

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

Implementation of Prototype Mobile Agent System Using Ruby
and Its Application to Remote Host Management

田中 秀明 井上 敦 富永 和人

Hideaki Tanaka Atsushi Inoue Kazuto Tominaga

東京工科大学工学部情報工学科

Dept. of Information Technology, Tokyo University of Technology

tominaga@cc.teu.ac.jp

ネットワーク環境での計算機管理の方法としては、SNMP のように管理対象機器の情報を収集して人間に提示し、問題への対処は主として人間が行なうという方法に対して、管理対象機器でプログラムを動作させ、能動的に管理対象を操作する方法がある。本研究ではそのような計算機管理を支援するシステムの構築を目的として、オブジェクト指向プログラミング言語 Ruby によってモバイルエージェントシステムを試作した。ランタイムシステムは Ruby で実現した。エージェントのコードはユーザが Ruby で記述する。そして計算機を監視するエージェントを作成し動作実験を行なった。

1 はじめに

近年、計算機の急速な普及に伴ってネットワーク環境が整備され、多数の計算機がネットワークに繋がれるようになった。企業や大学などでは組織内に大規模なネットワークをもち、その管理による負担が大きくなっている。

従来のネットワークを通じた計算機管理手法には、管理対象の計算機に管理者がリモートログインして管理作業を行なう方法や、SNMP によって管理対象の計算機から管理情報を収集し、管理者へ計算機の動作状況等の表示、計算機の制御を行なう方法がある。しかし、これらの手法には以下のような問題点がある。

- 通信量が多い
計算機の動作状況を知るために、管理者は管理情報を管理対象の機器からネットワークを介して取得する。また、管理機器の設定等もネットワークを介して行なう。そのため通信量が大きくなる。
- 管理者の負担が大きい
問題への対処は収集した情報に基づいて管理者が行なうため、管理にかかる人間の作業量が大

きい。
最近ではこれらの問題に対処するため、モバイル

エージェント技術を用いて計算機管理を行なうシステムの研究が行なわれている [1][2]。モバイルエージェントとは計算機間をエージェントが自律的に移動し、移動先で情報の収集と判断を行なうことでユーザから与えられたタスクを実行するソフトウェアである。エージェントは、ユーザから与えられたタスクを実行するためのプログラムと、収集した情報などのデータをもつ。モバイルエージェントを計算機管理に応用すれば、エージェントに移動先での情報の収集と判断を行なわせることで、トラフィックを抑えることができる。また、エージェントに移動先での処理を任せることで、管理者の負担軽減や負荷分散がはかれる。

エージェントはオブジェクトと見なすことが自然であるため、オブジェクト指向の言語およびシステムが多く作成され用いられてきた。既存のモバイルエージェントシステムにおけるエージェント記述言語には、Java [5] や独自言語 [4] などがある。近年、新しいオブジェクト指向言語として Ruby[3] が注目されている。Ruby は Java に比べ、組み込みクラスや標準クラスで用いられている識別子が短く、変数宣言が不要で、改行を使って文を終えられるなどスクリプトの書き易さと読み易さに特徴がある。Ruby はファイルやディレクトリの操作からソケットによるネットワーク機能まで豊富なクラスライブラリを

持ち、数多くの UNIX 上や DOS, Windows 上でも動作する。本研究では Ruby を用いてモバイルエージェントシステムを試作し、計算機管理へ応用した。

2 モバイルエージェントシステム

本研究で作成したモバイルエージェントシステム (以下エージェントシステム) では、ユーザが各ホスト上にランタイムシステムを動作させておき、エージェントはランタイムシステムを利用し、計算機間の移動、計算機上での処理を行う (図 1)。

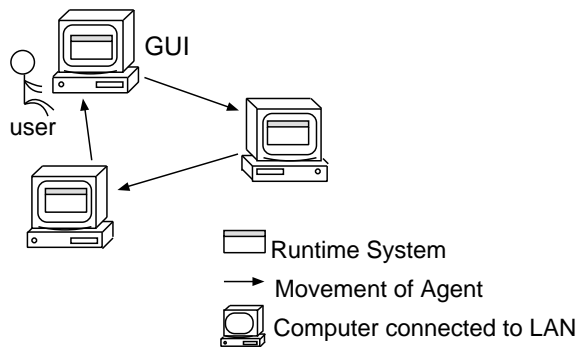


図 1: エージェントシステム

2.1 ランタイムシステムの機能と構成

ランタイムシステムは以下の機能を持つ。

- エージェント送受信
- エージェントの実行
- エージェントの停止
- ユーザインタフェース

ランタイムシステムの構成を図 2 に示す。ランタイムシステムを複数のスレッドで実現した。Receiver はエージェントを受信し、そのエージェントのプログラムを ProgramQueue に格納し、データを DataQueue に格納する。AgentContext は ProgramQueue からプログラムを取り出し実行する。ホスト上のリソースを利用する場合には RequestHandler を利用する (2.3 節で述べる)。実行を開始したエージェントは DataQueue からデータを取り出し、プログラムに基づく処理を行う。エージェントは処理を終えると、消滅するか移動する。移動する場合にはエージェントは実行状態を保存し、それを AgentContext が他のホストへプログラムと共に送信する。GUI はユーザ

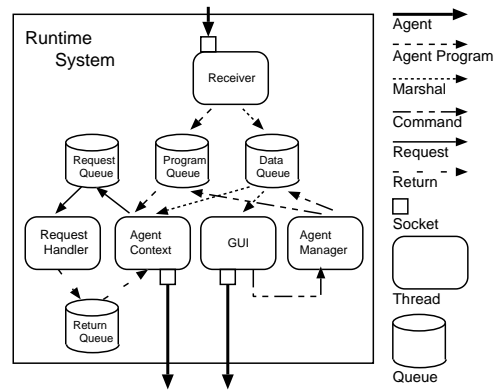


図 2: ランタイムシステムの構成

にエージェントシステムへのインタフェースを提供する。ユーザは GUI を利用してエージェントの送信や実行結果の取得、エージェントの停止などを行なう。エージェントの停止には AgentManager を用いる。AgentManager はユーザが指定したエージェントを AgentContext が実行する前に ProgramQueue から削除する。

2.2 エージェントの機能と構成

エージェントは Ruby により記述するため、さまざまな動作を行わせることができる。この動作は、2.3 節のランタイムシステムのセキュリティメカニズムによって制限される。

エージェントは、その振舞いを決めるプログラムと、実行状態などのデータから構成される。移動時には AgentContext は Ruby 組み込みモジュールの Marshal によりデータをシリアライズする。実行時には Marshal によりデータを復元する。

2.3 エージェントシステムのセキュリティ

モバイルエージェントシステムでは、プログラムとデータを計算機間で移動させるため、セキュリティ機能が必要である [6]。本エージェントシステムでは、以下をセキュリティ機能の目的とする。

- 通信路上のエージェントを盗聴者や改竄者から保護する
- 悪意のあるエージェントから他のエージェントを保護する
- 悪意のあるエージェントからランタイムシステムとホストを保護する

これらのセキュリティ要件を満たすため、このシステムに以下の機能を与えた。

- 通信路の暗号化
- エージェントへの電子署名と認証
- エージェントのホストへのアクセス制御

通信路の暗号化には Ruby の SSL Socket ライブラリを用いた。エージェントへの電子署名と認証には PGP (version 2.6.3ia) を使用した (図 3)。

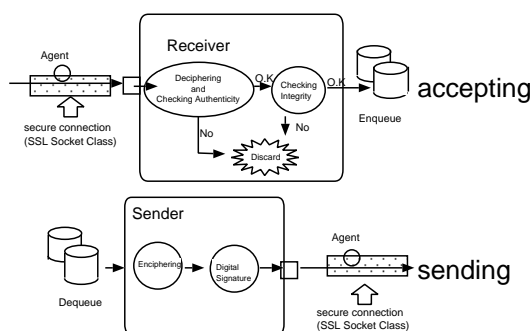


図 3: エージェントシステムの暗号化と認証のモデル

エージェントのホストへのアクセス制御は図 4 の仕組みで行なう。Ruby はスレッドごとに 5 段階のセーフレベルを設定でき、レベルに応じてそのスレッド内で行なうことのできる動作に制限を加えることができる。セーフレベルの設定にはスレッドローカルな \$SAFE 変数を使う。エージェントの実行を行なう AgentContext を \$SAFE=4 (信頼できないスクリプトの実行を行なうレベル) に設定し、エージェントの動作を制限する。これにより、エージェントはホスト上で自ら I/O を行なったり、外部コマンドを実行することはできない。

エージェントがホスト上のリソースを利用するには API オブジェクトを利用する。API オブジェクトには I/O を使うためのメソッドや、外部コマンドを使うためのメソッドなどを与えておく。例えば、エージェントがファイルをオープンするときには \$api.open(filename) を使う。\$api.open メソッドは、AccessController オブジェクトのメソッドを呼び出し、そのエージェントにそのファイルをオープンする権限があるか確認する。AccessController オブ

ジェクトは AgentInfoManager オブジェクトと PolicyFile からの情報によって許可もしくは不許可の判定を行ない、結果を返す。AgentInfoManager オブジェクトにはそのホスト上に存在するエージェントの認証情報を持たせ、PolicyFile にはエージェントの動作に関するセキュリティ要件を書いておく。その結果が許可の場合、\$api.open メソッドは Queue オブジェクトの requestq に request message を入れる。\$api.open メソッドは returnq に結果が返るまで待つ。RequestHandler は \$SAFE=0 (動作制限なし) のスレッドである。RequestHandler は requestq にある request message を取り出し、指定のファイルをオープンする。RequestHandler はオープンした File オブジェクトを returnq へ渡す。Queue オブジェクトは呼び出し元の \$api.open メソッドにその File オブジェクトを返す。\$api.open メソッドは File オブジェクトを呼び出したエージェントに返す。

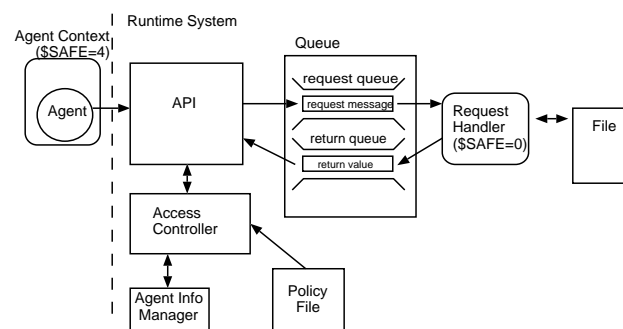


図 4: エージェントシステムのアクセス制御モデル

3 計算機管理システムへの応用

作成したエージェントシステムの応用として、LAN に接続している計算機の負荷の状態を監視する計算機管理システムを作成した。エージェントは、管理者が設定した監視順路にしたがって各計算機のロードアベレージを取得する。エージェントが取得したロードアベレージをランタイムシステムの GUI を用いて表示する。

3.1 監視エージェント

作成した監視エージェントは、監視順路のリストにしたがって計算機間を巡回する。監視エージェントは移動先で計算機のロードアベレージを取得してデータに加え、リスト中の次の計算機へ移動する。も

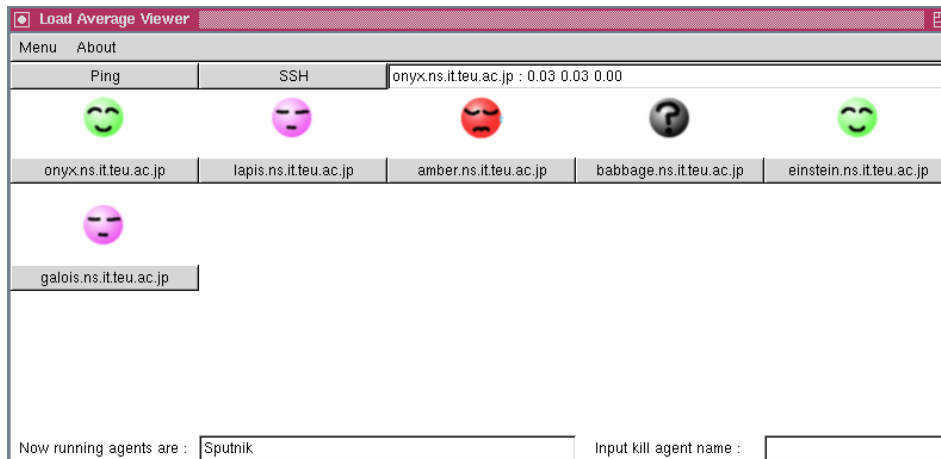


図 5: 計算機管理システムの GUI 画面

し、監視エージェントが移動先の計算機に移動できなかった場合には、その計算機に問題があると見なして、リスト中の次の計算機へ移動する。リスト中の最後の計算機で処理を終えた監視エージェントは再びリスト中の最初の計算機へもどる。

3.2 計算機管理システムの GUI

計算機管理システムの GUI では、監視エージェントの送信と停止、実行結果の表示を行なう。監視エージェントはウィンドウ上から管理者が送信ボタンを使って送信する。監視エージェントの停止をするには、管理者がウィンドウ上のテキストフィールドに停止させたいエージェント名を入力する。

3.3 実験と評価

作成した計算機管理システムの評価を行なうため、計算機管理システムの動作実験を行なった。実験は LAN に接続した FreeBSD マシンと Vine Linux マシンを使って行なった。図 5 はシステムの動作結果である。この表示からは管理対象の計算機の状態を知ることができる。マシン onyx のアイコンは緑色で表示されており、負荷が小さいことを表す。マシン amber のアイコンは赤色で表されており、負荷が大きいことを表す。また、マシン babbage は「?」のアイコンで表され、エージェントが babbage に移動できなかったことを示す。

本実験の監視エージェントは 40 行の Ruby スクリプトとして実現できた。このスクリプトには、エージェントが持ち運ぶデータを Marshal によって処理するコードや、エージェント自身の移動を処理する

コードが含まれている。これらのコードは共通処理としてエージェントクラスに与えるべきである。そうすることによってユーザが記述すべきスクリプトはより小さくなり、記述が容易になるだろう。

4 おわりに

本研究では、Ruby を用いてモバイルエージェントシステムを試作し、計算機管理への応用を行なった。監視対象の計算機をエージェントが巡回して各計算機のロードアベレージを取得し、それを管理者に表示するシステムを作成した。

今後の課題として、エージェントの動作を抽象的に記述する手段の提供や、監視対象計算機の異常発生を知ると自動的に管理者へ通報したり、自ら計算機の障害回復を行なうエージェントの作成などがある。

参考文献

- [1] 田中秀明: Ruby を使ったエージェントシステムの構築とその応用に関する研究, 東京工科大学卒業論文 (2002).
- [2] 小野木渡, 清水亮博, 大野浩之: ネットワークワームを利用したネットワーク管理手法, コンピュータソフトウェア, Vol16, No.3, pp.29-46 (1999)
- [3] まつもとゆきひろ, 石塚圭樹著: オブジェクト指向スクリプト言語 Ruby, アスキー出版局 (1999).
- [4] 東芝: 考えながら動き回るエージェント Plangent, <http://www.toshiba.co.jp/plangent/index.j.htm>
- [5] IBM: Aglets Software Development Kit Home Page, <http://www.trl.ibm.co.jp/aglets/index-j.html>
- [6] 小野康一, 大島満: 移動エージェント Aglets のセキュリティ・モデル, <http://www.brl.ntt.co.jp/itech/wit98/proceedings/ono/html/index.html>