L-010

モバイルエージェントシステムのためのリモートデバッガの検討
A Study of Remote Debugger for Mobile Agent System

尾崎 槙 † Shin Osaki
太田垣 真也 † Shinya Otagaki
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とは，エージェントと呼ばれる自律的なプログラムが，ネットワークに接続されたエージェント実行環境であるノードを移動しながら問題を解決していくシステムである．モバイルエージェントシステムは分散システムの構築に有効な技術であり，様々な研究が行われてきた [1][2]．しかし，モバイルエージェントシステムを用いたシステムが一般的に普及しているとは言い難い．その原因の 1 つとして開発におけるデバッグの困難性が挙げられる．

プログラムをデバッグする際にはエラーの原因を特定するために実行中のプログラムの状態を知る必要がある．このため，一般的にはデバッガのブレークポイント機能を用いて，プログラムを一時停止させて状態を確認することでデバッグを行う．このブレークポイント機能をモバイルエージェントシステムで用いる場合，エージェントは遠隔地のノードで動作しているため，リモートデバッグ機能を用いる必要がある．ところが，一般的なリモートデバッグ機能はリモート接続先が頻繁に変わることを想定していないため，ノード間を頻繁に移動するエージェントに対応することができない．また，分散システムのデバッグの手法として，分散システムに対するブレークポイント機能がすでに提案されている [3]．しかし，モバイルエージェントのように遠隔地の実行環境を移動するようなプログラムに対して利用可能な機能は実現されていない．

そこで本研究ではモバイルエージェントシステムにおけるデバッグの困難性を解決するために，モバイルエージェントの移動に合わせて動的にリモート接続先を変更することで, モバイルエージェントの移動に対応したデバッグ機能を開発する．

2 モバイルエージェントシステムのデバッグの問題点

モバイルエージェントシステムは，複数のエージェントが多数のノードに分散してノード間を移動しながら動作する特徴を持つ．この特徴により，クライアントとサーバの 2 つのプログラムを準備する必要がないことや耐障害性が高いこと，通信処理の簡易化ができるといった様々な利点がある．しかし一方で，開発時におけるデバッグを困難とする要因ともなっている．

†鳥取大学大学院 工学研究科 情報エレクトロニクス専攻

2.1 モバイルエージェントシステムにおけるデバッグの問題点

デバッグは一般的に以下のような手順で行われる．

1. **エラーの存在の認識**
   システム上でエラーが発生したことを認識する．
2. **エラー発生原因の特定**
   具体的にどの処理にエラーが発生しているかを特定する．
3. **修正方法の決定**
   特定したエラー発生原因に対して修正方法を決定する．
4. **修正・動作確認**
   修正を施し正しく動いているかを確認する．

これらのデバッグ手順をモバイルエージェントシステムに当てはめた場合，各ステップにおいて図 1 のような問題が発生する．

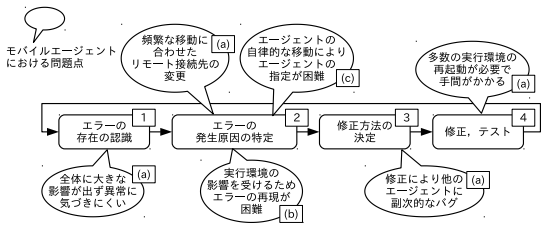

図 1: デバッグ手順とモバイルエージェントシステムにおける問題点

1. **エラーの存在の認識**
   モバイルエージェントシステムではエージェントが遠隔地のノードに移動して動作する．このため，エージェントにエラーが発生した場合はエージェントが滞在しているノードで予期せぬ動きやエラーメッセージの発生などの異常が発生する．他のノードにも影響を及ぼすほどの大きいエラーであれば発見は容易であるが，多くの場合はエラーの影響は狭い範囲にとどまってしまう．これは耐障害性の面で見れば優れている点であるが，デバッグの際にはエラーの発見が遅れる原因となってしまう．このため，発生したエラーを発見するには複数のノードを監視

する必要がある．しかし，システムの規模が大きくなればノードやエージェントの数は膨大になるため，各ノードの状態やエージェントの状態を把握することが困難となり，エラーの存在の認識が困難となる．

2. **エラー発生原因の特定**
エラーの原因を特定するにはまずどのような処理を行った時にエラーが発生するか，そして，具体的にソースコードのどこに誤りがあるかを特定する必要がある．一般的にはデバッガのブレークポイント機能を用いてプログラムを一時停止し，どの処理でどのような値を扱ったかなどの詳細な情報を取得することでエラーの発生原因を特定していく．
モバイルエージェントシステムではデバッグ対象のエージェントは遠隔地の実行環境に分散して動作しているため，ブレークポイント機能を用いるにはリモートデバッグ機能を用いる必要がある．このためにはデバッグ対象のエージェントが滞在しているノードとリモート接続する必要があるため，まずデバッグ対象のエージェントを見つけ出す必要がある．しかし，遠隔地のノード間を自律的に移動するエージェントの位置を常に把握することは難しい．このため，エージェントの指定が困難となる．
また，エージェントは頻繁にノード間を移動するため，その度に接続先を変更する必要があるが，一般的なリモートデバッグ機能では頻繁にリモート接続先が変わることは想定されていない．
さらに，エージェントはノード間を移動しながら，滞在しているノードのリソースを消費して動作するため，ネットワークの通信状態や滞在しているノードの状態から大きな影響を受ける．このため，同じエラーを再現することが困難となる．これらの原因によって，エラーの発生原因の特定が困難となる．

3. **修正方法の決定**
モバイルエージェントシステムではエージェント同士が協調しながら動作している．このため，あるエージェントを修正した場合に別のエージェントの動作にも影響を及ぼすことがある．このため，各エージェント同士の依存関係についても考慮する必要がある．しかし，エージェントは他のエージェントと協調はするものの，基本的には各エージェントが自律的に単体で動作しているため，エージェント同士の協調関係を把握することが難しい．このため，システムを修正した際に起こる影響の予測も難しくなり，修正方法の決定が困難となる．

4. **修正・動作確認**
一般的には修正した内容の動作確認をするには，プログラムのソースコードを変更し，その変更を適用するためにプログラムを再起動することが必要となる．しかし，モバイルエージェントシステムではエージェントの動作が他のエージェントにも影響を与えるため，修正の影響を受ける全てのエージェントに対して変更を適用する必要がある．このため，修正したエージェントと協調関係にある複数のエージェントを同期した上で中断し，修正後に再開する必要がある．

2.2 モバイルエージェントシステムのためのデバッガ

モバイルエージェントシステムのデバッグにおける問題を各ステップ毎にまとめると以下のようになる．

1. エラーの存在の認識
   (a) 複数のエージェントの監視が困難
2. エラー発生原因の特定
   (a) エージェントの指定が困難
   (b) エージェントが遠隔地のノード間を頻繁に移動することによるリモート接続先の変更
   (c) ノードの影響を受けるため，エラーの再現が困難
3. 修正方法の決定
   (a) 修正による他のエージェントの副次的なバグ
4. 修正・動作確認
   (a) 修正後の再起動が手間

モバイルエージェントシステムのためのデバッガにはこれらの問題を解決するための機能が必要である．本論文では，上記の 1．エラーの存在の認識と 2．エラー発生原因の特定を対象とする．

問題 1(a) に対しては，システム上のエージェントとノードを集約的に監視できるモニタにより対応する．これにより，複数のエージェントの監視を可能とし，エラーの存在の認識を支援する．

問題 2(a) に対しては，エージェントの識別と検索機能により，システム上から指定したエージェントを見つけ出すことを可能とする．問題 2(b) に対しては，遠隔地を移動するエージェントの移動に合わせて動的に接続先を変更可能なブレークポイント機能により対応する．これにより，動作中のエージェントをリモートデバッグ可能とし，エラー発生原因の特定を支援する．問題 2(c) に対しては，エージェントとノード間の通信のログ管理機能により，エラーの再現を行い易くする．

3 モバイルエージェントシステムにおけるデバッグ機能の設計

モバイルエージェントシステムでは多数のエージェントが遠隔地のノードに分散して動作する．それら全てのエージェントに対して個別にデバッグ処理を行うことは手間がかかるため，デバッグの際はそれらを一箇所から管理できることが望ましい．これを解決するための機能として，複数のエージェントとノードを集約的に監視で

きる機能を実現する. そして, 複数のエージェントから指定したエージェントを見つけ出すためのエージェントの識別・検索機能, 実行中のエージェントの状態を把握するためのエージェントの移動に対応したブレークポイント機能，そして，エラーの再現を支援するためのエージェントとノード間の通信ログ管理機能をそれぞれ実現する.

3.1 遠隔地からの一括管理

複数のエージェントを一括で監視するためには，各ノードで動作するエージェントの状態を一箇所から取得する必要がある．これは，各ノードから監視用モニタに対してエージェントの情報を送信させることで実現する．これにより，各ノードやエージェントの情報を一括に管理することが可能となる.

3.2 エージェントの検索と識別

エージェントは遠隔地のノードに分散して動作し，また，ノード間を移動しながら自律的に動作するため，エージェントの位置を常に把握しておくことは難しい．そこで，デバッグ対象として指定したエージェントを検索する機能が必要となる．また，検索を行うには検索の条件を指定する必要がある．デバッグ対象のプログラムは各エージェントであるため，全てのエージェントを一意に識別できる必要がある．そこで，各エージェントに固有の識別子を付与し，この識別子を検索の条件として検索することで目的のエージェントを指定する.

3.3 エージェントに対するブレークポイント機能

モバイルエージェントシステムにおいてデバッグ対象のプログラムは各エージェントであるため，提案するブレークポイント機能は単体のエージェントに対してブレークする機能を実現する.

エージェントは移動先のノードで動作するため，指定したブレークポイントの保持およびエージェントの処理の監視はエージェントが動作しているノードが行う．ノードはブレークポイントを指定されたエージェントの処理を監視し，ブレークポイントの処理を実行する直前にエージェントを一時停止する．これにより，単体エージェントに対してのブレークポイント機能を実現する.

移動後の継続したデバッグ　デバッグ中にエージェントが移動した場合もデバッグを継続するには移動後も設定したブレークポイントを保持する必要がある．そこで，エージェントが移動する際に移動元から移動先のノードに設定したブレークポイントを送信することで，移動後もブレークポイントを保持する.

また，エージェントの移動後は移動に合わせて監視先を変更する必要がある．エージェントが移動する度にエージェントを再検索するのでは手間がかかる上に，頻繁に移動するエージェントを捕捉しきれない．そこで，エージェントが移動した際に管理元に対して移動先の情報を送信し，管理元は受け取った情報に基づいてリモート接続先を変更する．これにより，エージェントの頻繁な移動に対応したリモート接続先の変更が可能となる.

3.4 エージェントとノード間の通信のログ管理機能

エージェントは滞在しているノードからの影響を受けて動作するため，エラーの発生はエージェントだけでなくノードの状態も関係する．このため，エラーを再現するには，エージェントに加えてノードからの影響も再現する必要があり，エラーの再現が困難となる.

ノードからの影響で発生するエラーは，ノードが保持する異常な値をエージェントが読み込むことで発生する．また，ノードが異常な値を保持する原因の 1 つはエージェントからノードに対する異常な値の書き込みである．よって，エージェントとノード間の通信のログを記録することで，エラーが起こるまでのノードの状態遷移を記録する．これにより，エラーを再現することを支援できる.

4 実験

モバイルエージェントシステムに対するデバッグ機能を我々が開発しているモバイルエージェントフレームワーク Maglog[4] に実装した．そして，開発したデバッグ機能の評価実験として，テスト用アプリケーションをデバッグする実験を行った．このアプリケーションにはあらかじめ第三者によりバグを含ませてあり，そのバグの内容を知らない状態でデバッグを行った．そして，デバッグ機能により発見可能なバグの確認を行い，その結果から，開発したデバッグ機能によりデバッグの困難性をどの程度軽減できるかを考察した.

4.1 テスト用アプリケーション

テスト用アプリケーションは次の種類のエージェントが動作している.

- round_agent
  システムに参加しているノードを巡回するエージェント．指定された時間間隔毎に，システムに参加しているノードを順番に移動する．このエージェントは，システムの動作中にシステムに参加しているノードが停止した場合，移動時にエラーが発生し，エージェントの動作が停止するバグを持つ.
- random_agent
  システムに参加しているノードをランダムに移動するエージェント．指定された時間間隔毎に，システムに参加しているノードからランダムに選択したノードに移動する．このエージェントは，システム起動直後は正常動作するが，他のノードがシステムに参加した直後に移動先をランダムに決める処理で必ずエラーが発生し，エージェントの動作が停止するバグを持つ.

このテスト用アプリケーションを用いた実験では，random_agent, round_agent の生成数はそれぞれ 2 つ，時間間隔は全て 5000 ミリ秒とした.

4.2 テスト用アプリケーションのデバッグ

デバッグ機能により発見可能であったバグとその発見時のデバッグ手順を示す.

round_agent のデバッグ　システムに参加していたノードを停止した直後に，他ノードで動作していた round_agent の処理でエラーが発生した．Maglog におけるエージェントの移動は，go(Asid) という組み込み述語で行われる．以降この述語を go/1 と呼ぶ．Asid とは各ノードのアドレスなどによって表されるノードの識別 ID であり，この関数ではエージェントの移動先ノードの識別 ID を示す．

移動処理の確認のために，round_agent の go/1 にブレークポイントを設定した．その状態で実行すると，go/1 を実行する直前で停止した．その際の Asid の値を確認すると，既に停止しているノードの Asid が引数として渡されていることが確認できた．次に，Asid の値が異常であった原因を特定するために go/1 の前にブレークポイントを設定し，そこからステップ実行により処理を進めて確認した．その結果，移動先のリストの更新処理が行われていなかったため他ノードが停止しても移動先のリストが正常に更新されず，本来なら消えるはずの移動先候補が残ってしまったものであると分かった．

random_agent のデバッグ　システムに複数のノードが参加した瞬間に，random_agent の処理でエラーが発生した．これも他のノードに依存しているエラーであるため，go/1 にブレークポイントを設定して確認した．

go/1 にブレークポイントを指定して実行すると，go/1 で停止することなくエージェントが異常終了した．これにより，移動処理より前にエラーが含まれていることが分かったため，移動処理の前にブレークポイントを設定し，ステップ実行で処理を進めていくと，移動先をランダムに決める処理で異常終了した．そこで，異常終了した処理を詳しく確認した．異常終了した処理は，移動先のリストからランダムに選択する処理を行っており，その際にリスト参照用の変数をデクリメントすべき処理でインクリメントしたことにより，移動先のリストのサイズを超えたものを参照したことがエラーの原因であると分かった．システムに参加しているノードが 1 つである場合はループ条件によりこのエラーの部分を処理しなかったが，他のノードが参加したことでエラー部分を処理することになったため，他のノードがシステムに参加した際に異常終了したとわかった．

4.3　考察

開発したデバッグ機能を用いてデバッグを行った結果，ブレークポイント機能及びステップ実行機能を用いることで遠隔地で動作するエージェントの実行中の状態を詳細に知ることが可能となった．また，エージェントが他の実行環境を移動した際も特別な操作を行うこと無く移動後のエージェントに対してデバッグを継続できた．これにより，単体のエージェントの処理は一般的なデバッガにおけるブレークポイント機能に近いレベルでデバッグが行えた．このため，他のエージェントから影響を受けず単体のエージェントのみが原因で起こるようなエラーであれば，本システムを用いる事で十分にエラーの原因特定の困難性を軽減できると考えられる．

今回行った実験において，複数のエージェントを同時に停止させて処理を確認したい状況や他のエージェントと協調動作するエージェントを停止させて処理を確認したい状況などがあった．しかし，そのようなエージェントに対してブレークポイント機能を用いる場合，複数エージェントの処理の順序関係などを考慮した上で，エージェントを停止させる必要がある．しかし，開発したブレークポイント機能ではそのようなことを考慮した上で用いる事が難しく，指定したエージェントの処理を確認することが十分にはできなかった．このため，バグの原因が複数のエージェントに及ぶ場合は今回開発したブレークポイント機能では特定することは難しいと考えられる．このため，複数のエージェントへの対応方法を検討する必要がある．

5　おわりに

本研究では，モバイルエージェントシステムのデバッグの困難性の問題のうち，エラーの存在の認識とエラー発生原因の特定の困難性を解決するために，モバイルエージェントのリモートデバッグ機能の開発とその評価を行った．その結果，単体エージェントの実行中状態を詳細に知ることができ，単体のエージェントが原因で起こるバグの特定において支援が可能であった．これにより，デバッグの困難性の軽減ができることを示せた．

今後の課題としては，本論文では対象としなかった修正方法の決定と修正，テストの問題への対応が挙げられる．修正方法の決定の問題については，エージェントの協調関係の把握を支援するための機能の開発により解決することを考えている．修正，テストの問題については，動作中のエージェントに対してホットコード置換を行う機能を開発することで解決することを考えている．

参考文献

[1] Ali R. Hurson, Evens Jean, Machigar Ongtang, Xing Gao, Yu Jiao, and Thomas E. Potok. Recent advances in mobile agent-oriented applications. In *Mobile Intelligence: Mobile Computing and Computational Intelligence*, pp. 106–139, 2010.

[2] Abdelkader Outtagarts. Mobile agent-based applications : a survey. *International Journal of Computer Science and Network Security*, Vol. 9, No. 11, pp. 331–339, 2009.

[3] Jerry Fowler and Willy Zwaenepoel. Causal distributed breakpoints. In *Proceedings of the 10th International Conference on Distributed Computing Systems*, pp. 134–141, 1990.

[4] Shinichi Motomura, Takao Kawamura, and Kazunori Sugahara. Logic-Based Mobile Agent Framework with a Concept of “Field”. *IPSJ Journal*, Vol. 47, No. 4, pp. 1230–1238, 2006.