移送コストを考慮した 環境適応型移動エージェントシステムの設計と実装

森 正一† 高汐 一紀††

† 電気通信大学大学院 電気通信学研究科
†† 電気通信大学 情報工学科

携帯端末での動作を考慮した，計算環境へ適応可能な移動エージェントシステムを提案する．本システムでは，エージェントをアプリケーションロジックを実装するコア部と，環境ごとに実装する機能モジュール部に分け，システムより与えられる環境情報によりそのモジュールを適切にロードすることにより，携帯端末に限らず様々な計算環境での適応動作を実現する．また，モジュールコードの動的取得機構により，移送コストの削減を実現する．これらにより，移動エージェントの適応範囲を携帯端末へ拡大し，PC-携帯端末間での統一的な計算環境を実現する．

Design and Implementation of an Environment-Aware Mobile Agent System Aimed to Reduce Migration Cost

Masakazu MORI† Kazunori TAKASHIO††

† Graduate School of Electro-Communications, The University of Electro-Communications
†† Department of Computer Science, The University of Electro-Communications

We propose the mobile agent system that can adapt its behavior to computing environment and aiming to run at portable devices. We divide an agent into a core part and function module parts. The core part implements application logic. Each function module implements the function which is different in every environment. This module part is implemented by every computing environment. By selecting the function modules part for each environment, the agent adapts itself to computing environment. Our system reduces cost of agent migration by using “on demand module loading mechanism”. Our agent model expands the field of mobile agent to portable devices and embedded system. It also realizes unified computing environment between personal computers and portable devices.

1 はじめに

ポスト PC 時代におけるユビキタスコンピューティング環境を実現する基盤技術として，移動エージェントソフトウェアアーキテクチャが注目されている [1, 2]．ユビキタスコンピューティング環境では，ユーザは自身の計算環境を詰め込んだ計算機を持ち歩く必要はない．ユーザはタスクの実行に必要な情報のみをパーソナリティとして携帯し，「今そこにある」計算機やデバイス，サービス[1] を必要に応じてパーソナライズすることで，それらをあたかも自分のものであるかのように利用する．パーソナライズ対象が PC であれば，ディスプレイ上に操作インタフェースを含めたユーザのデスクトップ環境が再現され，ホスト透過な操作性が保証されるだろう [3]．

移動エージェントは実行状態を維持したままホスト間を移動可能な計算実体であり，移動エージェントのフレームワークを用いてアプリケーションを構築することにより，ユーザに対してユビキタスコンピューティング環境上での移動透過かつシームレスなアプリケーション環境を提供することが可能となる．アプリケーションは，ユーザの移動に付き添うように，パーソナライズされた様々なデバイス上に移動，環境に適応することでタスクを継続的に実行

[1] 正しくはデバイスの集合としての計算環境

する．表示機能を持つデバイス上では，ユーザに経過を報告し，タッチパネル等の入力インタフェースを持つデバイス上では，ユーザに指示を求めることもあるであろう．

既存の移動エージェントシステムの多くは，Java 言語を用いて記述されており，プラットフォーム独立なアプリケーションの記述を可能としている [4, 5]．しかし，その多くが Java 2 Standard Edition (J2SE) 上での実装であり，PC 以外のデバイス，すなわち PDA や組込みデバイス等の J2SE の動作が困難なデバイスが実装のターゲットとみなされることはなかった．

我々の目的は，これらリソースに制限の多いデバイス上でも動作可能な「環境適応型移動エージェント」のフレームワークを設計し，様々なデバイス上にその実行環境を実装，ユビキタスコンピューティング環境実現の核となるソフトウェアインフラストラクチャを構築することにある．本稿では特に，手軽に携帯可能な計算端末である Palm OS 搭載 PDA[6] に注目し，PC と PDA の双方で動作可能な移動エージェントシステムを実装，統一的な計算環境の実現を目指す．

以下，本稿の構成は次のようになる．第 2 節では，現在の問題点を整理する．第 3 節で提案するシステムの設計について述べ，第 4 節でその実装，第 5 節で評価について述べる．

2 論点と方針

本節では，PC，PDA 双方で動作する移動エージェントシステムを設計する上で問題となる論点を整理し，本稿での方針について述べる．PDA で動作するエージェントシステムの問題点は，以下の 3 つが挙げられる．

- リソース差の吸収
- 動的クラスロード，シリアライズ機構の代替
- 移送コストの削減

以下では，それぞれについて詳細を述べる．

2.1 リソース差の吸収

PDA 上には現在，PersonalJava や Java 2 Micro Edition (J2ME) 等の Java 実行環境が実装されている．今回作成する移動エージェントシステムでは，PC 上の J2SE 環境に加え，PDA 上のこれらの環境で動作させる必要がある．PDA 上の Java 実行環境は，動作させる端末に必要なリソースを少なく抑えるため，J2SE に比べ使用できる API が限られる．

既存のシステムは，J2SE 上での動作のみを考慮したものが大多数である．これらのシステムでは，単一の Java 実行環境をターゲットとしている為に，Java 実行環境が変化し，提供される API が変化する状況に対処することは難しい．そこで，PC と PDA 双方の Java 実行環境に共通の基本 API を用いてシステムやエージェントを作成することも考えられる．しかし，PDA 上の実行環境として J2ME を考えた場合，基本 API はごく限られたものとなる．そのためアプリケーションとして非力なものとなり，システムとして現実的ではない．

これらの解決手段として，システムでラッパを用意して，ラッパでリソース差を吸収する方法が考えられるが，PC，PDA 間のリソースの違いが顕著であり，それらの違いを吸収する汎用的なラッパの作成は困難であると考えられる．

また，階層的なエージェント構造を用い，複数の機能エージェントを組み合わせることによって，アプリケーションを構成する方法も考えられる [7]．エージェントの構成を動的に変更することにより，アプリケーションを環境に適応させる．しかし，複数のエージェントを現在の PDA 上で同時に動作させることは処理速度の問題，後述するクラス名のバッティングの問題があり，実用的ではない．

そこで，本稿では，エージェントを各プラットフォーム独立の部分と，プラットフォーム依存の部分に分ける．プラットフォーム依存の部分は各 Java 実行環境上に個別の実装を行い，プラットフォーム独立の部分はすべての環境に共通の実装を行う．エージェントの実行環境変化に合わせ，プラットフォーム依存の部分を変更する方法を採用する．この方法により，マルチプラットフォームでの動作を実現する．

2.2 動的クラスロード，シリアライズ機構の代替

多くの Java エージェントシステムでは，エージェントの移送は，以下の手順で実現される．

1. Java シリアライズ機構を用い，エージェントインスタンスをバイト列に変換
2. エージェントをコードと共に移動先へ送信
3. 移動先でコードを動的にロード
4. 移動先でバイト列をデシリアライズしエージェントインスタンスを生成

しかし，PDA 上の Java 実行環境では 1，3 の機構の機構が必ずしも提供されているとは限らない．その為に，PDA 上で動作する移動エージェントを実現するには新たなフレームワークを提供する必要がある．

そこで，本稿では，シリアライズ機構に代えて，エージェント毎に状態をバイト列化するメソッドを実装し，状態の移送を実現する．クラスロードについては，エージェントのコードをデフォルトクラスローダでロード可能な形で PDA へ移送する．

2.3 移送コストの削減

PDA は PC に比べ極端にハードウェアリソースが少ない．その為に，エージェントの移動時にもリソースを圧迫しないよう，エージェント移送コストを軽減する必要がある．具体的には，可能な限りエージェントシステム及び，エージェント本体のファイルサイズをコンパクトにする必要がある．

たとえ，PC，PDA 両環境で動作するコードを完成させたとしても，その大きさが巨大であれば，PDA での動作は非実用的なものとなる．リソース資源にゆとりのある，PC 上でもエージェント移送コストの削減は重要な課題である．

第 2.1 節の通り，本稿ではエージェントのプラットフォームに依存する部分を各環境で個別に実装する方法を採用する．この方法では，エージェントが動作する際には，現在の環境に合ったプラットフォーム依存のコードのみが必要で，他の環境のコードは必要ない．つまり，エージェントに他のプラットフォーム用のコードを常に携帯させ，移送先で環境に合わせコード選択を行う手法では必要以上にコストが掛かる事になる．

そこで，エージェントの移送時に，プラットフォーム依存の部分を動的に取得，破棄出来るフレームワークを採用する．このフレームワークにより，移送時のコストの軽減を実現し，エージェント移送時の PDA 上のリソースの圧迫を防ぐ．

3 *m*-P@gent

今回我々は，前述で挙げた問題点を解決するべく，新しいエージェントシステム”*m*-P@gent” の設計と実装を行った．本システムの特徴は，Multi Platform に対応し動作可能なエージェントの作成が可能となる点である．

以下では，システム全体像を述べた後，特徴であるエージェントの構造と，エージェントの移送について詳しく述べる (Palm OS 搭載デバイス特有の問題点は第 4 節の実装で解説する)．

3.1 設計方針

本システムでは，エージェントのマルチプラットフォームに適応した動作と，移送コストの軽減を実現する為に以下の手法を採用した．

- エージェント機能のモジュール化

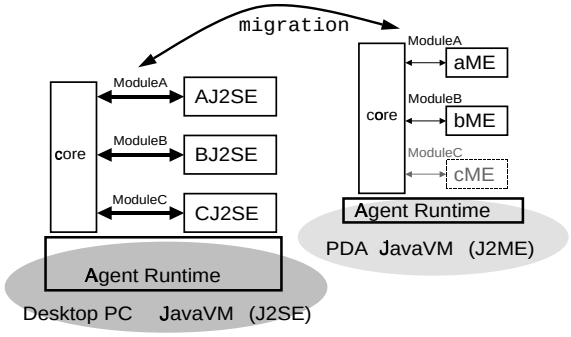

図 1: エージェント動作イメージ

- モジュールの動的取得

まず，エージェントの機能をアプリケーションロジックを実装するコアと，コアに記述された機能をプラットフォーム毎に個別に実装するアドオンモジュールに分割する．そして，プラットフォームに合わせ選択されたアドオンモジュール群とコアで，エージェントとして動作させる．

後者は，エージェントを移送する際に，アドオンモジュールコードを動的に移送する手法である．

3.2 エージェントの構造

本システムではエージェントを各々のプラットフォームに適応させる為に，エージェントシステムが計算環境の情報 (環境情報) を提供する．ここで提供する環境情報の粒度は，Java 実行環境の違い (J2SE, J2ME) 程度のものとする．

そして，エージェントを，各機能を環境毎に個別に実装する部分 “**アドオンモジュール**” と，アプリケーションロジックの部分 “**コア**” に分割し，エージェントに環境情報とモジュールの対応表 “**プロファイル**” を保持させる．プロファイルには，各計算環境共通のモジュールタイプとそれに対応する環境毎のモジュールを

[モジュールタイプ名]=[アドオンモジュール名]

と記述する．

エージェント記述者はコア，アドオンモジュール，プロファイルを用意し，エージェントを記述する．エージェントの起動時には適切なプロファイルが読み込まれ，更に，プロファイルに記述されているアドオンモジュールのコードも読み込まれる．そしてシステムから与えられる環境情報に合わせアドオンモジュールをロードし，エージェントを起動する．エージェント動作時のアドオンモジュールの指定はモジュールタイプを用いて行う．

図 1 は，プロファイルとして以下のリスト (リスト 1，2) を用意し，コアとして `core` を用いたエージェント例である．

```
ModuleA=AJ2SE
ModuleB=BJ2SE
ModuleC=CJ2SE
```

リスト 1: J2SE 用プロファイル

```
ModuleA=aME
ModuleB=bME
ModuleC=
```

リスト 2: J2ME 用プロファイル

core では，環境ごとに違うモジュール A，a の 2 つを ModuleA というモジュールタイプ名を用い指定し，アプリケーションロジックを記述する．ModuleC のように，環境により実装の存在しないモジュールの作成も可能である．

各機能をモジュールタイプ名で指定し，アクセスすることで，実装の違うモジュールに同じ手法でアクセス可能となる．それにより，モジュールの実体を環境ごとに変化させることで，環境に適応した動作が可能となる．また，再利用可能なモジュールを作成することにより，コーディングの手間を軽減する事も可能となる．

3.3 エージェントの移送

エージェント移送時に，常に全環境のモジュールを移送していては移送コストが増大してしまう．この解決手段としては，

- モジュールコードをシステムで用意し，全ホストにあらかじめ配置
- エージェント毎のモジュールコードを集中管理しエージェントの移動をスター型にする

が考えられる．

前者は全てのモジュールコードをあらかじめエージェントシステム側で用意し，エージェント記述者は，これのみを用いてエージェントを作成するという方法である．
しかし，独自のモジュールを作成することが出来ない為，拡張性に乏しく，前述の汎用的なラッパ作成の場合と同様，汎用的なモジュールを作成することは困難である．また，モジュール群はエージェントサーバに常に保持しておかなければならず，エージェントサーバ自体が巨大化してしまうし，モジュールの更新も困難である．よってこの方法は，PDA 上でエージェントを動作させることを考える，本稿のシステムには不向きと考えられる．

後者は，エージェントを生成する度に，モジュールコード群を集中管理サーバへ登録する．エージェント移動時には一度管理サーバへ戻り，移動先の環境に対応したモジュールを選択，選択されたモジュールを移動先のサーバへ移動する方法である．

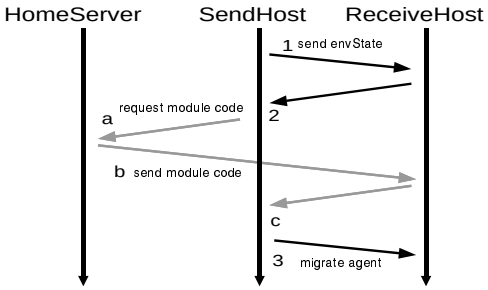

図 2: エージェント移送プロトコル

この方法では管理サーバにエージェント移送時の負荷の集中が起こる．これにより，管理サーバが正常に動作を行っている間はエージェントの動作は正常に行われるが，管理サーバがダウンした場合には全てのエージェントの動作が停止してしまう．また，エージェントの移送の際には，常に管理サーバという迂回路を通らなければならないという事になる．

そこで，本稿では第 3 の手段として，モジュールコードの管理をエージェントの生成サーバ (HomeServer) 毎に行う事にする．

エージェントの移送方法には，

- 移動先サーバが潤沢な環境の場合には全てのモジュールコードを移送
- 移動先サーバが貧弱な環境の場合には移動先に対応したモジュールコードのみを移送

の 2 パターンを用意する．

エージェントは，移送コストを考慮する必要の無い環境では前者の方法で移動する．ネットワークや，移動先のリソース等の問題により移送コストを軽減させたい場合には，後者の方法で移動し，その際に破棄したモジュールコードの再取得は HomeServer より行う．

具体的なエージェント移送時のプロトコルは以下のようになる (図 2)．

1. 移動先サーバの環境確認
   エージェントが対応可能な環境のリストを移動先サーバへ送信し，移動可能か確認

2. モジュールコード保持確認
   エージェントが移動先環境に必要なモジュールコードを現在所持しているか確認し，所持していない場合は以下を行う

   (a) HomeServer へ移動先環境に不足しているモジュールコードの送信を要求

(b) HomeServer が移動先サーバへモジュールコードを送信

(c) モジュールコード送信時にエラーが発生していないことを確認

3. エージェント移動
コアが移動先サーバへ移動し，移動完了

このプロトコルにより，移動先サーバが貧弱な環境の場合，モジュールコードによる移動先環境の圧迫を避ける事が可能となる．

また，各エージェントのコードを各 HomeServer で分散管理する為，負荷の集中を軽減する事が出来る．更に，HomeServer から移動先へ移動元を介さずに，コードを直接送信する事が可能となる．

4 システムの実装

本節では，*m*-P@gent システム実装の詳細と，簡単なアプリケーション例を示す．システムの実装に関しては，エージェントの動作部と，PC と PDA のシステムのそれぞれ特徴的な部分のみ述べる．

4.1 実装環境

本システムの実装は PC は，Windows2000 上の Java 2 SDK Standard Edition (Version1.3)，PDA は Palm 上の JAVA 2 Micro Edition Connected, Limited Device Configuration / KVM Palm (Version1.0.2 FCS) 上で行った．

また，Palm 上の追加ライブラリとして，kAWT[8] を利用することにより，イベントベースでのエージェント記述を可能とした．

4.2 エージェント動作部

エージェント本体であるコアとモジュールは，それぞれ，`IAgent`，`IModule` というインタフェースを実装した Java クラスである．また，同一モジュールタイプ間のアクセスには Java のインタフェースフレームワークを使用する．エージェントが複数になると，エージェント毎に多数のモジュールが存在するため，モジュール間やエージェント間のクラス名のバッティングが発生する可能性が高くなる．この問題を防止する為に，クラスローダをエージェント毎に用意した．

また，エージェント作成に際し，J2ME でコードを動作させる為に，コアコードと PDA 上のモジュールコードはあらかじめ，事前検証を行っておく必要がある[2]．

モジュールはエージェント起動時に外部プロファイルを読み込む事により指定する．外部プロファイルは，

[2]詳しくは CLDC ドキュメントを参照 [9]

[Agent 名]`@`[環境]`.env`

という名称のファイルとして用意する．プロファイル内では，以下の書式でモジュールクラスの指定を行う．

[モジュールタイプ]=[モジュールクラス名]

エージェント起動者は，必要な環境の数だけこのプロファイルを用意する．また同タイプのモジュールへ，インタフェースを利用しアクセスする場合には，モジュールタイプをインタフェース名とする仕様とした．実際には，システムでエージェント毎に用意される “`getModule(`モジュールタイプ`)`” メソッドでモジュールインスタンスを取得し，インタフェースへキャストし動作させる．

また，PDA (Palm) 上でオブジェクトのシリアライズが不可能な為に起こる状態移送の問題は，エージェント毎に，状態をバイト列として入出力するメソッドを，エージェント記述者が独自に実装する事で解決する．

4.3 @Desk

@Desk は，PC 環境上に実装された *m*-P@gent エージェントシステムである．PC 上のエージェントサーバ間の通信は Java の標準的なネットワーク API である RMI を使用し，エージェントの移送に関しては，Java のシリアライズ機構を利用することとした．

また，モジュールの取得に関しては，エージェントサーバに，エージェントの受信に際して，現在の環境以外のモジュールコードを受け入れるか受け入れないかを決定するため，「受け入れる，受け入れない」という選択フラグを持たせた．これにより，エージェントサーバ管理者による，エージェント受信動作の変更を実現する．

エージェントコードや，状態は移送コスト削減の為，移送時に Java Archive (JAR) ファイルにまとめて移送する．HomeServer が保持するモジュールコードも，エージェント生成時に JAR ファイルを生成し，管理する．

また，移送途中でエラーが発生した場合はエージェント移送はキャンセルされる．

4.4 @Palm

@Palm は，Palm OS 搭載 PDA (以下 Palm) 上に実装された *m*-P@gent エージェントシステムである．Palm OS 上で Java アプリケーションを実行する際には，JAR ファイルを PRC (Palm resource) 形式に変換する必要がある．また，Palm OS 独自のデータベース方式である PDB (Palm database) 形式でデータの保存を行う．このため Palm 上でシ

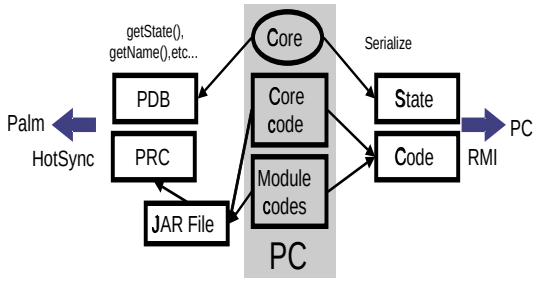

図 3: PC-Palm 移動と PC-PC 移動

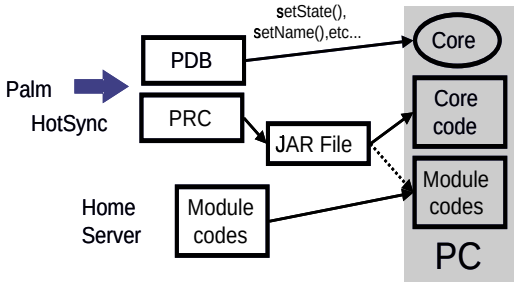

図 4: Palm-PC エージェント移動

ステムを動作させるには，これらに対応する必要がある．

Palm 上でオブジェクトのシリアライズを行えない為に起こる問題は，先に述べたエージェントへの状態の入出力を行うためのメソッドを実装することによって，オブジェクトの状態をバイト列として書き出し，更に，そのデータをエージェントの ID 等と共に PDB 形式で保存することにより解決した．

動的なクラスロードが不可能な問題は，エージェントのコードを，エージェントシステムのコード，`J2MEPalm` 環境のモジュールのコードと共に一つの JAR ファイルとしてまとめ，生成された JAR ファイルを PRC 形式に変換し，クラスロードをエージェント起動時にデフォルトクラスローダで行えるようにすることで解決した．

生成された PRC，PDB ファイルはそれぞれ，コード，状態に対応する (図 3)．エージェントは，生成された PRC，PDB ファイルを HotSync で Palm 上にインストールすることにより，再開される．HotSync は専用アプリケーションを個別に起動する必要はなく，PC のエージェントシステム上で，エージェントの移動要求から HotSync 終了までの一連の動作を自動的に行う事が出来る．

Palm から PC へ移動する際には，ユーザの要求により，移動先サーバが Palm 上の PRC，PDB ファイルを，HotSync を用いる事により取得する．そして，PRC ファイルを JAR ファイルに変換し，エージェントのコードを取得し，PDB ファイルを解析して，エージェントの ID や状態を取得する．

```
TextFrame=J2SEFrame
FileIO=J2SEIO
Clip=J2SEClip
```

リスト 3: LWEditor@J2SE.env

```
TextFrame=PalmFrame
FileIO=PalmIO
Clip=
```

リスト 4: LWEditor@J2ME.env

そして，PC 環境用のモジュールコードは，PRC ファイルには保存されていないので HomeServer へ転送を要求して取得する (図 4) ようにした．その後，エージェントを PC 上で再構成し，起動する．

また，共通のコードを動作させる際に問題となる API の不足部分 (`java.io.Serializable` 等) はエージェントシステムでダミークラスを作成し，ロードさせることで対処した．

4.5 エージェント例

ここで例に挙げる LWEditor は J2SE 環境，J2ME (kAWT 利用) 環境に適応するエディタである．プロファイルをリスト 3，4 に示す．

ここでは，テキストを表示する `TextFrame`，ファイル入出力を行う `FileIO`，クリップボード機能を提供する `Clip` モジュールを用いている．

例えば，モジュールタイプ名 `FileIO` に関連づけられている `J2SEIO`，`PalmIO` について，`J2SEIO` ではファイルダイアログを出し，ファイルへの入出力を行うが，`PalmIO` では，データベースへの書き込みを実装する．

コアである `LWAgent` は，`IAgent`，`java.awt.ActionListener` を実装し，ファイル入出力に関するイベントへの振る舞いや，エージェント到着時等のモジュールの動作，そして，状態をバイト列に変換するメソッド等を記述する．コードの抜粋をリスト 5 に示す．

5 評価

エージェントシステムと先の例のエディタエージェントの各環境毎のファイルサイズを表 1 に示す．評価に用いた環境は以下の通りである．

```
public void
    actionPerformed(ActionEvent e) {
  String c = e.getActionCommand();
  if (c.equals("Exit")) {exit();}
  else if (c.equals("Save")) {
    fileIO.saveText(textFrame.getText());}
     :
}
public void arrive() {
     :
  textFrame =
      (TextFrame)getModule("TextFrame");
  textFrame.setText(text);
     :
}
public byte[] getState() {
  return text.getBytes();
}
```

リスト 5: LWEditor.java

PDA	:	Palm Vx (Palm OS 3.5)
PC	:	Sony VAIO PCV-LX80/PBK Intel PentiumIII 866MHz Memory:240MB
HomeServer	:	Toshiba DynaBook SS3480 Intel PentiumIII 600MHz Memory:196MB
Network	:	LAN(100Base-TX)

表 1 の，エージェントシステムのファイルサイズは JAR ファイル形式，エージェントのファイルサイズは各環境に対応するモジュールの合計である．また，状態は PC 上では JAR ファイル内のシリアライズされたエージェントオブジェクト，プロファイル，及びエージェント名という 3 つのデータの合計であり，Palm 上では PDB ファイルの大きさである (エージェント自身のステート保存バイト列は 10Bytes).

表 1 より，Palm 上では PC 上の半分以下の大きさで動作させることが可能となる事がわかる．ここで，PC 間におけるエージェントの移送時に，より容量の小さい Palm 用の状態を利用しない理由は，Palm 用の状態生成に PC 用の状態生成以上にオーバーヘッドがかかるためである．

同環境のもとで，前述のエディタエージェントを PDA から PC へ移送する場合にモジュールコードの移送 (送信要求・コードパケット生成・送信の合計) に要した時間は 521ms であった．

	PC	PDA
エージェントシステム	61kB	7.4kB
サンプルエージェント (core は 3kB)	21.3kB	6kB
ステート	640B	320B

表 1: エージェントシステム，サンプルエージェントサイズ

また，Palm 上での動作は，同様にエディタエージェントで，起動時に PDB から ID 等のエージェントの状態を得るのに 1486ms かかる．この時間は，データベースを読む為に必要な時間である．エージェントの保持する状態の量にもよるが，他のエージェントでも同程度の時間が最低，必要となる．

同エージェントの Palm 上でのモジュールインスタンス作成時間を計測した．`LWEditor`，`PalmFrame`，`FileIO` 各インスタンスの生成にそれぞれ，843，11960，410ms の時間を要する．`PalmFrame` インスタンス生成に時間がかかるのは，kAWT のロードに時間がかかる為である．

6 関連研究

携帯端末をターゲットとしたエージェントシステムについては既に開発が進められているものも幾つか存在する．東芝の picoPlangent[10] は，独自のスクリプト言語を用い，知的移動エージェントを実現している．機能のコンポーネント化等ベースとなるアイデアは本システムと同じであるが，ターゲットが知的エージェントであるところが，本システムとは異なる．また，本システムが，Java 言語内で完結しているのに対し，独自のエージェント記述用言語の習得が必要となる．

NEC の携帯電話向け Java 小型エージェントシステム [11] は，J2SE，J2ME でのエージェントの実装を，各々の環境で異なるアプリケーション使用することによって実現している．この点で本システムとは大きく異なる．エージェントの移送は，エージェント管理サーバを用い，MIDP 環境へ転送するという方式を採用している．管理サーバの概念により，エージェントの非同期転送を実現している．

リソースに制限のある組み込み機器をターゲットとしたエージェントシステムとして TINIAgent[12] がある．TINIAgent は，TINI デバイス上で動作するエージェントシステムを設計している．しかし，TINI 上への実装は，現在完全には行われていない．TINI 上の次期 Java 実行環境では，シリアライズ，動的クラスロードがサポートされるため，その環境

上での動作を予定している．

また，エージェントの環境対応の点で，MobileSpace は，環境対応をエージェントを組み合わせることにより実現している．これを当システムへ適用することは，PDA の処理能力の問題から，難しいと考えられる．

7　今後の課題

Palm 上への複数のエージェントの移動については，現在はサポートしていない．これは，動的クラスロードが不可能であることから，エージェントを複数載せた場合のクラスのバッティングの問題が解決出来ない為である．現時点では，Palm 一台にエージェントシステム一つ，エージェント一つの動作に限定している．この点については改良が必要である．

また，サポートしている環境情報は，現時点では Java 実行環境の違い等，粒度の大きい環境情報だけだが，EIS/MA[13] 等を組み込む事により，より細かな環境への適応や，Palm 以外の他の携帯端末への対応等を行っていきたい．

エージェント作成を容易にする為の，モジュール作成のフレームワーク整備も重要な課題である．

その他，環境情報のネーミング，セキュリティ等問題が残っている．

8　まとめ

本稿では移送コストを考慮した環境適応可能なエージェントシステムを提案，設計し PC，PDA という異なる環境の Java 上に実装し評価した．これにより，エージェントのモジュール化による環境適応は，移動エージェントの適応範囲を携帯端末まで拡大し，移送コスト削減に有効であることを示した．

これは，携帯端末を利用した Follow-Me アプリケーションフレームワークである *f*-Desktop[2] 等様々なところで利用可能であろう．

個人が様々な計算環境を利用することが可能である今，それらの間をシームレスに移動できるアプリケーションの実現は早急に解決されるべき課題である．本研究では，その解決策の 1 つを示した．

参考文献

[1] 高汐一紀，中澤仁，大越匡，徳田英幸: ユーザモビリティの実現を目的とした移動エージェントフレームワーク, 日本ソフトウェア科学会第 16 回大会論文集, 1999

[2] Kazunori TAKASHIO, Gakuya SOEDA, Hideyuki TOKUDA: A Mobile Agent Framework for Follow-Me Applications in Ubiquitous Computing Environment, Proceedings of IEEE International Workshop on Smart Appliances and Wearable Computing (IWSAWC2001), 2001

[3] Kenneth R. Wood, Tristan Richardson, Frazer Bennett, Andy Harter, and Andy Hopper: Global Teleporting with Java: Toward Ubiquitous Personalized Computing, Computer, Vol.30, No.2, 1997

[4] Aglets web site, the aglets portal, `http://www.aglets.org/`

[5] Voyager web site, `http://www.objectspace.com/products/voyager/`

[6] Palm, Inc. web site, `http://www.palm.com/`

[7] Ichiro Satoh: MobileSpaces: A Framework for Building Adaptive Distributed Applications using a Hierarchical Mobile Agent System, Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS2000), pp.161-168, IEEE Press, 2000

[8] kAWT web site, `http://www.kawt.de/`

[9] Sun Microsystems, Inc.: Connected, Limited Device Configuration Specification Version 1.0 Java 2 Platform Micro Edition

[10] 長健太，林久志，加瀬直樹，大須賀昭彦: 携帯機器向け知的移動エージェント picoPlangent, 情報処理学会第 62 回全国大会，2001

[11] 前田直人，中島震，森本美子: 携帯電話向け Java 言語環境を用いた小型移動エージェントシステム，第 4 回プログラミングおよび応用のシステムに関するワークショップ SPA2001, 2001

[12] 重安哲也，浦上美佐子，三浦賢司，松野浩嗣: 組み込み機器用モバイルエージェントシステムの設計と実装，第 12 回 コンピュータセキュリティ研究会　第 101 回マルチメディア通信と分散処理研究会，2001

[13] Hendro Subagyo，高汐一紀: 動的な環境に適応する移動エージェントシステムの実装と評価, 第 4 回プログラミングおよび応用のシステムに関するワークショップ SPA2001，2001