Flying Emulator：携帯端末用ソフトウェアの開発試験環境

佐 藤 一 郎[†,††]

無線 LAN などを利用する携帯端末は接続先のネットワーク固有の情報やサービスを利用することがあり，そのアプリケーションの動作はネットワークに依存することになる．本論文ではこうしたネットワーク依存性を持つ携帯端末用アプリケーションの開発試験手法を提案していく．これは携帯端末の物理的な移動をモバイルエージェントの論理的な移動により再現するものである．具体的には，携帯端末の計算環境を再現するエミュレータをモバイルエージェントとして実現し，アプリケーションとともに携帯端末の接続先となるネットワークに移動させるものである．この結果，アプリケーションは携帯端末が移動先ネットワークに接続されているのと同様に，そのネットワーク上の情報やサービスを利用することができる．また，移動経路やネットワーク接続可否はエミュレータまたは外部システムから柔軟に制御できることから，複数のネットワークに移動・接続するアプリケーションまたはコンテンツの開発試験に有用である．本論文はこの手法の基本概念と Java 言語を利用したプロトタイプシステムを概説するとともに，その応用事例を紹介する．

Flying Emulator: Building and Testing Software for Mobile Computing

Ichiro Satoh[†,††]

This paper presents a mobile-agent framework for building and testing mobile computing applications. When a portable computing device is moved into and attached to a new network, the proper functioning of an application running on the device often depends on the resources and services provided locally in the current network. To solve this problem, this framework provides an application-level emulator of portable computing devices. Since the emulator is constructed as a mobile agent, it can carry target applications across networks on behalf of a device, and it allows the applications to connect to local servers in its current network in the same way as if they were moved with and executed on the device itself. This paper also demonstrates the utility of this framework by describing a Java-based prototype implemenation of the framework and the development of typical location-dependent applications and contents in mobile computing settings.

1. はじめに

携帯端末から Ethernet や IEEE802.11b，Bluetooth などの有線または無線 LAN が利用可能になっている．この結果，移動先からインターネットなどの広域ネットワークへの接続が可能となっているが，今後は移動先にあるローカルネットワークやその場所に依存した情報・サービスの利用も期待されている．たとえば，無線 LAN の到達範囲は数 m〜数百 m となるが，その範囲の地域情報を発信するデータベースや，ノート PC や PDA の近くにあるプリンタを利用することも可能となる．また，第 4 世代携帯電話では無線 LAN 技術との融合が想定されるなど，今後は小範囲の無線 LAN を前提にした携帯端末向けアプリケーションソフトウェア（以降はアプリケーションと呼ぶ）の需要は高まるといえる．

しかし，既存の携帯電話のように場所に依存しない広域移動体通信を利用する場合と比較して，ローカルネットワーク固有の情報やサービスを利用するアプリケーションの開発は困難となる．これは携帯端末の移動とともに接続先のローカルネットワークがかわり，結果としてアプリケーションが利用できる情報やサービスも変化するからである．したがって，アプリケーションの動作試験では携帯端末が移動・接続されうる場所ごとにその動作を確認しなければならない．また，携帯端末上のアプリケーションは UPnP [10] や Jini [2] などのサービス発見機構を利用することがあるが，これらの機構はマルチキャスト通信を前提としており，

† 国立情報学研究所
National Institute of Informatics
†† 科学技術振興事業団
Japan Science and Technology Corporation

利用範囲はマルチキャストパケットの到達可能なネットワークドメイン内に限定される．したがって，その動作試験では携帯端末をローカルネットワークに接続する必要があり，さらに各ローカルネットワークにより発見されるサービスも相違することがある．本論文では，これらの問題を解決する方法として，モバイルエージェント技術[12]を利用する手法（Flying Emulatorと呼ぶ）を提案する．これは携帯端末に対応したエミュレータをモバイルエージェントとして実現したものであり，開発対象のアプリケーションを携帯端末の移動・接続先となるローカルネットワーク（正しくはそのネットワーク上にある所定のコンピュータ）に運び，そのネットワーク内でアプリケーションを実行させる．よって，そのアプリケーションはローカルネットワーク固有の情報やサービスを直接利用できることになる．すなわち，ここで提案する方法は携帯端末の物理的な移動によってアプリケーションが受ける動作環境の変化を，エミュレータおよびアプリケーションというソフトウェアの論理的な移動により再現するものである．また，このとき開発者は携帯端末とともに移動する必要はない．

ここで本論文の構成を述べる．続く2章では提案するFlying Emulator手法の基本概念を述べる．3章ではそのプロトタイプシステムの設計と実装を，4章では応用例を説明する．そして5章では関連研究をまとめ，6章では結論と課題を述べる．

2. 方　　法

本論文では携帯端末上で動作するソフトウェアの開発・動作試験を行うための枠組みを提案する．ここで対象とするソフトウェアは携帯端末内で動作するアプリケーションソフトウェアである☆．そして，携帯端末のEthernetやIEEE802.11b，Bluetoothなどの有線または無線LANにより，移動先のローカルネットワーク上のデータベースやプリンタなどの情報サービスや計算資源を利用することができる．

2.1 携帯端末用アプリケーション開発・試験環境

一般にノートPCやPDA，携帯電話などの携帯端末が持つプロセッサやメモリは十分ではなく，さらにディスプレやキーボードなどの入出力装置にも制約がある．このため，携帯端末そのものを使ってアプリケーションの開発・試験を行うことは難しく，携帯端末と同様な計算環境をソフトウェア的に再現するエミュレータをワークステーション上に用意し，そのエミュレータ内でアプリケーションの開発・試験を行うことが多い．ただし，既存の携帯端末用エミュレータの多くはスタンドアローン実行されるアプリケーションを対象としている．このため，ネットワーク上の各種情報やサービスを利用するアプリケーションを扱えるエミュレータは少ないが，次の3つの方法が提案されている．

(1) エミュレータ内で携帯端末の移動・接続先となるローカルネットワーク上の情報・サービスを擬似的に再現する方法．
(2) 広域ネットワークを介して，エミュレータから移動・接続先となるローカルネットワーク上の情報・サービスを遠隔利用する方法．
(3) エミュレータを実行するワークステーションを移動し，携帯端末の移動先にあるローカルネットワークに接続する．そして，そのネットワーク上の情報・サービスを直接的に利用する方法．

ここで(3)は再現精度が高いが，開発者の移動を要求するなど負担も大きい．(1)と(2)は開発者は移動する必要がないが，(1)ではエミュレータ内に各ローカルネットワーク上の多様な情報・サービスを再現することが困難となる．(2)はFirewallなどのセキュリティ機構により，ローカルネットワーク上の各種サーバを利用できるとは限らない．また，マルチキャスト通信などの一部の通信はローカルネットワーク内だけに利用が限定されることが多く，これらの通信を利用するアプリケーションには対応できない．そこで，本論文では(2)と(3)の中間に位置する新たな方法を提案する．なお，この方法は上記の3つを補完するものであり，排他的に利用することを意図していない．

2.2 Flying Emulator フレームワーク

ここで提案する手法（Flying Emulator）は，**図1**のような携帯端末の物理的な移動とそれにともなう接続先ローカルネットワークの変化を，**図2**のようにモバイルエージェントの論理的な移動により再現するものである．まず，この手法では携帯端末のアプリケーション実行環境を再現するエミュレータをモバイルエージェントとして実現する．実際の携帯端末は移動とともに接続先のローカルネットワークが変わり，それにともない携帯端末上のアプリケーションが利用できる各種情報・サービスも変化することになる．一方，このエミュレータは携帯端末の接続先となるローカルネットワーク上の所定のコンピュータにアプリケーションとともにソフトウェアレベルで移動する（**図3**）．この結果，エミュレータ上のアプリケーションはそのロー

☆ ソフトウェアが利用する通信プロトコルもアプリケーションレイヤ以上とする．

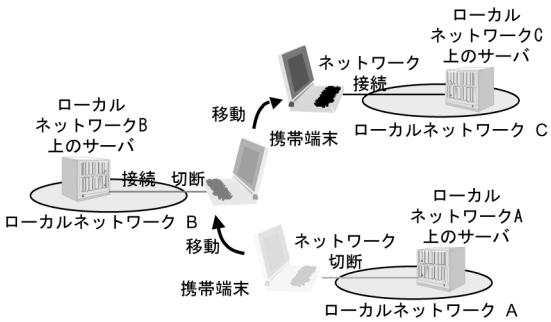

図 1 携帯端末の物理的移動
Fig. 1 Physical mobility of a portable device.

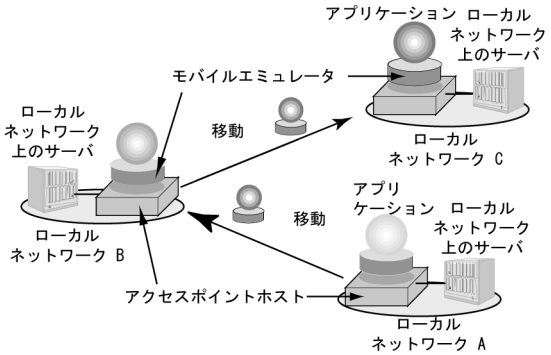

図 2 エミュレータの論理的移動
Fig. 2 Emulation of physical mobility by logical mobility.

カルネットワークの内側で動作することとなり，マルチキャスト通信を含むネットワーク上の各種サービスを利用することができる．また，モバイルエージェントはその特徴として，コンピュータ間を移動する際にプログラムが持つ変数内容などの実行状態を保持することができる．この結果，エミュレータとともに別のネットワークに移動したアプリケーションは移動前の実行状態から処理を再開することができる．これは携帯端末が別のネットワークに移動・接続したとき，その端末上のアプリケーションが移動前の計算途中結果から処理を再開することにほかならない．

このモバイルエージェントによる携帯端末エミュレータ（モバイルエミュレータと呼ぶ）は次の条件を満足する必要がある．

- モバイルエミュレータはアプリケーションに対して携帯端末と同様な計算環境を提供する．そして，エミュレータ上で動作試験が行われたアプリケーションは変更や再コンパイルすることなくそのまま携帯端末上でも動作する．
- 携帯端末の移動・接続順序に従って，モバイルエミュレータは接続先となるローカルネットワーク

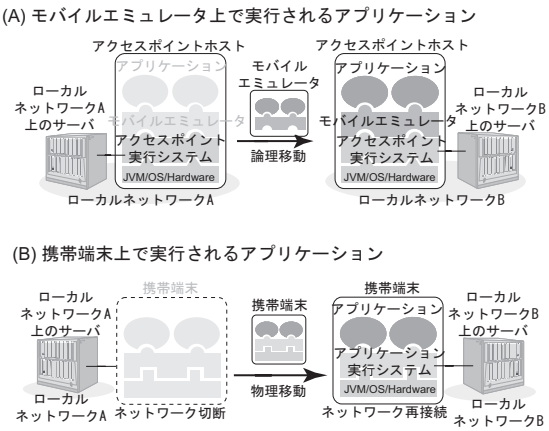

図 3 モバイルエミュレータによる携帯端末の移動の模倣
Fig. 3 Emulation of the movement of a mobile computer by migrating a Mobile Agent-based Emulator.

（にある所定のコンピュータ）にアプリケーションを移送する．そして携帯端末がローカルネットワークに接続した場合と同様に，アプリケーションをローカルネットワーク内の各種通信サービスを利用させながら動作試験を行う．

- 移動経路やアプリケーションに対するネットワーク接続可否はモバイルエミュレータ自身により制御可能とし，さらに所定の管理用コンピュータから集中的に監視制御できるようにする．

ところで携帯端末の移動・接続先となるローカルネットワークにはモバイルエージェントの実行環境となるコンピュータ（以降ではアクセスポイントホストと呼び，APH と略記）が必要となる☆．つまり，携帯端末があるネットワークから別のネットワークへの移動・接続する過程は，その端末に対応したエミュレータが，移動元ネットワーク上の APH から，移動先のネットワーク上の APH にアプリケーションを運び，そのアプリケーションをそれぞれの APH で実行・ネットワーク接続することにより再現される．なお，各ローカルネットワーク上の APH は広域ネットワークにより接続されているとする．

本論文では議論を明確化するために，対象となるアプリケーションを Java 言語によるプログラムに限定し，携帯端末と APH 間の動作環境の相違の多くは Java 言語の仮想機械により解決する．これは本論文の目的が，個々の携帯端末の内部動作を再現することではなく，エミュレータの移動性により接続先ネット

☆ APH に必要な計算負荷は大きくない．したがって，専用コンピュータは不要であり，既存のサーバ上に必要なソフトウェアを稼働させるだけでよい．

ワークの変化を再現する手法を提案することにあるからである．さらに，このエミュレータにおける携帯端末自体の再現方法はモバイルエージェントとして実現される以外は一般のエミュレータと同じである．なお，Java 言語はそのプラットフォーム非依存性や実行安全性から，携帯電話や PDA におけるアプリケーション記述方式として広く利用されていることから，Java 言語プログラムに限定するこによってこの手法の一般性が失われることはない．

3. Flying Emulator フレームワーク

Flying Emulator 手法は次の 4 つのソフトウェアから構成される．

- **モバイルエミュレータ（Mobile Agent-based Emulator）** アプリケーションに対して携帯端末内部の動作環境を再現するモバイルエージェントであり，アプリーションとともに移動先ネットワーク上の APH 間を移動する．
- **アクセスポイント実行システム（Access Point Runtime System）** 各 APH においてモバイルエミュレータの実行・移動環境を提供し，到着したエミュレータ内のアプリケーションに対してその APH が接続されているネットワーク内の通信サービスを利用することを許す．
- **遠隔制御サーバ（Remote Control Server）** モバイルエミュレータの移動とアプリケーションのネットワーク接続を集中制御するためのコンピュータであり，開発者がインタラクティブにエミュレータを操作・監視することを可能にする．
- **アプリケーション実行システム（Application Runtime System）** 携帯端末上で動作し，携帯端末およびアプリケーションの実行を監視するミドルウェアである．

図 4 はモバイルエミュレータによる開発試験時のシステム構成を示したものである．ところで現在の実装では上記はすべて Java 言語により実現されており，実行プラットフォームを選ばない☆．

3.1 モバイルエミュレータ

エミュレータは携帯端末の種別ごとに用意され，その構成は**図 5** のようになる．主要な機能について説明する．

3.1.1 携帯端末移動のエミュレーション

モバイルエミュレータは APH 上のアクセスポイント実行システム間を移動するが，その移動先 APH は遠隔制御サーバによる明示的な制御以外に，それ自身で移動経路を定めることもできる．これは携帯端末の移動順序に応じて APH 名（ネットワークアドレス）を並べたリストをエミュレータに設定することにより，エミュレータがそのリストに従って移動するものである．このリストにはさらに各 APH 上の滞在時間やネットワーク接続・切断設定，実行するアプリケーション名なども定義することができる．

ところで，携帯端末の移動ではアプリケーションはネットワーク接続または切断されたり，また，バッテリ消費を防ぐため携帯端末とともに休止状態になったりすることがある．そこで，アプリケーションの実行モードとして次の 3 つを導入する．

- **接続実行モード**（networked running）携帯端末が稼働中かつネットワークに接続されている状態に対応する．エミュレータではアプリケーションは携帯端末の接続先となるローカルネットワーク上の APH で実行され，さらにその APH を介してローカルネットワークに接続することが許され

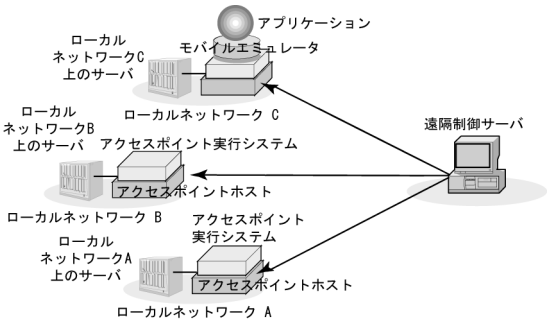

図 4 Flying Emulator フレームワークの構成
Fig. 4 Flying Emulator architecture.

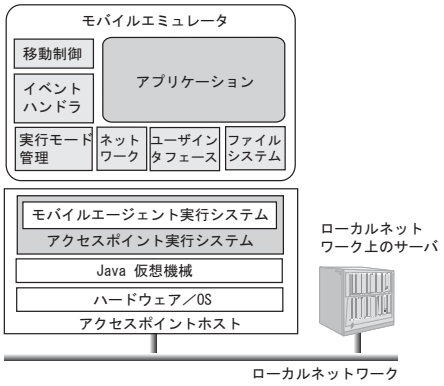

図 5 モバイルエミュレータの構成
Fig. 5 Mobile Agent-based Emulator.

☆ ただし，携帯端末上で動作するアプリケーション実行システムは端末動作変化を監視するために，Java 言語のネイティブメソッド（JNI）を利用することがある．

ている状態である．

- **単独実行モード**（isolated running）アプリケーションは実行中であるが，携帯端末が無線 LAN 範囲外に移動するなどして，ネットワーク接続が断たれている状態である．エミュレータ内のアプリケーションは APH 上で実行されているが，ネットワーク接続が制限されている状態として再現される．
- **休止モード**（suspended）携帯端末とともにアプリケーションが休止している状態であり，休止後は直前の状態から再開する．これはエミュレータを構成するモバイルエージェントをアプリケーションとともにデータ化（整列化または直列化）することにより対応する．

これらの実行モードはモバイルエミュレータ，または携帯端末上のアプリケーション実行システムにより管理される．そして，そのモード変更の前後には付録に示したコールバックメソッドを呼び出すことにより，アプリケーションに変更を通知する．

ここで実際の携帯端末との相違を述べておく．多くの携帯端末のように同時に接続するネットワークはたかだか 1 つとする．ただし，携帯端末ではハンドオーバ技術により，明示的なネットワーク切断を経ずに接続先のネットワークが切り替えられることがある．一方，モバイルエミュレータは他のネットワークに移動する前にアプリケーションを休止モードとする必要があり，アプリケーションを接続実行モードのまま移動させることはできない．ただし，この論文で扱うアプリケーションはネットワーク依存性が高いもの，つまり携帯端末の移動先ローカルネットワークのサーバなどに接続するクライアントであり，接続先ネットワークが替わるときにはサーバへの通信切断などの明示的な処理が必要である．また，バッテリ消費防止や振動の影響を減らすために携帯端末を休止させたうえで移動させることも多い．以上からこれらの相違によって一般性を失うことはない．

3.2 アプリケーション実行環境のエミュレーション

モバイルエミュレータはアプリケーションに対して携帯端末内部の実行環境を再現する．なお，再現方法自体はスタンドアローン型エミュレータと同様になることから，移動性に起因する特徴的な部分のみを説明する．

3.2.1 携帯端末の計算環境

エミュレータを実行する APH と携帯端末ではハードウェアおよび OS が異なることが多い．この場合，エミュレータは携帯端末側プロセッサの命令を解釈・実行するインタプリタを用意し，そのインタプリタ上でアプリケーションを実行することとなる．ただし，本論文ではアプリケーションはすべて Java 言語プログラムとなることから，同言語の仮想機械がそのインタプリタに相当する．また，エミュレータ内のアプリケーションの実行速度は APH 上でエミュレータを介さずに実行した場合とほぼ同じである．

3.2.2 ユーザインタフェース

携帯端末が提供する入出力装置は，画面サイズや解像度，キーボードやポインティング装置などに制限が多い．このため，携帯端末向けのアプリケーションの開発ではこれらの制限を考慮する必要がある．そこで，典型的な携帯端末，たとえば小型のノート PC，タブレット PC，PDA の出力装置それぞれに対応する基本 Java 言語クラスを提供し，個々の端末の出力装置はそれらのサブクラスとして容易に実現できるようにしている．なお，これらのクラスは Java 言語のウィンドウ用クラス（`java.awt.Window`）を基に画面サイズなどの制限を加えたものとして実現されている．一方，入力装置に対してはキーボードなどの各装置に対応したクラスを提供しており，それらを使い分けることにより対応する．なお，ユーザインタフェースをともなう動作試験の自動化のために，キー入力およびポインティング操作を擬似的に行う機構を Java 言語のユーザ入力再現用クラス（`java.awt.event.Robot`）を利用して提供している．また，後述するように遠隔制御サーバにユーザインタフェースを表示し，そのサーバから操作することもできる．

3.2.3 ファイルシステム

携帯端末のファイルシステムを再現するために，エミュレータはその内部にファイルシステムに相当するデータベースを提供している．これは各ファイルをファイル名（ディレクトリ名を含む）とファイルコンテンツの組により保持するものである．したがって，アプリケーションが利用するファイルはエミュレータ内で管理・格納され，エミュレータの移動前後もそのまま保存されることになる．また，ファイル生成，書き出し，消去などの基本操作は Java 言語のファイル操作用クラスと互換性を持つ API により行うことができる．

3.2.4 ネットワーク

APH 上のアプリケーションは APH が提供するネットワーク用 API，たとえばソケット通信や遠隔オブジェクトメソッド呼び出し（RMI）を利用することができ，エミュレータのネットワークアドレスは APH のアドレスを継承する．ただし，前述のようにエミュ

レータとともにコンピュータ間を移動するとき，アプリケーションは明示的に接続・切断処理を行う必要がある．ネットワーク接続・切断時にはアプリケーションが持つ所定のコールバックメソッドを呼び出すことができるので，アプリケーションはそのメソッド内に必要な接続・切断処理を記述することになる．なお，一般の携帯端末の移動と同様に移動透過性は保証しない．つまり，エミュレータ移動後に移動元 APH に到着したパケットを移動先に転送することはない．また，DHCP などによる動的アドレス割当てを利用する携帯端末と同様に，エミュレータが他の APH に移動する際にはそのネットワークアドレスも変わることになる．

3.3 アクセスポイント実行システム

APH は携帯端末が移動・接続するローカルネットワーク上のコンピュータであり，次の 2 つから構成される．

3.3.1 モバイルエージェントシステム

本論文で提案した枠組み自体は特定のモバイルエージェントシステムに依存するものではないが，今回の実装では Java 言語によるモバイルエージェントシステムである MobileSpaces[13] を利用している☆．この MobileSpaces はプロトコルのトンネリング機構により HTTP および SMTP などの一般的なプロトコルを介してエージェントを移送することができる．したがって，ローカルネットワークに Firewall が設けられていても柔軟に対応することができる．また，MobileSpaces システムはエージェント移動に際してエージェントの認証を行うことが可能であり，これにより不正なモバイルエミュレータの到着を防ぐことができる．

ところで，モバイルエージェントの移動における実行状態のデータ化・転送では，ヒープ領域を対象にする弱移動性と，スタック領域やプログラムカウンタも含める強移動性がある[5),12)]．MobileSpaces は他の多くの Java 言語モバイルエージェントシステムと同様に Java 言語の直列化機構を利用しているために弱移動性に基づいている．そこでエミュレータはアプリケーションのデータ化直前と，到着後の逆変換直後にアプリケーションに用意されたコールバックメソッドを呼び出すことにより，アプリケーション自身による明示的な状態保持および実行再開を可能にしている．なお，これらの制約の一部は強移動性に対応したモバイルエージェントシステムを用いることにより解決することができるが，強移動性に基づく既存システムは計算性能的な制約があり，さらにネットワークを含む計算資源は移動透過とならないことから，根本的な解決にはならない．

3.3.2 アプリケーションの実行補助

エミュレータ内のアプリケーションは APH を介してネットワーク上のサーバや通信サービスを利用することができる．このほか，APH 上の実行システムは HTTP サーバとしての機能を持ち，HTTP リクエストとして遠隔制御サーバからコマンドを受け取り，それをエミュレータに転送する機構を持つ．なお，モバイルエージェントシステム自体の拡張を避けるために，この機構もエージェントとして実装され，モバイルエミュレータにはエージェント間通信でコマンドが転送される．

多くのアプリケーションはグラフィカルユーザインタフェース（GUI）を持つことが多く，これらを介したアプリケーションの動作試験が必須となる．そこで，現在の実装では GUI を開発者の手元のコンピュータで表示・操作する手段として，IBM の Remote Abstract Window Toolkit（RAWT）[6] または X-Windows を利用している．これにより APH 上で実行されるアプリケーションの GUI を遠隔制御サーバに転送し，そのサーバ上の画面に表示することができるようになる．逆にそのサーバからのキーボードやポインティング装置などによる入力操作を APH 上でのアプリケーションに転送することもできる．したがって，アプリケーションがエミュレータとともに遠隔 APH で実行される場合でも，開発者は遠隔制御サーバ上でそのアプリケーションの GUI を表示・操作することが可能である☆☆．

3.4 遠隔制御サーバ

これは複数の APH を集中管理するとともに，モバイルエミュレータを制御するコンピュータである．各 APH はエミュレータが到着した直後と移動する直前にこのサーバに通知することにより，サーバは各エミュレータの現在位置を把握する．また，このサーバは各エミュレータに対して**図 6** のような制御用 GUI を提供することから，アプリケーション開発者はエミュレータに対して，たとえば他の APH への移動，アプリケーションの起動や実行モードの変更をインタラクティブに操作することができる．なお，エミュレータの制御はエミュレータが存在する APH 上にあるコマ

☆ MobileSpaces システムはエージェントの階層化やシステム自身の拡張性が特徴となるが，ここではこれらの機能を利用していない．

☆☆ 音声入出力には対応していない．また，遠隔アクセスポイントホスト上で実行されるアプリケーション内部情報を監視・変更する際には遠隔デバッグシステムなどを組み合わせる必要がある．

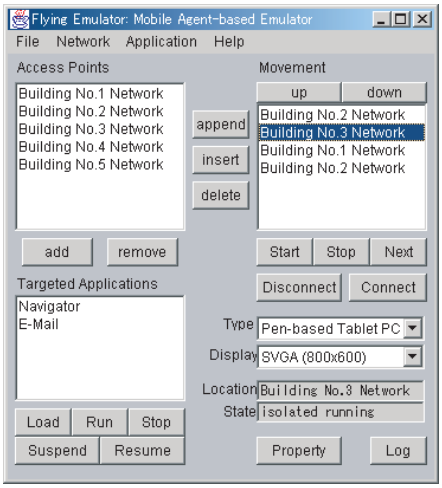

図 6 モバイルエミュレータのユーザインタフェース
Fig. 6 User interface of a Mobile Agent-based Emulator.

ンド受信用 HTTP サーバに対してリクエストを送ることにより実現している．

3.5 携帯端末用アプリケーション実行システム

この実行システムは携帯端末上で動作する Java 言語によるミドルウェアであり，Java 言語の JDK 1.1 以上または Personal Java の上で動作する．これは携帯端末の実行状態やネットワーク接続を監視するとともに，アプリケーションの実行モードを管理する．そして，携帯端末の状態が変化したときには実行モードの変更として，モバイルエミュレータと同様にアプリケーションのコールバックメソッドを呼び出す．したがって，この実行システムとモバイルエミュレータはアプリケーションに対して同一の実行環境を提供することになり，エミュレータ上で動作確認をしたアプリケーションはプログラム改変や再コンパイルをすることなく，携帯端末上でも同様に動作することができる．

4. 応 用 事 例

Flying Emulator 手法を利用した携帯端末向けアプリケーションの開発試験事例を示す．

4.1 ネットワーク依存ディレクトリサービス

ここでは Jini[2] を利用したプリンタやデータベースの発見システムの動作試験を考える．このシステムではビル内の各フロアにサブネットワークがあり，それぞれのサブネットワークには Jini のサーバと，1 台以上のプリンタやデータベースサーバが接続されている．ユーザはノート PC を持ってフロア間を移動し，Ethernet によりサブネットワークに接続する．そして Jini の Lookup サービスを利用してそのフロア上のプリンタやデータベースを発見・利用する．なお，Jini はマルチキャストにより Lookup サーバのアドレスをクライアントであるノート PC に通知することから，マルチキャスト通信の到着可能ネットワークドメイン（たとえば，各サブネットワーク）の外から Lookup サービスを利用できるとは限らない．したがって，これまでの動作試験では実際にノート PC を移動させてサブネットワークに接続する必要があった．一方，Flying Emulator 手法を利用することにより，アプリケーションはノート PC に対応したモバイルエミュレータとともに各サブネットワーク上の APH に移動することから，サブネットワークの内側で動作し，Jini の Lookup サーバの検出も可能となる．また，その際にノート PC を移動させる必要はない．

4.2 ユーザナビゲーションシステム

無線 LAN を利用したユーザナビゲーションシステムを考える．これは既存システム[1),3)] と同様に，無線 LAN の到達範囲に入った携帯端末に周辺地域情報を配信するシステムであり，ここではそのアプリケーションおよびそのコンテンツ開発試験に Flying Emulator 手法を利用した．なお，このシステムはビルやそのフロアなどを単位にした複数のサブネットワークに IEEE802.11b 基地局と DHCP や HTTP サーバを配置するものであり，同時に UDP マルチキャストにより HTTP サーバのアドレスをサブネットワーク内に定期発信している．ユーザは無線 LAN が利用可能なタブレット PC を持ちながらビル内などに移動すると，そのタブレット PC が HTTP サーバを発見して，そのビルの案内図などを自動的にダウンロード・表示するものである．

そこでタブレット PC に対応したモバイルエミュレータを導入し，タブレット PC 上の案内図表示アプリケーションの通信処理および案内図表示について動作試験を行った．まず各サブネットワーク上のコンピュータを APH として利用するため，Java 言語の実行環境に加えて，MobileSpaces システムの実行システムおよびアクセスポイント実行システム（両システムを合わせて約 500 KB の Java 言語クラス群）を導入する．モバイルエミュレータは案内図表示アプリケーションを各サブネットワーク内の APH に運び，そのアプリケーションはマルチキャスト UDP を受信することにより HTTP サーバを特定し，そこから案内図をダウンロードすることができた．なお，このモバイルエミュレータ（サイズ 140 KB）が 100 Mbps Ethernet で接続された 2 つの APH（Pentium-III 600 MHz）間の移動に要する時間は 120 ms となる．**図 7** は遠隔制御サーバの画面である．ここで (A) は APH 上のモバ

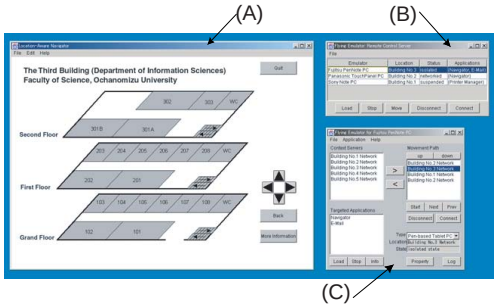

図 7 モバイルエミュレータによるユーザナビゲーションシステムの動作試験における遠隔制御サーバの画面
Fig. 7 The screenshot of the remote control server, when the user navigation system runs on the Mobile Agent-based Emulator.

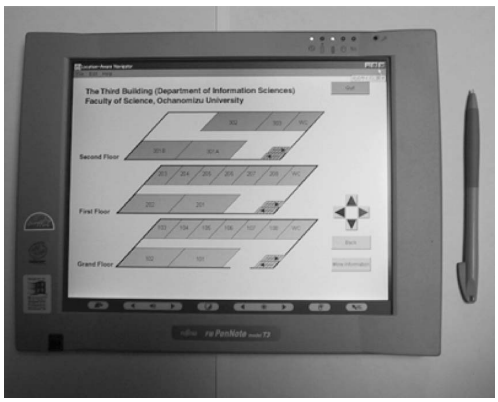

図 8 タブレット PC で動作するユーザナビゲーションシステム
Fig. 8 The user navigation system running on a Pen-based Tablet PC.

イルエミュレータ内で動作する案内図表示アプリケーションのウィンドウであるが，Remote AWT により遠隔制御サーバ上で表示・操作が可能となっている．(B) はモバイルエージェントシステムの制御ウィンドウ，(C) は遠隔制御サーバのエミュレータ制御用ウィンドウである．そして，**図 8** はタブレット PC 上で動作する案内図表示アプリケーションであり，図 7 (A) と同様の表示・操作ができることが分かる．なお，エミュレータ上で動作試験をされたアプリケーションは書き換えや再コンパイルをする必要はない．

特に美術館などの案内では，コンテンツは現在地点によって決まるだけでなく，各ユーザが移動してきた経路に応じてコンテンツを替えることがあり，その動作試験は煩雑となるが，この Flying Emulator 手法では各ユーザの移動経路に対応した APH リストをモバイルエミュレータに設定することにより，さまざまな移動経路で動作確認を自動化することもできることから，コンテンツ開発にも有用となる．

5. 関 連 研 究

前述のように，携帯端末向けアプリケーションの開発・試験では携帯端末のハードウェア上の制約が障害となる．このため，アプリケーションの開発試験はワークステーション上で動作するエミュレータ上で行うことが多い．しかし，これらのエミュレータは携帯端末内部の動作をアプリケーションに対して再現するものであり，ネットワークを介して提供される各種情報・サービスを正確に再現することは不可能に近い．さらに携帯端末はその移動にともないネットワーク切断や接続先ネットワークが変化することがある．なお，ネットワークの変更を吸収して携帯端末上で動作するアプリケーションからはその相違を隠蔽する移動透過性に関する技術がいくつか研究されているが[7),11)]，すべての切断や相違を隠蔽することは困難であり，逆にネットワークの相違を積極的に利用したアプリケーション，たとえば無線 LAN 到達範囲内の地域依存情報の提供なども増えている．

この問題の解決方法の 1 つは，アプリケーションを携帯端末，またはその端末用エミュレータを稼働する可搬型ワークステーションとともに移動・接続させながら動作試験を行う方法である．しかし，開発者への負担が大きく，開発の最終段階以外は利用すべきではない．もう 1 つの解決方法は固定型ワークステーション上のエミュレータ内でアプリケーションを動作させながら，そのエミュレータを介して端末移動先ネットワーク上の各種情報やサービスを遠隔利用する方法である．これには UC Berkeley の InfoPad プロジェクト[9)] や Lancaster 大学のネットワークエミュレータ[4)] などがある．しかし，マルチキャストを利用したサービスのようにそのローカルネットワークの内部だけで有効な通信も多い．また，Firewall などのセキュリティ機構により，ローカルネットワーク上の各種サーバを外部ネットワークから利用できるとは限らない．さらにサーバとの通信データ量が多いアプリケーションの試験では，広域ネットワークの通信トラフィックが増えるという問題もある．一方，本論文で提案した方法はエミュレータとなるソフトウェアの論理的移動を利用することにより，開発者および携帯端末の物理的移動は不要であり，さらにアプリケーションは端末移動先となるネットワーク内で動作するために上述のようなネットワーク上の制限が少ない．

なお，本論文では携帯端末エミュレータをモバイルエージェントとして実現した．これまでモバイルエー

ジェントの実行システム（プラットフォーム）や分散システムの効率化・管理手法などへの応用事例は数多くあるが[5),8),13)]，携帯端末向けアプリケーション開発に利用する研究事例は皆無である．本論文の手法は著者の前稿[14)]をもとにしているが，対象となるアプリケーションはモバイルエージェントに限られるなどの制約があった．ところで，モバイルエージェントの実際的な応用ではセキュリティ上の問題が障害となっているが，ここで提案した方法はアプリケーション開発段階でのみ利用することから，他の応用分野と比較してセキュリティ的な問題が深刻化しにくい．

6. おわりに

携帯端末向けアプリケーションの開発試験手法として，モバイルエージェントによる携帯端末エミュレータを用いた手法を提案した．これは携帯端末の移動にともなう接続先ネットワークの変更・切断を，それのエミュレータとアプリケーションを複数ネットワーク上で移動させることにより再現するものである．この手法により，アプリケーションは携帯端末内の動作環境だけでなく，携帯端末の移動経路も再現し，さらに携帯端末が移動先ネットワークに接続されているのと同様に，そのネットワーク上のサーバを利用しながら動作試験が行える．このため，接続先ネットワークに依存したアプリケーションやコンテンツの開発試験において有用となる．

最後に今後の課題について述べる．今回の実装ではJava言語によるアプリケーションを対象としたが，他の実行形式の場合でもその動作環境を再現するエミュレータをモバイルエージェントとして実現することにより対応可能である．今後は携帯端末のネイティブ実行形式など Java 言語以外のプログラムにも対応する予定である．また，本論文で提案した方法と 2 章で紹介した 3 つの試験方法は，アプリケーション開発過程に応じて使い分けるべきものである．したがって，他方式も利用可能な統合開発システムを検討している．

謝辞 有意義なコメントをいただいた査読者の方々に感謝する．

参考文献

1) Abowd, G.D., Atkeson, C., Hong, J., Long, S., Kooper, R. and Pinkerton, M.: Cyberguide: A Mobile Context-Aware Tour Guide, *ACM Wireless Networks 3*, pp.421–433 (1997).
2) Arnold, K., Wollrath, A., Scheifler, R. and Waldo, J.: *The Jini Specification*, Addison-Wesley (1999).
3) Cheverst, K., Davis, N., Mitchell, K. and Friday, A.: Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project, *Proc. ACM/IEEE Conference on Mobile Computing and Networking* (*MOBICOM'2000*), pp.20–31 (2000).
4) Davies, N., Blair, G.S., Cheverst, K. and Friday, A.: A Network Emulator to Support the Development of Adaptive Applications, *Proc. USENIX Symposium on Mobile and Location Independent Computing*, USENIX (1995).
5) Fuggetta, A., Picco, G.P. and Vigna, G.: Understanding Code Mobility, *IEEE Trans. Soft. Eng.*, Vol.24, No.5 (1998).
6) International Business Machines Corporation: Remote Abstract Window Toolkit for Java (1998). http://www.alphaworks.ibm.com/
7) Kistler, J.J. and Satyanarayanan, M.: Disconnected Operation in the Coda File System, *ACM Trans. Computer Systems*, Vol.10, No.1, pp.3–25 (1992).
8) Lange, B.D. and Oshima, M.: *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley (1998).
9) Le, M., Burghardt, F. and Rabaey, J.: Software Architecture of the Infopad System, *Workshop on Mobile and Wireless Information Systems* (1994).
10) Microsoft Corporation: Universal Plug and Play Device Architecture Version 1.0 (June, 2000). http://www.upnp.org/UpnPDevice_Architecutre_1.0.htm
11) Perkins, C.: IP Mobility Support, Internet Request For Comments RFC 2002 (1996).
12) 佐藤一郎：モバイルエージェントの動向，人工知能学会論文誌，Vol.14, No.4, pp.598–605 (1999).
13) Satoh, I.: MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, *Proc. International Conference on Distributed Computing Systems* (*ICDCS'2000*), pp.161–168, IEEE Computer Society (April 2000).
14) Satoh, I.: Flying Emulator: Rapid Building and Testing of Networked Applications for Mobile Computers, *Proc. Conference on Mobile Agents* (*MA'2001*), LNCS, Vol.2240, pp.103–118, Springer (2001).

付　録

本論文の開発対象アプリケーションは Java 言語プログラムであり，特別な制限はない．しかし，下記のインタフェースを実装クラスとなり，各メソッドに必要な処理を定義する必要がある．

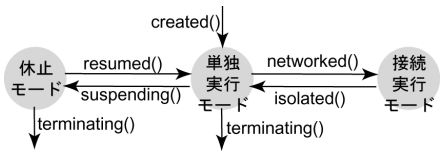

図 9 実行モード遷移とコールバックメソッド
Fig. 9 Callback method invocations in execution mode transition

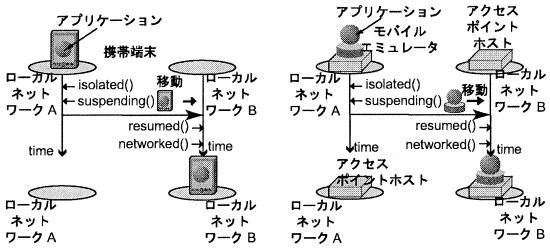

図 10 携帯端末およびモバイルエミュレータ移動
Fig. 10 The movement of portable devices and the migration of Mobile Agent-based Emulator.

```
1:  interface MobilityListener
2:    created() // invoked after creation
3:    terminating() // invoked before termination
4:    networked() // invoked after network enabled
5:    isolated() // invoked after network disconnected
6:    suspending() // invoked before suspension
7:    resumed() // invoked after resumption
8:  }
```

上記はコールバックメソッドとなり，アプリケーションの実行モードが変化した際に，そのアプリケーションの実行環境となるモバイルエミュレータまたはアプリケーション実行システムから呼び出される．その呼び出しタイミングは**図 9** のようになる．

たとえば，携帯端末またはモバイルエミュレータが移動する際には**図 10** のようにアプリケーションのメソッドが同様のタイミングで呼び出される．ここで独立実行モードから接続実行モードに変わる直後にメソッド `networked()` が呼び出され，再び独立実行モードになるとその直後にメソッド `isolated()` が呼び出される．また，休止状態モードから独立実行モードに移行する直後にメソッド `resumed()` が，休止状態モードに移行する直前にメソッド `suspending()` が呼び出される．なお，遠隔制御サーバから APH 上のエミュレータ内アプリケーションに対してメソッド呼び出しコマンドを送ることができる．これにより，たとえば電池切れなどの異常状態をコールバックメソッド呼び出しとしてアプリケーションに通知することができる．

(平成 14 年 3 月 25 日受付)
(平成 14 年 10 月 7 日採録)

佐藤 一郎（正会員）

1991 年慶應義塾大学理工学部電気工学科卒業．1993 年同大学大学院理工学研究科計算機科学専攻前期博士課程修了．1996 年同専攻後期博士課程修了，博士（工学）．1996 年お茶の水女子大学理学部情報科学科助手，1998 年同学科助教授．2001 年より国立情報学研究所助教授．このほか，1994 年〜1995 年まで Rank Xerox 訪問研究員．1999 年〜2002 年まで科学技術振興事業団さきがけ研究 21（「情報と知」領域）研究員．1996 年度情報処理学会論文賞，1999 年同学会山下奨励賞，1998 年日本ソフトウェア科学会高橋奨励賞ほか受賞．分散システム，ミドルウェア，プログラミング言語に関する研究に従事．日本ソフトウェア科学会，電子情報通信学会，人工知能学会，IEEE，ACM 各会員．