共生・寄生モデルにおけるモーバイルエージェントに対するデバッガ

2H－2

矢崎 友久†, 飯島 正‡, 山本 喜一‡, 土居 範久‡
†慶應義塾大学大学院理工学研究科　‡慶應義塾大学理工学部

1　はじめに

モーバイルエージェントの振舞いはエージェント動作記述言語のプログラムで定義される．このことを考えた場合に問題となることの一つに，モーバイルエージェント特有の問題として，遠隔地で動作するプログラムのデバッグがある．リモートノードで動作しているエージェントの挙動を観察するためには，通常のデバッグツールとは違った仕組みが必要となる．

本研究では，ネットワーク上を動き回るエージェントの挙動を，特定のノードでユーザが観察・制御することを可能にするための機構を提案する．そのために，エージェントが階層構造を構築することができる共生・寄生モデル [1] を基礎モデルとして使用する．すなわち，デバッグ対象のエージェントに対し，その振舞いを検出する Monitor Agent を動的に起動させ，その情報をユーザのいるホストに位置する User Interface Agent へ通信させる．

2　対象システム

デバッガの構成を示す前に，その対象となるシステムについて概説する．はじめに，基礎モデルとして利用している共生・寄生モデルについて述べ，その後，共生・寄生モデルに基づくシステム上に構築したエージェントの内部構造について述べる．

2.1　共生・寄生モデル

共生・寄生モデル上において，概念的にエージェントは物理的なネットワーク構成とは独立したソフトウェアモジュールとして捉えられる．エージェント同士はメッセージを交換することにより，通信することができる．エージェントの中に複数のエージェントが入れ子状に寄生して，階層構造を形成することもある．

階層構造は，実際には図1のように，階層の順に通信ストリームがつながることによって実現される．また，構造の形成，再形成は，通信ストリームの付けかえ，またはエージェントのネットワーク上の移動によって達成される．

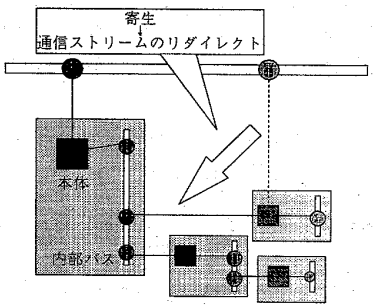

図 1: エージェントの概念アーキテクチャ

宿主エージェントは，寄生しているエージェントの通信を観察したり，フィルタリングしたりすることが可能である．

2.2　エージェントの内部構造

共生・寄生モデルの特徴を利用してデバッガを実現するために，エージェントの内部のインタプリタは，プログラムや変数を保持する部分，プログラムを実行する部分などを，それぞれサブエージェントとして寄生させているモーバイルエージェントとして構成した (図 2).

この4つのサブエージェントが協調して動作することにより，全体が1つのエージェントとして振舞うことができる．

エージェントの動作はインタプリタ言語によって記述されており，そのプログラムは Program Carrier Agent が保持している．Interpreter Agent はプログラムを実行するエージェントで，Program Carrier Agent からプログラムを渡されることにより動作する．プログラムは，テキストの形のままでメッセージの中に埋め込まれ，渡される．Variable Carrier Agent はプログラムの中の変数の値を保持している．File Handler Agent はファイルを操作する役割を持っている．

それぞれのサブエージェントは，エージェント間メッセージ通信によって必要な情報をやりとりする．例えば，Interpreter Agent がプログラムを実行している時に変数の値を更新する必要が出てきた場合，Interpreter Agent から Variable Carrier Agent に対して変数更新要求のメッセージが送られる．そのメッセージを受けとった Variable Carrier Agent は，自分の中の変数と値の対応表を更新する．

"A debugger for mobile agents on the Symboitic/Parasitic Model", Tomohisa Yazaki, Tadashi Iijima, Yoshikazu Yamamoto and Norihisa Doi, Keio University

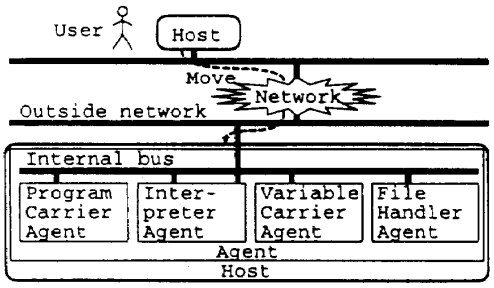

図 2: エージェントの構成 (通常時)

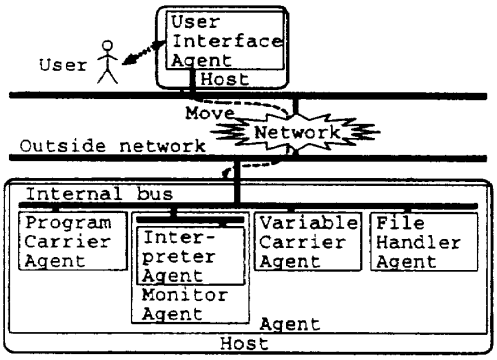

図 3: エージェントの構成 (デバッグ時)

3 デバッガの構成

図3に示すように，デバッガは 2 つの部分から構成される．一つは Monitor Agent で，デバッグ対象となるエージェントの内部に寄生して Interpreter Agent を自分の内部に取り込み，対象エージェントを観察する．もう一つは，User Interface Agent で，ユーザの指示を受けとり，Monitor Agent が手に入れた情報をユーザに対して提示する．

4 デバッガの動作

Monitor Agent は Interpreter Agent が送ったり受けとったりするメッセージを傍受することによって，その挙動を観察する．例えば，先ほど挙げたような変数更新のメッセージを傍受することにより，Monitor Agent はいつ変数が更新されたかを知ることができる．更新された変数が，ユーザによって監視したい変数として指定されたものであれば，この変数が更新されたという情報が Monitor Agent から User Interface Agent にメッセージによって伝えられる．それと同時に Interpreter Agent は動きを止め，ユーザからの指示を待つ．このようにして，変数の監視という，デバッガの機能の一つが実現される．

Monitor Agent の 2 つめの役割は，プログラムを改変し，行番号に関する情報をプログラム内に埋め込むことである．プログラムの改変は，Program Carrier Agent が Interpreter Agent にプログラムコードを渡そうとする時に起こる．その時 Monitor Agent はプログラムコードが含まれたメッセージを傍受する．そして，実行中のそれぞれの時点で何行目のコードを実行しているかを User Interface Agent に知らせるようなコードを，プログラムの中に埋め込んでから Interpreter Agent に渡す．この結果として出来上がったコードを Interpreter Agent が実行することにより，ユーザが指定した任意の行で実行を止めることができるようになる．

ユーザの指定した条件が満たされ，Interpreter Agent が動きを止めたとき，ユーザはインタプリタの状態を調べるだけでなく，プログラムコードや変数の値を変更することができる．これは Monitor Agent が Program Carrier Agent や Variable Carrier Agent に，変更依頼のメッセージを送ることで実現される．

5 Monitor Agent が寄生するパターン

Monitor Agent が寄生する方法は 3 通りある．1 つめは Interpreter Agent の起動時から寄生しているパターンで，デバッグ対象エージェントとデバッガを同時に起動することにより，このパターンの寄生が行なわれる．2 つめは すでに起動しているエージェントに対してデバッガを起動する (動的に寄生させる) パターンであり，この場合，Monitor Agent が遠隔地にいる対象エージェントのもとに派遣されることになる．3 つめは，Monitor Agent が待ち伏せをするパターンである．対象エージェントが Monitor Agent の待っているノードを訪問すると，Monitor Agent が寄生する．

3 つめの待ち伏せのパターンは，場所指定によるブレークポイントとみなすことができる．これはいままでデバッガにはなかった，モーバイルエージェントのデバッギングに特有な機能であり，移動経路が予測不能もしくは変わりやすいエージェントの挙動を観察するのに有用である．

6 まとめ

共生・寄生モデル上において，ネットワーク上を動き回るモーバイルエージェントを特定のノードで観察するための，デバッガを提案した．このようなツールは，実用化しつつあるモーバイルエージェントの開発において不可欠なものである．

また，場所指定によるブレークポイントの概念を示した．これは移動経路が予測不能もしくは変わりやすいエージェントの挙動を観察するのに有用である．

参考文献

[1] 飯島 正，山本 喜一，土居 範久: 「拡張可能なエージェントのための共生・寄生モデル」電子情報通信学会，信学技報 (KBSE-97-33), pp. 25-31(1998)