副 テ ー マ

モバイルエージェントによる
経路監視機構に関する研究

副テーマ指導教官　敷田 幹文 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

酒井 潤

2002 年 10 月 15 日

目 次

図 目 次

表 目 次

第 1章

序論

1.1 はじめに

インターネットの急速な普及に伴い、モバイルエージェントと呼ばれる、主にネットワーク上で自律的かつ協調的に振る舞うことにより人間の活動の支援を行うソフトウェアが注目を集めている。また、インターネットの複雑化、大規模化に伴い、ネットワークの構成が動的に変化する。そのような環境において、ネットワーク経路監視は、管理をする面において困難となっていくだろう。そこで、本研究は、新しい経路監視システムとして、モバイルエージェントを用いた経路監視システムを提案・構築し、その有効性を検討する。

1.2 本論文の構成

以降、2章では従来方式と提案方式について概観する、また問題点に付いて述べる。3章では本研究で用いたモバイルエージェントについて、また、その一種である Aglets について概説する。5章ではシステム評価方法ついて述べる。6章では現状でのモバイルエージェントの課題、将来の展望について述べる。

第 2章

従来方式・提案方式

2.1　従来方式

ネットワークにおける監視システムの代表的な一つとしてSNMPがある。これは、UDP/IPをベースとし、ネットワークに接続された機器をネットワーク経由で監視するためのプロトコルである。管理システム（マネージャー）と、ルーターやハブなどの各種の管理対象となるネットワーク構成機器（エージェント）の間で管理に必要なデータを享受する方法を決めている。

しかし、従来方式であるSNMPでは、ネットワーク単体だけの監視となる。この方式は、ネットワーク監視において経路監視までも正しく監視することはできず、クライアント側からの監視が行いにくいと問題がある。

2.2　提案方式

本研究では、従来方式により行いにくいとされる経路状態をネットワーク機器単体だけでなく周辺の経路状態をも監視するために、モバイルエージェントを用いた経路監視機構を提案する。

第 3章

Mobile Agentについて

3.1 Mobile Agent 概説

エージェントとは、自律性 (autonomy), 社会性 (social ability)、反応性 (reactivity)、自発性 (pro-activeness) の４特性を備えたソフトウェアあるいはハードウェアに立脚したシステムであるとされている。

また、モバイルエージェントとはエージェントの１機能分類であり、「人間の代行として自律的に計算機間を移動しながら、仕事の遂行をする計算主体」と定義することができる。このことの様子を図を 3.1 に示す。モバイルエージェントは自律的に計算機間を移動し、到着した計算機上で処理継続するプログラムである。移動する際には、プログラムコードはもちろん実行状態も転送される。

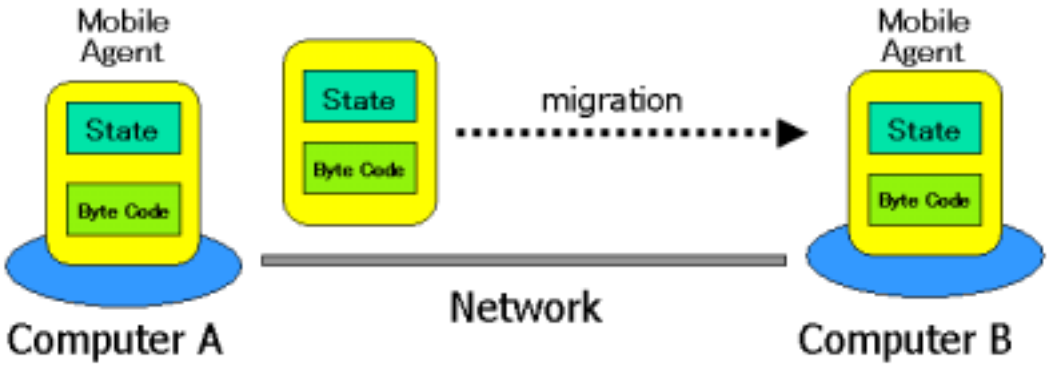

図 3.1: MobileAgent

3.2 Mbile Agent の利点

- ネットワーク負荷低減
  通常のサーバー・クライアント間コミュニケーションは、ネットワークを経由してメッセージを交換することによって行われる。しかし、エージェントがサーバーの存在するホストに移動すれば、ネットワークを経由せずに直接サーバーと通信を行うことができ、ネットワーク通信量を削減できる可能性が生じる。
- 通信路の切断による非同期可能
  コストや通信路の特性から通信路の切断を余儀なくされる場合がある。このような場合に、エージェントを切断される通信路の向こうに移動させておくことにより、通信路が切断された状態でもエージェントの実行を継続することができる。
- ソフトウェアの保守管理の容易化
  計算機には、モバイルエージェントプラットフォームだけを事前にインストールすればよく、計算機ごとのソフトウェアのインストールおよびアップグレード作業が不用であるため、保守管理が容易となる。
- 分散プログラミングの容易化
  サーバー/クライアント方式のプログラム設計の場合に必要な通信プロトコルの決定および通信プログラムの記述の必要がないため、プログラムが容易となる。
- 処理機能の動的配置
  移動先ホストに、期待する処理機能がインストールされていない場合を考える。このような場合でも、エージェントがその機能に対応するプログラムを内包して移動し、移動先ホストでそのプログラムを実行することが可能である。

3.3 Mobile Agent の欠点

- テストデバッグが困難
  全ての移動先の環境を把握するのは困難である。そのためデバックが困難である。
- キラーアプリケーションが無い
  「モバイルエージェントでなければできない」というアプリケーションは今のところ見つかっていない。
- セキュリティ機構
  モバイルエージェントが移動先コンピュータの各種リソースを不正にアクセスする。

また、エージェント自身が悪意を持ったエージェントによって攻撃される場合がある。エージェントが移動した先で、不正なホストがモバイルエージェントを攻撃するということも考えられる。

モバイルエージェントの応用としては、分散検索、負荷分散、並列実行、電子商取引、ネットワーク管理システム、ワークフロー、グループウェアなど様々な分野で利用されている。

3.4 Agletsについて

Aglets Workbench は IBM が研究開発を続けているモバイルエージェントのためのフレームワークである。Aglets は移動エージェントの一種で、ネットワーク上で移動・実行可能なオブジェクトである。プログラマは Java のクラスライブラリィとして提供される抽象クラスを継承することにより、プラットフォーム独立で、ネットワーク上を移動しながら実行されるエージェントプログラムを容易に作成することができる。Applet とは違い、エージェントの実行環境があるところに Java のオブジェクトを送り込んでオブジェクトを実行させるものである。また、エージェントの生成、削除、その他の制御のためのビジュアルな環境として GUI である Tahiti も用意されており、インターネット上を動き回るエージェントの制御を比較的簡単に行うことができる。マルチプラットフォームである Java の高い移植性のためにほとんどの計算機上で Aglets を実行できる利点がある。

3.4.1 Agletsの利用形態

エージェントは固有の環境で実行でき、今はインターネット/イントラネットの TCP/IP を通信できるレイヤーの上で実現しているが、他の通信が可能なレイヤーをインプリメントすれば他の通信環境でも通信が可能になるようにデザインされている。

3.4.2 移動実行の原理

エージェントはクラス定義、データをスナップショットとして考えることができる。 実行環境の AgletContext は Java で書かれていて、ハード上の JavaVM で実行される。エージェントが移動するときには実行をいったん止め、クラスとデータに分けてネットワーク

上をストリームとして送る。この通信のプロトコルを ATP(Agent Tracefer Protocol) と呼んでいる。

ATP(Agent Trancfer Protocol) とは エイジェントとシステムの間の独立した方法でエイジェントの移送のために設計されたアプリケーション・レベルのプロトコルである。インターネット上にあるネットワークで結ばれたコンピュータ間でエージェントを移動・実行させるためのプロトコルを提供する。

3.4.3 他のモバイルエージェント

- Telescript
  GeneralMagic 社による最初のモバイルエージェントシステム
- Voyager
  ObjectSpace 社による CORBA 対応の Java ベースのモバイルエージェントシステム
- Plangent
  東芝が開発するプランニング機能を重視した Java ベースのモバイルエージェントシステム
- Bee-gent
  東芝によるエージェント間通信に重点をおいたモバイルエージェントシステム
- Odyssay
  GeneralMagic 社による Telescript を Java ベースに移植したモバイルエージェントシステム

3.4.4 Aglets の基本概念

- Aglet
  自律的 (Autonomous) かつ反応的 (reactive) な Java オブジェクト．自分自身がスレッドであり，移動した後も自分自身で動き出すので自律的である．メッセージに対して反応することができるので反応的である。
- AgletProxy
  AgletProxy は Aglet の代理である．Aglet へのダイレクトなアクセスから Aglet を保護する．また，Aglet の位置透過性を実現する（つまり，AgletProxy によって，Aglet の本当の場所を意識せずに済む)。

- AgletContext
  Aglet の workplace である．基本的には，一つの **AgletServer** に対して一つの Aglet-Context がある（一つの AgletServer に複数の AgletContext を存在させることも可能である）。
- Message
  Aglet 間で交換される Java オブジェクト。

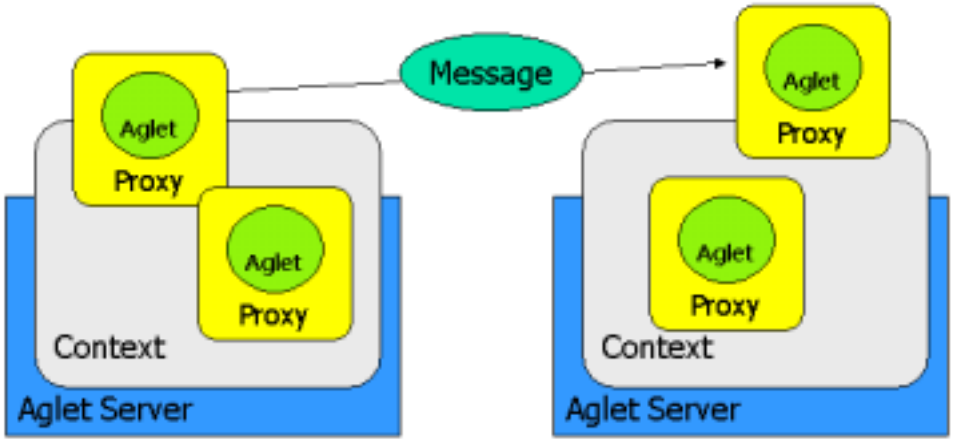

図 3.2: Agelt の基本概念

3.4.5 エージェントのイベント

モバイルエージェントは必要に応じて、制約なく望みの場所に移動できることが望ましい。エージェントが移動することに意味があるならば、エージェントの移動時に環境の変化にともなう処理が要求されることもある。AWB では移動および状態の変化に対するエージェントの対応の記述の容易さを重視した。エージェントの状況に変化が起こる前後にはエージェントへのイベントを送り、そこに処理の記述を行うことが可能になっている。

次のリスト一覧はエイジェントの基本的演算を要約している。すなわちそれらは創生 (Create)、クローニング (Clone)、発送（Dispatch）、撤回・撤収（Retract）、不活性化、活性化 (Activate and Deactivate、処分 (Dispose) である。これらのことを図 3.3 に示す。

- 生成 (Create)

Aglet の創生はコンテクストにおいて生じる。新しい Aglet は識別を割り当てられ、コンテクストに挿入され、そして初期立ち上げが行われる。Aglet は、Aglet が成功裏に初期立ち上げされるや否やすぐ、実行を開始する。

- クローニング (Clone)
  Aglet のクローニングは同じコンテクストにおいて元の Aglet と殆ど同一のコピーを生成する。元の Aglet とコピーの Aglet の唯一の差異は割り当てられる識別と、実行が新しい Aglet において再び開始するという事実である。
- 発送 (Dispatch)
  Aglet を 1 つのコンテクストから他のコンテクスト発送することは Aglet を現在のコンテクストから取り去り、そこで Aglet が実行を再び開始する。目的地のコンテクストへ Aglet を挿入することになる。
- 撤回・撤収（Retract）
  Aglet の撤回はその現在のコンテクストからそれを引き戻し、それを、そこから撤回が要求されたところのコンテクストへ挿入する。
- 活性化と不活性化 (Activate and Deactivate)
  Aglet の不活性化は一時その実行を停止し、その状態を 2 次ストーレジに蓄える。Aglet の活性化は同じコンテクストにおいて Aglet を回復する。
- 処分 (Dispose)
  Aglet の処分はその現在の実行を停止し、その現在のコンテクストから Aglet を取り去る。

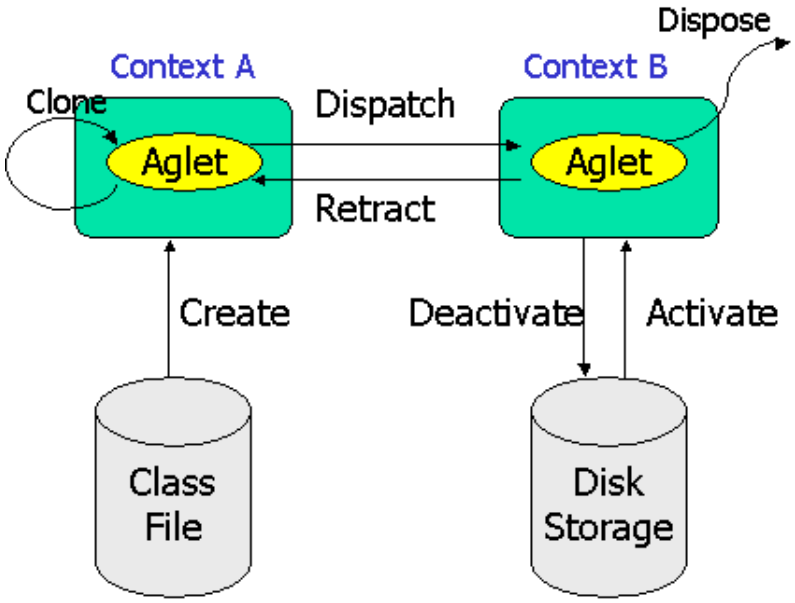

図 3.3: イベント駆動型エージェント

第 4章

システム構成

4.1 実行環境

本システムの実装方針と、各構成要素に対する実装の概要について述べる。

4.1.1 実装方針

まずエージェントはマルチプラットフォーム性が求められる。エージェントはネットワーク上の様々なホスト上を移動するため、OS やハードウェアに依存した実装は望ましくない。したがって実装には Java 言語を採用した。Java とは、中間コードと呼ばれるバイトコードが Java 仮想マシン内のインタプリタによってネイ ティブコードに変換されてから実行される。そのため Java はハードウェアや OS に依存しない。また、プラットフォームに依存せず、ネットワー クを意識したプログラミング言語をサポートしている機器であれば、Java で書かれた同じプログラムを動作させることができる。したがって、Java は各ホスト間を移動するモバイルエージェントを記述するための言語としては非常に望ましい。

4.1.2 Aglet システム構成

Aglet におけるシステム構成を図 4.1 に示す。

OS の上に JavaVM、Security Layer を生成しその上に Aglet サーバを立てる。移動する計算機上にも同様に Aglet サーバを立てることにより Aglet が移動できるシステムを構成できる。エージェントの移動には TCP/IP を用い、プロトコルとして ATP を用いている。

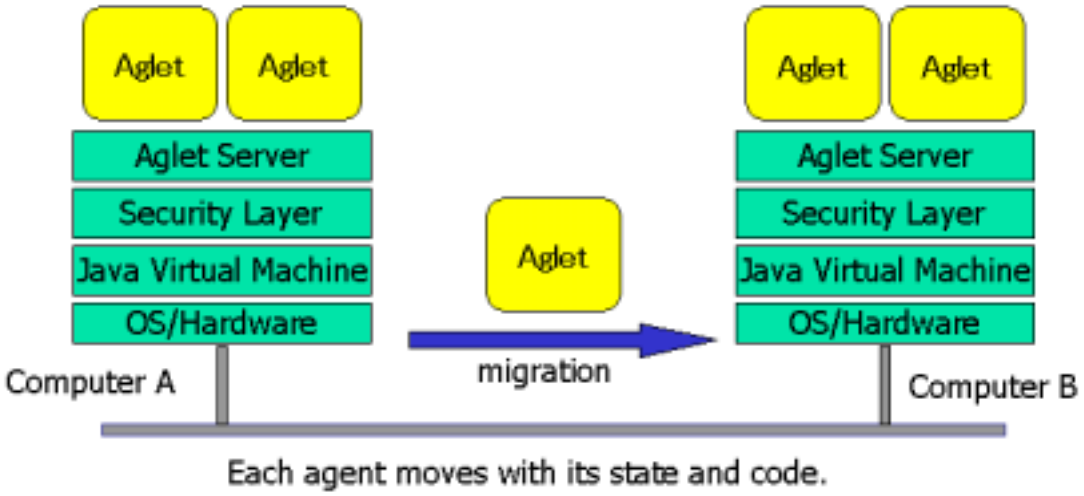

図 4.1: システムの概要

4.1.3 エージェントプラットフォーム

エージェントプラットフォームは、エージェントの生成や移動ならびに消滅、エージェント間通信などを司るサーバプログラムである。すなわち、エージェントプラットフォームはユーザ側・データソース側のいずれにも存在する。エージェントの到着を待ち受けるサーバ部分と、内部に存在する各エージェント状態を管理するモニタ部分から構成される。

4.1.4 処理のためのモジュール

処理のためのモジュールはエージェントプラットフォーム内に存在し、エージェントによって動的にロードされ、処理を実行する。

4.1.5 動作手順

まず、ユーザは自分のホスト上でエージェントプラットフォームを起動し、これを通じてエージェントを生成する。そして、取得するデータや期待される加工・変換処理、表示形式を指定する。エージェントは対象とするデータが存在するホストの、エージェントプラットフォーム上に移動する。そして、データソースに対して通信を行い、目的とする

データを取得する。次にエージェントは必要に応じて、取得したデータに対して、必要な処理を行う。最後にエージェントは生成されたコンポーネントを保持してユーザ側ホストに戻り、表示など処理を行う。

4.1.6 エージェント本体

本研究におけるエージェント本体には、以下の機能要件をみたすクライアントプログラムを実装した。

- データ
  移動するホストのリストを所持
  巡回するサーバのリストを所持
  サーバまでのルータのリストを所持
  監視対象までのパケット到着時間の閾値を所持

- 処理動作
  保持したリスト通りに各ホストを巡回する機能
  ホストの調査を行う機能
  サーバへの経路調査を行う機能
  障害情報をマネージャへ報告する機能

4.2 経路監視システム

本経路監視システムを図 4.2 に示す。まずマネージャはエージェントである Aglet を生成・送出し、Aglet は保持しているホストのリストを参照して各ホストを巡回していく。また Aglets は一定期間ごとにホスト間を巡回する。各ホストでは経路情報を取得するプログラム traceroute を実行する。その所得した情報を元にエージェント自身が自律的に経路状態を監視し判断するというシステムになっている。

次に監視対象のものと処理動作をあげる。

- ホスト
  まず、移動するホストに異常があった場合は Aglet はそのホストを飛ばし、次のホス

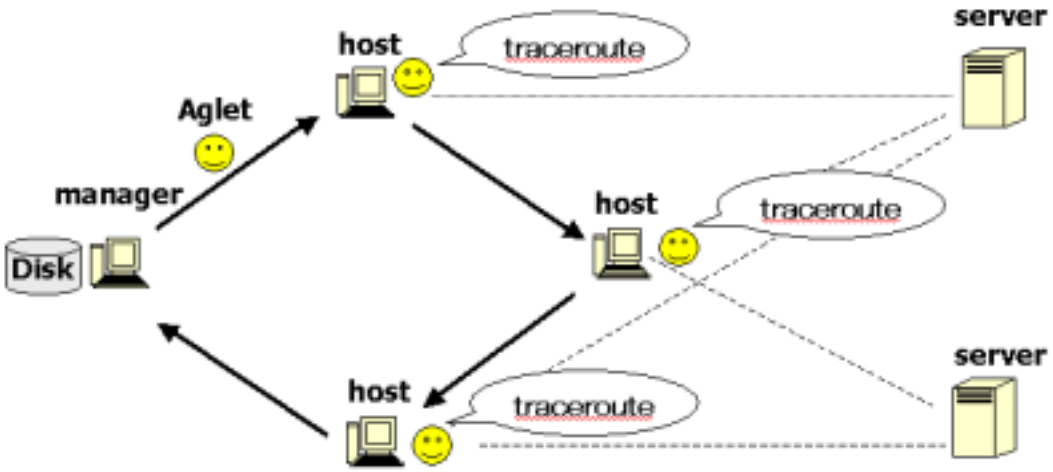

図 4.2: 経路監視システム

トへ移動する。その際、移動できないことをエージェントが判断しその旨をマネージャに即座に送信する。送信する際には、メッセージというオブジェクトを生成しそのオブジェクトをマネージャへ送信する。このようにして、ホストの異常に対して処理を行った。

- サーバ

  サーバに対しては、まず管理対象となるサーバまで traceroute プログラムを実行する。そのサーバに異常があった場合には、その旨をマネージャーへメッセージとして送信する。また traceroute プログラムは経路情報を取得するために各対象物までパケットを送信するためパケットが到着するまでの時間の閾値を設定し、その閾値以上の時間がかかった場合にもマネージャへメッセージを送信するように処理をさせた。

- ルータ

  traceroute プログラムはサーバまでの経路途中のルータ情報も取得することができる。そのためサーバ同様に、経路途中にあるルータの異常とルータまでのパケット到着時間を監視、判断させる処理を行わせた。

本研究では、以上のように経路状態をネットワーク機器単体だけでなく周辺の経路状態をも監視する、経路監視機構を実装した。

第 5章

評価

5.1 試用実験と考察

実装したモバイルエージェントによる経路監視プログラムの有効性や効果を調べるために、実装したプログラムを用いて、ネットワークで各ホストを移動するモバイルエージェントの旋回時間と通信量を測定した。今回測定に使用したマネージャは is14e0u55 とし、旋回したホストとして is24e0u55,is16e0u32,is34e1u18 とした。また監視するサーバとしては is14e0u39 とした。はじめに各ホストを数を徐々に増やし旋回時間と通信量を Ethereal を用いて測定した。また測定回数は 30 回の平均値をとり、各ホスト数の増加と旋回時間の関係について調べた。結果を表 5.1 に示す

結果からほぼホスト数の増大に対して直線的に旋回時間と通信量がともに増加していることがわかる。このことからホストが増大してもある程度の旋回時間、通信量が予測することができる。しかし、今回の実装したプログラムでの旋回時間においては、ホストの巡回時間はほぼ直線的に上がっていたが、旋回するホストの中で遠隔にあるホストが存在する場合、結果として旋回時間は大きく増加することが予想される。そのため、旋回順序を変えて実行したところ旋回時間に多少の差異が現れた。今後の検討事項として、近いホス

表 5.1: 実験結果

ホスト数	Time(m/sec)	Traffic(M/byte)
1	0.3315	8969
2	0.6632	9150
3	1.1018	9514

トから自動的に旋回していくアルゴリズムを提案し、旋回時間を減少させる必要がある。また同様に通信のトラフィックが少ない回線から旋回していくアルゴリズムも必要となるだろう。

Ethereal とは、ネットワークを流れるプロトコルをキャッチし、その内容を解析して表示する。多岐にわたるプロトコルに対応すると同時に、パケットを抽出するフィルタリング機能やパケットを見やすくするマーキング機能を備えるな ど、パケットをモニタリングするために十分な機能を備えたソフトである

第 6章

まとめ

6.1　展望

セキュリティ対策 セキュリティ対策は、モバイルエージェント普及のための重要な課題である。モバイルエージェントのためのセキュリティ対策は、数多く提案されている。セキュリティ確保は無視できない問題である。特に悪意のあるホストによるモバイルエージェントへの攻撃は、大きな問題である。

障害時の信頼性 ネットワーク社会の進展に伴って問題となってくるのが、ネットワーク内に分散した大量の情報の共有化、ネットワークを介したシステムの連携、ユーザ個人に合わせたサービスの提供、サービスの所在と使用方法の明確化・標準化等である。こうした問題への取り組みにおいて、モバイルエージェントという枠組みは非常に適した枠組みではないかと思う。モバイルエージェントの今後の課題としては、開発方法の確立と支援環境の改善、柔軟性と信頼性・安全性、プログラム自体が移動することから必須となってくる高度なセキュリティの確保、異なる環境下でも処理を行い得るような異なるモバイルエージェント間の相互運用性の実現、等がある。モバイルエージェントが次代のネットワーク社会の、オブジェクト指向に次ぐ概念として広まっていくことが望まれる。

6.2　むすび

本研究は、新しい経路監視システムとして、モバイルエージェントを用いた経路監視システムを提案・構築し、その有効性を検討した。しかし、今回の研究では測定ホスト数が

学内ネットワーク内での近いホストのみの実験であったため、旋回時間は直線的になったが、遠隔にあるホストが存在する場合、多くの旋回時間がかかると予想される。今後の課題は、モバイルエージェントの自動旋回のアルゴリズムの検討である。

参考文献

[1] DANNY B.LANGE, “Programming And Deploying Java Mobile Agents With Aglets”,ADDISON-WESLEY,August 1998