L-009

エージェント間の部分的な協調関係に着目した モバイルエージェントシステムのデバッグ手法の提案

Proposal of Debugging Focused on Partial Interaction of Agents for Mobile Agent System

太田垣 真也 † Shinya Otagaki
尾崎 槙 † Shin Osaki
東野 正幸 † Masayuki Higashino
高橋 健一 † Kenichi Takahashi
川村 尚生 † Takao Kawamura
菅原 一孔 † Kazunori Sugahara

1. はじめに

モバイルエージェントシステムは，エージェントと呼ばれる自律的なプログラムがネットワークで接続された計算機上の実行環境間を移動しながら協調して動作することで機能するシステムである．これを用いることでクライアント／サーバ方式のプログラム設計に必要な通信プロトコルの決定及び通信プログラムの記述が必要ないため，分散システムのプログラミングが平易となる．また，無線による通信で通信路の切断が余儀なくされる場合，切断される前にエージェントを通信先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．このような利点があるため，モバイルエージェントは分散システムの構築に有効な技術であり，実用化が望まれている．

このようなモバイルエージェントを用いたシステム開発において，デバッグは必要不可欠であるが，モバイルエージェントのデバッグの研究はあまり進んでいない．JADE[1] では，エージェントの協調関係をシーケンス図で確認することができるが，モバイルエージェントのような移動を伴う協調関係はサポートされていない．また，Aglets では実行環境とエージェントの状態を確認するためのツール Tahiti[2] が開発されている．しかし，Tahiti では，エージェントの情報を個別に確認することができるが，どのように協調しているかを確認することはできない．

モバイルエージェントシステムの設計では，複数のエージェント間の協調関係を定義することでシステムの動作を定義していく．このため，エージェント間の協調関係を把握することにより，システムの動作を知ることができる．このことから，モバイルエージェントシステムをデバッグするためには，複数のエージェントがどのような協調関係を持ち動作しているかを把握する必要がある．しかし，モバイルエージェントシステムの規模が大きい場合，計算機やエージェントの数が増大し，協調関係が複雑化する．このため，エージェントの協調関係を把握することは難しくなる．また，通信遅延や通信帯域の不足により，広範囲に渡る協調関係の把握は難しい．さらに，運用中の大規模なモバイルエージェントシステムのデバッグを行うことを考えると，システム全体の中断・再開を行うことは大きなコストを伴う．

†鳥取大学大学院 工学研究科 情報エレクトロニクス専攻

そこで本稿ではモバイルエージェントシステムの協調関係の把握が難しいことによるデバッグの困難性の問題を解決するために，協調関係全体ではなく，協調関係を部分的に確認し，協調関係を追跡しながら確認範囲を移動させていくデバッグ手法を提案する．

2. モバイルエージェントのモデル化

モバイルエージェントによって構成されたシステムにとって，バグの原因となるのは異常な状態となったエージェントだけとは限らない．異常な状態となったエージェントと協調して動作していたエージェントがバグの原因である場合も多い．そこで，モバイルエージェントシステムの動作の流れにおいて関わり得るオブジェクトの関係性について考察する．

2.1. モバイルエージェントのモデル

エージェントの協調関係を追跡するために，MASIF[3] や FIPA[4] を基にエージェントのモデルを検討する．

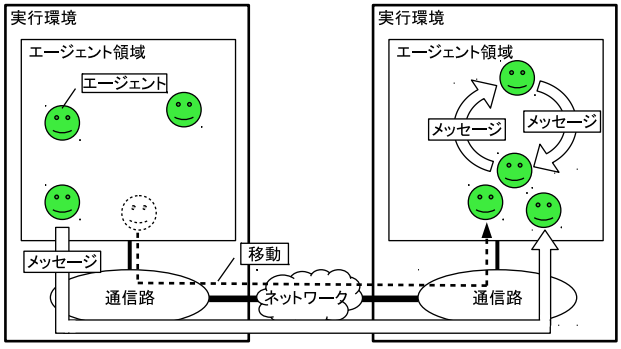

図 1: モバイルエージェントのモデル

モバイルエージェントシステムの動作の流れに関わるオブジェクトとしては以下のものがある．

- **エージェント**
  メッセージを解釈して自律的に判断し処理を行うプログラムである．また，エージェントは，自身のプログラムコードと実行状態を持ち，これらを他の実行環境に転送することで移動を行う．
- **エージェント領域**
  エージェントが存在するための場所である．エージェントは，エージェント領域において処理の実行

や他のエージェントとのメッセージのやり取りを行う．モバイルエージェントシステムにおいて，エージェントが処理を実行するために必要である．このエージェント領域は，MASIF では Agent System にあたり，FIPA では Agent Platform と Agent Mnagment System にあたる．

- **通信路**
  エージェントを他の実行環境上のエージェント領域へ移動，また，他の実行環境のエージェント領域に存在するエージェント同士がメッセージの交換を行うための機能を提供する．この通信路は，MASIF では ORB にあたり，FIPA では Agent Communication Channel にあたる．

- **実行環境**
  エージェントを動作させるためのサーバプログラムである．エージェント領域と通信路を持つ．

- **メッセージ**
  メッセージは，エージェント間でやり取りされる命令である．エージェントが他のエージェントと協調するためやデータのやり取りをするために必要である．

2.2. エージェントの移動

モバイルエージェントシステムのデバッグを難しくする要因の一つにエージェントの移動がある．エージェントの移動とは，エージェントが他の実行環境上のエージェント領域に移動することである．他のエージェントと協調して動作するエージェントが移動することにより，デバッグに必要な協調関係が複雑になり，デバッグが難しくなる．そこで，モバイルエージェントシステムの動作の流れにおけるエージェントの移動を定義する．これにより，デバッグのために協調関係を把握する際に考慮すべきエージェントの移動がどういうものかを明確にする．

2.3. 協調関係の追跡

提案するデバッグ手法において，エージェントの協調関係を追跡する必要がある．この協調関係とは，エージェント同士の協調して動作する互いの関係性のことであり，図 1 におけるエージェント同士のメッセージのやり取りが行われた関係のことである．協調関係の広がり方には，同じ実行環境上での広がりと，別の実行環境への広がりの二種類がある．これはあるエージェントが協調して動作する場合，同じ実行環境上の別のエージェントとメッセージのやり取りを行う場合と，別の実行環境に滞在しているエージェントとメッセージのやり取りを行う場合があるということである．別の実行環境上のエージェントとのメッセージのやり取りの際には，滞在している実行環境から別の実行環境上のエージェントとメッセージをやり取りする場合と，相手のエージェントが滞在している実行環境に移動してメッセージのやり取りを行う場合がある．つまり協調関係の追跡とは，エージェントがメッセージのやり取りを行った相手が同じ実行環境上に存在するか，別の実行環境上に存在するか判断し，その処理の相手であるエージェンを見つけ出していくということである．これにより，異常な状態となったエージェントが発生した処理において関わりのあるエージェントを見つけ出し，追跡していくことができる．

3. 提案手法

モバイルエージェントシステムのデバッグの支援のために，異常な状態となったエージェントからそのエージェントの協調関係を追跡し，協調関係の把握する範囲を徐々に拡大していくことで，協調関係を持つエージェントのバグの原因部分の特定を支援するデバッグ手法について提案する．

3.1. 協調関係の追跡によるデバッグ手法

モバイルエージェントシステムは，複数のエージェント間の協調関係を定義しシステムの動作を決定して構築される．このため，エージェント間の協調関係を把握することによりシステムの動作を把握することができ，デバッグの際のバグの原因の特定の支援を行うことができる．しかし，複雑なエージェントの協調関係を把握することは困難である．そのため，異常な状態となっているエージェントを見つけ，まずその単体のエージェントだけの狭い範囲の協調関係を把握し，バグの原因が隠れている協調関係を把握する．この時把握する協調関係とは，エージェント同士のメッセージのやり取りである．最初に異常な状態となったエージェントを調べ，バグの原因が見つからなければ，メッセージのやり取りにより異常な状態を引き起こしたデータがもたらされたことになる．そこで，最初に異常な状態となったエージェントと協調して動作するエージェントをデバッグの対象とし，バグの原因がないか調べる．

ここで，Master-Slave モデルを例に協調関係の追跡を説明する．Master-Slave モデルは，Master エージェントが Slave エージェントを一つ以上生成し，Slave エージェントが他の実行環境に移動して行った処理の結果を元いた実行環境に戻って Master エージェントに報告するというモデルである．Slave エージェントがさらに Slave エージェントを生成して Master エージェントとなることで，階層的に協調関係が構築される場合もある．

図 2 では，Master エージェントが Slave エージェント Slave1，Slave2 を生成し，Slave エージェントはそれぞれ別の実行環境に移動し動作する．また，Slave1 は Slave1-1，Slave1-2 を，Slave2 は Slave2-1，Slave2-2，Slave2-3 を生成し，この時生成された Slave エージェントはそれぞれ別の実行環境に移動し動作する．それぞれの処理が終わると，Slave エージェントはそれぞれの生成元に結果を報告する．この時，Master エージェントに異常が発見されたとする．

この時，まず，異常な状態となった Master 自体にバグの原因がないかを調べる．これは，エージェント単体が最小の協調動作の関係性の範囲だからである．Master 自体にバグの原因がない場合，Master がメッセージのやり取りを行い協調関係を持っているエージェントを調べる．この場合は，生成と結果の報告というやり取りのある Slave1，Slave2 が Master と協調関係を持つ

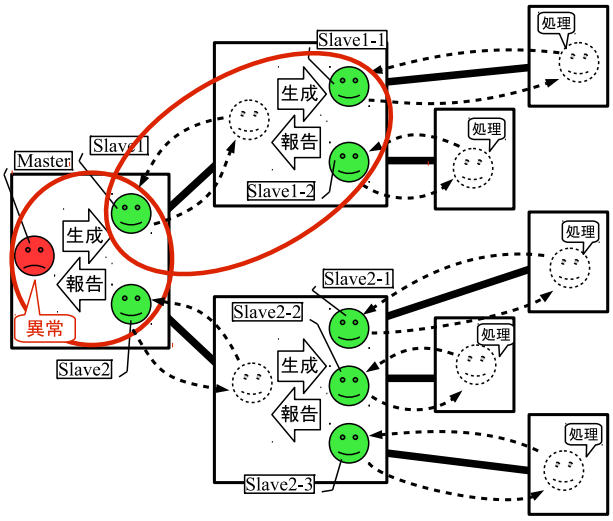

図 2: 協調関係の追跡

エージェントであり，バグの原因を調べるために追跡する協調関係を持つエージェントである．Slave1，Slave2 を調べ，バグの原因を持つエージェントがいない場合は，さらに調べる範囲を移動する．この時，バグの原因ではないが手掛かりが見つかった場合は，その手掛かりを見つけたエージェントを優先的に調べていく．このように把握する協調関係の範囲を移動していき，その範囲内のエージェントを調べることで，協調関係を持つエージェントのデバッグの手間を減らすことができる．

4. デバッガの設計

デバッグ手法を実現するには，協調関係の追跡機能の実現が求められる．この協調関係の追跡機能によりバグの原因の特定の支援を行うためには，追跡の取っ掛かりとなる異常な状態のエージェントを見つける必要がある．この異常な状態のエージェントを発見するための仕組みとして，エージェントの検索機能が考えられる．エージェントの実行状態の内，異常な状態といえる実行状態を条件としてモバイルエージェントシステムに対して検索をかける．これにより，協調関係の追跡機能を用いるために必要な取っ掛かりとなる異常な状態のエージェントを見つける．

このエージェントの検索機能によって見つけた異常な状態のエージェントに対し協調関係の追跡機能を用い，バグの原因の特定の支援を行う．

4.1. 協調関係の追跡によるデバッグ手法

エージェントの協調関係の追跡機能を実現するために，各実行環境でエージェントの動作を監視しログを採取する．採取するログの内容は，各エージェントの他のエージェントとのメッセージのやり取りである．このログを元に協調関係を部分的に確認し，協調関係を追跡しながら確認する範囲を移動する際に利用する．メッセージのやり取りには，データの受け渡しや処理の依頼だけでなく，エージェントの生成や削除等も含まれる．そのため，一連の処理の途中で削除されたエージェントがいてもメッセージのやり取りのログを頼りにその削除されたエージェントを特定し，そのエージェントにバグの原因がないかを調べることができる．

また，全てのエージェントがそれぞれ一つの処理のために生成されるのではなく，複数の処理に関わるエージェントも存在する．協調関係を追跡している途中で，このような複数の処理に関わるエージェントと協調して動作していた場合，その複数の処理に必要とされるエージェントは多くのエージェントと協調関係を持つ．そのため，確認する協調関係の範囲が大きく広がり，バグの原因の特定が難しくなる．そこで，メッセージのやり取りのログを取る際に処理毎の ID を付与しておく．これにより，協調関係の追跡の際に協調関係の中に複数の処理に関わるエージェントが含まれていても，バグの原因がないか調べるエージェントの数が大きく増加することを防ぐ．

このようにシステムの動作時に採取したメッセージのやり取りのログを頼りに確認する協調関係を追跡し，確認範囲を移動していくことで，バグの原因の特定を支援し，モバイルエージェントシステムのデバッグの手間を緩和する．

5. おわりに

モバイルエージェントシステムの協調関係の把握が難しいことによるデバッグの困難性の問題を解決するために，協調関係を部分的に把握し，協調関係を追跡しながら把握する範囲を移動させていくデバッグ手法を提案した．今後は，各機能を更に詳細に検討するために，モバイルエージェントのモデルにおいて，どういった内部状態やコンテキストを持つかということを定義する必要がある．また，各機能を実装し評価する．

参考文献

[1] JADE: Java Agent DEvelopment Framework. http://jade.tilab.com/.

[2] Tahiti. http://www.research.ibm.com/trl/aglets/tahiti/tahiti.htm.

[3] MASIF. http://www.omg.org/.

[4] FIPA. http://www.fipa.org/.