動的に構成を変更可能な仮想スマート環境の構築

榎堀 優[†1] 西尾信彦[†1]

ユビキタス環境を構築するための様々な機器を敷設した空間が，オフィスや家庭，駅ナカなどに敷設されていくと予想され，本稿ではこの空間をスマート環境と呼ぶ．そのうえで，サービスフレームワークを稼働してユビキタス環境を構成したとき，単一のユビキタス環境の構成を，複数のスマート環境で利用するのが難しいという問題や，各入出力機器のアクセス制御が難しいという問題，各ユビキタス環境を構築するサービスフレームワーク間で利用機器が衝突するという問題がある．これらに対し本稿では，動的に構成を変更可能な仮想スマート環境を提案する．仮想スマート環境は，実スマート環境を構成する各機器を仮想化し，組み合わせ方と仮想化した機器へのアクセス制御を用いて，ユビキタス環境ごとに適切なスマート環境を提供する．実装は Java 言語を用いて行い，Unstructured P2P 型の仮想スマート環境構築機構として実現した．評価は，仮想スマート環境の構築時間，実スマート環境と比較した性能低下，複数の仮想スマート環境の共存性の観点から行い，実スマート環境の構成に依存しない実行環境がユビキタス環境に提供できることと，機器の切断・再接続時に復帰処理を行うサービスフレームワークを用いることで実スマート環境に割り当てるユビキタス環境を内部状態を保持したまま切り替えられることを確認した．

A Virtualization of Smart Environment for Flexible Structure Reconfiguration

Yu Enokibori[†1] and Nobuhiko Nishio[†1]

In the future, there will be environments composed of various devices for establishing smart environments in some places such as offices, homes, and stations. In this paper, we discuss two issues of hardware side of smart environments. The first issue is the difficulty of using same software structure of smart environment at different hardware structure. The second issue is access control and collisions between devices among several smart environments established on the same hardware structure. In order to address these issues, we propose a virtual environment with a flexible structure allowing optimization for each smart environment. The virtual environment provides optimized environments for each smart environment, using combination of virtualized devices with controlled access. We developed an unstructured peer-to-peer framework using Java language. The framework is evaluated from three viewpoints: speed of the framework, a comparison of performance degradation in the actual smart environment, and a study of what kinds of collisions happen when virtual environments are used to establish more than one smart environment on the same hardware structure. These results show that the framework can provide a flexible environment for smart environments which is tolerant of unexpected disconnection of devices.

1. はじめに

家庭内やオフィスなどの至るところにコンピュータやセンサ，家電など配置して，それらを統合制御するサービスフレームワークを用いた連携によって利用者を支援するユビキタスコンピューティングが提唱されている．本稿では，家庭内やオフィスなどの至るところにコンピュータやセンサ，家電などを配置したサービスフレームワークの入れ物となる環境を**スマート環境**，スマート環境にサービスフレームワークを導入して，利用者の状況を判断しつつ適切なサービスを提供したり，複数のサービスや機器を連携させて利用者を支援したりする環境を**ユビキタス環境**と呼ぶ．ユビキタス環境を構築するサービスフレームワークの研究[1)–5)]は多数行われており，利用者の場所に応じてサービスを提供する機器を変更したり，行動履歴に応じてサービスを提案したりといったことを可能としている．また，情報家電などで使われる UPnP[⋆1]や Bonjour[⋆2]など

†1 立命館大学
Ritsumeikan University

⋆1 UPnP Forum: Web Page, http://www.upnp.org/

⋆2 Apple, Inc.: Bonjour Overview, http://developer.apple.com/documentation/Cocoa/Conceptual/NetServices/

も広義の意味でユビキタス環境を構築するためのサービスフレームワークといえる．ユビキタス環境は今後，オフィスや家庭，駅の構内に配置された商業スペース（駅ナカ）などに敷設されていくと予想され，建築時に配線やセンサなどを埋め込んだ建造物の登場も考えられている．しかし，スマート環境に対する研究は十分ではなく，本稿では現状のスマート環境における 2 つの問題を対象とする．

1 つ目は，特定のユビキタス環境の構成を，別のスマート環境で利用するのが難しいことがあげられる．これは，研究室内で構築したユビキタス環境を駅ナカのデモ会場へ移設するなどの場合が相当し，スマート環境が空調やモニタなどの場所に依存した入出力機器を内包するために，個々の物理構成が異なることに起因する．例として，配線の取り回しの制約で，研究室内では 1 つの制御ホストで管理していた入出力機器群が，駅ナカでは複数の制御ホストで管理している場合などが相当する．ほかにも，研究室内で利用できたキーボードなどの入出力機器が，駅ナカでは別の機器で代用する必要がある場合などがある．スマート環境が大規模である場合や，その用途によっては，利用するユビキタス環境の構成に合わせてこれらの物理構成を変更することが難しい．また，建築時に配線やセンサなどを埋め込んだ場合は，小規模であっても物理構成を変更することが難しい．

2 つ目は，各入出力機器のアクセス制御が難しいという問題や，複数のサービスフレームワークを同時に利用するときに利用機器が衝突するという問題があげられる．この問題は，ホテルの広間をイベント会場として複数の企画が次々と利用したり，同時に複数の企画を開催したりするように，単一のスマート環境で企画に応じてユビキタス環境を切り替えて利用したり，同時に複数種のユビキタス環境を構築したりする場合に発生する．ほかに，研究室にゲストが来訪したときに，スマート環境の一部を開放する場合などにも発生する．同一のサービスフレームワークを用いたスマート環境間であれば，セキュリティの確保や衝突回避も可能だが，サービスフレームワークを特徴に応じて適切に選択する運用はできない．

そこで本稿では，**動的に構成を変更可能な仮想スマート環境**を提案する．本研究では実スマート環境の物理機器構成を仮想化[⋆1]し，ユビキタス環境を構成するサービスフレームワークが要求する物理機器構成を動的に作り出す．仮想スマート環境は，実マシンに対する仮想化である仮想マシンのように，実スマート環境に対する仮想化を提供して，実スマート環境とサービスフレームワーク間の依存性を解決する．仮想スマート環境を用いた場合，仮想マシン利用時の OS のように，各サービスフレームワークは仮想スマート環境上で動作する．このほかにも，実スマート環境に割り当てる仮想スマート環境を切替え可能とすることで，利用目的に合わせたサービスフレームワークの動的切替えを可能とする．また，利用者ごとに仮想スマート環境で利用できる機能や機器を制限することで，ゲスト独自のユビキタス環境の構築などを安全に行うことを可能とする．

たとえば，実スマート環境の物理機器構成として，モニタ制御を担当するホストが 1 台，空調制御（シリアル経由）を担当するホストが 1 台，ライト制御（シリアル経由）を担当するホストが 1 台あり，これらのリソースは UPnP で管理されているとする．この実スマート環境に対して，独自に Bonjour ベースで作成したプレゼンテーション特化のサービスフレームワークを用いたい状況にあり，特化サービスフレームワークは物理機器構成として，ライト制御（シリアル経由）とモニタ制御が共にできるホストを 1 台，空調制御（シリアル経由）を担当するホストが 1 台，制御サーバ用途に 2 台のホストを要求するとする．このとき，本研究では，各機器と各ホストの結び付きを仮想化し，各機器と各ホストの接続を仮想的に組み替え，また，本例のように実ホスト数以上のホストが要求されたときには仮想ホストを作成し，利用するサービスフレームワークが必要とする物理機器構成を，仮想スマート環境として動的に作り出す．

以下，2 章で我々の提案している仮想スマート環境を実スマート環境と比較しつつ述べ，3 章で本稿で実装した機構について述べる．4 章で実装した仮想スマート環境について，仮想スマート環境の構築時間，実スマート環境と比較した性能低下，複数の仮想スマート環境の共存性の観点から評価を行い，5 章で関連研究を紹介し，6 章でまとめを行う．評価の結果，仮想スマート環境を用いて，実スマート環境の構成に依存しない実行環境をユビキタス環境に提供できることを確認した．また，機器の切断・再接続時に復帰処理を行うサービスフレームワークを用いることで，実スマート環境に割り当てるユビキタス環境を，その内部状態を保持したまま切り替えられることを確認した．

⋆1 仮想化：物理的境界を隠蔽し，論理的単位で分割や統合，配置の変更などを可能とすること

2. 仮想スマート環境の提案

仮想スマート環境は，実スマート環境とサービスフレームワーク間の依存性の解決を目的としており，実スマート環境内の物理機器構成によらず，サービスフレームワークに適切な構成を提供できる必要がある．また，実スマート環境上で利用目的に適したサービスフレームワークへの切替えを可能とするために，単一の実スマート環境上に複数の仮想スマート環境を構築可能であり，実スマート環境と仮想スマート環境の割当てを動的に変更できる必要がある．さらに，ゲスト用に制限された仮想スマート環境などの構築を可能とするため，利用者別に利用できる機器構成の管理を行えることが求められる．

本章では，まず，実スマート環境の構成について述べ，次に，実スマート環境の構成と比較しつつ，動的に構成を変更可能な仮想スマート環境の構成を定義する．その後，運用時の動作として構築・停止・休止・再開について述べ，実スマート環境に割り当てる仮想スマート環境の切替え動作と，利用者別の仮想スマート環境の権限管理について明らかにする．最後に，仮想スマート環境を利用する場合の制約について述べる．

2.1 実スマート環境の構成

図 1 に実スマート環境の構成の概要を示す．まず，実スマート環境中のリソースは，制御機器と制御機器に接続されたリソースに分けられる．以後，制御用の PC なども制御機器と表記する．制御機器以外のリソースは，さらに入出力機器と他の機器に分けられる．

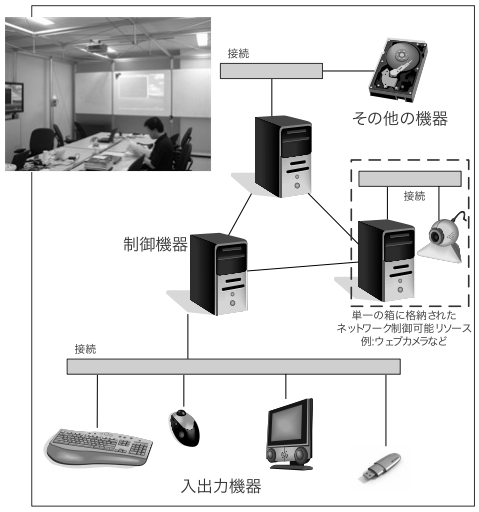

図 1 スマート環境の内部構成
Fig. 1 Simple structure of Smart Environment.

- 制御機器
  各種入出力機器が接続された PC や遠隔制御可能なモニタ切替え器などが相当し，通常，ユビキタス環境を制御するサービスフレームワークは制御機器上で動作する．
- 入出力機器
  人物や空間の状態を検知するセンサや情報を出力するモニタなどが相当する．
- 他の機器
  外部記憶装置などが相当する．物理的位置に依存しないリソースであり，仮想的に作成されていてもよい．

実際にスマート環境を構築する場合には，PC などの明確な制御機器とそれに接続されたリソース以外にネットワーク経由で利用可能な可能なリソースとして，ウェブカメラや NAS，UPnP 制御のセンサや情報家電などが存在する．これらのリソースは，制御機器に接続されていない独立のリソースに見えるが，制御機器とデバイスが単一の箱に格納されているもの，と考えることもできる．たとえば制御機器にカメラが接続され単一の箱に納められているのがウェブカメラであると考えられる．

以上のことから，本稿ではスマート環境の構成を制御機器間のネットワークによって構成され，制御機器以外のリソースはいずれかの制御機器に接続されているものとして簡略化して扱う．

2.2 仮想スマート環境の構成

前節で述べたように，実スマート環境の物理機器構成は制御機器間のネットワークによって構成され，制御機器以外のリソースはいずれかの制御機器に接続されている．よって，動的に構成を変更可能な仮想スマート環境の実現は，制御機器と制御機器以外のリソースの仮想化と，制御機器と制御機器以外のリソースの組合せを，接続ホストが異なるなどの実スマート環境の物理機器構成にとらわれずに組み替え可能とすることで実現できる．仮想スマート環境の内部構成を**図 2** に示す．

仮想スマート環境は，仮想化された制御機器（**仮想制御機器**）と，仮想化された制御機器以外のリソース（**仮想リソース**）によって構成される．仮想リソースは仮想制御機器とは独立であり，どの仮想制御機器へも割り当てることができる．目的の仮想スマート環境の構築に不要な仮想リソースは，仮想制御機器に接続されず，空き仮想リソースとして仮想リソースプールに保持される．

仮想化された制御機器は利用者と物理的接触を持た

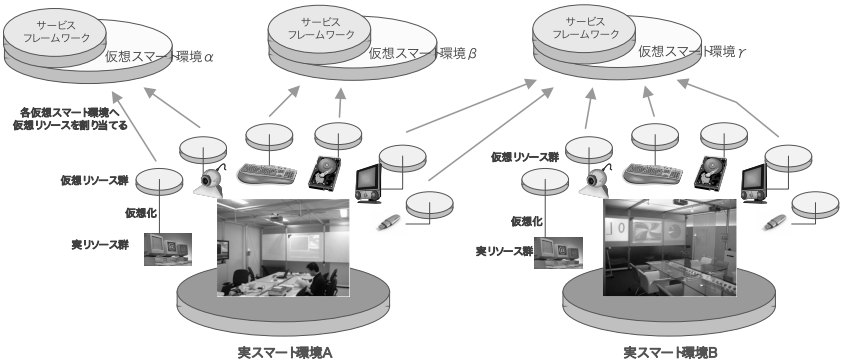

図 3 仮想リソースの組合せによる仮想スマート環境の構築
Fig. 3 Examples of Virtaul Smart Environments composed of selected Virtual Resources.

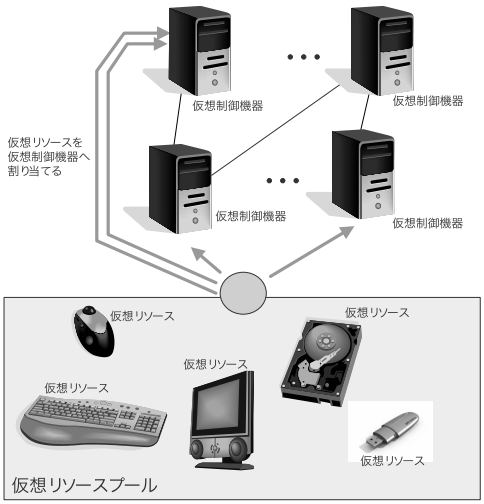

図 2 仮想スマート環境の内部構成
Fig. 2 Structure of Virtual Smart Environment.

ないため，外部記憶装置の容量や CPU の処理能力が許す限り，必要個数が仮想的に作成される．一方で，仮想スマート環境内の入出力機器は実スマート環境内の入出力機器と結び付けられる必要があるため，実スマート環境内に存在しない入出力機器を利用したり，実在個数を超えて利用したりすることはできない．不足している入出力機器をエミュレーションなどで補うこともできるが，種々あるデバイスすべてを個別にエミュレートすることは現実的ではないため，本稿ではこれを扱わない．

以上のことから，単一の仮想スマート環境は，仮想制御機器の設定（種類や個数，配置するサービスフレームワークの情報）と，各仮想リソースと仮想制御機器の組合せとして表現される．

2.2.1 実スマート環境との関係

仮想スマート環境では，各仮想スマート環境を構成する仮想リソースの組合せを制御することにより，単一の実スマート環境に複数の仮想スマート環境を構築したり，異なる実スマート環境のリソースを組み合わせて単一の仮想スマート環境を構築したりすることを可能とする（**図 3**）．図 3 中では仮想スマート環境 α と β として，実スマート環境 A 上に複数の仮想スマート環境を構築しており，異なるユビキタス環境を単一の実スマート環境で稼働させることが可能となっている．また，図 3 中の仮想スマート環境 γ のように，複数の実スマート環境の仮想リソースを組み合わせた単一の仮想スマート環境を構築することで，単一のユビキタス環境のみで複数の実スマート環境にまたがったサービスを作成可能としている．

2.3 仮想スマート環境の運用

本節では，実際の仮想スマート環境の構築・停止・休止・再開の手順について述べる．構築時の手順で，利用者別に適切な権限管理が行われた仮想スマート環境の構築動作について述べ，停止・休止・再開時の手順で，仮想スマート環境を実スマート環境に割り当てる場合の動作について述べる．

2.3.1 仮想スマート環境の構築

仮想スマート環境の構築時に利用者は次に示す 4 段階の手順を踏む．

(1) 認証
(2) 利用可能な仮想リソース一覧の取得
(3) 利用する仮想リソースの選択
(4) 仮想スマート環境の構築

認証処理 (1) により，仮想スマート環境の管理者は利用者を特定でき，適切な仮想リソースのみを利用者へ提供できる (2)．利用者側は自己の判断で不適切なリソースを除外した後に，仮想スマート環境を構築する (3)．以上の手順を踏むことにより，仮想スマート環境を双方の適切な判断の下で安全に運用することが

できる．

仮想スマート環境の構築 (4) は，構築するユビキタス環境で必要となる数の仮想制御機器を構築し，それぞれに必要となる仮想リソースを接続することで行う．構築後の仮想スマート環境の各仮想制御機器に必要なサービスフレームワークを導入して，利用するユビキタス環境を構築する．

2.3.2 仮想スマート環境の停止・休止・再開

仮想スマート環境の停止は，利用している仮想リソースの開放と，仮想制御機器の停止によって行われる．まず，利用している仮想リソースが開放され，仮想制御機器が停止される．

仮想スマート環境の休止は，仮想スマート環境の構成情報と仮想制御機器の動作状態を保存することで行う．これにより，状態を維持したまま運用中の仮想スマート環境を保存することが可能となる．仮想スマート環境で動作しているユビキタス環境が厳格なプロトコルを利用している場合には，仮想スマート環境を構成する複数の機器の情報を完全に同期しつつ保存することが求められる．

再開時には，利用者は構築時と同様の手順を踏み，仮想制御機器へのサービスフレームワーク導入時に，休止時に保存した仮想制御機器の動作状態を復元する．仮想スマート環境の再構築は，認証で得られた利用可能な仮想リソースの範囲内で行うため，利用者の権限の変更や実スマート環境の物理機器構成の変更などにより，利用可能リソース中に再構築に必要な仮想リソースが存在しないことがある．この場合，利用者に通知し，代用仮想リソースの提案，該当する仮想リソース抜きでの再構築，再構築の停止，などの対処を行う．実際に利用可能な手法は，実装に依存する．

2.4 仮想スマート環境の制約

仮想スマート環境は実スマート環境と比較して，同一構成のユビキタス環境を簡単に作成でき，リソース別の適切な権限管理が可能である．一方で，仮想スマート環境の構築に使えるリソースが，仮想化機構が利用している仮想化技術によるものに制限され，さらに仮想化のために各処理性能が低下する．仮想化技術を用いるためにこれらの制限は避けられないが，処理性能の低下は実装時に十分に高速な仮想化技術を用いることで低下の幅を抑えることはできる．

また，2.2 節では，スマート環境の構成を簡略化して，ネットワーク経由で利用可能な可能なウェブカメラやセンサ，NAS なども，単一の箱の内部に制御機器とデバイスが接続されつつ格納されているものとして扱ったが，組み込み型の制御機器を仮想化することは難しく，実際にはこれらが利用できないことが多い．組み込み型制御機器の仮想スマート環境への取り込みは仮想スマート環境内の仮想制御機器間に仮想ネットワークを構築し，その仮想ネットワークへの参加の可否もって行うことも可能ではある．しかし，制御機器として仮想化ができるわけではなく，また，仮想ネットワークを作成することは仮想スマート環境の独立性を高めるが，実スマート環境との協調性は低下させてしまう．本稿では実環境との協調性を重視し，組み込み済み制御機器の取扱いは今後の課題とする．

3. 実　　装

本章では，仮想スマート環境を構築するために作成した，**仮想スマート環境構築機構**（以下，**仮想化機構**と呼ぶ）について述べる．仮想化機構は Java 言語を用いて実装し，Sun Microsystems 社の J2SDK 1.5 上で動作するように設計した．本章ではまず，仮想化機構の全体構造について述べ，その後，実装における各リソースの仮想化方法について述べる．最後に，仮想スマート環境の構築時に関する実装と，作成したクライアントプログラムについて述べる．

3.1 全 体 構 造

実装の全体構成を**図 4** に示す．仮想化機構は，仮想制御機器インタフェースと仮想リソースインタフェースを持ち，仮想制御機器や仮想リソースの構築に利用する仮想化技術をモジュール単位で必要に応じて追加できる．本実装では，制御機器として年々小型化が進む PC を想定し，仮想制御機器を仮想マシン（VMware[⋆1]）を用いて実装した．制御機器以外のリソースとしては，PC に接続されているキーボードやモニタ，マウスと，

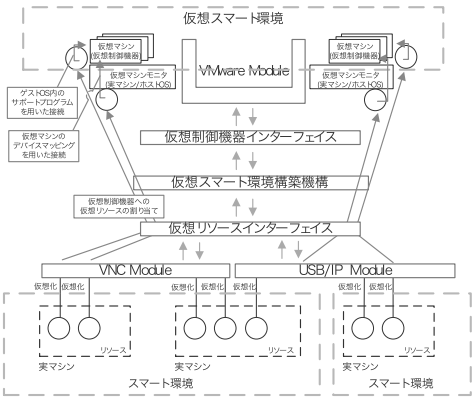

図 4 実装の全体構成
Fig. 4 Structure of the implementation.

⋆1 VMware, Inc.: VMware, http://www.vmware.com/

USB デバイスを用い，仮想化には VNC[6] と Linux の USB over IP 実装である USB/IP[7] を用いた．USB を利用できるため，USB-シリアル変換デバイスを用いることで，多彩な機器を接続することができる．

モジュール化により，現存する多彩な仮想化技術の取り込みや，新しい仮想化技術の取り込みを簡単に行うことができ，仮想制御機器に速度を求める場合は運用に制約があるが高速な仮想マシンとして User Mode Linux[8] や Xen[9] の仮想制御機器モジュールを追加する，仮想制御機器に多彩な CPU エミュレーションを求めるなら QEMU[10] の仮想制御機器モジュールを追加する，仮想リソースにで SCSI ストレージを扱いたければ iSCSI[11] の仮想リソースモジュールを追加する，などの利用目的や仮想化技術の特徴に合わせた拡張が考えられる．

各仮想リソースは SHA-1 を用いた識別子（以下，**UID**）を持ち，UID は単一の仮想化機構内で 1 度確定した後は不変である．各仮想リソースの情報は仮想リソースを提供する各ノードに保存され，検索は Unstructured P2P 型で行う．これにより，仮想リソースを提供するノードが離脱したときに，離脱ノードが提供していた仮想リソースの情報が仮想化機構全域から削除される．一般に Unstructured 型の検索より，Structured 型の検索の方が性能が良いことが知られている[12] が，本研究で対象としている仮想スマート環境では内包するノード数が少ないため Structured 型の利点が生かしきれず，また，一般的な Structured 型では値がどのノードに補完されているのかは自明ではなく，突発的離脱時の情報の削除が難しいことから，本実装では Unstructured P2P 型を用いた．同様に仮想化機構全体も各ノードの突発的な参加や脱退への耐性が必要であり，また，前述の仮想リソース情報管理との相性から仮想化機構全体も Unstructured P2P 型で実装した．

3.2 仮想スマート環境の構築

ユビキタス環境を構築するサービスフレームワークの仮想制御機器への導入は，サービスフレームワークを設定済みの仮想マシンディスクイメージを利用した．各ディスクイメージは仮想制御機器の初回構築時に，スケルトンからコピーして利用する．1 度登録されたディスクイメージには UID が設定され，次回からの構築（主に設定情報を用いた自動再構築）時には，構成情報中に保存された UID の指定により利用される．UID は単一の仮想化機構内で確定した後は不変である．

各仮想リソースの仮想制御機器への接続は 2 つの形態があり，仮想マシンのゲスト OS 上で動作しているサポートプログラムを通じて行うか，仮想マシンのホスト OS 側*1 に接続した後に，仮想マシンのデバイスマッピング機構をもって行う．前者の場合，利用可能な仮想化技術がゲスト OS に依存するが，後者の場合よりも良い性能が期待できる．後者の場合，ゲスト OS の種類に制限されずに仮想リソースを提供可能であるが，利用可能な仮想化技術がホスト OS および仮想マシンの実装に依存し，仮想化が 1 段階多く行われるために前者の方法より性能が低下する．実装では，これらを併用可能とし，構築するユビキタス環境に合わせて使い分ける形をとった．たとえば，USB デバイスを Windows から利用する場合，USB/IP は Windows で利用できないため，Linux を用いたホスト OS 上に仮想制御機器を構築し，USB デバイスをホスト OS へ USB/IP を使って接続した後に，VMware のデバイスマッピングを使って Windows 上のサービスから利用した．これら仮想リソースの接続は，仮想化機構に参加している各ノード間で構築するトンネル内を経由し，トンネルの構築を制御することで適切な権限下での接続を可能としている．

このほかに，仮想スマート環境の構築をサポートするための，GUI クライアントを実装した．作成した GUI クライアントを**図 5** に示す．GUI クライアントでは，スケルトンからの仮想制御機器の新規構築/削除や各仮想制御機器へ仮想リソースの接続が行える．作成した仮想スマート環境の構成情報は，UID を用いて表現した XML 形式の設定ファイルに保存でき，保存した構成情報は仮想スマート環境の再構築時に利用で

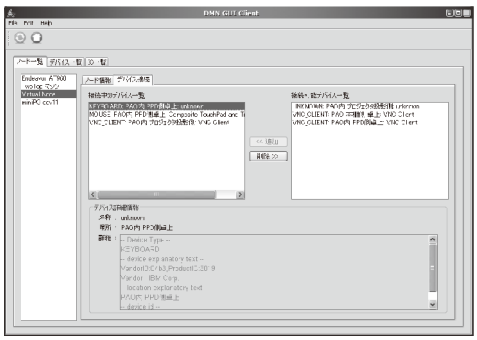

図 5 GUI クライアント
Fig. 5 GUI client.

*1 仮想マシン技術によってはホスト OS という概念がないものもある．ここでは制御用の OS を指してホスト OS とする．

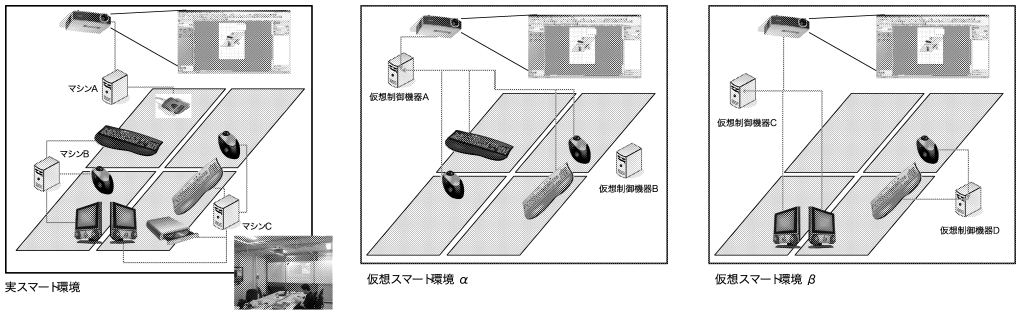

図 6 実験時の構成
Fig. 6 Structures of Physical/Virtual Smart Environments used at evaluation.

表 1 実験時に配置した実リソース
Table 1 Physical resources used at experiments.

マシン	リソース	利用モジュール
マシン A	プロジェクタ	VNC MODULE
	カードリーダ	USB/IP MODULE
マシン B	モニタ	VNC MODULE
	キーボード	USB/IP MODULE
	マウス	USB/IP MODULE
マシン C	モニタ	VNC MODULE
	キーボード	USB/IP MODULE
	マウス	USB/IP MODULE
	DVD-ROM ドライブ	USB/IP MODULE

きる．再構築時に構成情報に記載されているリソースが見つからない場合，利用者へ通知して，特定リソースを省いた形で仮想スマート環境の構築を続行するか，構築を中止するか，の判断を仰ぐ．また，他のクライアントプログラムとして，FeliCa に対応付けた構成情報による環境構築プログラム，USB メモリに保存した構成情報からの環境構築プログラムを作成した．

4. 評　　価

本稿では，実スマート環境上で仮想スマート環境を運用する場合の影響について，仮想スマート環境の構築時間，実スマート環境と比較した性能低下，複数の仮想スマート環境の共存性の 3 点から評価を行った．

4.1 実 験 環 境

研究室内の環境と，運用時に構築した仮想スマート環境を**図 6** に示す．実験に利用した実スマート環境には，**表 1** に示すリソースを配置した．また，実験時に利用した各マシンの性能を**表 2** に示す．表 2 中の各マシンは OS として Ubuntu 6.10 を使用し，JavaVM として J2SE build 1.5.0_08-b03 を用いた．各マシンは単一のスイッチングハブを用いて 100BASE-TX の LAN を構築した．仮想制御機器はすべてマシン C 上で動作させ，仮想マシンとしては VMware Server 1.0.1-29996 を用いた．構築済み仮想制御機器と仮想制御機器のスケルトンとしては**表 3**，**表 4** に示すものを用いた．

表 2 実験時に利用した各マシン性能
Table 2 Performances of machines used at experiments.

	マシン A
CPU	Core Duo T2600 (2.16 GHz)
メモリー	2 GB
HDD	Maxtor 6V300F0 270 GB SATA
	マシン B
CPU	Pentium(R) M 1.73 GHz
メモリー	512 MB
HDD	HTS541040G9AT00 40 GB ATA
	マシン C
CPU	Celeron(R) 2.53 GHz
メモリー	512 MB
HDD	SAMSUNG HD160JJ 160 G

表 3 構築済み仮想制御機器
Table 3 Composed virtual control devices used at experiments.

	仮想マシン A-C	仮想マシン D
仮想イメージサイズ	8 GB	8 GB
メモリ	256 MB	512 MB
OS	Ubuntu 6.06	Windows XP sp2
Guest 拡張	USB, VNC	無

表 4 仮想制御機器のスケルトン
Table 4 Skeletons of virtual control devices used at experiments.

	Ubuntu	Windows XP sp2
仮想イメージサイズ	8 GB	8 GB
メモリ	256 MB	512 MB
OS	Ubuntu 6.06	Windows XP sp2
Guest 拡張	USB, VNC	無

本実験で仮想スマート環境を 2 組構成し，それぞれで異なるサービスを実行した．作成した仮想スマート環境の構成は図 6 に示している．仮想スマート環境 α は複数人で編集可能な共有議事録を運用する構成であり，4 つの入力機器を USB/IP モジュールを用いて集

約し，1 つのモニタを VNC モジュールを用いて接続した 1 台の仮想制御機器と，サービスフレームワークが稼働するサーバ用途に利用した 1 台の仮想制御機器が存在する．仮想スマート環境 β は多画面プレゼンテーションを行う構成であり，3 つの画面を VNC モジュールを用いて集約した 1 台の仮想制御機器と，プレゼンテーションの進行制御用に 2 つの入出力機器を USB/IP モジュールを用いて集約した 1 台の仮想制御機器が存在する．これらの仮想スマート環境の構築は GUI クライアントを用いて行い，切替えには FeliCa に対応付けた構成情報による環境構築クライアントを用いた．構成設定の XML ファイル，仮想制御機器を構築するためのディスクイメージ，スケルトンはすべてマシン C に配置した．

4.2 仮想スマート環境の構築時間

仮想スマート環境の構築時間の計測には，仮想スマート環境 α の構成を用いて 20 回の計測を行った．仮想マシンが停止した状態から仮想スマート環境を構築した場合の平均は 96 秒であった．このうち，仮想マシンに依存した時間を除くと，平均は 3 秒程度となる．このことから，本実装の動作の大部分は利用する仮想マシンの動作に依存していることが分かる．

この 3 秒程度は，ほぼ USB/IP の接続の待ち時間である．USB/IP を用いたデバイスの着脱を短い間隔で連続して単一ホストに対して行った場合にエラーが発生したため，本実装では各デバイスの着脱を 1 秒以上の間隔を空けて行っている．実際の間隔は処理負荷などによって多少前後する．本実験で利用した仮想スマート環境 α では，単一仮想制御機器に接続する USB デバイスは 4 つであり，合計で着脱時の待ち時間は 3 秒程度となる．よって，仮想化機構のメッセージング部などはミリ秒単位で終了している．

環境未構築からの構築時間の計測も，同様に仮想スマート環境 α の構成に対して 20 回の計測を行い，全必要時間は平均で 575 秒であった．このうち，スケルトンからコピーを行う時間が大半を占め，コピー時間と仮想マシンに依存する時間を除くと平均で 6 秒以内に収まった．この処理には起動時と同様の処理が含まれるため，コピー時間と仮想マシンに依存する時間を除いた平均 6 秒のうち，3 秒程度は前述した起動時に必要とされる時間である．残る 3 秒は安定化のためにとっている各処理間の待ち時間であり，複製後の処理開始までの待ち時間，仮想マシンの登録後の待ち時間，初回起動設定時の各ローカルコマンド呼び出しの待ち時間などの合計となり，各待ち時間を適切な値とすることで短縮できると見込まれるが，今回は最適化を行っていない．

環境構築済みからの起動，環境未構築からの構築のどちらの場合も，仮想化機構の動作時間は仮想マシン依存の時間と比較して 1/30，1/95 と小さく，仮想化機構の性能は利用している仮想化技術や外部記憶装置の性能に大きく依存することが分かる．

停止処理の場合，稼働中の状態からは全体で平均 29 秒，休止には全体で平均 7 秒がかかった．休止中からの再開には，全体で平均 33 秒かかった．停止/休止/再開の各処理についても起動/構築処理と同様の傾向が示された．

4.3 実スマート環境と比較した性能低下

仮想スマート環境における性能の低下は，利用した仮想化技術による性能低下と，デバイス接続時に用いたトンネルによる性能低下が考えられる．このうち，利用した仮想化技術による性能低下は本稿で議論しない．トンネルについて確認しておくと，トンネルは Java を使って実装されており，仮想リソースの接続はトンネル内部を経由して行う．

以上の条件でトンネルを経由して仮想リソースを接続した場合と，トンネルを経由せず直接接続した場合の速度比較を行った．マシン B - マシン C 間にトンネルを作成し，仮想リソースとして USB/IP を用いて接続した CD-ROM ドライブを用いて，CD-ROM 中に保存した 703 MB のデータを転送した．20 回の平均値は，トンネル経由時で 248.435 秒，2.8 MB/s，通常接続時で 255.327 秒，2.8 MB/s となり，有意な差は見られなかった．これは転送速度が 100BASE-TX の実測値を飽和していないためである．そのため，単純なデータ転送プログラムを C 言語で作成し，通常接続時とトンネル経由時の最良値を計測した．その結果，通常接続時は 9.8 MB/s，トンネル経由時は 9.1 MB/s となり，トンネル経由時の最大速度が 0.7 MB/s 低下していることが分かった．しかし，トンネル経由時の最大速度は，利用した仮想化技術の最大速度を超えず問題にはならなかった．よって，実スマート環境と比較した場合の性能低下は利用する仮想化技術に依存し，適切な仮想化技術の選択が求められることが分かる．

4.4 複数の仮想スマート環境の共存性

実スマート環境に割り当てる仮想スマート環境を切り替えたり，実スマート環境上に複数の仮想スマート環境を構築したりする場合，双方の仮想スマート環境で同時に利用できないリソース（たとえば，マウスは双方に接続しても正常に動作しない）が競合していない場合は，切替えや並列動作は問題なく動作する．

一方で，競合しているリソースが複数の仮想スマー

ト環境で同時に利用できないリソースである場合，どちらかが実リソースを占有する．単一の仮想リソースが複数の仮想スマート環境から同時に利用できるかは，実装に依存する問題なので議論しない．なお，本稿の実装では衝突のあるリソースは同時に利用できず，仮想スマート環境が実スマート環境に割り当てられるときに仮想化機構が必要に応じて仮想リソースの接続先を切り替える．そのため，USB/IP モジュールが競合した仮想スマート環境の切替えを行った場合，仮想リソースが切断された方の仮想スマート環境から該当リソースが検出できなくなり，カードリーダや DVD-ROM ドライブ上のデータを読み出していたサービスが影響を受け停止した．本問題は，仮想スマート環境上で動作するサービスフレームワークを変更し，デバイス断絶を検出して自動的に読み込み処理を休止し，再接続時に再開処理を行うような拡張を行うことや，利用者が問題を意識しつつ，切替え前に影響のあるサービスを停止するなどの運用を行うことで抑止できた．

仮想スマート環境の切替え時の動作時間を利用する仮想リソースが競合している図 6 の仮想スマート環境 α と β を同時に稼働させつつ，実入出力機器との接続を動的に切り替えて行ったところ，必要となった処理時間は平均で 3 秒程度であった．切替えには各環境と結び付けられた FeliCa を利用し，平均処理時間の 3 秒に操作時間は含まれていない．

4.2 節で示した構築動作時と同じく，この平均処理時間 3 秒は，ほぼ USB/IP の接続待ち時間である．実験で利用した仮想スマート環境で USB/IP で接続するデバイスの最大個数の平均は $(4(\alpha) + 2(\beta))/2 = 3.0$ である．よって，平均処理時間は 3 秒のうち，USB/IP の実装に依存する処理時間を除いた，仮想化機構だけでの動作時間は最大でも 1 秒程度である．突発的に停止したノードがあり，再接続を内部的に数回試みた場合に 1〜6 秒程度余分にかかることがあり，これが平均処理時間を増加させている．

4.5 休止/再開時のサービスの継続性に対する問題

休止処理は仮想スマート環境を構成する仮想制御機器を同時に休止させることで行われる．しかし，この休止処理はすべてが同時に終わることはなく，仮想マシンを動作させている実マシンの性能などにより，各仮想制御機器で数ミリ秒から数秒のずれが生じ，再開時にも同様にずれが生じる．これにより，仮想制御機器間の内部状態にずれが生じ，サービスフレーム側に補正機能がない場合は，同期を要求するサービスなどで障害が発生する．実験では，2 台の制御機器間で FTP 通信を行っている状態で停止/再開処理を行ったところ，接続を維持できず，処理が中断されてしまった．完全に同期しつつ休止/再開が可能な仮想マシンを用いることで，この問題は解決できるかもしれない．しかし，現在の実装では不可能である．利用する仮想マシンの変更の検討も含め，今後の課題とする．

4.6 評価のまとめ

実験の結果から，仮想化機構の性能は利用する仮想化技術に大きく依存することと，実スマート環境に割り当てる仮想スマート環境の切替え時に競合リソースがある場合や，仮想スマート環境の休止/再開時のサービスの継続性に一部問題が残るものの，実スマート環境に対する仮想スマート環境を用いたサービスフレームワークの切替えや，単一スマート環境に対する複数の仮想スマート環境の構築を可能としていることが分かった．

環境構築済みからの起動，環境未構築からの構築，それぞれの平均所要時間は 96 秒，575 秒であった．仮想スマート環境の起動や構築は，スマート環境の各機器を起動や，各機器への OS のインストールと同様の処理であり，実スマート環境の起動や構築に比べ大きな差は見られない．また，どちらの場合も，仮想化機構の動作時間は仮想マシン依存の時間と比較して 1/30，1/95 と小さく，仮想化機構の性能は利用している仮想化技術や外部記憶装置の性能に大きく依存することが分かる．よって，仮想化機構の高速化には，高速に起動可能な仮想マシンを用いたり，RAID を構成した高速な外部記憶装置を用いたりすることなどが必要となる．たとえば，仮想制御機器で動作する OS が Linux に限定可能ならば，VMware より高速な UML を利用するといった方法が考えられる．また，外部記憶装置に RAID0 のストライピングを適用して書き込み性能をほぼ 2 倍とした場合，環境未構築からの構築であれば，外部記憶装置依存時間は，全処理時間（575 秒）− 起動平均時間（96 秒）− 初回起動時依存時間（3 秒）= 476 秒となり，その半分の 238 秒の短縮が見込まれ，約 41%の性能向上が見込まれる．また，他の仮想化機構に依存する性能低下は，トンネルによる最大性能の低下があるが，トンネル経由時の最大速度は利用した仮想化技術の最大速度を超えず実装上は問題となっていない．そのため，デバイスの通信速度の性能改善を求める場合，利用している仮想化技術の最適化を行うことが求められる．

現在の実装では，実スマート環境に結び付ける仮想スマート環境の切替え時に競合リソースがある場合や，仮想スマート環境の休止/再開時のサービス継続性に一部問題が残る．このうち，仮想スマート環境の切替

え時の問題に対しては，サービスフレームワークにデバイス断絶を検出して自動的に読み込み処理を休止して再接続時にレジュームを行う変更を加えたところ，発生を抑止できた．また，利用者が問題を意識しつつ，切替え前に影響のあるサービスを停止する運用を行うことで，サービスを継続しつつ仮想スマート環境を切り替えることは可能であった．休止/再開時の問題は，現在の実装では解決できない．利用する仮想マシンの変更の検討も含め，今後の課題とする．

5. 関 連 研 究

ユビキタス環境を構築するサービスフレームワークとして様々なものが研究されており[1)–5)]，モバイル端末用に利用者の位置に合わせて利用するデバイスを順次切り替えていく拡張が行われたもの[13)]も存在する．これらは，実スマート環境の柔軟な運用を行うが，本研究とは対象としているレイヤが異なる．本研究は実スマート環境に対する仮想化を提供し，これらのサービスフレームワークと実スマート環境の間で動作する．実スマート環境と本研究，サービスフレームワークの関係は，実マシンと仮想マシン，OS の関係にたとえることができ，実スマート環境に対してサービスフレームワーク選択の柔軟性，サービスフレームワークに対して実スマート環境の構成に依存しない柔軟性を提供する．

Plan9[14)] は Bell 研究所で開発されている分散 OS である．Plan9 は単一のラップトップ上で動作することも，分散されたリソース上で動作することもでき，ネットワーク上に分散して配置された各種リソースを用いて単一マシン上で稼働している Unix と同様に扱える．これは本研究中の仮想制御機器に近い．そのため，Plan9 を仮想制御機器として複数動作させることで，仮想スマート環境に近い構造をとることができる．しかし，Plan9 は OS であるため実台数以上の仮想制御機器を作成することや，仮想制御機器で利用する OS の選択ができない．一方，仮想制御機器は仮想マシンに対して各種リソースを接続する形で構成され，リソースの許す限り必要台数を作り出すことや，利用する OS を選択することができる．また，現存するサービスフレームワークは同一の OS 上で開発されていないため，仮想スマート環境は異なる OS で構成された仮想制御機器を作成できる必要がある．

Virtual Network Appliance[15)] では，複数の情報家電の機能を合わせて，仮想的な情報家電（VNA）を作り出す．利用者は VNA Markup Language で VNA を構成するための機能の組合せを表現する．VNA は本稿の定義ではサービスフレームワークであり，VNA 上で様々なサービスフレームワークを動作させることを目的としていない．仮想スマート環境の仮想制御機器は，その上で動作するサービスフレームワークを制限しない汎用性を持っている．また，VNA は単体の VNA を作り出すこと目的としており，複数の仮想制御機器を取り扱う仮想スマート環境とは異なる．

VMware Infrastructure 3[⋆1]（以下，VI3）は，VMware 社が提供している仮想環境ソリューションである．VI3 は仮想マシン制御機構であり，複数台のマシンによって構成されたクラスタ上に，システムリソースが許す限りの仮想マシンを構築でき，その集合をもって仮想環境を構築できる．また，構築された仮想マシン間の仮想ネットワークも構築できる．一方で，VI3 はサーバの運用を目的としたものであり，仮想スマート環境が要求する種々の入出力機器を含めた仮想化や共有，たとえば，別々の実マシンに接続された 2 つのモニタを使って，2 つのモニタが接続された 1 つの仮想マシンを作り出す処理などは扱っていない．

6. お わ り に

本稿では，ユビキタス環境の実行環境となる実スマート環境を仮想化し，簡単な導入や複数のユビキタス環境の共存を可能とする**動的に変更可能な仮想スマート環境**を提案した．仮想スマート環境では，実スマート環境中の制御機器や入出力機器などのリソースを仮想化し，必要に応じて組み合わせることによって，動的に再構成可能な仮想スマート環境を実現している．

実装は Java 言語を用いて行い，Unstructured P2P 型のスマート環境構築機構を実装した．制御機器の仮想化には VMware を，他のリソースの仮想化には VNC と USB/IP を用いた．評価を，仮想スマート環境の構築時間，実スマート環境と比較した性能低下，複数の仮想スマート環境の共存性の 3 点について行った．評価の結果，仮想化機構の性能は利用する仮想化技術に大きく依存することと，実スマート環境に割り当てる仮想スマート環境の切替え時に競合リソースがある場合や，仮想スマート環境の休止/再開時のサービスの継続性に一部問題が残るものの，実スマート環境に対する仮想スマート環境を用いたサービスフレームワークの切替えや，単一スマート環境に対する複数の仮想スマート環境の構築を可能としていることが分かった．

⋆1 VMware, Inc.: VMware Infrastructure 3, http://www.vmware.com/products/vi/

今後は，ネットワーク機器への対応と仮想スマート環境を用いたユビキタス環境運用システムの構築を行っていく．

参 考 文 献

1) Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. and Nahrstedt, K.: Gaia: A Middleware Infrastructure to Enable Active Spaces, *IEEE Pervasive Computing*, Vol.1, No.4, pp.74–83 (2002).
2) Kawaguchi, N.: Cogma: A Middleware for Cooperative Smart Appliances for Ad hoc Environment, *Proc. International Conference on Mobile Computing and Ubiquitous Networking* (*ICMU2004*), pp.146–151 (2004).
3) Brumitt, B., Meyers, B., Krumm, J., Kern, A. and Shafer, S.: EasyLiving: Technologies for Intelligent Environments, *Handheld and Ubiquitous Computing*, pp.12–29 (2000).
4) Kidd, C., Orr, R., Abowd, G., Atkeson, C., Essa, I., MacIntyre, B., Mynatt, E., Starner, T., Newstetter, W., et al.: The Aware Home: A Living Laboratory for Ubiquitous Computing Research, *Cooperative Buildings*, pp.191–198 (1999).
5) Enokibori, Y. and Nishio, N.: Realizing a Secure Federation of Multi-institutional Service Systems, *Ubiquitous Computing Systems, Second International Symposium, UCS 2004*, Tokyo, Japan, November 2004, pp.146–156 (2004).
6) Richardson, T., Stafford-Fraser, Q., Wood, K. and Hopper, A.: Virtual network computing, *IEEE Internet Computing*, Vol.2, No.1, pp.33–38 (1998).
7) Hirofuchi, T., Kawai, E., Fujikawa, K. and Sunahara, H.: USB/IP: A Transparent Device Sharing Technology over IP Network, *IPSJ Digital Courier*, Vol.1, pp.394–406 (2005).
8) Dike, J.: A user-mode port of the Linux kernel, *Proc. 2000 Linux Showcase and Conference*, Vol.2, No.4 (2000).
9) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the art of virtualization, *Proc. 19th ACM Symposium on Operating Systems Principles*, pp.164–177 (2003).
10) Bellard, F.: QEMU, a Fast and Portable Dynamic Translator, *Proc. USENIX Annual Technical Conference, FREENIX Track*, pp.41–46 (2005).
11) Satran, J., et al.: iSCSI, Technical report, RFC 3720 (Apr. 2004).
12) Castro, M., Costa, M. and Rowstron, A.: Peer-to-peer overlays: Structured, unstructured, or both?, Technical report, MSR-TR-2004-72, Microsoft Research Cambridge (2003).
13) Chetan, S., Al-Muhtadi, J., Campbell, R. and Mickunas, M.: Mobile Gaia: A middleware for ad-hoc pervasive computing, *2nd IEEE Consumer Communications and Networking Conference* (*CCNC 2005*), pp.223–228 (2005).
14) Pike, R., Presotto, D., Thompson, K. and Trickey, H.: Plan 9 from Bell Labs, *Computing Systems*, Vol.8, No.3, pp.221–254 (1995).
15) Nakazawa, J., Okoshi, T., Mochizuki, M., Tobe, Y. and Tokuda, H.: VNA: An object model for virtual network appliances, *ICCE 2000 Digest of Technical Papers, International Conference on Consumer Electronics*, pp.364–365 (2000).

(平成 19 年 4 月 13 日受付)
(平成 19 年 10 月 2 日採録)

榎堀　優（学生会員）

1983 年生まれ．2004 年立命館大学理工学部情報学科卒業．2006 年同大学院理工学研究科博士前期課程修了．現在，同大学院博士後期課程に在籍．ユビキタスコンピューティング，街中コンピューティング，アドホックネットワーク等に興味を持つ．

西尾　信彦（正会員）

1962 年生まれ．1986 年東京大学工学部計数工学科数理工学コース卒業，1988 年同大学院理学系研究科情報科学専攻修士課程修了．同博士課程単位取得退学後，1992 年より（有）アクセス研究開発室，1993 年より慶應義塾大学環境情報学部および政策・メディア研究科に勤務．博士（政策・メディア）．2000〜2004 年 JST さきがけ研究 21「協調と制御」領域研究者．2003 年より立命館大学に勤務．現在，情報理工学部教授．自律分散協調システム，ユビキタスコンピューティングとセンシングネットワークの研究開発に従事．1994 年山下記念研究賞．情報処理学会 UBI 研究会運営委員．ACM，IEEE 各会員．