信頼性の高い移動エージェントシステムの構成方法

新堀 健治

北陸先端科学技術大学院大学 情報科学研究科

1999 年 2 月 15 日

キーワード: 移動エージェント, 非ビザンチン障害, 障害処理, イベント, チェックポイント.

1 本研究の目的と背景

本研究の目的は、信頼性の高い移動エージェントシステムを構成するための方法を考案することである。本研究でいう信頼性とは移動エージェントにおける計算機環境の物理的な障害に対応できる性質とする。移動エージェントの技術を現実的なアプリケーションに利用するにあたってセキュリティと信頼性の確保をする必要がある。本研究では信頼性の問題について取り扱う。移動エージェントはネットワークで接続されたホスト間を移動し、様々な計算機環境上で処理をする。その様な移動エージェントは処理を達成するために、通信障害やホストダウンの障害に対処する必要がある。しかし、既存の移動エージェントシステムは、計算機環境の障害に関して十分考慮された設計になってない。そのため、計算機環境に障害が発生した場合、既存の移動エージェントシステムで作成された移動エージェントは、その目的を完結することができない。例えば、1) ノートパソコンのバッテリー切れによるホストダウンが発生し移動エージェントが消滅する、2) 移動エージェントが必要とする計算機資源が不足しその処理が継続できない、などの問題が発生する。そのような問題を解決するために、計算機環境に障害が発生した場合、その障害を検出し対処することができる移動エージェントシステムの機構を考案する必要がある。本研究では移動エージェント上で発生する計算機資源の枯渇問題 (resource starvation problem) も障害として扱うことにする。

2 Coop

本研究で提案する、信頼性の高い移動エージェントシステムを作成するための基本フレームワークを Coop と名付ける。Coop は、非ビザンチン的ハードウェアの障害を検出し、移動エージェントがその障害に対処することができるフレームワークである。Coop が扱う障害として、1) ホストがダウンする場合、2) 移動エージェントが、移動先で期待していた計算機資源（CPU やメモリなど）が使えない場合、3) ネットワークが切断される場合、4) 移動エージェントのオーナーが時間内に処理結果を得られない場合、が挙げられる。Coop に基づいた移動エージェントシステムは、これらの障害に対して以下の障害処理をする。上記 1 におけるホストダウンには、予測できる場合とそうでない場合の２種類がある。予測できる場合は、バッテリー切れや UPS（無停電電源装置）の動作、検出可能なシャットダウンなどがある。この場合の障害に対しては、移動エージェントを別のホストに移動させるか二次記憶装置に永続化するなどの対処をする。予測できない場合としては、停電などによる突然のホストダウンなどがある。この場合の障害に対しては、チェックポインティングを用いる方法で対処する。移動エージェントは一定間隔の時間で、その実行状態を二次記憶装置に保存するチェックポインティングを行う。これによりホストが回復した後、チェックポイント（記録された実行状態）からその処理を再開することができる。上記 2 の障害に対しては、計算機資源の解放を待つか、別のホストに移動するなどの対処をする。上記 3 は、移動前の通信障害と移動中の通信障害がある。移動前の通信障害に対しては、移動先を変更するか、通信障害が回復するまで待機するなどの対処をする。移動中の通信障害に対しては、移動する前に移動エージェントのレプリカを作成し、確実に移動先に到着するまで繰り返し転送するなどの対処をする。上記 4 は、ホストダウンが回復しなかった場合や、チェックポイントした実行状態が壊れてしまった場合など、移動エージェントの処理を再開することができない場合が考えられる。この障害に対しては、オーナーがいる開始ホストで移動エージェントのレプリカを残し、それを再送するなどの対処をする。

Coop のアーキテクチャは、a) 障害対処機構と b) 管理モジュールの２つから構成される。障害対処機構は各々の移動エージェントに組み込まれており、障害処理メソッドとイベントモジュールの２つの処理部分から成る。障害処理メソッドとは、移動エージェントに定義された障害に対処するメソッドである。イベントモジュールとは、移動エージェントの実行環境を監視し、イベント（障害）が発生するとそのイベントに対応する障害処理メソッドを呼び出す。また、障害処理機構は柔軟なカスタマイズが可能なアーキテクチャで構成されている。実際に既存の移動エージェントシステムに障害対処機構を組み込むだけでは、すべての障害処理を行うことはできない。例えば、移動エージェントは自分自身を活性化（二次記憶装置に保存してある状態からの復活）することはできない。障害処理として移動エージェントを活性化させる場合、その処理を行う管理モジュールが必要である。管理モジュールとは、複数の移動エージェントを管理する位置固定エージェントで、１つのノードに対して１つしか存在しない。

3 実装

本研究はCoopをJavaのクラスライブラリとして設計し、実装した。実際に作成した例題は、システム情報取得エージェントと回覧板エージェントである。システム情報取得エージェントとは、各ホストのシステム情報を取得し、システム管理者にその情報を表示する移動エージェントである。回覧板エージェントとは、現実世界の回覧板を計算機上で実現した移動エージェントである。また、巡回エージェントに限り、クラッシュ障害に対応した単純な分散アルゴリズムを提案し、実装した。本研究で提案した分散アルゴリズムを使用するか、しないかはプログラマの自由である。この分散アルゴリズムは回覧板エージェントで利用している。Coopの実験プラットフォームとしてJavaベースの移動エージェントシステムである佐藤氏のAgentSpaceを利用した。障害検出する部分でネィティブメソッドを使用しているため実行環境はWindows95,98,NT4.0に限定される。実際には、Coopのアーキテクチャそのものはプラットフォーム独立なものである。この研究における実験のための実装にWin32上のJavaを使ったのである。

4 まとめ

信頼性の高い移動エージェントシステムを構成するためのフレームワークCoopを提案し、Javaのクラスライブラリとして実装した。Coopの利点は、1)Coopのアーキテクチャは移動エージェントシステムのプラットフォームに依存しない、2)Coopで提供した障害対処機構は柔軟なカスタマイズが可能である、3) 障害処理部分が独立しているため、モジュールとして取り扱うことができる、などの利点がある。Coopに基づく移動エージェントの問題は、各々の移動エージェントがイベントモジュールを持ち、イベントを監視するため、その負荷が高い点にある。しかし、各々の移動エージェントがイベントモジュールを持つことで、移動エージェントの目的に適したイベントの検出をカスタマイズすることができる利点があり、この問題に関しては、Coopの柔軟性とのトレードオフだと考えている。