ソフトウェア工学 135−8
（２００１．１１．２２）

ユビキタス計算環境向け軽量移動エージェントシステム

前田 直人　　中島 震
NEC ネットワーキング研究所

あらまし　移動エージェント技術は無線ネットワークと携帯情報端末から構成されるユビキタス計算環境を対象としたシステム開発のための良い枠組みを与える．本稿では，ユビキタス計算環境向け軽量移動エージェントシステムを提案する．本システムは，エージェント管理サーバを中心とする Star 型のエージェント移動を前提とし，エージェント消滅に対応するためのバックアップ機能と，限られた計算機資源の中でも動作するように軽量化されたエージェント実行環境を特徴とする．

キーワード:　移動エージェント，分散システム，MIDP，ユビキタス

Lightweight Mobile Agent System for Ubiquitous Computational Environment

Naoto MAEDA　　Shin NAKAJIMA
Networking Research Laboratories
NEC Corporation

Abstract　Mobile agent technology can provide a good framework to develop distributed systems for ubiquitous environment composed of wireless networks and mobile information devices. In this paper, a lightweight mobile agent system for ubiquitous environment is proposed. The system adopts the star agent mobility pattern where an agent management server is centered. The system provides backup facilities to restore vanishing agents and a lightweight agent runtime environment for the poor computational resources.

Keywords:　Mobile Agents, Distributed Systems, MIDP, Ubiquitous

1 はじめに

移動エージェントは，“ある場所” から “別の場所” へ移動し，移動先で処理を行うことにより，与えられた目標を達成する計算実体である．移動エージェント技術を用いることにより，ネットワーク切断状態での処理の継続が可能になり，ネットワークの通信トラヒック削減効果が期待できる．そのため，エージェント技術は，無線ネットワークと携帯情報端末から構成されるユビキタス計算環境向けのシステム開発に有効であると考えられている [1].

一方，これまで提案されてきた移動エージェントシステムは有線ネットワークとデスクトップ計算機からなる安定した計算環境を想定していた [2, 10]．どのように移動エージェントを実現するか，その方法は様々であり，対象とする計算環境やアプリケーション領域に応じて適切な機能/構成を選択しなければならない．

我々は，まず，デスクトップ計算環境とユビキタス計算環境の差に注目し，携帯電話向けの標準 Java 言語環境 [6] を用いてユビキタス計算環境を対象とした移動エージェントシステム [3] の試作を行った．本システムは，耐故障性と軽量化に主眼を置き設計されている．本稿では軽量エージェントシステムの主要な機能/構成を説明し，次に，軽量化のための実行コード抽出手法について述べる．

2 想定する移動エージェント

エージェントを記述する言語としてオブジェクト指向言語を想定し，エージェント計算の継続手法として *Weak Migration* を想定する．

このような移動エージェントシステムでは，エージェント移動は次の三つの手順で実現される．第一に，エージェントが移動要求を発行すると，実行基盤はエージェントの状態を表す情報を取得する．エージェント状態を表す情報はエージェントを構成するオブジェクトのインスタンス変数の値より構成される．第二に，移動元の実行基盤は，エージェントの実行コードと先に取得したエージェント状態とシステムの管理情報等を移動先の実行基盤に転送する．第三に，移動先の実行基盤は送られてきたエージェントの情報からエージェントを構成するオブジェクトを復元し，エージェントの特定の Call-back メソッドを呼び出すことにより，エージェントの処理を再開させる．以上の処理により，移動元でのエージェントの計算を移動先で継続させる．

3 軽量移動エージェントシステム

3.1 移動の制御

状況に応じた自律的な移動は，移動エージェントの一つの特徴として考えられている [10]．一方，システム開発の観点からは，システムの複雑さを軽減し，耐故障性や信頼性を高めるために，エージェント移動を制御することが重要である [8]．我々は，エージェント管理サーバを中心とした Star 型の移動を前提とし，エージェント移動を制御することにより，ユビキタス計算環境がもたらす障害に対応する．

エージェント管理サーバはデスクトップ計算機上で動作し，以下の二つの役割を果たす:

1. エージェントの待避場所
   ユビキタス計算環境では，エージェントが移動要求を発行した時点で，移動先へエージェントを転送できるとは限らない．サーバは移動に失敗したエージェントを受け入れ，移動可能になった時に移動先へエージェントを転送する機能を備えた，エージェントの待避場所を提供する．

2. 障害対応
   ネットワークで転送されている途中，または移動先で計算途中に障害が発生し，エージェントが消滅する可能性がある．エージェント障害に対応するために，サーバはエージェントを管理し，エージェントが消滅した際にそのバックアップを起動する機能を提供する．

携帯情報端末から携帯情報端末へのエージェント移動の様子を図 1 に示す．エージェントの

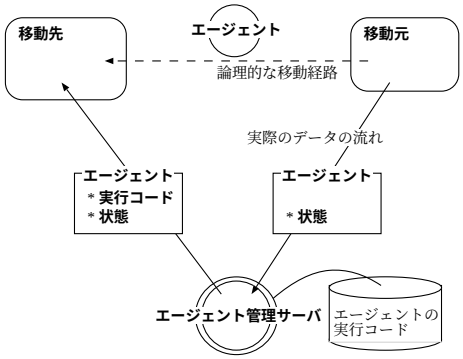

図 1: Star 型のエージェント移動

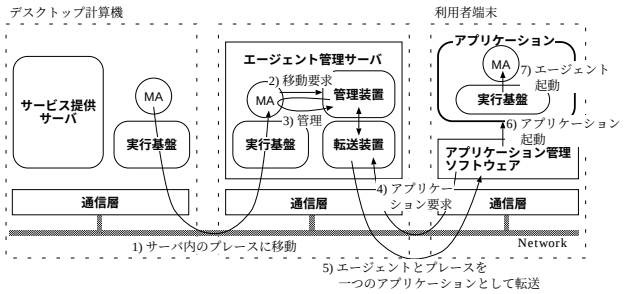

図 2: エージェント転送方式

情報は，まず移動元からサーバへ転送され，その後，サーバから移動先へ転送される．通常のエージェント移動はエージェント状態だけでなく実行コードの転送も含む．しかし，本システムではサーバにエージェントの実行コードを管理させておくことが可能であるため，移動元からサーバへのエージェント移動の際には実行コードを転送する必要はない．

3.2 エージェント転送方式

エージェントは実行基盤が提供する実行環境上で動作する．そのため，既存の移動エージェントシステムでは，利用者が，最初に自分の端末に実行環境をインストールすることを前提としていた．しかし，この方法は利用者の負担が大きい．我々は，実行基盤の配布方法がエージェント技術を採用する際の一つの課題であると考え，利用者が事前に基盤をインストールせずに済むエージェント転送方式を開発した．以下，エージェント転送方式について説明する．

エージェント転送方式を図 2 に示す．図中の矢印は，デスクトップ計算機から携帯情報端末へのエージェント移動時に発生する一連の処理を表す．デスクトップ計算機にはサービス提供サーバと実行基盤が配置される．一方，利用者側の携帯情報端末には，実行基盤を配置しておく必要はない．ただし，端末は外部からアプリケーションを取得し実行する機能を持った “アプリケーション管理ソフトウェア” を備えるとする．この条件のもと，エージェント管理サーバをサービス提供マシンと利用者端末の間に配置する．サーバは，サービス提供マシンから利用者端末へ移動するための機能を移動エージェントに提供する．実行基盤を持たない利用者端末にエージェントを移動させるために，サーバ内の転送装置は，エージェントと実行基盤を一つの “ アプリケーション” としてまとめ，アプリケーション管理ソフトウェアの機能を利用して利用者端末に送り込む．

3.3 実行基盤

本システムは携帯電話や PDA 等の携帯情報端末を動作プラットフォームとして想定している．これらの計算環境では，従来のデスクトップ計算環境と比較して，メモリや入出力デバイス，ネットワーク機能等，多くの点で制限されており，また，種類によって利用可能な機能が大きく異なる．従って，多くのプラットフォームで共通に動作可能なコア機能のみを備える実行基盤を設計する必要がある．

我々は以下の二つの機能がエージェント基盤として最低限必要であると判断した:

1. ライフサイクル管理機能
   エージェントの生成，消滅，中断，移動前，移動後等の各イベントに応じて，エージェントの適切な Call-back メソッドを呼び出す．

2. 移動機能
   実行基盤は，サーバから送られてきたエー

ジェント状態を表す情報からエージェントを復元し，エージェントの処理を再開させる．また，エージェントから移動要求を受け付けると，エージェントの状態を取得し，その状態を表す情報をエージェント管理サーバへ転送する．

エージェント間通信機能は実行基盤に含めない．エージェント間通信機能は，Telescript[10] や Aglets[2] 等，代表的な移動エージェントシステムにおいて基本機能として提供されている．しかし，エージェント間通信機能を利用するエージェントは，ネットワークが利用できない間，その処理を中断しなければならない．ユビキタス計算環境が利用する無線ネットワークの性質を考慮した場合，エージェント間通信機能を実行基盤に含めることは適切ではない．

実装では，コア機能を持つ実行基盤に必要に応じて他の機能を付加することが出来るビルディング・ブロック形式の枠組みを提供することにより，アプリケーション開発者の負担を軽減することが出来る．実行コードのサイズと消費メモリ量に関しては，実行基盤を実装する際に注意して小さく押さえる．

3.4 実装

3.4.1 エミュレータを利用した実装

携帯電話向けの Java 言語環境である MIDP [6] を利用して，軽量移動エージェントシステムを試作した．Sun Microsystems が提供する MIDP の参照実装に含まれる携帯電話エミュレータを用いて動作を確認した．

図 3 にエージェント実行環境を示す．携帯電話の利用者はエージェント管理サーバが提供するサービスを利用してエージェントを携帯電話内に移動させる．移動エージェントはエージェント管理サーバを経由して，携帯電話から携帯電話へ，携帯電話からサービス提供サーバへ，またはその逆向きに移動することが出来る．

実装した実行基盤は，2 つのインタフェースと 6 つのクラスから構成され，Jar により圧縮したサイズは，7.4Kbyte である．

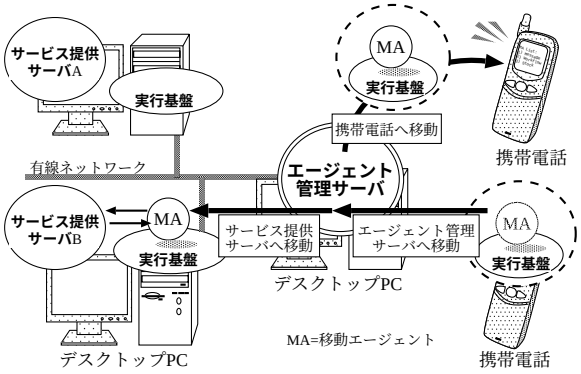

図 3: エージェント実行環境

3.4.2 携帯電話実機を利用した実装

次に，我々は携帯電話実機と公衆網を利用した実験を行うために，試作システムを i-mode 対応 Java[5] に移植した．MIDP と i-mode 対応 Java は，共に J2ME(Java2 Micro Edition) の定める枠組みに従って作成され，基礎となる Java 仮想機械 (KVM) 基本ライブラリ群 (CLDC: Connected Limited Device Configuration) は，同一の仕様を利用している．そのため，移植は容易であった．移植作業は，主に，GUI 関連クラスの修正と，アプリケーションの開始や終了などをフックしたアプリケーション・フレームワークが定める Call-back メソッドの名前の違いへの対応である．

i-mode 対応 Java の制限により，アプリケーションを構成する Jar アーカイブのサイズは 10Kbyte 以下でなければならない．そのため，実行基盤を移植後，実機上で動作させるため，更にクラス数の削減や機能の削除を行った．実行基盤は 3 つのクラスから構成され，Jar により圧縮したサイズは 5.4Kbyte である．

実際に携帯電話にエージェントを移動させたところ，エージェント移動に要する時間は，場所と時刻に応じて変化した．例えば，エージェント管理サーバから携帯電話実機へ転送するエージェント情報 (Jar ファイル) のサイズが 6Kbyte の時，エージェント転送時間を測定すると，測定場所，時刻に応じて約 15〜40 秒の間でばらついた．また，40 秒を越える場合は転送に失敗する結果となった．

i-mode 対応 Java 準拠の携帯電話，及び，MIDP 参照実装のエミュレータに含まれるアプリケーション管理ソフトウェアは，共に Pull 型のアーキテクチャを採用している．そのため，利用者がエージェントの移動を明示的に要求した場合にのみ，携帯電話へエージェントが転送される．

4 実行コード抽出

無線ネットワークは有線と比較してバンド幅が狭い．エージェント移動時に発生する転送量をより少なく押さえる必要がある．本節では，転送量を削減する手法としてエージェントの実行コード抽出 [4] を説明する．

4.1 アイデア

軽量移動エージェントシステムではエージェントが移動する際，必ずエージェント管理サーバを経由する．エージェント管理サーバは，移動元からエージェント状態を受け取り，移動元へ実行コードとエージェント状態を転送する．この時，エージェント状態を用いて実行コードを静的に解析し，得られた解析結果を用いて移動先で必要となる実行コードを抽出する．元の実行コードの代りに，抽出した実行コードを転送することにより，転送量を削減する効果が得られる．

4.2 既存のコード抽出・圧縮技術

オブジェクト指向言語を対象としたコード抽出/圧縮手法の研究は既にいくつか行なわれている [7, 9]．これらの研究では，アプリケーションの実行コードを静的に解析し，(1) 到達不能なクラス，メソッド，フィールドの検出，(2) パッケージ名，クラス名，メソッド名，フィールド名の短縮，(3) クラスの融合，これら三種類の処理を行なうことによりアプリケーションのコード量を削減する．抽出された実行コードは，抽出前の実行コードと同じ振る舞いをし，抽出前の実行コードの任意の可能な実行系列と同じ実行系列を持つ．

4.3 エージェント実行コードの抽出

移動エージェントの場合を考えると，二つの点で通常のアプリケーションのコード抽出と異なる．第一に，エージェントの一つのライフサイクルの中で，移動の度に，エージェントの実行コードが起動される．移動後に起動される実行コードのエントリポイントが常に同じとは限らない．第二に，エージェントを構成するオブジェクトは，エージェント状態を用いて復元されるため，エージェントの処理があるエントリポイントから開始された時点で，既にエージェント状態に含まれる変数の値は定義されている．

以上の性質から，次のようにエージェントのコード抽出を行う．まず，移動後に呼出されるエントリポイントを決定し，そのエントリポイントから Call Graph を作成する．その結果から，到達不能なクラス，メソッド，フィールドを判別する．次に，実行コードのデータフロー解析と制御フロー解析を行い，エージェント状態として与えられる変数の値を利用して，さらに厳密に到達不能なクラス，メソッド，フィールドの判別を行う．

前節で述べた試作システムとメッセージを持って指定された宛先を巡回するエージェントを対象に，コード抽出の実験を行った．その結果，携帯電話に転送される Jar アーカイブのサイズは 9%〜45%削減された [4]．実験では，フロー解析は行わず，エントリポイントを定め Call Graph を作成する方法のみを利用した．

5 まとめと今後の課題

ユビキタス計算環境向け軽量移動エージェントシステムを提案した．本システムは，エージェント管理サーバを中心とする Star 型のエージェント移動を前提とし，エージェント消滅に対応するためのバックアップ機能と，限られた計算機資源の中でも動作するように軽量化されたエージェント実行環境を特徴とする．

今後の課題として，エージェントの実行コード抽出手法を実装し評価する必要があると考えている．

参考文献

[1] R. Gray, et al: Mobile agents: Motivations and state-of-the-art systems, Dartmouth Univ., TR2000-365, April (2000).

[2] D.B. Lange, and Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley (1998).

[3] 前田 直人, 中島 震: 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, SPA2001 オンラインプロシーディング (2001).

[4] 前田 直人, 中島 震: コード抽出を行う移動エージェントシステム, 日本ソフトウェア科学会第 18 回大会論文集 (2001).

[5] NTT DoCoMo Web Site: ｉモード対応 Java, http://www.nttdocomo.co.jp/i/java.html.

[6] Sun Microsystems, Inc.: Mobile Information Profile, JSR-37, September (2000).

[7] 関口 龍郎, 大岩 寛, 米澤 明憲: オブジェクト指向言語によって記述された，携帯電話，PDA のアプリケーションプログラム圧縮方式, PPL2001 オンラインプロシーディング (2001).

[8] Tahara, Y., Ohsuga, A., and Honiden, S.: Agent System Development Method Based on Agent Patterns, Proc. ICSE'99, pp.356-367 (1999).

[9] F. Tip, C. Laffra and P. F. Sweeney: Practical Experience with an Application Extractor for Java, Proc. OOPSLA'99 (1999).

[10] White, J.E.: Mobile Agents, In Bradshaw M.J. (edt.), Software Agents, AAAI Press/The MIT Press, pp.437-472, April (1997).