ピア・ツー・ピアネットワークにおけるモバイルエージェント間メッセージ配送プロトコルとその応用事例

浅井 伸一† 堀田 英一†† 箱崎 勝也†

†電気通信大学大学院 情報システム学研究科
††日本電信電話株式会社 情報流通プラットフォーム研究所

概要

ピア・ツー・ピアネットワークにおいてモバイルエージェントを利用する場合，エージェントの位置の把握が一般的なネーミングサービス等では対応することができない．そのため，エージェントを発見できないことによりメッセージの配送が困難になる問題が生じる．

我々はこの問題を解決するために転送方式と照会方式の２つのメッセージ配送プロトコル提案している．本稿では本プロトコルの検証を行い，メッセージの配送およびエージェントの移動が確実に行えることを証明する．また，本プロトコルを使用したアプリケーションの実装と評価を通じてその有効性を確認する．

Message Delivery Protocols for Mobile Agents and Their Applications in Peer-to-Peer Networks

Shinichi Asai † Eiichi Horita †† Katsuya Hakozaki †

†Graduate School of Information Systems, The University of Electro-Communications
††NTT Information Sharing Platform Laboratories

Abstract

In peer-to-peer networks, it is often difficult or impossible to locate mobile agents and deliver messages to them, when using only a general-purpose naming service such as DNS. To solve this problem, two protocols are proposed for delivering messages to mobile agents in peer-to-peer networks. The first one utilizes redirection of messages by a proxy; the second one utilizes a lookup service for agent locations. It is verified that the two protocols ensure message delivery and agent mobility under appropriate conditions of networks. The effectiveness of the protocols are demonstrated by implementing and evaluating two application systems based on the protocols.

１ はじめに

昨今，計算機の性能向上による処理の分散化と環境構築の容易さから，ピア・ツー・ピアネットワークが注目されている．また，ピア・ツー・ピアネットワークにモバイルエージェントを利用してシステム／アプリケーションの効率化，高機能化，作成の容易化等を図ろうとする研究が行われている．しかし，ピア・ツー・ピアネットワークにおいて，自律的に移動しなおかつ移動頻度が高いモバイルエージェントを利用すると問題が発生する．これは，エージェントの位置の把握が一般的な分散システムのネーミングサービス等では対応しきれないため，エージェントの発見が難しくなり，エージェント間におけるメッセージの配送が困難になる．

現在，モバイルエージェントの移動先を追跡する一般的な方法[1]として，以下の３つがある．しかし，ピア・ツー・ピアネットワークにおいてはそれぞれ問題が発生する．

(1)エージェントが移動する際にエージェントサーバ内に移動先のログを残し，これを順に追跡していく方法(図１(a))
→エージェントの通過点となるサーバが１つでも停止してしまうと追跡ができなくなる
(2)エージェントが見つかるまですべてのエージェントサーバに問い合わせ，探し出す方法(図１(b))
→すべてのサーバへ総当たり的に問い合わせるため効率的ではない
(3)エージェントが移動するたびに，分散環境内のネームサーバに移動先の情報を登録し，これに問い合わせることで探し出す方法(図１(c))
→サーバへの処理の集中，移動のたびにリモートのサーバへ登録することによるオーバーヘッド，集中型サーバの設置が難しい，サーバを分散すれば情報更新に時間がかかる

そこで我々は，ピア・ツー・ピアネットワークにおける新しいモバイルエージェントの位置決定方式とエ

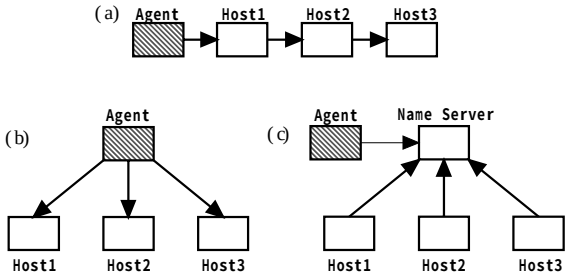

図１　移動先の追跡方法

ージェント間でのメッセージ配送方法として，ホームサーバを用いた転送方式と照会方式の２つのプロトコルを提案している[2]．

以下，２章では本プロトコルについて述べ，３章では本プロトコルの検証について述べる．そして，４章では実装と評価から考察を行い，５章では応用事例としてアプリケーションの提案と実装を行う．最後に６章で本稿をまとめる．

2　プロトコルとその特徴

2.1　ホームサーバ

ホームサーバＳ(HomeServer)は自分をホームとするエージェントの位置や名前等の情報を管理する．それとともに，他のエージェントからのメッセージ配送要求や問い合わせにも対応する．

各エージェントには，ホームサーバに対する移動許可の要求と移動後の報告を義務付ける．エージェントは移動しようとすると，ホームサーバに対し移動の許可を求める(moveReq)．ホームサーバは，エージェントが移動しても問題がないと判断すれば，移動の許可を与える(movePermission)．移動の許可を受けたエージェントは次のホストへと移動し，移動後ホームサーバに自分の情報を伝える(moved)．

しかし，エージェントがメッセージを受信する間もなく移動を繰り返してしまうと，永久にメッセージの配送を行うことができない．そこで，エージェントが移動の許可を求めてきたときに，ホームサーバはメッセージを確実に配送するため，移動を制限する役割も持つ．

2.2　転送方式(Redirection)

転送方式(図２)は，エージェントＦ(Finder)がエージェントＴ(Traveler)にメッセージを送信する際，エージェントＴのホームサーバＳへメッセージを送信する(message)．ホームサーバＳはエージェントＴへメッセージを転送し(deliverMsg)，メッセージの配送完了をエージェントＦへ報告する(msgSent)．

この時，エージェントＴが絶えず移動し続けていたり，移動中にホームサーバがメッセージを受け付けると，メッセージを配送できない．そのため，ホームサーバは moveReq を受信した時に，転送するメッセージを持っていれば，移動の許可を出す前にメッセージを転送する．

このプロトコルは，プロトコルの階層や用途が全く異なるものの，メッセージ(パケット)を転送するという点において，仕組みは Mobile IP[3]に近い．Mobile IP におけるノードがこのプロトコルにおけるエージェントと見ることができる．

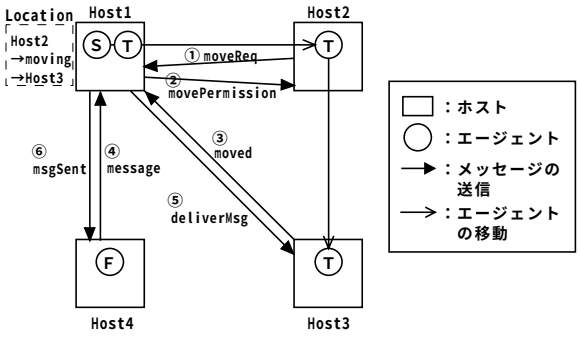

図２　転送方式

2.3　照会方式(Lookup)

照会方式(図３)は，エージェントＦがエージェントＴの滞在中のホストをホームサーバＳへ問い合わせる(lookup)．ホームサーバＳはエージェントＴの位置情報をエージェントＦへ回答する(place)．そして，その情報に基づき，直接メッセージを送信する(deliverMsg)．その後，メッセージの送信完了をホームサーバへ報告する(msgSent)．

この時，エージェントＴが絶えず移動し続けていたり，エージェントＴからホームサーバへの問い合わせ中に移動してしまうと，エージェントＴの位置情報が回答されずメッセージを配送できない．そのため，ホームサーバは moveReq を受信した時に回答待ちの問い合わせがあれば，回答後 msgSent を受け取るまで移動の許可を保留する．

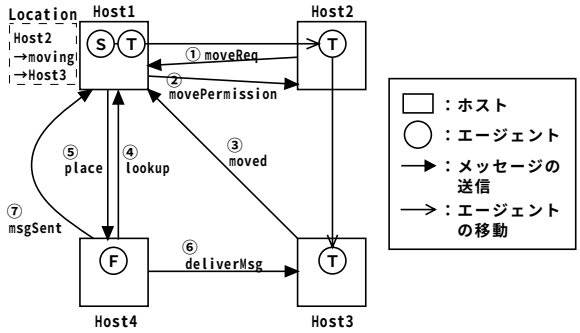

図３　照会方式

2.4　２つのプロトコルの特徴

両プロトコルの共通点として，以下のようなことが挙げられる．ホームサーバは自分をホームとしているエージェントについての情報を責任を持って管理するということと，他のエージェントからのメッセージ配送において，ホームサーバは自分をホームとするエージェントの移動するタイミングを調整し，メッセージのより確実な配送を行うようにしているということである．

また，両プロトコルの相違点としては，以下のようなことが挙げられる．転送方式においては，ホームサーバがメッセージを一旦受け付けた後，受信側エージェントの移動についての必要な調整を行いながらメッセージを転送する．よって，簡単なプロトコルで確実なメッセージ配送が実現できる．また，メッセージの

配送はホームサーバを経由して行われるため，受信側エージェントの位置情報を，送信側エージェントに知られることなく配送することが可能である．一方，照会方式においては，送信側エージェントが受信側エージェントの位置情報をホームサーバへ問い合わせる．照会した位置情報を元に直接メッセージを送信し，かつ位置情報の再利用を図ることで，通信量の点から効率的である．

3　プロトコルの検証

本プロトコル保証したい性質として以下のものがある．

<1>送信者のメッセージがシステムに受け付けられれば，必ず相手に配送される

<2>エージェントが移動要求を出せば，必ず移動を許可される

プロトコルの性質記述論理 ACTL(Action-Based Computational Tree Logic)[4]により，記述すると次のようになる．

$AG_T([\overline{acceptedByNS}]AF_T(<\overline{receiveMsg}>(T)))$…<1>

$AG_T([\overline{moveTraveler}]AF_T(<\overline{moveOK}>(T)))$　…<2>

ここで，T は恒真を表す記号，$\overline{acceptedNS}$ はメッセージが受け付けられたことを表す動作，$\overline{receiveMsg}$ はメッセージが相手に配送されたことを表す動作，$\overline{moveTraveler}$ はエージェントの移動要求がシステムに受け付けられた動作，$\overline{moveOK}$ はエージェントがシステムから移動許可を受けた動作である．また，$AG\chi(\phi)$はアクション式χを満たすアクションでラベル付けられた全てのパスの全ての状態でϕが成り立つことを表し，$AF\chi(\phi)$はアクション式χを満たすアクションでラベル付けられた全てのパスのある状態でϕが成り立つことを表す論理演算子である．これらは不動点演算子を用いて次のように表現される．

$AF\chi(P) = \mu X.P \vee (<\chi>(T) \wedge [\chi](X))$

$AG\chi(P) = \eta X.P \wedge [\chi](X)$

<1>の性質の検証について，CCS[5, 6]によるプロセスの記述を分析するツールである Concurrency Workbench(CWB)[7]を利用した．CWB は CCS によるプロセスの記述と命題μ計算で与えられた論理仕様に対しモデル検査を行い，仕様を満たすかどうかを検証する．図４，図５，図６にエージェントが移動可能なホストの数が２の場合における，本プロトコルの CCS によるプロセス式を示す．現在，一組の送信者と受信者がいる場合を検証し，本プロトコルでこの性質が保証されることを確認している．

現在のプロトコルに対して<2>の性質のを検証するためには，可能動作についての強公平性の制約を入れる必要がある．これは，MoveKeeper に moveReq が受け付けられても moveReq0 が NS1_nomsg 等に受け付けられるとは限らないからである．CWB ではこの公平性の制約を入れることが困難であるため，モデル検査プログラムである SPIN[8]を使用した．SPIN は PROMELA 言語[9]によるプロトコルの記述と線形時相論理で与えられた論理仕様に対してモデル検査を行い，仕様を満たすかどうかを検証する．現在，一組の送信者と受信者がいる場合を検証し，本プロトコルで性質<2>が保証されることを確認している．

今後の課題として，送信者と受信者が複数のいる場合に<1>，<2>の性質を検証することがある．公平性

```
RedirectSystem  =def (Finder | NS1_nomsg | MoveKeeper | Traveler1) \ L
                   where L = {moveReq, moveReq0, movePermission, moved1, moved2,
                              message, deliverMsg1, deliverMsg2, msgSent}
NS1_nomsg       =def message.acceptedByNS.deliverMsg1.msgSent.NS1_nomsg + moveReq0.movePermission.NS1_moving_nomsg
NS1_moving_nomsg=def moved2.NS2_nomsg + message.acceptedByNS.NS1_moving_msg
NS1_moving_msg  =def moved2.deliverMsg2.msgSent.NS2_nomsg
NS2_nomsg       =def message.acceptedByNS.deliverMsg2.msgSent.NS2_nomsg + moveReq0.movePermission.NS2_moving_nomsg
NS2_moving_nomsg=def moved1.NS1_nomsg + message.acceptedByNS.NS2_moving_msg
NS2_moving_msg  =def moved1.deliverMsg1.msgSent.NS1_nomsg
Finder          =def message.acceptedByNS.accepted.msgSent.Finder
```

図４　転送方式のCCS によるプロセス式

```
LookupSystem        =def (Finder | NS1_noLookup | MoveKeeper | Traveler1) \ L
                         where L = {moveReq, moveReq0, movePermission, moved1, moved2,
                                    lookup, place1, place2, msgSent1, msgSent2, deliverMsg1, deliverMsg2}
NS1_noLookup        =def lookup.acceptedByNS.place1.msgSent1.NS1_noLookup + moveReq0.movePermission.NS1_moving_noLookup
NS1_moving_noLookup =def lookup.acceptedByNS.NS1_moving_lookup + moved2.NS2_noLookup
NS1_moving_lookup   =def moved2.place2.msgSent2.NS2_noLookup
NS2_noLookup        =def lookup.acceptedByNS.place2.msgSent2.NS2_noLookup + moveReq0.movePermission.NS2_moving_noLookup
NS2_moving_noLookup =def lookup.acceptedByNS.NS2_moving_lookup + moved1.NS1_noLookup
NS2_moving_lookup   =def moved1.place1.msgSent1.NS1_noLookup
Finder              =def lookup.accepted.(place1.deliverMsg1.msgSent1.Finder + place2.deliverMsg2.msgSent2.Finder)
```

図５　照会方式のCCS によるプロセス式

```
Traveler1  =def τ.deliverMsg1.receiveMsg.Traveler1 + τ.moveReq.T1_moving
T1_moving  =def deliverMsg1.receiveMsg.T1_moving + movePermission.moveOK.τ.(moved2.Traveler2)
Traveler2  =def τ.deliverMsg2.receiveMsg.Traveler2 + τ.moveReq.Traveler2
T2_moving  =def deliverMsg2.receiveMsg.T2_moving + movePermission.moveOK.τ.(moved1.Traveler1)
MoveKeeper =def moveReq.moveTraveler.moveReq0.MoveKeeper
Property1  =def AGT([acceptedByNS](AFT(<receiveMsg>(T))))
Property2  =def AGT([moveTraveler](AFT(<moveOK>(T))))
```

図６　両方式で共通な部分のCCS によるプロセス式

の制約を入れれば現在のプロトコルにおいて<1>，<2>とも検証可能であると予想され，この検証を進めている．同時に公平性の制約を必要としないプロトコルを検討している．

4　プロトコルの実装

実装については，モバイルエージェントフレームワークである Aglets Software Development Kit(ASDK) [10, 11]と Java 言語を用いた．そして，本研究で検証した２つのプロトコルに従ってメッセージ通信を行うエージェントプログラムを作成した．

4.1　評価と考察

作成したエージェントプログラムを使用した評価を文献[2]で行っている．この評価結果からの考察として，以下のことが挙げられている．

(1)照会方式の方が必要となる制御用メッセージ数が多いため，配送時間が転送方式に比べて遅い
(2)実際のネットワーク環境では，制御用メッセージが多くなるほどトラフィックの量やネットワークの速さが影響してくる可能性がある
(3)今回の評価実験ではメッセージ内容が１語の文字列であったが，これが大きくなるとメッセージ内容がホームサーバを経由することによる転送方式での遅延の発生が懸念される
(4)照会方式では，一度ホームサーバへ問い合わせたエージェントＴの位置情報を保存・再利用するが可能であり，この効果で転送方式よりも配送時間が短くなることも考えられる．
(5)MOA[12]におけるエージェント間のメッセージ通信時間と比較すると，メッセージ配送時間が遅い
→エージェントへの移動制限および移動中による遅延と，Aglets がメッセージを送信するたびにコネクションを張り直していることによる遅延

そこで，文献[2]と同様の実験環境でエージェントＴを移動しないようにして測定した結果を表１に示す．エージェントＴの移動による影響を受けていないため，先程の結果よりもメッセージ配送時間が短縮されている．また，メッセージ通信用のコネクションを保持したままにするよう Aglets を変更して測定した結果を表２に示す．コネクションを保持している場合のメッセージ配送時間が，毎回コネクションを張り直す場合の３分の２程度になっていることがわかる．

表１　コネクション張り直す場合のメッセージ配送時間

	メッセージ間隔(ｍｓ)		
	10	100	1000
転送方式	82	75	72
照会方式	175	161	127

表２　コネクションを保持する場合のメッセージ配送時間

	メッセージ間隔(ｍｓ)		
	10	100	1000
転送方式	65	58	42
照会方式	120	106	65

今回，評価を行ったシステムでは，UDP でパケットを送信するだけで 5ms ほどかかっているので，TCP でメッセージを送って ack を返す，ということを行っている今回のメッセージ配送はこの程度の時間がかかると考えられる．

4.2　ネーミングサービス

現在利用している Aglets はネーミングサービスを持っていない．本プロトコルで提供しているのはエージェントに対するロケーションサービスであり，ネーミングサービスの提供までは行っていない．そのため，本プロトコルを実装するにあたり，LDAP を用いたネーミングサービスを実装した．

まず，エージェントから LDAP サーバへの問い合わせをエージェント作成者とクラス名で行う．ここで，エージェント作成者は個人または個人のグループでエージェントを作成し，その動作について責任を持つものである．この問い合わせに対して，LDAP サーバはホームサーバのアドレスと AgletID を返す．なお，AgletID はエージェント生成時にシステムによって自動的に付加されるユニークな ID である．(図７)

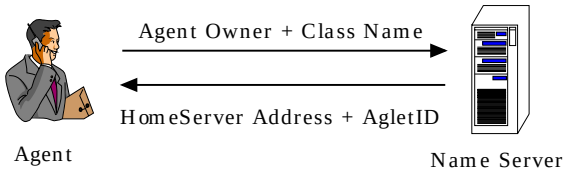

図７　ネーミングサービス

5　応用事例

本プロトコルを利用したアプリケーションとして，回覧板エージェントと情報配信・管理エージェントを提案する．回覧板エージェントは転送方式を利用し，情報配信・管理エージェントは照会方式を利用したアプリケーションである．5.1 および 5.2 ではこれらのアプリケーションについて述べる．

5.1　回覧板エージェント

転送方式を利用したアプリケーションとして，回覧板エージェントを提案する．エージェントが回覧板(の内容)を持ち，登録されたユーザのホストをまわり回覧板を見せる．回覧板を見せたらその人から確認印をもらい，回覧板の管理者に報告する．(図８)

5.1.1　背景

回覧板を回した場合，途中の人がいないとそこで止まってしまうことになる．その間，誰も回覧板がどこで止まっているか把握することができず，回覧板の管理者(回した人)は誰が回覧板を見たのかも確認することができない．

そこで，モバイルエージェントと転送方式を利用して回覧板を回し，この問題の解決を図る．

5.1.2　動作

エージェントは回覧板を見て欲しい人のホストを順にまわり，その人がいれば回覧板を見せ確認を受ける．そして次の人(ホスト)へと移動して行く．もし，不在の人(ホスト)がいればそこを飛ばし，次の人へ先に見せる．このことは管理者にも伝えておき，後でまたこの人へは回覧板を見せる．図９に回覧板エージェントの状態遷移図を示す．

ユーザが回覧板を見た場合には確認印をもらう．確認印として，自分の名前の入力し表示したボタンを押

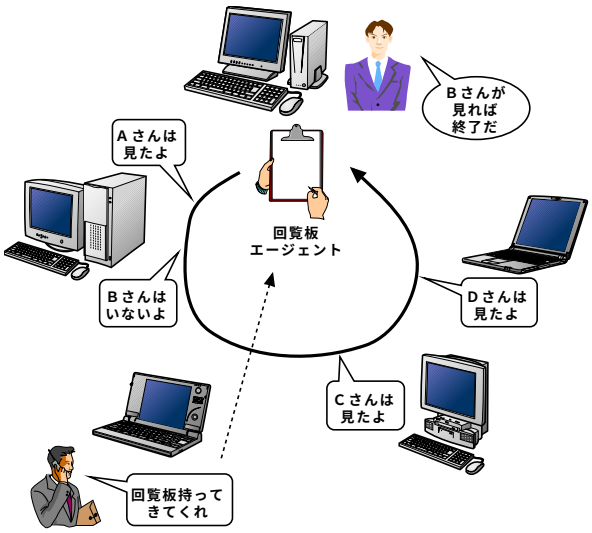

図８　回覧板エージェント

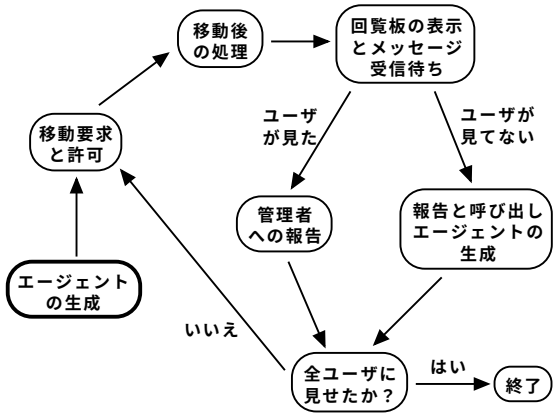

図９　回覧板エージェントの状態遷移図

してもらうようにする．確認を受けたら，管理者(回覧板を回した人)に報告(メッセージ送信)する．また，次の場合はユーザが不在と判断する．

・エージェントが移動しようとして移動できなかった場合
・移動して回覧板を表示したのにもかかわらず，確認印をもらえない場合

この場合には，その人を飛ばして次の人のところへ行く．この時，不在で回覧板を見せられなかった人のことを管理者に報告する．また，不在だった人へは一通り巡回した後でもう１度回覧を試みる．また，不在だった人から連絡を受けた場合にはすぐに回覧板を持っていくようにする．

移動先でネットワークが切断された場合，Aglets の非活性化(Deactivate)機能を利用して，再びネットワークに接続されるのを待つ．非活性化をするとローカルのストレージにエージェントが保存され，指定時間後に再び活性化(Activate)する．指定時間前に Aglet Server が終了した場合には，次回の Aglet Server 起動時に活性化する．活性化した後は巡回の続きを行う．

5.1.3　特徴

管理者は，回覧板エージェントとのメッセージ通信が保証されていることから，必要なときに現在誰が見ているのか(どこのホストにいるのか)，今までに誰が回覧板を見たのかなどの情報を得ることができる．また，こちらから命令を送ることで，エージェントの動作の変更(回覧板を見せる人の変更，途中での回覧板エージェントの強制回収など)を行うことができる．

不在で回覧板を見ることができなかった人は，転送方式を利用することで回覧板エージェントに連絡を取ることができる．これにより，自分の都合のよいときに回覧板を持ってきてもらうといったことが可能となる．

5.2　協調型パーソナル情報管理エージェント

照会方式を利用したアプリケーションとして，協調型パーソナル情報管理エージェントを提案する．各ユーザに対応したエージェントはそのユーザのパーソナル情報を管理し，他のユーザのエージェントとの間で情報を送受信することで各々のパーソナル情報の協調的な管理を支援する．また，エージェントは各ユーザの移動にともない，ネットワーク上の計算機間およびノートパソコン等の持ち運び可能な計算機に保存されて移動を行う．(図１０)

5.2.1　背景

グループ内での LAN 環境における情報の共有は既存のシステムでも可能であるが，移動を繰り返すユーザに情報の送信を行うという点では問題がある．仮に，電子メールで情報を送信するとする．この場合，移動先のユーザがメールサーバをチェックしない限り気が付くことはないし，大きな添付ファイルがある場合にはあまり好ましいことではない．また，移動を繰り返すユーザにしてみれば移動先のネットワークに接続した自分の計算機でも，移動先にある計算機でも同じ環境で情報管理をして作業を継続したいという要望がある．

そこで，モバイルエージェント型のパーソナル情報管理アプリケーションと照会方式を利用することでこの問題の解決を図る．なお，今回は扱うパーソナル情報としてスケジュール情報を取り上げる．

5.2.2　動作

照会方式を利用して配信された情報(メッセージ)はファイルとして保存しておく．また，自分で登録した情報および他のエージェントから配信された情報に対する編集・削除・送信といった操作が可能となっている．

エージェントはユーザの行動にともない，二種類の移動の仕方がある．まず，ユーザが他の計算機に移動して作業をする場合，エージェントはユーザの指示によりネットワーク上を移動し，他の計算機に移る．この時，移動元と移動先にある情報(ファイル)について同期をとり，この２つの環境を同一のものとする．次に，ユーザがエージェントをノートパソコン等の持ち運び可能な計算機に入れて移動する場合，以下のような手順を踏む．エージェントはネットワーク上を移動するときと同様にホームサーバから移動の許可を受ける．その後，エージェントは自分自身を非活性化し，

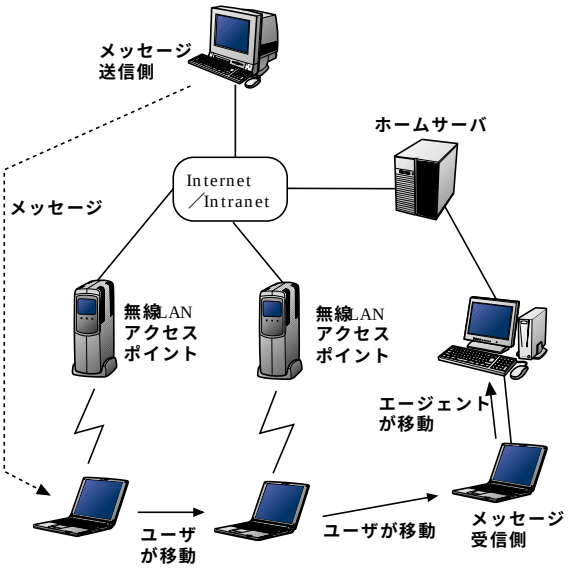

図１０　パーソナル情報管理エージェント

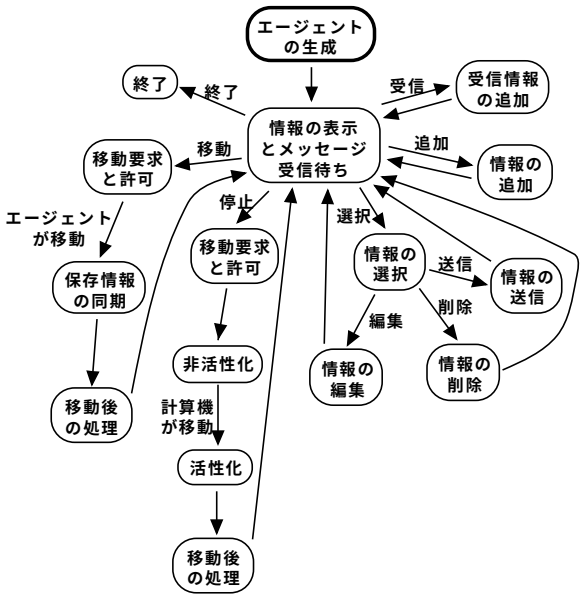

図１１　パーソナル情報管理エージェントの状態遷移図

ローカルストレージに保存されて移動する．そして，ユーザにより移動後エージェントは活性化され，移動が完了したことをホームサーバへ報告する．ユーザの移動が長時間に及ぶ場合，ホームサーバで問い合わせが滞留する可能性がある．そのため，問い合わせの受付後一定時間が経過した場合には，メッセージ送信先が移動中であることを回答する．図１１にパーソナル情報管理エージェントの状態遷移図を示す．

5.2.3　特徴

エージェントがユーザにともなって物理的もしくはネットワーク上を移動しても他のユーザは追跡が可能になるため，ユーザは移動してもネットワークへの接続中は確実に情報を受け取ることができる．さらに，別の計算機での作業時にも今まで使用していたエージェントによって情報の配信を受けることが可能となり，作業を継続することができる．

また，照会方式を利用することにより，メッセージとして配信する情報量が多い場合においても転送方式により効率的に配送することができる．

６　まとめ

今回，我々が提案しているプロトコルについて２つの性質が成り立つかどうかを検証し，本プロトコルでこの性質が保証されていることを確認した．また，評価を通じて明らかになった本プロトコルの特徴から，応用事例として２つのアプリケーションの提案と実装を行った．

今後の課題として，メッセージの送信者と受信者が複数の場合の検証が挙げられる．また，今回提案・実装したアプリケーションはまだプロトタイプの段階である．そのため，実際の利用においてはさらにアプリケーションの完成度を高める必要があると考えている．

参考文献

[1] Yariv Aridor and Mitsuru Oshima: Infrastructure for Mobile Agents: Requirements and Design, in Proceedings of Workshop on Mobile Agents MA'98, Lecture Notes in Computer Science, Vol.1477, pp.38-49, Springer, 1998.

[2] 浅井伸一, 堀田英一, 箱崎勝也: Peer-to-Peer ネットワークにおけるモバイルエージェントの位置決定およびメッセージ配送方式, 情報処理学会 DICOMO シンポジウム論文集, pp.409-412(2002)

[3] C. Perkins: IP Mobility Support for IPv4, RFC3220, The Internet Society.

[4] R. De Nicola and F. Vaandrager: Action versus state based logics for transition systems, in Proceedings of LITP Sprint School on Theoretical Computer Science, Lecture Notes in Computer Science, Vol.469, pp.407-419, Springer, 1990.

[5] Robin Milner: Communication and Concurrency, Prentice Hall, 1989.

[6] Glenn Bruns: Distributed Systems Analysis with CCS, Prentice Hall, 1996.

[7] Rance Cleaveland, Joachim Parrow and Bernhard Steffen: The Concurrency Workbench: A Semantics-Based Tool for the Verification of Concurrent Systems, ACM Transaction on Programming Languages and Systems, 15(1):36-72, 1993.

[8] Spin - Formal Verification: http://spinroot.com/spin/

[9] Gerard J. Holzmann: Design and Validation of Computer Protocols, Prentice Hall, 1991

[10] Aglets Software Development Kit: http://www.trl.ibm.com/aglets/

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

[12] D. S. Milojicic, W. LaForge, D. Chauhan: Mobile Objects and Agents (MOA), in Proceedings of the USENIX Coonference on Object-Oriented Technologies and Systems(COOTS), April 1998.