防火壁などによるエージェント移動の阻害を回避する移動エージェントシステムの構築*

田苗見 理紗　税田 竜一　冨永 和人

東京工科大学 工学部 情報工学科

概 要

インターネットが社会的に広く利用されるようになるにつれ，安全性などの観点から各組織のネットワークは防火壁を持つようになった．このため，移動エージェントシステムにおけるエージェントの移動が防火壁によって妨げられる場合が増えてきた．また携帯型の計算機などがインターネットの構成要素として加わったことにより，このようなホストで移動エージェントを用いる際には，ホストがオフラインである場合を考慮しなくてはならない．我々はオブジェクト指向スクリプト言語である Ruby に基づく移動エージェントシステム Rubiret を開発してきた．本研究では，エージェントのプログラマ及び利用者にとって透過的にこれらの問題を解決することを目的とし，システムを設計し，実現した．具体的には，防火壁に囲まれていないインターネット上にエージェントが一時的に待機できる場所を置くこと，またエージェント移動を HTTP によって行なうという手法を用いた．これらの対処により，上に挙げたエージェント移動の問題が緩和された．

1　はじめに

インターネットが高性能かつ普遍的になるにつれて，それを有効に利用するシステム構築パラダイムが考案され利用されてきた．単なるデータ通信から始まり，遠隔手続き呼出し，クライアント／サーバモデル，ピアツーピアモデルなど，いずれもが現在適材適所に利用されている．

移動エージェントもそのようなシステム構築パラダイムのひとつであり，これまで数多くの移動エージェントシステムが開発されてきた．そしてそれらのシステムを土台として情報検索や計画支援などといったさまざまな応用が提案および実用化されている [12]．

その一方，インターネットが社会的に広く利用されるようになるにつれ，安全性などの観点から各組織のネットワークは防火壁を持つようになり，現在ではインターネット上の全てのホストが他の全てのホストと任意の通信ができるという環境ではなくなっている．移動エージェントパラダイムの本質は，プログラムを持つエージェントの実体がホストからホストへ移動できることにあるが，その移動が防火壁によって妨げられることになる．さらに，インターネットに参加する機器として，従来のような固定型の計算機のみではなく，携帯型計算機や個人用携帯情報端末 (PDA) が増えてきた．これらのホストはネットワークに常時接続しているわけではないため，移動エージェントの移動時に問題となる場合がある．しかしながら多くの移動エージェントシステムは，エージェントの移動がこれらのような要因で妨げられないことを仮定している．

我々は移動エージェントシステム Rubiret を開発してきた [15]．このシステムは，エージェント記述言語としてオブジェクト指向スクリプト言語である

*Constructing a mobile agent system coping with obstructions to agent migration caused by firewalls and offline hosts

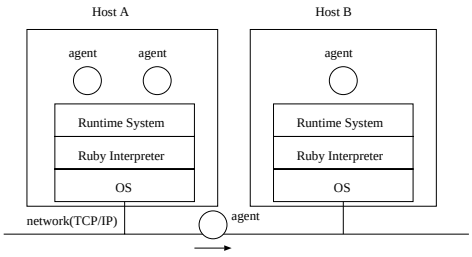


図 1: Rubiret システムの構成

Ruby を採用している．スクリプト言語を採用することにより，利用者が簡単にエージェントのプログラムを書いてそのエージェントを利用できるというシステムを目指している．そのため前述の問題は利用者の負担を増やさずに解決する必要がある．

そこで我々は，エージェントのプログラマ及び利用者にとって透過的な，これらの問題を緩和する枠組みを設計し，それを利用したシステムを開発中である．本稿では問題の解決方針と枠組み，およびシステムの設計と実装について述べる．

2 移動エージェントシステム Rubiret

移動エージェントシステム Rubiret（以下 Rubiret システム）はオブジェクト指向スクリプト言語 Ruby によって実装された移動エージェントシステムである．Rubiret システムにおけるエージェントの移動には，コードとデータを移動させるがスタックやプログラムカウンタなどの実行状態は移動させない弱いマイグレーションを採用している．Rubiret システムはエージェントとエージェントの動作プラットフォームとなるランタイムシステムからなる．

Rubiret システムではエージェントが活動するホストで，あらかじめランタイムシステムを動作させておく．エージェントはランタイムシステムを利用し，必要に応じて複数のホスト間を移動しながら与えられた処理を実行していく（図 1）．

利用者が Rubiret システムのエージェントを作成する際には，システムに附属の Rubiret クラスを継承してプログラムを記述する．このクラスには移動エージェントの基本的な機能が与えられている．

さらに Rubiret クラスには Ruby のプログラムで使われる標準的なメソッドと同じ名前のメソッドが，エージェント用に再定義されている．したがって利用者はエージェントのプログラムを Ruby のプログラムを書くのと同じ感覚で作成できる．例えば Ruby の組み込みクラス Kernel が持つ `puts` メソッドは端末（標準出力）に文字列を表示するのに以下のように用いられる．

Ruby 標準の `puts`

```
puts "Hello, Ruby!!"
```

Rubiret クラスはこのメソッドを再定義し，エージェント固有の標準出力ウィンドウに文字列を表示するメソッドとしている．リスト 1 のプログラムを実行すると，標準出力ウィンドウに「`Hello, Ruby!!`」と表示される．

リスト 1: Rubiret クラスの `puts` の利用

```
class SampleAgent < Rubiret
  def run
      puts "Hello,␣Ruby!!"
  end
end
```

Rubiret システムでエージェントを移動させるには，Rubiret クラスに定義されている `moveto` メソッドに移動先ホスト名を指定する．例えばホスト aaa.rubiret.net にいるエージェントが bbb.rubiret.net に移動するためのコードはリスト 2 のようになる．

リスト 2: エージェントの移動

```
class MoveAgent < Rubiret
  def run
    onhost("aaa.rubiret.net") {
      moveto("bbb.rubiret.net")
    }
  end
end
```

3　エージェント移動における通信路に関する問題と解決方針

移動エージェントシステムを実際のインターネット上で利用する場合，エージェントの移動に関して以下のふたつが大きな問題となる．

ひとつめの問題はホストがオフラインになる問題である．仕事などで使用される携帯型計算機や PDA などの端末は頻繁に停止／起動されたり，またネットワークから離れることも多い．このようなホストに向かってエージェントが移動しようとした場合の処理が必要である．

ふたつめの問題は防火壁の存在である．防火壁は急増する不正アクセスやコンピュータウイルスによる被害などから組織のネットワークを守るために導入されることが一般的になっている．しかしながら移動エージェントは現在のところ電子メールや WWW と比較するとまだ普及しておらず，移動エージェントのための通信は防火壁で妨げられることが多い．

これらの問題に対し，我々は以下の方針をとる．

1. エージェント待機所の設置
   ホストのオフラインや防火壁の制限によってエージェントが移動できない場合にそのエージェントが待つことができる一時的な待機所を置く．

2. HTTP の利用
   一般的に広く使われ，かつ防火壁で通信を制限していることが少ないプロトコル HTTP を利用し，エージェントを移動させる．さらにこれによって HTTP プロキシサーバの使用が可能となり，防火壁が通信路上にある場合でも外部との通信できる可能性が増える．

続くふたつの節ではこれらの実現について述べる．

4　エージェント待機所

エージェント待機所とは送信先のランタイムシステムがオフラインであったり防火壁の内側にあるなどしてエージェントを送ることができない場合に，そのエージェントを一時的に保管するための場所である（図 2）．保管されたエージェントは送信先ホストからの送信要求があった時に送り出される．したがって各ランタイムシステムは起動時にエージェント待機所に自分宛のエージェントがないか問い合わせる．また，その後も一定時間毎に問い合わせを行なう（自分が防火壁の内側にある場合）．エージェント待機所も，預っているエージェントの宛先ホストに対して，一定時間毎に送信を試みる．

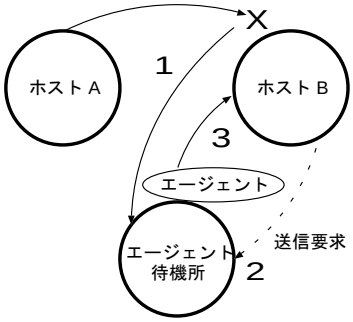


図 2: エージェント待機所を利用したエージェントの移動

図 2 においてホスト A がホスト B へエージェントを送信する手順を以下に示す．

1. ホスト A はまずホスト B へエージェントを送信しようとする．しかしこの時に B はオフラインであった．その場合に A はエージェント待機所にエージェントを送信する．

2. ホスト B はオンラインになるとエージェント待機所に問い合わせを行なう．

3. エージェント待機所はホスト A から預ったホスト B 宛てのエージェントを問い合わせへの返答として B へ送信する．

エージェント待機所は全てのランタイムシステムで共通とする．各ランタイムシステムの設定ファイルにエージェント待機所の位置（URI で指定される）を記述する．なお，エージェント待機所が停電や故障などのために利用できなくなる場合を考え，複数のエージェント待機所を用意できるようにする．こ

の場合，各ランタイムシステムはそれら全てのエージェント待機所の URI を保持する．

エージェント待機所には記憶容量の制限を設け，それを超える要求があった場合にはエージェントを受け取らずにエラーを返す．

5 HTTP によるエージェント移動

エージェント待機所を導入したため，Rubiret システムでのエージェントの移動形態には 2 通りがある．ひとつはあるホストが相手ホストへエージェントを送る場合である（場合 1 とする）．この場合には要求メッセージの中にエージェント本体を埋め込む必要がある．もうひとつはエージェント待機所に問い合わせをしてエージェントを受け取る場合である（場合 2 とする）．この場合，応答メッセージにエージェント本体が入ることになる．

HTTP メソッドにはいくつかあるが，データの送信に使用されるメソッドは通常 `POST` であり，データの受信に使われるメソッドは `GET` である．よってエージェントの送信（場合 1）には `POST` メソッドを用い，エージェント待機所へのエージェント送信要求（場合 2）には `GET` メソッドを使用する．

まず場合 1 の動作手順を図 3 に示す．これはホスト A からホスト B へエージェントを送信する場合である．

1. ホスト A が `POST` メソッドを用いてホスト B へエージェントを送信する
2. ホスト B は受け取った旨を `POST` メソッドへの応答として返信する．そしてホスト B は受け取ったエージェントを動作させる

この場合の要求メッセージの一例をリスト 3 に，それに対する応答メッセージの例をリスト 4 に示す．ここで送信元はホスト aaa.rubiret.net，送信先は bbb.rubiret.net である．

リスト 3: 要求メッセージ

```
POST /bbb.rubiret.net/Rubiret/1.1␣HTTP
    /1.1
Host: bbb.rubiret.net: 7007
User-Agent: Rubiret/1.1
Accept:␣text/plain
Content-Length: 1063
Content-Type: text/plain
Connection:␣close

<rubiret>
［テキスト化されたエージェント本体］
</rubiret>
```

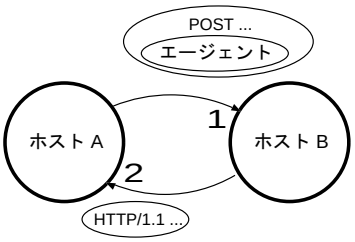


図 3: 動作手順（通常の動作)

リスト 4: 応答メッセージ

```
HTTP/1.1␣202␣Accepted
Date:␣Wed,␣30␣Sep␣2004␣15:21:33␣GMT
Server:␣Rubiret/1.1
Content-Length: 10
Content-Type: text/plain

Accepted
```

次にエージェント待機所を利用した場合（場合 2）の動作手順を示す．この場合の動作は図 4 に示すようになる．なおメッセージリスト中では，ホスト A が aaa.rubiret.net，ホスト B が bbb.rubiret.net，エージェント待機所は relay.rubiret.net となっている．

1. ホスト A がホスト B にエージェントを送信しようとするが失敗する．そしてホスト A はエージェント待機所へエージェントを送信する．この時は `POST` メソッドを用いる．送信メッセージをリスト 5 に，その応答をリスト 6 に示す．
2. ホスト B は起動時から一定時間間隔でエージェント待機所に `GET` メソッドを用いて問い合わせを行なう．問い合わせメッセージをリスト 7 に示す．

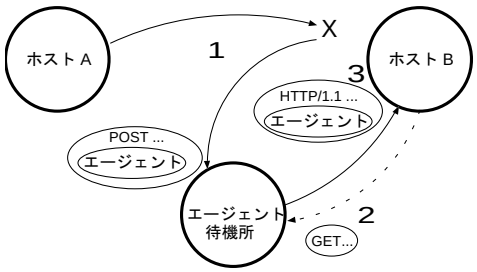


図 4: エージェント待機所を併用した際の動作手順

3. エージェント待機所は問い合わせに対し，応答としてエージェントをホスト B に返す
   この時の応答メッセージ例をリスト 8 に示す．

リスト 5: エージェント待機所へのエージェントの送信

```
POST /bbb.rubiret.net/Rubiret/1.1␣HTTP
    /1.1
Host: relay.rubiret.net: 7007
User-Agent: Rubiret/1.1
Accept:␣text/plain
Content-Length: 1063
Content-Type: text/plain
Connection:␣close

<rubiret>
［テキスト化されたエージェント本体］
</rubiret>
```

リスト 6: エージェント待機所からの受付応答

```
HTTP/1.1␣299␣Agent␣pool
Date:␣Wed,␣30␣Sep␣2004␣15:21:33␣GMT
Server:␣Rubiret/1.1
Content-Length: 12
Content-Type: text/plain

Agent␣pool
```

リスト 7: エージェント待機所への問い合わせ

```
GET /bbb.rubiret.net/Rubiret/1.1␣HTTP/
    1.1
Host: relay.rubiret.net: 7007
User-Agent: Rubiret/1.1
Accept:␣text/plain
Content-Length: 0
Content-Type: text/plain
Connection:␣close
```

リスト 8: エージェント待機所からのエージェントの送信

```
HTTP/1.1␣200␣OK
Date:␣Wed,␣30␣Sep␣2004␣15:21:33␣GMT
Server:␣Rubiret/1.1
Content-Length: 1063
Content-Type: text/plain

<rubiret>
［テキスト化されたエージェント本体］
</rubiret>
```

以上の通信は通常の HTTP を用いているため，HTTP プロキシサーバを経由したエージェントの移動も可能である．ただしその場合には要求 URI を絶対アドレスにする必要がある．例えば図 4 においてホスト A (aaa.rubiret.net) からエージェント待機所 (relay.rubiret.net) へのメッセージをプロキシサーバを経由して送る場合には，要求 URI を

```
http://relay.rubiret.net:7007/bbb.
rubiret.net/Rubiret/1.1
```

とする．

6 動作画面例

エージェントの動作画面の例を，簡単な通信を行なうエージェントによって示す．このエージェントは，指定したホストへ文字列を持って移動し，到着したら表示する．図 5 (a) 及び (b) に送信元ホスト（名前は pinokio.dyndns.ws）の画面を示した．図 5 (a) が送信直前の状態である．エージェントはこれからホスト yosaku.mine.nu に文字列「こんにちは」を持って移動しようとする．この時点で受信側ホスト yosaku.mine.nu は動作していない．よって送信元ホストはエージェント待機所にエージェントを送る．その直後の画面が 図 5 (b) である．右下にあったエージェントのウィンドウが消えており，画面上部にあるランタイムのログ画面に，エージェン

ト待機所 pc5.hachioji.wide.ad.jp に移動したことを表わすメッセージが示されている．受信側ホストを起動し，ランタイムを立ち上げると，ランタイムはエージェント待機所から自分宛のエージェントを受け取る．エージェントは持っている文字列を表示する（図 5 (c)）．なおこの例ではウィンドウを見やすくするために，位置を手で調整してある．

7　関連研究と議論

利用者の端末がオフラインになる問題に対しては主に以下のふたつの解決方針が採られる．

そのひとつは，オフラインになる端末と常時接続のホストの役割を分けるやりかたである．具体的には，端末にエージェントによる計算結果を利用者に提供する機能を主として持たせ，ランタイムシステムを持つホストと区別する．例えば MAP [9] では MAP サーバという常時接続ホストがエージェントのランタイム機能を提供し，利用者の端末は基本的にエージェントによる処理結果を利用者が受け取るために機能する [10]．TeaTray [13] では，エージェントは常時接続のポータルサーバから動作を開始する．利用者はクライアントを利用してポータルサーバに要求を送り，結果を受け取る．これにより利用者の移動や端末のオフライン化を支援している．Mobeet [11] も同様に，利用者にインタフェースを提供する端末（UIA が動作するホスト）と，その他の常時接続のホスト（エージェントプールがあるホストなど）を分けて扱っている．この解決方針は，利用者のホストでエージェントを作成し，そこからエージェントを動作させるという使用法を考えている我々のシステムには不向きである．また，MAP や TeaTray における実現方法では，利用者端末がオンラインになった場合の動作が利用者に透過的でない（MAP ではエージェントの再活性化の是非を問われる，TeaTray では明示的に結果をポータルサーバに受け取りにいく）ため，我々のシステムにとって望ましくない．

もうひとつは，エージェントが移動できない場合のためにエージェントを一時的に待機させておく場所を設けるという方針であり，本研究ではこの方針を採用している．既存のシステムの中では Agent Tcl [3] や Aglets [1] などがこの方針を採っている．Agent Tcl ではオフラインになる端末それぞれに対し，対応する常時接続のドック (dock) マシンを与える．ドックマシンは対応する端末がオフラインのときに到着したエージェントを保持し，端末がオンラインになった時に端末にそのエージェントを送ることができる．この動作はエージェントの利用者にとって透過的である．しかしながらこの方法を実現するには各端末に対応するドックマシンを設置する必要があり，それにはコストがかかる[1]．Aglets には，ドックマシンと同様の機能を提供するエージェント箱 (agent box) という概念がある．エージェント箱は端末が他の常時接続ホストに持つことができる私的なエージェントの待ち行列である．これに対して本研究のエージェント待機所はホスト間で公に共有される場所である点が異なる．

なお，これらのシステムのうちいくつかは利用者あるいは端末の移動を支援しているが，本研究のシステムは未対応である．今後は DNS などによってそれを支援することを考えている．

防火壁の問題に対する主たる解決法は，防火壁の内側から外側への接続確立のみを使ってエージェントを移動させる方法である．Aglets では，前述したエージェント箱と，エージェント移動プロトコルである ATP [4] の `dispatch` および `retract` 操作によって，防火壁を越えてエージェントを移動させられる [1]．これら 2 つの操作は，外向きの接続確立のみを用いて，内から外および外から内への両方のエージェント移動を可能にする．

この方法に加えて HTTP を用いることで，防火壁の問題をさらに緩和できる．HTTP は組織外（防火壁外）への要求に一般的に使われるプロトコルであるため，防火壁を越えた通信ができるように防火壁が設定されていることが期待できるからである．あるいは防火壁によって通信が妨げられない HTTP プ

[1] ひとつのホストが複数の端末のドックマシンになることは可能である．ただひとつのドックマシンが全ての端末を担当するという極限が我々の方法と見ることができる．

ロキシサーバを利用することができる可能性がある．Lingnau ら [5] はエージェントを送るのに HTTP の `POST` メソッドを，状態を要求するのに `GET` メソッドを使えることを示唆し，エージェントを HTTP によって送るための内容型 (content type) を提案している．前述した Aglets では，ATP パケットを HTTP でくるむことによってエージェントを送受信できる [7, 8]．aZIMAs [6] では WWW サーバ上に移動エージェントのランタイムシステムを乗せたものがひとつの単位であり，エージェントは単位の間を HTTP の `POST` メソッドによって送られる．我々のシステムも `POST` メソッドを用いるが，さらに定期的に `GET` メソッドを使った要求を行なうことで，エージェントが自発的に防火壁外から内へ移動することを支援している．Foukarakis らのシステム [2] も aZIMAs と同様に WWW サーバ上にランタイムシステムを乗せたものを一単位とするが，通信に HTTP 上の SOAP を使うところが異なる．

なおセキュリティに関しては，本システムはランタイム間の直接の通信を SSL で保護する機能と，ホスト上の資源へのアクセスをエージェントを生成して送り出した利用者（エージェントの所有者）に応じて制御する仕組みを持っている [14]．ただし現在のところ，利用者及びランタイムの認証は不完全であるので，今後公開鍵暗号系を用いた認証を導入する予定である．

8 まとめと今後

移動エージェントシステム Rubiret において，端末がオフラインになる問題や防火壁によって通信が阻害される問題の解決方法について述べた．具体的には，防火壁に囲まれていない公のインターネット上にエージェント待機所を置くこと，またエージェント移動を HTTP によって行なうことでそれらの問題に対処した．

今後はセキュリティを強化することに加え，利用者の移動を支援すること，視覚的なインタフェースを改善することによってシステムの有用性を高めたい．

参考文献

[1] Yariv Aridor and Mitsuru Oshima. Infrastructure for mobile agents: Requirements and design. In *Mobile Agents - Second International Workshop, MA '98*, Vol. 1477 of *Lecture Notes in Computer Science*, pp. 38–49. Springer, 1998.

[2] I. E. Foukarakis, A. I. Kostaridis, C. G. Biniaris, D. I. Kaklamani, and I. S. Venieris. Implementation of a mobile agent platform based on web services. In *Mobile Agents for Telecommunication Applications - 5th International Workshop, MATA 2003*, Vol. 2881 of *Lecture Notes in Computer Science*, pp. 190–199. Springer, 2003.

[3] Robert Gray, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. Mobile agents for mobile computing. Technical Report PCS-TR96-285, 1996.

[4] Banny B. Lange and Yariv Aridor. Agent Transfer Protocol – ATP/0.1, 1997. http://www.trl.ibm.com/aglets/atp/atp.htm.

[5] Anselm Lingnau, Oswald Drobnik, and Peter Dömel. An HTTP-based infrastructure for mobile agents. In *WWW Journal - 4th International WWW Conference Proceedings*, 1995.

[6] Amar Nalla, Abdelsalam (Sumi) Helal, and Vidya Renganarayanan. aZIMAs - almost Zero Infrastructure Mobile Agent System. In *Proceeding the IEEE Wireless Communications and Networking Conference*, 2002.

[7] Kouichi Ono, 2004. Personal communication.

[8] Mitsuru Oshima, Guenter Karjoth, and Kouichi Ono. Aglets specification 1.1 draft 0.65, 1998. http://www.trl.ibm.com/aglets/spec11.htm.

[9] Antonio Puliafito, Orazio Tomarchio, and Lorenzo Vita. MAP: Design and implementation of a mobile agents platform. *Journal of System Architecture*, Vol. 46, No. 2, pp. 145–162, 2000.

[10] Orazio Tomarchio, Lorenzo Vita, and Antonio Puliafito. Nomadic users' support in the MAP agent platform. In *Mobile Agents for Telecommunication Applications, Second International Workshop, MATA2000*, Vol. 1931 of *Lecture Notes in Computer Science*, pp. 233–241. Springer, 2000.

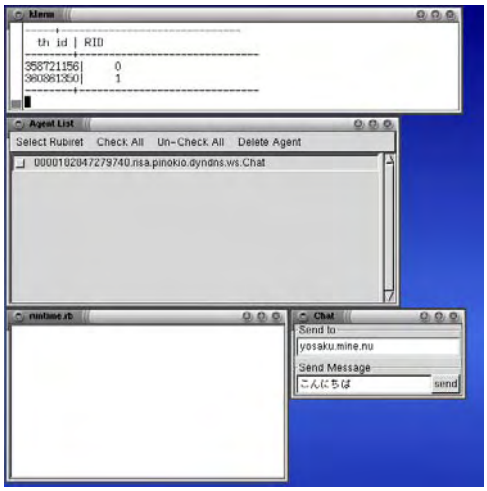
[11] Nobukazu Yoshioka, Akihiko Ohsuga, and Shinichi Honiden. A multi-agent framework for ubiquitous information systems: Mobeet framework. In *Proceeding the 6th International Bi-Conference Workshop on Agent Oriented Information System (AIOS-2004)*, pp. 103–118, 2004.

[12] 長尾確. エージェントテクノロジー最前線. 共立出版, 2000.

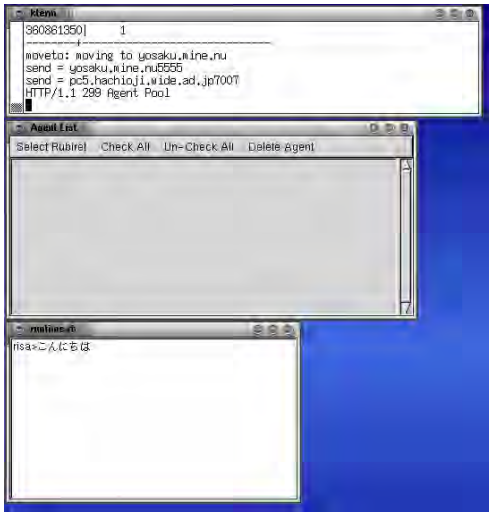
[13] 藤波香織, 長岡亨. MobileAgent による高機能情報サービスプラットフォームの提案. 第 61 回情報処理学会全国大会論文集, 2000.

[14] 田中秀明, 井上敦, 冨永和人. Ruby によるモバイルエージェントシステムの試作と計算機管理への応用. 日本ソフトウェア科学会第 19 回大会論文集, 2002.

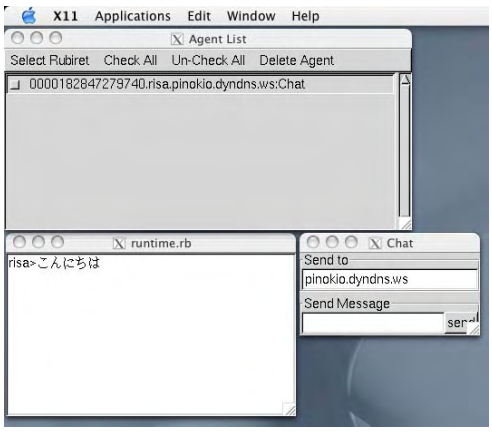
[15] 井上敦, 竹内祐一, 冨永和人. オブジェクト指向スクリプト言語によるモバイルエージェントシステムの実装とその応用. 電子情報通信学会技術研究報告, Vol. 102, No. 602, pp. 19–23, 2003.



(a) 送信元ホスト：送信直前の状態



(b) 送信元ホスト：送信後



(c) 受信側ホストがエージェントを受信した

図 5: 動作画面例