rdev: 仮想化環境を用いた遠隔デバイス連携

管 文鋭* 河口 信夫†

Abstract

近年、情報家電の普及、計算機の小型化、無線 LAN などネットワーク技術の発展などにより、ユビキタス環境の実現として日常生活中の様々な機器間連携を実現するスマートスペースに関する研究が盛んである。多くの既存研究では、主にそれぞれの環境内における機器連携やサービスに着目し、遠隔利用やスマートスペース間の連携まで論じられていない。一方、我々は、スマートスペース間をユーザが自由に移動し、移動先においてその場のデバイスを使用したり、遠隔地である自宅内のデバイスにアクセスするというようなスマートスペース間連携に着目する。そのような連携を実現するために、本稿では、個々のスマートスペースを構成するプライベートネットワークに接続されたデバイス群を、任意の場所からアクセスするための手法、複数スマートスペース連携環境を複数のユーザによる共有手法、ユーザ認証手法を提案する。また、提案手法ではスマートスペース間におけるデバイス制御の違いを吸収する仕組みも導入する。これらを実現するために、本システムは仮想化環境と仮想マシンを利用し、プロトタイプ実装では、PlanetLab と User Mode Linux を用いる。

1 はじめに

近年、ネットワーク通信機能を持つ情報家電が普及し始め、それらを相互接続したり、PC・携帯電話などの情報端末と連携することにより、新しいサービスの提供が期待されている。また、ユビキタス環境の実現として日常生活中の様々な機器間連携を実現するスマートスペースに関する研究が盛んである。

一方、ノート PC の小型化・軽量化、スマート・フォンの登場により、ユーザが情報端末を持ち歩く機会が多くなりつつある。さらに、無線 LAN などの無線通信技術の普及により、外出先のホテルや駅、飲食店などにおいて、手軽に携帯型情報端末を介したインターネットへのアクセス環境が整いつつある。

このような傾向の中、ユーザが移動先にあるデバイスを使用したり、移動先から自宅にあるデバイスにリモートアクセスすることができるならば、より多様かつ高度な機器間連携サービスの実現を期待できると考えられる。

代表的な情報家電や情報機器の連携プロトコルとして UPnP[1] や Bonjour[2]、DLNA[3]、ECHO-NET[4] などがある。それらの多くは、デバイスを接続しただけで連携できるようになるゼロ・コンフィギュレーションを特徴として持つ。しかし、それらのプロトコルはローカルネットワークで構成される単体のスマートスペースにおける利用を想定しているため、単体では、遠隔地のデバイスを連携させることは困難である。

遠隔地のデバイスとの連携を考慮し、グローバルネットワークに対応したスマートスペース関連携もいくつか提案されている [9, 10]。これらは、VPN やゲートウェイを利用して異なるスマートスペースを接続することで、デバイスの遠隔操作や連携を可能にする。これらは、スタティックに指定された異なる拠点が相互接続され、その間でデバイス連携が可能となるが、それ以外の拠点に移動する場合は、新たに拠点を接続する必要がある。

我々は、スマートスペース間をユーザが自由に移動し、任意の移動先においてその場のデバイスを使用したり、遠隔地である自宅内のデバイスにアクセスするというようなスマートスペース間連携に着目する。そのような連携を実現するために、以下のことを提案する。

1. 個々のスマートスペースを構成するプライベートネットワークに接続されたデバイス群を任意の場所からアクセスするための手法
2. スマートスペースを相互接続するために整備された環境を複数のユーザにより共有する手法
3. ユーザ認証手法や異なるユーザが個々のデバイスに対するアクセスを制御する手法
4. デバイスの違いを吸収する手法

個々のスマートスペースを任意の場所からアクセス可能にし、それらを相互接続するために整備された環境を複数ユーザが共有可能にするために、仮想化環境をそれぞれのスマートスペース中のゲートウェイに導入する。また、デバイスによる違いを吸収するために、仮想マシンを用いる。

本手法を用いることにより、整備されたデバイス連携環境を複数ユーザが利用でき、かつ、ユーザ・デバイスごとのアクセス制御が可能となる。さらに、本手法はデバイスの違いを吸収するので、ユーザはデバイス環境に合わせ下位ソフトウェアを用意したり、上位ソフトウェアを変更する必要がない。これらの特徴により、本手法は、端末や複数ユーザが任意に移動することに適した遠隔デバイス連携となる。

*名古屋大学大学院 情報科学研究科
†名古屋大学大学院 工学研究科

今回のプロトタイプ実装では、仮想化環境として PlanetLab を利用し、仮想マシンとして User Mode Linux を用いる。

本稿の構成は以下の通りである。2 節では、スマートスペース間におけるデバイス連携の定義を明確にし、3 節では、本手法の特徴と、利用シナリオとしてユーザが外出先にてその場にある FAX 機を使って自宅にきた FAX を受ける例を紹介する。4 節では、動機に副うようなシステム設計を提案する。5 節では、実装と、それに用いる PlanetLab と User Mode Linux を紹介する。6 節では、システムの動作例を順に追って説明する。

2 スマートスペース間におけるデバイス連携

本稿では、ユーザが任意の移動先にあるスマートスペース中のデバイスや自宅スマートスペース中のデバイスをシームレスに使用でき、連携させることを考える。

異なるスマートスペース間におけるデバイスのシームレスな連携とは、ユーザが任意にスマートスペース間を移動し、デバイスの場所を意識せずに、それらを使用し、スマートスペース間のデバイス群を連携することをいう。

このようなシームレスなデバイス連携を考えるために、まず我々は、想定するスマートスペース環境、デバイスの種類とユーザの移動に伴うデバイスの利用形態を分類した。

- スマートスペース環境
  1. **Home**：ユーザの自宅、または、ユーザが所有するスマートスペース（プライベートネットワーク）
  2. **Away**：ユーザが移動した先のスマートスペース（プライベートネットワーク）
  3. **Home device**：Home 中のデバイス（ネットワーク接続・ノード搭載・周辺デバイス含む）
  4. **Away device**：Away 中のデバイス（ネットワーク接続・ノード搭載・周辺デバイス含む）
- デバイスの種類
  1. **Network device**：スマートスペースを構成するプライベートネットワークにネットワーク経由で接続されているデバイス
  2. **Node device**：スマートスペースを構成するプライベートネットワーク中のノードに搭載されているデバイス
  3. **Peripheral device**：スマートスペースを構成するプライベートネットワーク中のノードにシリアルなどにより接続されている周辺デバイス
- デバイスの使用形態
  1. **Local device Access**:Away において、Away device にアクセス。
  2. **Remote device Access**：Away において、Home device にアクセス。または、Home において、Away device にアクセス。
  3. **Device cooperation**：Home/Away において、両方にあるデバイスにアクセスし、連携を行う。

デバイス連携の利用例としては、ホテルのスマートスペースに接続されているプラズマテレビを自宅のテレビの番組予約に合わせること、外出先から自宅スマートスペース中の DVD レコーダを参照すること、出張先と自社両方のプリンタに同じ印刷物を出力するなどのことが考えられる。

3 仮想化環境を用いた遠隔デバイス連携

2 節にて述べたように、ユーザが任意の移動先において Away Device や Home Device にアクセスしたりそれらを連携させることを実現するために、我々は、仮想化環境を用いた遠隔デバイス連携システム、rdev: Remote Device Environment using Virtualization を提案する。

3.1 要求仕様

1. **マルチユーザに対応した スマートスペース連携環境構築**
   任意の Away において、ゲートウェイの役割を果たすノードの配置など、異なるスマートスペース間を接続するのに必要な環境をユーザ自身が自由に構築し、サービスを利用可能。
   マルチユーザに対応しているため、複数人がそのような環境を共有可能。そのためのユーザ認証機構を持つ。
   Web インタフェースにより、環境を手軽に構築可能。
2. **デバイスによる違いの吸収**
   異なる拠点間において、デバイスの違いを吸収。ユーザは普段使用するアプリケーションをそのまま使用可能。
3. **デバイスの対応付け**
   異なる拠点においても、ユーザソフトウェアがアクセスを意図するデバイスと、その場に存在する実デバイスが自動的に対応付け可能。

4. **デバイスアクセス制御**
   ユーザがデバイスに対してのアクセス権限を管理し、アクセス制御可能。
5. **セキュリティ (リソースの干渉)**
   ネットワークなどのリソースを共有するユーザ間で干渉が行われないこと。

3.2 利用シナリオ

提案システムにおいて、以下のような利用シナリオ例を挙げる。

- Home 宛ての FAX を Away の FAX 機から受け取る

Home または Away の FAX 機の接続状態として、Home/Away を構成するプライベートネットワークにネットワーク接続されているか、あるいは、Home/Away を構成するプライベートネットワーク中にあるノードにシリアルなどにより接続されている。

1. ユーザAは、Away（たとえば、ホテル）に到着し、自分の PC・小型端末もしくは Away にある端末の Web ブラウザから、後節で説明する手法により、Away を構成するプライベートネットワークに参加する。
2. ユーザAは、端末から、Away プライベートネットワーク中にあるアクセスを許可されている FAX 機を利用できる。その際に、ユーザAは自分の端末に利用する FAX 機に対応したドライバを用意する必要はない。FAX 機に対する権限は Away の管理者によって管理される。
3. ユーザAは、Web ブラウザから、Away と Home プライベートネットワークを接続する。（実際には、1. において、ユーザAが Away プライベートネットワークに参加した時点で、同じユーザ空間にある Away と Home プライベートネットワークは自動接続される）
4. ユーザAは、Web ブラウザから、Home の FAX 機の出力が Away の FAX 機の入力に繋ぐように設定する。
5. Home の FAX 機が受信をすると、受信は Away の FAX 機に転送され、ユーザAは Away の FAX 機から Home に来た FAX を受け取る。

このように、ユーザは簡単に異なる拠点間のデバイス連携を利用できる。

4 rdev の設計

3節の要求仕様を踏まえ、システムの設計を行った。この節では、主な設計法と本システムを構成するコンポーネントについて説明する。

システム構成の概要を図 1 に示す。

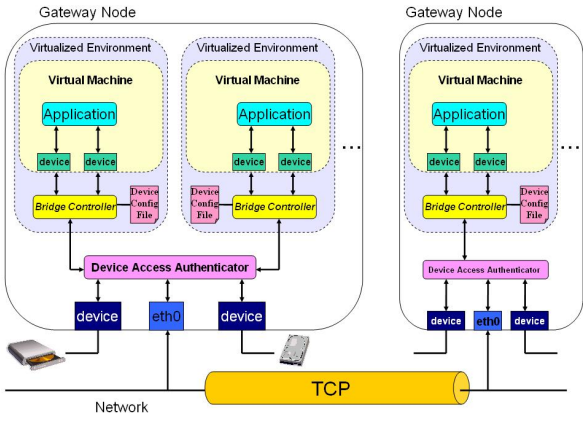

図 1: System overview of rdev

4.1 手軽な環境構築とマルチユーザ — 仮想化環境

異なるスマートスペースを相互接続するには、各スマートスペースにゲートウェイの役割を果たすノードが 1 つ存在する必要がある。以下、これをゲートウェイノードと呼ぶ。

本手法では、すべてのスマートスペースにおいて、ゲートウェイノードが 1 つずつ設置されると仮定する。ただし、この時点において、ゲートウェイノードは物理的に設置されているだけで、異なるスマートスペース間を接続する機能を持つことは仮定しない。そのような機能は本システムが提供する。

たとえば、図 2 の例では、それぞれに 1 つゲートウェイノードを持つ 6 つのスマートスペースがインターネット上に存在する。

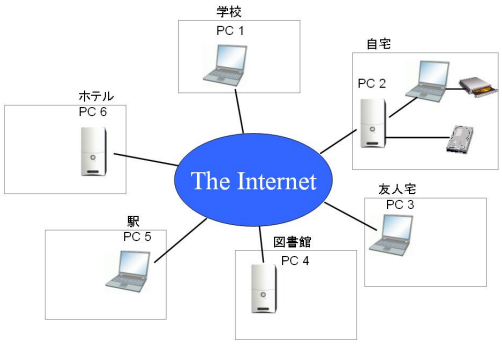

図 2: The set of private networks

このような環境が大きくなればなるほど、つまりスマー

トスペース数が多ければ多いほど、環境を複数ユーザで共有されるべきである。

また、環境が大きければ、すべてのスマートスペースを常に連携可能状態にする必要はない。ユーザがある時点において、すべてのスマートスペース間でデバイス連携をすることはあまり考えられないからである。たとえば、図 2 の例では、ユーザがホテルにおいてホテルと自宅デバイスを連携することは考えられるが、同時に図書館や友人宅にあるデバイスをも連携するシチュエーションはあまり考えられない。

我々は、接続可能なスマートスペース中から、ユーザが自分の用途に合わせ、任意なスマートスペースを選択可能にする。たとえば、図 3 の例では、ホテルと自宅のスマートスペースを選択することで、その 2 つのスマートスペース間のみにおいてデバイス連携が可能となる。

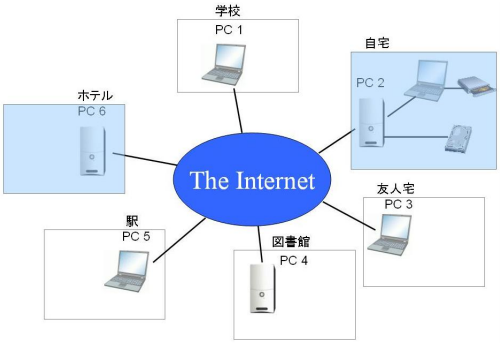

図 3: Selection of arbitrary private networks

我々は、マルチユーザとユーザによるスマートスペースの選択を実現するために、スマートスペースの仮想化に着目した。(図 1 中の Virtualized Environment)

それぞれのゲートウェイノードに複数の仮想化環境を用意し、仮想化環境同士を結ぶ。ユーザは自分のユーザ空間にある仮想化環境のみに対してアクセス権限を持つ。たとえば、図 4 では、自宅の VE1(Virtualized Environment) とホテルの VE1 は同じオーバレイ上にあり、友人宅の VE2 とホテルの VE2 は別のオーバレイ上にある。User1 は VE1 に対してアクセスでき、User2 は VE2 に対してアクセスできる。

以上の検討を踏まえ、マルチユーザへの対応を考慮し、仮想化環境を用いた環境構築するための手順を以下に示す。

1. 前提
   すべてのスマートスペース中にゲートウェイノードとして使用できるマシンが配置されている。
2. ユーザ空間の定義
   ユーザは、自分のユーザ空間を定義し、ほかのユーザのと区別する。図 4 の例では、User1 は VE1、User2 は VE2 というユーザ空間を持つ。

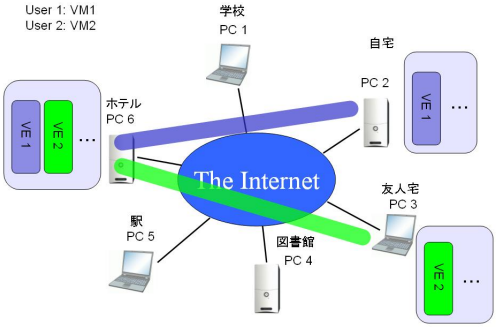

図 4: Support for multi-user by virtulization

ユーザ空間は、仮想化環境を用いて実現する。図 4 の例では、VE1 と VE2 がそれぞれ仮想化環境である。

3. スマートスペースの選択
   ユーザは、自分のユーザ空間に対して、接続したいスマートスペース中のゲートウェイノードを選択する。それぞれのユーザ空間において、選択されたゲートウェイノードが属するスマートスペースのみが相互接続される。図 4 の例では、User1 は、ホテルと自宅のゲートウェイノードを選択し、ホテルと自宅のスマートスペースが接続される。User2 は、ホテルと友人宅のゲートウェイノードを選択し、ホテルと友人宅のスマートスペースが接続される。
4. ゲートウェイノードの仮想化
   ユーザによって選択されたゲートウェイノード中には、それぞれユーザ空間に対応した仮想化環境が配置される。図 4 の例では、ホテルスマートスペースのゲートウェイノード中において、VE1 と VE2 という異なる仮想化環境、つまりユーザ空間が配置される。

本システムは、以上のことをユーザが簡単にできるようなインタフェースを提供する。

4.2 デバイスによる違いの吸収 — 仮想マシン

デバイスを利用するには、それに対応したデバイスドライバなどの下位ソフトウェアが必要であるが、ユーザが数多くの拠点を移動する場合、その都度デバイスに合わせて下位ソフトウェアをインストールしたり、上位ソフトウェアを変更することを避けたい。我々は、個々のデバイスの違いを吸収するようなシステムを考案した。それにより、デバイスに合わせて、ドライバなどのソフトウェアをインストールしたり上位ソフトウェアの設定を変更せずに、いかなるデバイス環境においても、ユーザが普段使用するアプリケーションを利用可能となる。

これを実現するために、本システムは、仮想マシンを用いる。(図 1 中の Virtual Machine) 実際にデバイスを利用するユーザのアプリケーションを仮想マシン中に置き、デバイスに依存するドライバなどの下位ソフトウェアをホスト OS に配置することにより、仮想マシン中のユーザアプリケーションを変更することなく、どのようなホスト OS でも動作させることが可能になる。

ただし、仮想マシンとホスト OS の受け渡し部分を適切に設定する必要があり、それは後述する Bridge Controller が提供する。

4.3 仮想マシンとホスト OS のデバイスのブリッジ — Bridge Controller

仮想マシンから、それが起動しているホスト OS 上のデバイスにアクセスするには、両者のデバイスをブリッジする機構が必要である。我々は、Bridge Controller と呼ばれるソフトウェアにより、これを実現する。Bridge Controller は主に次の 2 つのタスクを担う。

1. ユーザアプリケーションが、ホスト OS の Node device や Peripheral device にアクセスする場合は、仮想マシンのデバイスとホストデバイスを直接ブリッジする。
2. ユーザアプリケーションが、Network device、Home 中の他ノードの Node device、Peripheral device や Away device にアクセスする場合は、データを TCP パケットにカプセリングし、Away または Home 中の他ノードの Bridge Controller に転送する。

4.4 デバイスの対応付け — Device Config File

ユーザが異なる拠点に移動することを考慮すると、ユーザソフトウェアがアクセスしようとするデバイスが常にその場にある実デバイスと一致するとは限らない。たとえば、Home では/dev/ttyS1 をプロジェクタとしていても、Away では/dev/ttyS1 は照明器具にあたる可能性がある。それらを対応させるためには、ホストデバイスの実体を仮想化環境中のユーザに認識させることが必要である我々はそのような対応付けを行う機構として Device Config File を考案した。

ゲートウェイノードの Node device や Peripheral device の場合は、ホスト OS 上のデバイス情報を仮想化環境内にコピーされ、対応付けはユーザによって設定される。スマートスペース中の他ノードの Node device や Peripheral device の場合は、各スマートスペースの管理者が各ノードとデバイスの情報を合わせて記述し、ゲートウェイノードのホスト OS 上に保存し、それはさらに仮想化環境内にもコピーされる。

4.5 デバイスアクセス権限 — Device Access Authenticator

個々のデバイスに対するアクセス権限はユーザ毎に異なるため、アクセス権限を管理し、デバイスへのアクセスを制御する機構が必要である。

本システムでは、Device Access Authenticator によりこれを実現する。Device Access Authenticator は、ユーザ毎のアクセス権限に基づきデバイスへのアクセスを制御する。

アクセス権限は、それぞれのスマートスペースを管理する者が管理し、認証に用いる情報は個々のゲートウェイノード中に保存する。デバイスに対するアクセス権限はユーザ毎に定義されるが、マルチユーザに対応した本システムにおいて、すべてのユーザ情報をそれぞれのスマートスペースの管理者が把握するのは現実的ではない。そのため、Device Access Authenticator では、ゲストアカウントに相当するものを定義する。ユーザが Away Device にアクセスする場合は、ゲストアカウント権限の元でアクセス制御される。

ユーザが Device Access Authenticator にアクセスすることを防ぐために、Device Access Authenticator はゲートウェイノードのホスト OS の中、つまりユーザの仮想化環境の外側に置く。ユーザは仮想化環境の外側に対してはアクセス権限を持たないので、Device Access Authenticator に触れることはできない。

4.6 セキュリティ(リソースの干渉) — VLAN によるセグメント分離

マルチユーザに対応する場合、複数ユーザがネットワークなどのリソースを共有することになる。その際、それぞれがお互い干渉しないようにセキュリティを高める必要がある。つまり、ネットワークによる通信はほかのユーザから隠蔽するべきである。

これを実現するために、VLAN 技術を利用する。ユーザが小型端末を手に Away にやって来ると、小型端末はネットワークにより、Away にある仮想化環境が置かれるゲートウェイノードと繋がるが、その際に VLAN 技術を用いて、ユーザ毎に違うネットワークセグメントを割り当てる。つまり、ユーザの小型端末が、Away のプライベートネットワークに存在する VLAN 対応スイッチに接続すると同時に、制御プログラムが自動的にスイッチをコンフィグして新たな VLAN を作成し、Away から去る際にその VLAN を破棄する。

これにより、ユーザはセグメントにより分離され、それぞれを干渉することはない。

4.7 携帯性

実際にユーザが持ち運ばなければならないものを、なるべく少なく小さくすることが望ましい。これは利用形態によって２つのパターンに分かれる。

たとえば、ホテルや出張先に行く場合は、あらかじめ Away が決まっているので、現地のゲートウェイノード情報を知ることができる。その場合は、現地に向かう前に、現地のゲートウェイノードを自分のユーザ空間に取り入れると同時に自分のソフトウェア環境をそこに転送できる。実際には、Home と Away のゲートウェイノードに仮想化環境を入れ、それぞれに仮想マシンを転送する。これにより、ユーザは余分なものを携帯せずに、スマートフォンなどの小型端末から直接現地のデバイス環境を利用できる。

次に、Away が決まらないアドホックな場合は、自分の仮想マシンなどをスマートフォンなどの小型端末に保存して持ち歩けばよい。現地に到着した時点で、現地のゲートウェイノードに自分のユーザ空間を取り込むと同時に、小型端末から現地ゲートウェイノードに仮想マシンなどをローカルネットワーク経由で転送し、起動する。

5 実装

本システムのプロトタイプでは、仮想化環境として PlanetLab[8](Private PlanetLab[1])、仮想マシンとして User Mode Linux[7] を実装に用いた。実装を簡略化するため、今回のプロトタイプでは、ユーザが利用するデバイスをシリアル接続性を持つデバイスに限定する。つまり、ゲートウェイノードに接続されているシリアルデバイス、または、スマートスペース中のノードに接続されているシリアルデバイスである。

PlanetLab と UML を用いた本プロトタイプの全体構成を図 5 に示す。

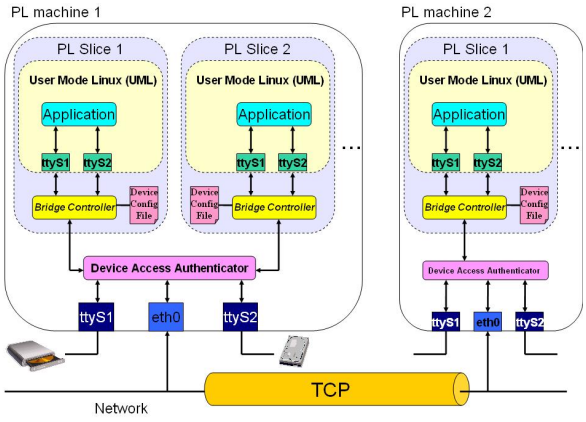

図 5: The implementation using PlanetLab and UML

5.1 仮想化環境 — PlanetLab

PlanetLab を利用することで、ユーザは PlanetLab 上のすべてのノードを共用できる。ノード共用のために、Slice という考え方が使われている。Slice とは個人の実験単位或いはユーザ空間のことで、利用者は Slice に対して PlanetLab 上の任意のノードを割り当てることができる。つまり、自分のユーザ空間に対して、自由にノードを割り当てることができる。それぞれのノード上の Slice は Linux VServer[2]によって実装され、１台のマシンで複数の Slice を持つことができる。一つのノード上の複数ユーザを Slice によりそれぞれ違うユーザ空間に閉じ込めるため、複数ユーザが一つのノードを共有できる。つまり、一つの PlanetLab ノードは複数の仮想化環境を中に持つ。さらに、PlanetLab は Slice の作成・管理やノードの管理、ユーザ認証を Web インタフェースにより一括に提供する。

このように、PlanetLab は我々の主な要求を満たすため、本手法のプロトタイプ実装では、異なるスマートスペース間を接続するゲートウェイノードを、PlanetLab ノードとして実装する。ユーザは Away のゲートウェイノードの存在を把握していれば、いつでも自由に Away のゲートウェイノードを自分の環境に取り入れることができる。つまり、自分の Slice 中に Away の PlanetLab ノードを追加することである。たとえば、ホテルや出張先に行く場合は、あらかじめ Away が決まっているので、現地のノード情報を知ることができる。その場合は、移動する前に、現地の PlanetLab ノードを自分の Slice に指定することで、そのノードを任意の場所から使用できることになる。Away が事前に決まらない場合でも、現地で PlanetLab ノードを Slice に指定するだけで、使用可能となる。

5.2 仮想マシン — User Mode Linux

本手法のプロトタイプでは、仮想マシンとして User Mode Linux[7](以下、UML) を用いる。UML は、Linux 上で動作する仮想マシンであり、Linux カーネルをユーザモードプロセスとして実行できる。UML カーネルやファイルシステムはホストマシンの環境によらず自由に変更することができるので、ユーザは好きにカスタマイズしたものを使用できる。

本来ならば、扱えるデバイスの種類を増やすため、VMWare のような、多くのホスト環境でも実行でき、か

[1]小規模の実験などに用いる、プライベートネットワークに閉じられた PlanetLab 環境。PlanetLab Consortium によって運営される PlanetLab と同等の機能を持ち、カスタマイズも可能である。

[2]Linux-VServer はホスト OS のカーネルで動作する仮想サーバで、そこから見える仮想ファイルシステムはホスト OS の実ファイルシステムの一部であり chroot の応用技術で実装されている。

つ、多くのゲスト環境を提供するような仮想マシンが望ましい。しかしながら、今回のプロトタイプでは、コンパクトさや起動の機敏さなどに重点を置き UML を用いた。

5.3 PlanetLab と UML の融合

UML を PlanetLab 上の Slice に乗せることで、マルチユーザに対応したアドホックな環境構築が可能で、かつ、デバイスの違いを吸収するようなシステムを構築できる。ユーザソフトウェアは UML 中にあり、UML は PlanetLab のユーザ Slice 中で動作する。

PlanetLab は、Slice の作成やノードの割り当てなどの基本的な機能を提供しているが、本手法では、Slice 中にユーザの UML を転送・起動したり、Bridge Controller などのコンポーネントを追加する必要がある。PlanetLab の基本機能はすべて Web インタフェースで提供されているので、我々が用意する追加機能もその中で実装することにより、ユーザビリティに優れたインタフェースを提供できる。

6 システム動作例

この節では、Away があらかじめ決まっている場合に、Away において、Local device access と Remote device access 両方のシステム動作を順を追って説明する。

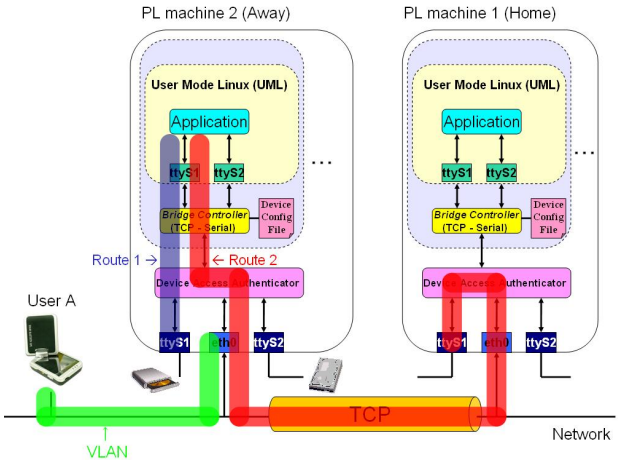

図 6: Sample scenario

1. 前提

   Home と Away のゲートウェイノードが PlanetLab に登録され、両マシンが同じ Slice に割り当てられ、ユーザがその Slice に対してアクセス権限を持つ。

   以下、Home の PlanetLab ノードを PL1, Away の PlanetLab ノードを PL2 とし、このユーザの Slice を Slice1 とする。

2. Home において

   (a) ユーザ A は PL1、PL2 両方に自分の UML を転送・起動し、Bridge Controller などのソフトウェアも起動する。

   (b) Device Access Authenticator がユーザ A の Home, Away デバイスに対する権限を調べる。

   (c) Slice1 中の Bridge Controller はユーザ A のデバイスアクセス権限に基づき、繋がれているデバイスと UML 上のデバイスアクセスチャンネルをブリッジする。

   (d) 起動された UML 中のソフトウェアは Device Config File に基づき、デバイスの種類を対応付ける。

3. Away において

   (a) ユーザ A はスマートフォン端末を手に、Away にやってくる。

   (b) スマートフォン端末は自動に Away の無線ネットワークにつながり、VLAN によって、PL2 と同じセグメントに配置される。この時点で、ユーザ A は Home、Away 両環境のデバイスを利用できるようになる。

4. Local device access

   PL2 に直接繋がれているローカルデバイスを利用する場合は、UML のデバイスアクセスチャンネルと PL2 の実デバイスが直接ブリッジされているので、ユーザ A がそのデバイスにアクセス権限を持っていれば、アクセス可能である。(図 6 中の Route1)

5. Remote device access

   Home の PL1 に繋がれているリモートデバイスを利用する場合は、PL2 の Bridge Controller はデバイスに対するアクセスデータを TCP パケットにカプセリングし、PL1 に転送する。PL1 はこのようなパケットを受け取り、デバイスに対するアクセス権限があれば、自分に繋がれている実デバイスにデータを渡す。(図 6 中の Route2)

この時、ほかのユーザが同じように、Away にきてシステムにアクセスしても、VLAN により、ユーザ A とは別のセグメントを与えられるので、お互いに干渉することはない。

7 関連研究

グローバルネットワーク環境における UPnP 機器連携の実現 [9] では、LAN 内の機器連携 UPnP を拡張したグ

ローバルにおいての機器連携を提案している。各 LAN にゲートウェイを設置し、ゲートウェイ上の Web サービスを利用し連携先の LAN 内の機器にメッセージを転送するアプローチが用いられている。

ワームホールデバイス [10] では、家庭内における利用に限定された DLNA を拡張した複数家庭間における機器連携を提案している。ワームホールデバイスと呼ばれるソフトウェアを各家庭に導入すると共に SIP サーバを利用する。ワームホールデバイスは、相互接続に必要なユーザ認証、NAT トラバーサル、DLNA 機器情報の管理、UPnP 通信の中継などを一括に行う。

nextD[5] では、環境の変化による影響を隠蔽するようなシームレスな遠隔デバイスアクセス機構を提案している。ファイアウォールによるアクセス制限のあるサブネットワーク間の遠隔デバイスアクセス、IP アドレスやデバイス接続先計算機の変化に対してサービス継続が可能であることなどが特徴である。

USB/IP[6] では、すべてのデバイス操作コマンドを IP パケットにカプセリングするようなデバイスドライバを使って、ローカル感覚でリモートデバイスを操作することを提案している。しかし、あくまでリモートデバイスにアクセスすることが目的で、デバイスの連携は考慮しない。

8 今後の課題

現時点において、プロトタイプの実装はまだ部分的であり、Device Config File、Device Access Authenticator、VLAN によるセグメントの分離、ユーザインタフェースなどが未着手である。

プロトタイプ実装完了後は、仮想マシン中に定番ソフトウェアを置き、本システム上で動作させ評価したいと考える。

現時点では、Device Config File と Device Access Authenticator の設計は初期段階にあり、今後それについての議論を深める。

また、一つのスマートスペースにおいて、デバイスが新たに追加されたり、削除されることがしばしば起こると考えられる。今後はそのようなデバイスの追加・削除を動的に検知し、Bridge Controller や Device Access Authenticator などに反映させる仕組みについて検討する。

9 まとめ

本稿では、スマートスペース間をユーザが自由に移動し、移動先においてその場のデバイスを使用したり、遠隔地である自宅内のデバイスにアクセスするというようなスマートスペース間連携に着目した。我々は、異なるスマートスペースを接続できるように整備された環境を複数ユーザにより共有し、ユーザが任意の移動先において選択的にスマートスペースを接続することが可能なシステムを提案した。また、本システムはデバイスの違いを吸収するため、ユーザは移動先においても普段使用するアプリケーションを変更する必要がない。

本システムを用いることにより、ユーザは自由に端末を持ち歩き、決まった拠点やデバイスの場所に縛られずに、移動先や自宅のデバイスを使用したり、それらを連携させることができるようになる。

参考文献

[1] *UPnP Forum:* “*Universal Plug and Play Device Architecture Version 1.0*”, http://www.upnp.org/resources/documents/CleanUPn-PDA101-20031202s.pdf, 2000.

[2] *Bonjour*, http://www.apple.com/jp/macosx/features/bonjour/

[3] *Digital Living Network Alliance,* “*DLNA Networked Device Interoperability Guidelines*”, expanded; March 2006, p.595, 2002

[4] *ECHO-NET*, http://www.echonet.org/

[5] 尾崎亮太, 日高宗一郎, 児玉和也, 丸山勝己, 計算機移動やデバイス移動に対してもサービスが継続可能な遠隔デバイスアクセス機構, 電子情報通信学会論文誌, Vol. J89-B No.8, 2006.

[6] T. Hirofuchi, E. Kawai, K. Fujikawa, H. Sunahara, *USB/IP - a Peripheral Bus Extension for Device Sharing over IP Network*, USENIX Annual Technical Conference, FREENIX Track, 2005.

[7] J. Dike, *User-Mode Linux*, http://user-mode-linux.sourceforge.net/

[8] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, *PlanetLab: An Overlay Testbed for BroadCoverage Services*, http://www.planet-lab.org/, 2003.

[9] 小川将弘, 早川裕志, 小板隆浩, 佐藤健哉, グローバルネットワーク環境における UPnP 機器連携の実現, マルチメディア, 分散, 協調とモバイル (DICOMO2007) シンポジウム, 2007

[10] 武藤大悟, 吉永努, ワームホールデバイス: DLNA 情報家電の遠隔相互接続支援機構, マルチメディア, 分散, 協調とモバイル (DICOMO2007) シンポジウム, 2007