Mobiquitous Environment:

適応型次世代ユビキタス基盤環境の構築

Mobiquitous Environment: Adaptive Ubiquitous Environment for Next Generation

永田 智大　村瀬 正名　権藤 俊一　鈴木 源太　守分 滋
Tomohiro Nagata　Masana Murase　Shun'ichi Gondo　Genta Suzuki　Shigeru Moriwake

慶應義塾大学 政策・メディア研究科（〒252-8520 神奈川県藤沢市遠藤 5322 慶應義塾大学
デルタ棟 S213 徳田研究室　E-mail: ngt@ht.sfc.sfc.keio.ac.jp）

ABSTRACT. Our project developed a platform software for the Mobiquitous Environment. Mobiquitous Environment is a next generation of Ubiquitous Environment which supports adaptation to mobility of people and applications, and changes in devices state. Continuous interaction between users and devices through change in service availability is realized.

1．背景

近年，携帯電話をはじめ無線 LAN など無線技術の進展が目覚しい．W-CDMA，CDMA2000，IEEE802.11a など広帯域無線ネットワーク技術の開発によって，人と人のコミュニケーション手段は音声通話，電子メールだけでなく映像のやりとりも可能なった．加えて，Bluetooth 技術によって人が所有する機器同士を無線ネットワークで接続し，複数の機器を同時に操作することも可能である．こうした無線技術の発達は，移動しながらでも作業を行えるモバイルコンピューティング環境を実現しつつある．また，オフィス，家庭，公共空間などあらゆる環境でインターネット接続可能な機器が設置され，ネットワークを介して制御できるようになってきた．こうした環境の出現によって，人々は屋外から家の機器を操作するだけでなく，周囲に存在する機器を組み合わせて作業を行うことも可能になった．さらに，センサがネットワークに接続され，センサ同士がネットワークを構成する研究が行われている．無線通信の機能を持ったセンサが動的にネットワークを構成し，各センサが取得するデータを収集するセンサネットワークである．将来的にはセンサからのデータをもとにネットワークに接続された機器が自身の動作を変更できるような環境が出てくるだろう．このように今後，移動しながら人と人，人と機器，機器と機器がネットワークを介して協調作業を行うことは重要なことになっていくと考えられる．

2．目的

本プロジェクトでは，ユビキタス環境下での人やアプリケーションの移動，機器の動作変更に着目し，移動に伴う環境の変化を動的に発生させて協調作業を行いやすい環境にし，また移動によって生じる様々な障害に対し動的に適応して継続的な協調作業を行うための支援基盤環境である ”Mobiquitous Environment ”(Mobiquitous は Mobile と Ubiquitous からの造語)を確立する．その際，新たなデバイスを作成するのではなく，既存のデバイスを利用し，その上で動作するソフトウェアによって適した機器を利用するよう動的に適応可能なサービスローミング技術を実現していく．

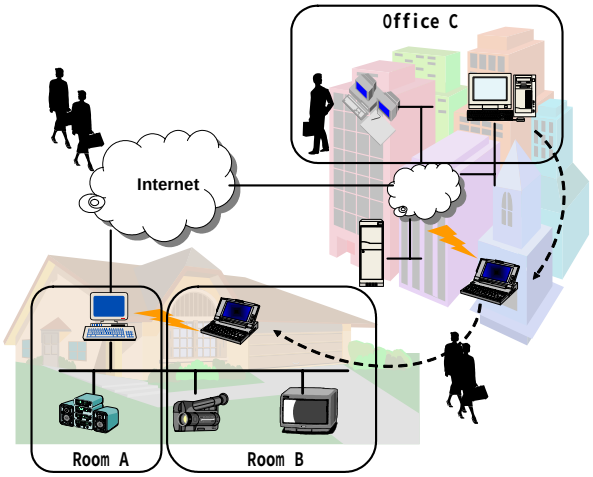

図 1　Mobiquitous Environment イメージ

利用者･機器･アプリケーションが移動しながら人と人，人と機器，機器と機器の協調作業(Human/Device Interaction)を円滑に行えるよう支援する新たなユビキタス環境を構築する．Mobiquitous Environment では，利用者が携帯の容易なウェアラブルコンピュータや PDA(これらを Mobile Client Host，簡単にクライアントと呼ぶ)を常に所有し，そのクライアントが中心となり，周辺にある情報家電やセンサ，ネットワーク上のコンテンツやサーバ(これらを Ubiquitous Service Provider，簡単にサービスプロバイダと呼ぶ)などを利用し，人と機器との協調

作業をおこなっていく．この Mobiquitous Environment を構築するにあたり，本プロジェクトでは 3 つの目的を挙げる．

- 人の移動によるマイナスな変化への動的適応
  第一に Mobiquitous Environment では，人や機器の移動により引き起こされるマイナスな変化に人や機器が動的に適応し，円滑な人や機器との協調作業を継続して行うことを実現する．ユビキタス環境における人や機器の移動は様々な変化を生じる．それまで利用できていた機器が利用できなくなったり，接続するネットワークが切れたりする．その結果，人や機器の間で行われていた協調作業に障害が発生する．そこで,人や機器の移動によって変化するサービスプロバイダの利用状況の変化に伴い，利用するサービスプロバイダとの通信方法を動的に変え，ユーザの要求を継続的に処理することが重要となる.たとえば, 利用者がある部屋で近傍にあるスピーカを利用して音楽を聴いていた．その利用者が部屋から出て別の部屋へ移動したとする．すると，さっきまで利用していたスピーカから音楽が流れていても意味がない．Mobiquitous Environment では利用者の移動によって利用する機器や通信方法を動的に変更できる．それにより，移動した先にある別のスピーカを利用して音楽を継続して聴ける．
- 人の移動によるプラスな変化への動的適応
  第二に Mobiquitous Environment では，人の移動や機器の動作の変化を動的に発生させることで，人の要求をより高いレベルで実現できる環境を構築する．これらの変化は上記のようなマイナスの変化を引き起こすだけとは限らない．人の位置や機器の動作が動的に変化ことで，今まで利用できなかった機器が利用できるようになる．また，より効率よく人の要求を実行できる状況を作り出すこともできる．その結果，人や機器の間で行われていた協調作業がより効率よく行われる．たとえば，利用者が現在いる場所で利用可能なサービスプロバイダが存在しない場合を考える．その場合，その場から一番近くに存在するサービスプロバイダを利用者は見つけ出し，発見されたサービスプロバイダが存在する場所へ移動することが可能となる．また，その場に新たなサービスプロバイダが物理的に移動してくると，利用者は移動せずともそのサービスプロバイダを用いて作業を行うことが可能となる．
- アプリケーションの移動による変化への適応
  第三に Mobiquitous Environment では，人や機器の移動に合わせてアプリケーションも移動することで，負荷の分散を目指す．従来のコンピューティング環境のように負荷のバランスが固定的なクライアント・サーバモデルとは異なり，アプリケーションが自分自身を実行する機器を動的に変えることで，計算能力の低い情報家電やウェアラブルコンピュータなどの負荷を動的にバランシングする．たとえば，計算能力の高いサーバマシーンが利用可能なら，利用する機器に直接関係のないアプリケーションの一部をそのサーバマシーンに移動させることで，機器やウェアラブルコンピュータの負荷を軽減できる．

Mobiquitous Environment では，人や機器，アプリケーションが移動しながら，人と人，人と機器，機器と機器の協調作業を円滑に行え，協調作業が円滑に行えなくなったら，積極的に人や機器が移動して円滑に行える．このような環境を既存の機器を用いて動的適応，動的変化などの新たなソフトウェア技術を構築することによって実現していく．

３．Mobiquitous Environment

本プロジェクトは，先に示した目標を達成するため 3 つの小タスクから構成される．タスク 1 は他のタスクに対しサービス利用の総合的な管理を, タスク 2 においては, 人の移動による利用可能サービスの変化およびサービス負荷分散を実現するためのサービス再構築フレームワークを実現する．タスク 3 では継続的なサービス利用を実現するソフトウェアの開発を行い，人の移動によるマイナスな変化に対し動的に適応する．

- タスク 1：適応的サービス利用管理機構
  ユーザの移動やサービスの開始，停止といったサービス利用状況の環境の変化に対し，アプリケーションが動的に適応して利用するサービスを切り替えることを支援するシステムを構築する.本ソフトウェアはディレクトリサーバ，サービスプロバイダ用ミドルウェアおよびクライアント用ミドルウェアから構成される。ディレクトリサーバはサービスプロバイダの登録ステートとクライアントの検索ステートを管理する．環境の変化が生じた際にディレクトリサーバはそれを検知し，関係のあるクライアントに変化の通知を行う利用状況変化通知機構を提供する．
- タスク 2：機器間協調動作フレームワーク
  機器間協調動作フレームワークは，サービスプロバイダの協調動作を支援するアプリケーションフレームワークである．機器間協調動作フレームワークでは，移動可能な分散モジュール（移動分散モジュール）によってアプリケーションを作成し，各モジュールは適切なサービスプロバイダ上で動作する．本プロジェクトにおいて，ユーザの移動に併せてサービスプロバイダを切り替えるサービスローミング機能，移動分散モジュールを協調動作させる機能，さらにサービスプロバイダ制御用 API を Java 言語によって開発，提供する．
- タスク 3：適応型タスク継続機構
  ユーザの「その時の環境」において「その時の状態」でタスクの継続性を実現する．これを実現するため，アプリケーションレイヤ，セッションレイヤ，トランスポートレイヤ, ネットワークレイヤの情報を集約し，アプリケーション動作状態や通信状態を抽象化した Commutext を生成する. そして, Commutext を用いて，アプリケーションが通信する環境の変化や，ユーザが利用する機器の変更に対して，適応的なタスク継続処理を可能とする．本プロジェクトにおいては，Commutext を生成する状態抽象化機構，管理する状態管理機構，適応的な処理を行う状態適応機構の開発，および，異なる機構間で Commutext を交換するために用いる通信プロトコルの規定などを行う．

次章以降では個々の機能について述べていく．

４．適応的サービス利用管理機構

本タスクでは，利用者やサービスプロバイダの物理的移動に応じて，その場で利用できる適切なサービスを動的に選択できるよう支援する ASAMA (Adaptive Service Availability Management) について述べる．本ソフトウェアはクライアントとサービスプロバイダに対するミドルウェアと，サービスプロバイダのサービス利用可能状況を管理するディレクトリサーバから構成される．

（1）設計

本タスクでは，以下の目標を実現する．

- 利用者の移動によって生じるサービス利用状況の変化(能動的変化)を検知できる

- サービスの移動やサービス提供の開始, 終了などによるサービス利用状況の変化(受動的変化)を検知できる
- ネットワーク障害や負荷上昇によるサービスプロバイダ利用不可能状況の検知
- プログラミング言語非依存なシステムを構築する
- ネットワーク切断・接続時のサービス中断・再開処理の支援

この 4 つの目標を実現するにあたり，4 つの機能を提供できるよう，設計した．

（2）実装

本システムは J2SE 1.3.1 と C 言語を用いて OpenLDAP を拡張することで実装を行った．図 2 にシステム構成図を示す．

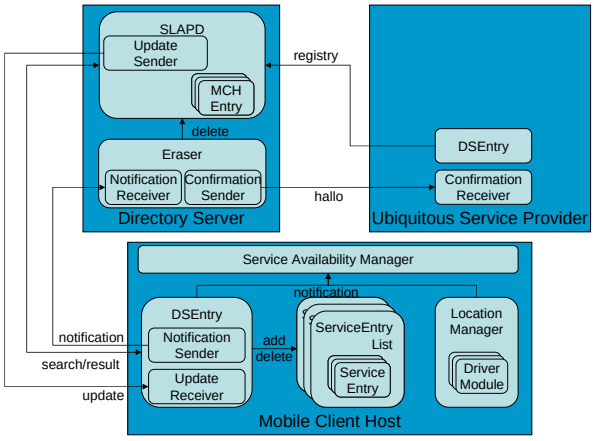

図 2　ASAMA システム図

クライアントの DSEntry モジュールはディレクトリサーバとの通信を管理し，サービスプロバイダの検索，アップデート，ノーティフィケーションメッセージの処理を行う．検索されたサービスプロバイダの情報は検索に利用されたクエリの種類ごとに ServiceEntry オブジェクトとして管理される．ServiceAvailabilityManager モジュールは能動的変化と受動的変化を監視し，変化が生じた場合，アプリケーションが用意する必要な処理を実行させる．ディレクトリサーバの SLAPD はその場所で利用可能なサービスプロバイダのエントリを管理しており，クライアントから検索要求がくると，検索ステートを MCHEntry として保存する．サービスプロバイダの DSEntry モジュールもディレクトリサーバとの通信を管理し，自身の登録・削除を行う．

- 能動的変化の検知機構

能動的変化の検知に関して，クライアントの LocationManager モジュールは利用可能な位置情報センサごとに DriverModule を用意し，ASAMA が定義する位置情報記述スキーマに変換，管理する．もし，利用者の位置情報が変化した場合，その旨を ServiceAvailabilityManager モジュールに通知する．

- 受動的変化の検知機構

受動的変化の検知に関して，サービスプロバイダの DSEntry モジュールからディレクトリサーバの SLAPD へ登録要求もしくは削除要求が行われると，SLAPD は MCHEntry から関係のあるクライアントを検索する．該当するクライアントが発見されると UpdateSender モジュールを通じてクライアントの UpdateReceiver モジュールへ通知が行われる．通知をうけた UpdateReceiver は ServiceAvailabilityManager モジュールに通知を行い，能動的変化が生じたことを教える．

- 協調的サービスプロバイダ管理機構

クライアントがディレクトリサーバに登録されているにもかかわらず，利用不可能なサービスプロバイダを発見すると，NotificationSender モジュールを通じてディレクトリサーバの NotificationReceiver モジュールへ通知を行う．通知をうけたディレクトリサーバは ConfirmationSender モジュールを通じてサービスプロバイダの ConfirmationReceiver モジュールへハローメッセージを送信する．プロバイダはハローメッセージを受信すると，応答をディレクトリサーバに返し，自身が利用可能であることを証明する．もし，ディレクトリサーバは応答を受信できないと，該当するサービスプロバイダが利用不可能状況であると判断し，そのエントリを削除する．

- 無線 LAN によるネットワーク切断予見機構

クライアント派利用者の位置情報を取得するセンサとして無線 LAN を利用することも考えられる．そこで，無線 LAN 用の DeviceModule では位置情報を取得するだけでなく，ネットワーク切断予見・再接続検知を行えるようにした．この DeviceModule が切断予見・再接続検知を行うと，アプリケーションが用意した切断事前処理，接続再開処理を実行させる．

４．適応的サービス利用管理機構

本サービス再構築ソフトウェアは、機器間の協調動作を実現する。たとえば、あるユーザが携行する端末上で音楽再生プログラムを実行していることを想定する。このユーザが屋外にいる場合には、音の出力はヘッドフォン経由になる。ここで、オフィスあるいは自宅など屋内に移動することで新たに高品質のステレオスピーカの利用が可能になる場合がある。この場合、音の出力先をヘッドフォンからステレオスピーカに切り替えることで、ユーザは音楽再生プログラムを柔軟かつサービス品質を向上させて利用できる。

（1）設計

先述のアプリケーションを実現するには、以下の要件を満たさねばならない。

- 機器の切り替え

  機器を切り替えることにより、サービス品質（例えば VGA から SVGA の画面を利用できるなど）の向上が見込まれる。

- 位置透過的なモジュール制御

  Sun RPC（Remote Procedure Call）や Java RMI（Remote Method Invocation）のような遠隔手続き呼び出しは複数の機器の制御を容易にする。

- サービスプロバイダ制御 API

  サービスプロバイダ制御用の API を統一化することで、アプリケーションプログラマは、サービスプロバイダ依存プログラムを意識せずに制御要求をサービスプロバイダに発行できる。

（2）実装

以上の設計方針に基づいて、本ソフトウェアである MoCA を図 3 のように実装する。

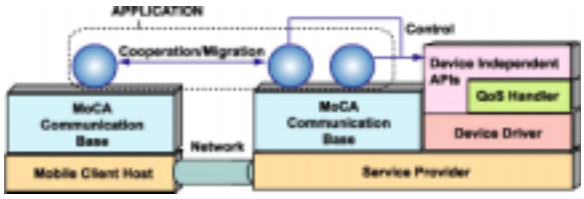

図 3　Moca システム図

本ソフトウェアでは、アプリケーションは複数の移動分散モジュールより構成される。移動分散モジュールは、サービスプロバイダを操作するためのプログラムであり、機器の切り替えに合わせた移動や CPU 負荷などの状況に応じた移動が行える。また、本ソフトウェアが想定するアプリケーションは、移動分散モジュールの他に Config モジュールと呼ばれるアプリケーション定義モジュールが存在する。このモジュールには、移動分散モジュールの構成および、移動分散モジュールが必要とするサービスプロバイダなどが記述されている。モバイルクライアントホストは、Config モジュールを読み込み、ディレクトリサービスと連携して、適切なホスト上に移動分散モジュールをロードあるいは移動させる。
本ソフトウェアでは、移動分散モジュールを容易に扱える機能を提供しており、分散モジュールを保管する機構や分散モジュールを移送する機構、さらに分散モジュールの協調動作を実現する機構がある。これらは通信部（MoCA Communication Base）として提供される。サービスプロバイダの制御を行うには、サービスプロバイダ制御 API を通じてデバイスドライバ（注意：OS で提供されるデバイスドライバとは異なる）を操作するサービスプロバイダ制御部を利用する。

- モジュールデータベース
  モジュールデータベースは各ホスト上で動作し、移動分散モジュールの実体、移動分散モジュールの位置情報（IP アドレスなどのネットワークアドレス）、移動分散モジュールの属性情報（利用できるサービスプロバイダのタイプ、モジュールの ID など）を管理している。またモジュールデータベースにおけるモジュール情報の取得はモジュール ID を検索キーに取得できる。
- 位置透過メソッド呼び出し機構（LTMI）
  位置透過メソッド呼び出し機構では、各モジュール同士の制御に利用される機構である。本機構では、モジュール同士の相対距離関係をモジュールデータベースと協調して算出し、相対距離が 0 の（同一ホストに存在する）場合にはローカルメソッド呼び出しを実行する。相対距離がそれ以外の（2 つのモジュールが別々のホスト上に存在する）場合には、リモートメソッド呼び出しを実現する。
- サービスプロバイダ制御 API
  本システムが提供するサービスプロバイダの制御方法は、既存の OS に見られるデバイスドライバに類似している。サービスプロバイダ提供者は、サービスプロバイダに依存した制御プログラムをデバイスドライバとして実装し、本システムに登録する。アプリケーションは、本システムに登録されているデバイスドライバを動的にロードし、それを利用してサービスプロバイダの制御を行う。これによって、アプリケーションプログラマはサービスプロバイダの実装方法を意識せずに、アプリケーションの構築を行える。

5．適応型タスク継続機構

本ミドルウェアは，こうした想定のもとにおいて，通信作業の適応動作に柔軟性を考慮し，継続的なサービス利用を支援する機構の実現を目的とする．継続的なサービス利用とは，環境変化に対する適応動作において，ユーザが意図するタスクの同一性と一貫性を保証することである．これは，特定のデバイスやアプリケーションの動作それ自体に同一性と一貫性を保証する，連続的なサービス利用支援とは異なる．すなわち，作業状態が時間的に途切れることのなくするための連続性を実現するよりも，環境変化の前後でユーザが実行する作業の一貫性を実現することに，本ミドルウェアの主眼を置いている．

（1）設計

本タスクでは，以下の目標を実現する．

- 環境変化に対して, ユーザタスクを柔軟で適応的に継続すること
- デバイスの移動や切り替えに対して, 適応動作を統一的とすること．
- OS やプログラミング言語，ハードウェアなどの実装方法に依存しないこと
- 抽象化した通信の状態を用いて適応動作をすること

まず，多様で異質なユビキタスコンピューティング環境を前提として，様々な環境変化に対応してユーザ作業の一貫性を実現するため，ユーザタスク継続の柔軟性と適応性を実現することが必要となる．つぎに，環境変化の要因として考えられる 3 つの場合について，差異なく動作することを可能とする．これは，ユーザとともにデバイスが移動する場合，ユーザが移動して異なるデバイスに切り替える場合，そして，ユーザの利用空間においてデバイスが登場・消滅する場合である．そして，これらの場合で統一的に動作する適応機構が必要である. また，サービスの実装方法に多様性を認めるため，特定のオペレーティングシステムやプログラミング言語などに依存しないアーキテクチャとして実現する必要がある．これらを実現するため, 抽象化したサービスの状態を用いて，サービス利用の継続性を実現する．

（2）実装

本システムの全体構成図を図 4 に示す．

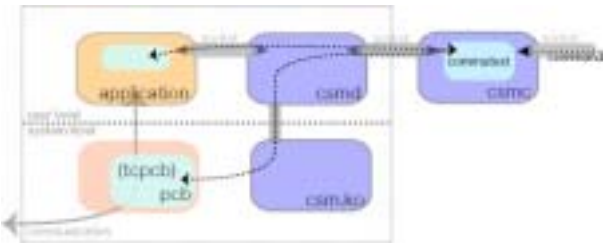

図 4　commutext システム図

以下, 図に示した各機構の実装について説明する. なお，以下に示す例は，相手先ホストのデバイスが変更された場合の動作である．

- 状態抽象化機構と状態反映機構

これらの機構は，ユーザレベルネットワークデーモン csmd を中心にして実装した．このデーモンは，システム内部の状態を操作するためのカーネルモジュール csm.ko とデバイスファイルを用いて通信し，アプリケーションに対する外部インタフェースとして制御ポートを開く．

アプリケーションの状態は, csmd の制御ポートからメッセージをやり取りして操作する．データの型変換はアプリケーションの実装依存であるが，C 言語で実装する場合には，save_comtxt, restore_comtxt の 2 つの関数を利用可能である．
アプリケーションから転送された状態情報は，エンドポイントのソケットペア(ローカルアドレスとリモートアドレス)を識別子としてリンクドリストに保存する．また，システムレベルの状態は，制御コマンドからの命令に応じて，カーネルモジュール(csm.ko)を通じ内部変数を操作する. 内部変数としては TCP コントロールブロックの値を用いた．

- 状態管理機構

この機構は，アプリケーションからの利便性を向上させるために, ネットワークデーモン csmc として実装した．まず，ユーザタスクを制御するプログラムなどに対する外部インタフェースとして制御ポートを開き，そこから受信した制御コマンドに応じて csmd に対する制御コマンド発行する．こうして，デバイスの csmd との間で状態の取得と反映をおこなう．状態は csmc 内で，csmd から読み込んだエンドポイントのソケットペアを識別子として管理し，外部コマンドからの指定もこの識別子を用いる．

- 状態の抽象化

サービスの通信状態を抽象化して管理するため，XML を用いて記述した．このデータは，csmd と csmc の間で交換する. 記述内容は, エンドポイントのソケットペア, 内部変数 TCP コントロールブロックの値, アプリケーション依存値から構成される．

- 適応的継続機構

適応的継続は以下の図 5 で示す 3 通りの場合で可能である．

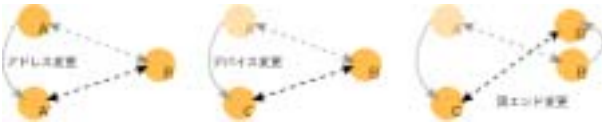

図 5　継続的作業の分類

これを実現するため, システムレベルでの TCP コントロールブロックとアプリケーションレベルでの状態をまとめた Commutext で管理し，様々な環境変化に対して多段階の柔軟な適応処理を可能としている．

5．本製成果を用いた応用アプリケーション

本プロジェクトの成果であるミドルウェア群を用いた応用アプリケーションを作成し，実際のユビキタスコンピューティング環境で動作させ, ミドルウェア群のテスト，および有効性について示していく．

（1）応用アプリケーションの実験環境について

本プロジェクトの成果を用いた応用アプリケーションを実際のユビキタスコンピューティング環境である慶應義塾大学 Smart Space Lab(以下 SSLab と呼ぶ)を使って実装を行った．SSLab を仮想的に複数のユビキタスコンピューティング環境と見立て，その間を利用者やサービスプロバイダが移動する．図 6 に SSLab の俯瞰図を示す．

図 6　SSLab 俯瞰図

SSLab の壁・床・天井はすべて 2 重構造になっており，様々な機器を利用者から隠蔽して設置することが可能である．今回は壁の中に PC を設置し，利用者にはディスプレイしか見えないようになっている．また，スピーカなどは天井に設置されている．
利用者やサービスプロバイダの位置情報取得のためのセンサとして，Intersense 社の IS-600 を利用した．IS-600 から取得できるセンサデータをサービス利用基盤ソフトウェアが定義する位置情報記述スキーマに変換し，利用者とサービスプロバイダの位置を管理する．

（2）サービスローミング技術を用いた TV 電話システム

サービスローミングの例として，今回は TV 電話システムを実装した．利用者は移動しながら周辺に存在するサービスプロバイダを適宜利用し，同じ相手と映像と音声を利用した会話が可能になる．図 7 にサービスローミング技術を用いた TV 電話システムの概要を示す．

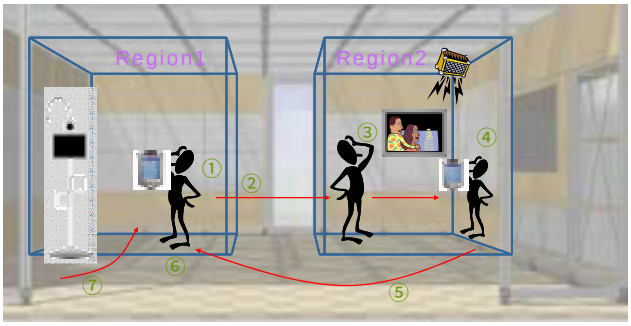

図 7　サービスローミング技術を用いた TV 電話システム

まず利用者は SSLab 内に仮想的に作られたリージョン 1 にいる(①)．そこでは周辺のサービスプロバイダが利用できないため，利用者が装備する iPAQ 上の MoCA から起動されたモジュールを用いて音声による会話を行う．iPAQ には ASAMA が提供するサービス管理のための GUI が表示されてあり，それを利用して隣のリージョン 2 に利用可能なサービスプロバイダが存在することを知り，移動する(②)．リージョン 2 に入ると，ディスプレイサービスとスピーカーサービスが利用できることがわ

かり，MoCA からそれらを利用して映像と音声を用いた会話を行う(④)．再び，利用者はリージョン 1 に移動する(⑤，⑥)．そこに持ち可能な SmartFuniture がリージョン 1 に持ち込まれる(⑦)．SmartFuniture にはディスプレイが接続されており， ASAMA のサービス管理 GUI は新たなサービスプロバイダが利用可能になった旨を利用者に表示する．利用者はそれをもとに SmartFuniture に接続されたディスプレイを用いて相手と会話を行う．

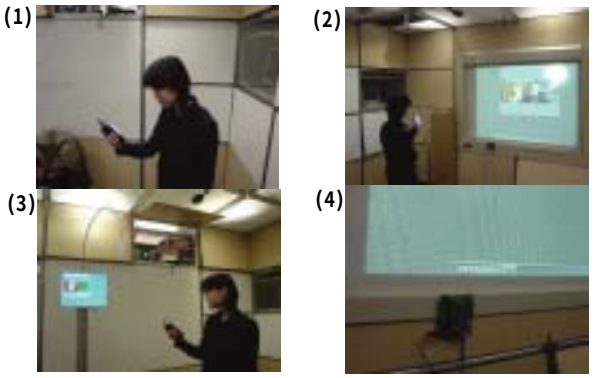

図 8　SSLab 内での TV 電話の様子

図 8 に実際の TV 電話システムを利用している様子を示す．図 8-(1)はリージョン 1 で iPAQ を用いて会話をしている様子，図 8-(2)はリージョン 2 でディスプレイサービスとスピーカーサービスを利用した様子，図 8-(3)はリージョン 1 で SmartFuniture を利用した様子である．また，図 8-(4)はリージョン 2 で利用者が利用するディスプレイがどのディスプレイなのかを確認するために，点灯される LED である．ASAMA のサービス管理 GUI でサービスプロバイダのエントリを選択すると，該当するサービスプロバイダに設置された LED が光り，利用者が実際のサービスプロバイダがどれなのか確認できるようになっている．

（3）利用者の位置に応じた動的ポスターシステム

公共の場(以後，パブリックスペースと呼ぶ)におけるサービスプロバイダのモデルとして，利用者の位置に応じた動的ポスターシステムが考えられる．利用者がサービスプロバイダに近づいてくると，自動的に広告のためのポスターが表示され，利用者がサービスプロバイダを利用し，離れた場合，ポスター表示を終了する．
利用者がリージョン 1 からリージョン 2 に移ると，サービスプロバイダは自分が存在する空間に利用者が新たに入ってきたことを検知する．それによって動的にポスターを自身のディスプレイに表示させ，音楽をスピーカから流し始める．これにより，サービスプロバイダがその環境で利用可能であることを利用者にアピールできる．
利用者がサービスプロバイダを利用し始めると，表示されているポスターや音楽は利用者にとって邪魔であるため，ポスター処理を終了する．同様に，利用者がその空間から離れる場合も処理を終了する．
パブリックスペースでは，サービスプロバイダを設置するために，魅力的なモチベーションを提示し，設置を行う企業などに積極的にその利点を示す必要がある．この動的ポスターシステムは，そういった企業に対し，パブリックスペースにおける広告というモチベーションを与え，Mobiquiotus Environment の普及のための糸口になると考えている．

（4）音声と位置情報を利用したライト制御システム

Mobiquitous Environment で使用できるユーザインタフェースは GUI だけとは限らない．音声認識が可能なサービスプロバイダが存在すれば，音声によるサービスプロバイダの制御も可能となる．図 9 に音声と位置情報を利用したライト制御システムの概略を示す．

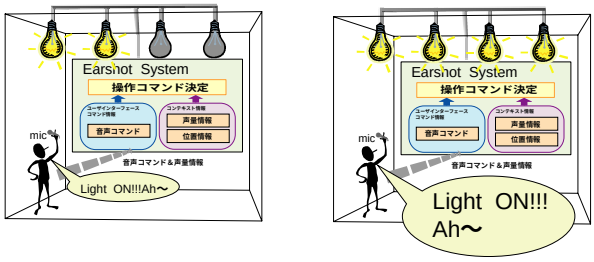

図 9 音声と位置情報を利用したライト制御システム概要

このライト制御システムでは発言内容と音声の大きさ，高低を利用し，位置情報を付加して SSLab 内にある電源の on/off および明るさの調節が可能な三菱製調光ライトの制御を行う．まず，音声の大きさと位置情報によって，空間内にある複数のライトのどれを制御するのかを指定する．ライトに対して発せられた制御用コマンドの音声が小さければ利用者から近いところに設置されたライトのみが制御され，音声が大きければ利用者から離れたライトまでも制御できる．またライトの明るさを制御する場合に音声の高低を利用する．制御コマンドの後に利用者が低い声から高い声を発した場合，ライトの明るさも音声の変化に応じて明るくなる．逆に利用者が高い声から低い声を発した場合，暗くなる．図 10 に SSLab 内でのライト制御の様子を示す．

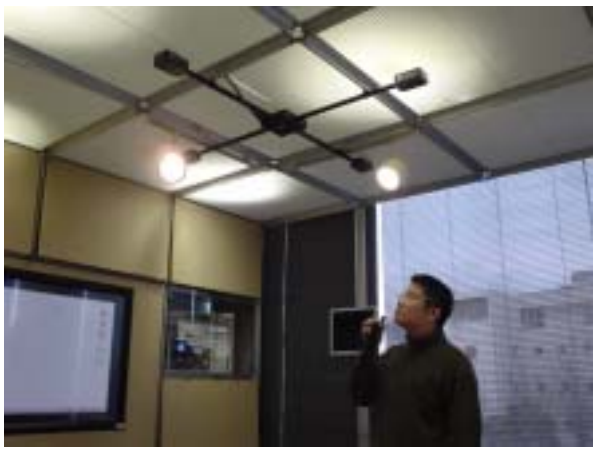

図 10　SSLab 内でのライト制御の様子

このような形でライトを制御するために，音声の大きさや高低の変化を利用者にわかりやすく示すよう，図 11 に示すような GUI を表示し，フィードバックを行う．

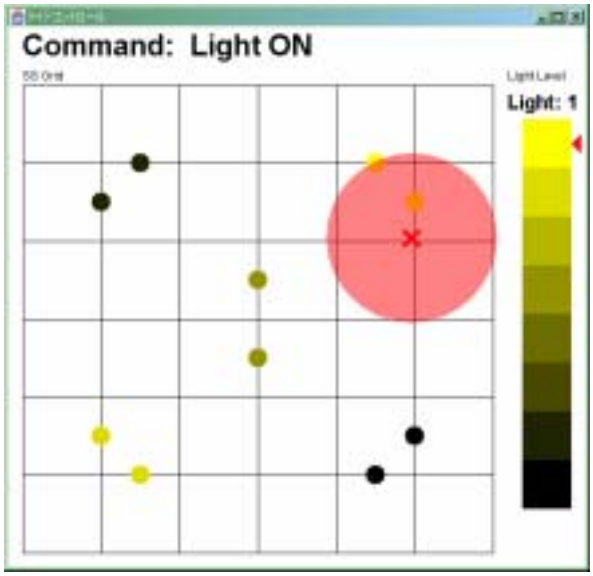

図 11　ライト制御システム用 GUI

本プロジェクトの成果を用いることにより，このような利用者の位置に応じたサービス利用アプリケーションは容易に作成できる．アプリケーションプログラマは利用者の移動やどこにどのライトが設置されているのかを考慮する必要がなくなり，より自由度の高い動的適応アプリケーションを作成可能となる．

４．まとめ

本プロジェクトでは，新たなユビキタスコンピューティング環境として，Mobiquitous Environment を提案し，サービスローミング技術によって利用者は移動しながら継続的なサービス利用を実現できるようになった．

これまでのユビキタスコンピューティング環境では，利用者がある場所に入って，その場にどのようなサービスがあるかわからない状態から，即興的にその場にあるサービスを用いて利用者の様々な要求を実行することが目的とされてきた．しかし，ホットスポットの普及や，研究機関におけるユビキタスコンピューティング環境同士の相互接続など近年の潮流を考えると，複数のユビキタス空間を利用者が渡り歩くことが考えられる．

本プロジェクトでは，利用者が複数のユビキタスコンピューティング環境を移動しながら，継続的な作業を実現できるよう，Mobiquitous Environment を提案した．

Mobiquitous Environment では特殊な機器を用意することなく，すでに流通している機器を使用した．また，特定のプログラミング言語に依存しないように設計されているため，どのハードウェアにも移植することが比較的容易である．機器上で動作するアプリケーションを作成する際に，本プロジェクトが作成したミドルウェア群を利用することにより，利用者だけでなく，サービスやアプリケーションも移動し，3 つの移動に適応しながら利用者に対して継続的な作業空間を提供できるようになった．

また，本プロジェクトは Mobiquitous Environment の中心となる技術として，利用者やサービスなどの移動によって生じる変化に動的に適応し，適切なサービスを利用できるサービスローミング技術を実現した．

これまでのモバイルコンピューティング環境では，利用者が携帯する計算機のみが移動することを前提としていた．OSI7 層モデルでいう，物理層からトランスポート層までの移動透過性を保証する研究は多い．しかし，移動によって変化するサービスの利用可能状況はアプリケーション層で解決する必要があると考えられる．また，Mobiquitous Environment での移動はサービスも移動することもあり，通信の両端それぞれが変化することが考えられる．

本プロジェクトでは，サービス利用基盤ソフトウェアによって，利用者が移動した場合でも，それによって生じるサービスの利用可能状況の変化を検知し，利用者に対して適切なサービスプロバイダを選択できるよう，支援する．また，サービス再構築ソフトウェアによって，アプリケーションの一部をサービスへ容易に移動することが可能となり，簡単に利用するサービスを切り替えるよう，支援する．支援する．継続的サービス利用支援ミドルウェアによって，利用者のいる環境で使用できる既存技術を利用しながら，サービスプロバイダとクライアントとの通信の切り替えを支援する．これらのミドルウェアを利用することで，サービスローミングが行えるアプリケーションを容易に作成できるようになった．

今後の課題として，本プロジェクトで実装した継続的サービス利用支援ミドルウェアのプロトタイプをもとにより機能が充実した完成版を作成することが第 1 にあげられる．第 2 に，ユビキタスコンピューティング環境で重要となってくるセキュリティについて考えていく．

今後の展望として，本プロジェクトの成果を学会や論文を通じて世界に発表していき，他の研究者からの意見などを取り入れつつ，より完成度の高いものを目指していく．

５．参考文献

［1］　"適応的ディレクトリサービスのおけるステート管理方法" 永田 智大, 西尾 信彦, 徳田 英幸 情報処理学会コンピュータシステムシンポジウム 1999 年 11 月 pp.57-64

［2］　"ウェアラブルネットワークにおける移動適応型分散通信機構" 権藤俊一, 永田智大, 岩本健嗣, 西尾信彦, 徳田英幸 情報処理学会マルチメディア, 分散, 協調とモバイル(DICOMO 2001)シンポジウム Vol.2001(7) 2001 年 6 月 pp.211-216

［3］　"ウェアラブルネットワーク環境に適したアプリケーションフレームワーク" 村瀬正名, 永田智大, 西尾信彦, 徳田英幸 情報処理学会マルチメディア通信と分散処理研究会(DICOMO) Vol.2001(7) 2001 年 6 月 pp.199-204

［4］　"ASAMA: 適応的なサービス利用管理機構" 永田 智大, 西尾 信彦, 徳田 英幸 情報処理学会論文誌 Vol.42(6) 2001 年 6 月 pp.1557-1569

［5］　"ユビキタスコンピューティング環境のための作業継続可能なユーザインタフェース切替機構" 守分滋, 村瀬正名, 永田智大, 西尾信彦, 徳田英幸 情報処理学会全国大会 2002 年 3 月

［6］　"ユビキタス空間を融合するネットワーク技術への課題" 徳田英幸, 中澤仁, 岩井将行, 由良淳一, 村瀬正名 情報処理学会情報処理学会会誌 Vol.43(6) 2002 年 6 月 pp.623-630

［7］　"サービス利用状況の変化に対する適応支援構" 永田 智大, 西尾 信彦, 徳田 英幸 情報処理学会システムソフトウェアとオペレーティングシステム研究会 Vol.2002(60) 2002 年 6 月 pp.1-8

［8］　"サービス利用状況の変化に対する適応支援機構" 永田 智大, 西尾 信彦, 徳田 英幸 情報処理学会論文誌 Vol.44(3) 2003 年 3 月

［9］ "Wearable Network: Architecture and an Implementation" Nobuhiko Nishio, Takeshi Iwamoto, Tomohiro Nagata, Hideyuki Tokuda IEEE International Workshop on Networked Appliances (IWNA'00) 2000 年

［10］ "ASAMA: Adaptive Service Availability Management" Tomohiro Nagata, Nobuhiko Nishio, Hideyuki Tokuda Proceedings of IEEE International Workshop on Networked Appliances (IWNA'01) 2001 年 3 月 p14--19

［11］ "Implementation and Evaluation of Wapplet Framework" Masana Murase, Takeshi Iwamoto, Tomohiro Nagata, Nobuhiko Nishio and Hideyuki Tokuda IEEE International Workshop on Networked Appliances (IWNA'02) 2002 年 1 月 p275--284