 1 上でアプリケーションを走らせ GUI を表示しているとする. ユーザが計算機 B へ移動しても, アプリケーションを再起動したりデータを計算機 1 から取り寄せたりしなくとも, 計算機 2 で VNC クライアントを起動して計算機 1 の VNC サーバにアクセスすると, 計算機 B に計算機 1 を利用していた際の GUI 環境がそのまま表示され, 操作可能となる.

AT&T ケンブリッジ研究所では VNC と Active Badge[5] を組み合わせ, Active Badge によりユーザーの移動を検出し, ユーザに一番近いディスプレイにデスクトップ環境を表示できるようなシステムを構築している.

図 3 に示すように VNC サーバと複数の VNC クライアントが動作しているマシンが存在する. ユーザーの位置はユーザーの持つ PDA が VNC クライアントの動くマシンに近づいたことで検出する.

この場合, コンポジションスペックにコンポジションするデバイスとして “VNC サーバー”,“PDA に距離が一番近い VNC クライアント” というように記述しておくことで, ユーザー (PDA) が移動しても近くの VNC クライアントにデスクトップが表示することができる. コンポジションスペックに記述するデバイスの指定方法として “デバイス ID XX 番” というように固定 ID を指定するのではなく “[デバイス A]-[一番近い]-[デバイス B]” といった, 相対的な関係を実現することが重要であると考える.

各デバイスの位置情報は, 環境サーバ[6] で管理し, 各オブジェクトの相対的関係が変化した場合は環境サーバの “Notify” 機能によって通知され, デバイスの再コンポジションが行われる.

6.　議論と今後の課題

本節では, 我々のアーキテクチャと HAVi との比較をおこなったあと, 我々のフレームワークの今後の課題に関して述べる.

我々のフレームワークは, 各デバイスのコンポジションを柔軟におこなうことが可能である. つまり, HAVi では, デバイスがもつ機能間の接続を変更したり, 異なるデバイスが持つ機能間を接続することはできない. そのため, 既存のデバイスを新たなサードパティのデバイスの機能を用いて拡張したり, 既存のデバイスをアドホックに接続することにより新しいデバイスを生成したりすることは困難である. しかし, 我々のフレー

ムワークでは，ユーザがコンポジションスペックを記述することによりテレビデバイスのチューナとアンプの間に別なデバイスが持つ音声処理モジュールを挿入することが可能である．また，ユーザの近くに存在するカメラやディスプレイを用いることでアドホックなテレビ会議システムを構築することが可能である．

また，我々が提案するフレームワークでは，コントロールの方法をカスタマイズしたり，ユーザが持っている特殊なコントローラを用いてデバイスを制御することも可能である．カスタマイズを実現するため，ユーザの好みや行動の履歴を用いたり，位置情報やユーザの状況を用いることでコントロール可能なデバイスをフィルタリングすることも可能である．現在の HAVi では，このような柔軟なコントロールの方法を提供することは困難である．

また，HAVi では，コントローラが IEEE1394 に接続されたデバイスやそのデバイスを制御するためのリモコンを用いてしか制御することができない．そのため，リモコンの制御に関しては個々のデバイス依存であるため我々のフレームワークが提供するような統一的な管理を提供することは困難である．

次に，現状の我々のフレームワークの問題点に関して述べる．はじめの問題点は，どのようにして離れた位置にあるデバイスを制御するかである．つまり，外出先から家庭内のテレビを制御したりすることをいかにに可能にするかである．また，複数ユーザが異なるリモコンを用いて同一のデバイスを制御するときどのようにして排他制御をおこなうかに関しても検討する必要がある．

次の問題点は GUI スペックとコンポジションスペックの記述の問題である．コンポジションスペックの問題は，デバイスをどのような抽象度をもった名前を用いて記述するかがシステムの拡張性を大きく決定することである．つまり，抽象度が高い名前を用いた場合は，アドホックなデバイスの接続を容易に実現するが，特定のデバイス間の接続を明示的に指定することは難しい．また，GUI スペックに関しては，GUI パーツの表示の仕方やどの GUI パーツがユーザに興味があるかをどのように指定するかなどに関して検討する必要がある．

最後の問題点は，デバイス間の接続の指定をどのようにおこなうかに関してである．つまり，各デバイスはが持つ A/V データ用のポートをどのように接続し，それらのデバイスをどのように同時に制御するかに関して検討する必要がある．将来的には，MIT の VuSystem[7] や JAIST のマルチメディアツールキット [8] で実現されているように，各機能間の接続を明示的に接続しストリームを定義することを可能とする必要がある．ストリームを定義可能にすることにより QOS 制御やメディア間同期などの高度なマルチメディアデータの処理をおこなうことが可能となる．

7. まとめ

本論文で提案したフレームワークでは，1 つのリモコンを用いて様々なデバイスが制御可能なこと，複数のデバイスを 1 つのデバイスとして扱うことが可能なこと，デバイスのコントロールの方法をユーザの好みや状況に応じてパーソナライズ可能なことを示した．また，2 つの例を用いて我々のフレームワークの柔軟性を示した．現在，我々が本論文で提案したフレームワークを Java 言語を用いて実装中である．また，我々のフレームワークを HAVi 上のアプリケーションとして構築することを検討中である．

参考文献

1) McCanne, S. and et al: Toward a Comon Infrastructure for Multimedia-Networking Middleware., *7th Intl. Workshop on Network and Oerating Systems Support for Digital Audio and Video (NOSSDAV'97)* (1997).
2) 佐藤一朗, 高橋美奈子, 棚橋杏子, 吉野裕子: モバイルエージェントの階層的な構成と移動, 日本ソフトウェア科学会全国大会 (1998).
3) IBM alphaWorks: *Beans Markup Language(BML)*. http://www.alphaWorks.ibm.com/formula/BML.
4) Richardson, T., andKenneth R. Wood, Q. S.-F. and Hopper, A.: Virtual Network Computing, *IEEE Internet Computing*, Vol. 2, No. 1, pp. 33–38 (1998).
5) Harter, A. and Hopper, A.: A Distributed Location System for the Active Office., *IEEE Network*, Vol. 8, No. 1 (1994).
6) Nakajima, T., Aizu, H., Kobayashi, M. and Shimamoto, K.: Environment Server: A System Support for Large Scaled Distributed Computing, *The 2nd International Conference on WorldWide Computing and Its Applications'98* (1998).
7) Lindblad, C. and Tennenhouse, D.: The VuSystem: A Programming System for Compute-Intensive Multimedia, *IEEE Journal of Selected Areas in Communications* (1996).
8) Nakajima, T.: A Toolkit for building Continuous Media Applications, *IEEE International Workshop on Real-Time Computing, Systems and Applications(RTCSA)* (1997).