B-031

モバイルエージェントシステム AgentSphere における通信プロトコルの開発

Development of Communication Protocol for Mobile Agent System AgentSphere

山本 拓哉† 山口 大祐ǂ 甲斐 宗徳ǂ
Takuya Yamamoto Daisuke Yamaguchi Munenori Kai

1 はじめに

モバイルエージェントシステム AgentSphere[1]では、エージェントと呼ばれるプログラムが自律的に AgentSphere と呼ばれるエージェント実行管理システム間を移動し処理を行う。エージェントは AgentSphere 間を移動するため、エージェント同士が協調動作を行いたい場合、情報を交換する必要が出てくる。そこで、エージェント間通信プロトコルの開発を行った。

エージェント間通信は、AgentSphere 上にある他のエージェントとメッセージの送受信を可能にし、エージェント同士の連携を行えるようにする機能である。この機能を実装することによってエージェントの開発者はエージェント間通信を、また AgentSphere の開発者は AgentSphere を構成するモジュール間の通信が出来るようになる。相手が同じマシンにいる場合はそのままメッセージを渡せるが、他のマシンにいるエージェントに補助なしでメッセージを送る時、自らが AgentSphere が動いているマシンを移動して相手を見つけて届けなければならない。これを行うと、探している間タスクエージェントの仕事が止まってしまう。さらに、作業中のエージェントが移動すると通信に必要なデータ以外のデータを送信してしまうため通信のトラフィックが増加してしまう。この点を考慮し、メッセージを送る場合は、タスクエージェントの仕事が止まることなく、移動に関する負荷も最小限にするために、配達用エージェントとしてメッセンジャを作成して届けさせる機能を開発することにした。

2 実装方法の比較

図 2-1 のように、通信を実現するためには二通りの方法がある。A 方式は、あるクラスが直接他のクラスへ渡す方法である。この方法の利点は、目的のインスタンスへ直接データを渡せることである。一方で二つの欠点がある。一つは、目的のインスタンスへ渡すまで、送信側は処理を停止しなければいけないことである。もう一つは送受信するエージェントのペアごとにプロトコルを定める手間が必要となることである。B 方式は A 方式のデメリットを解消することができる。A 方式一つ目の欠点は、仲介役となる Server にデータを預けることにより解決することができる。また A 方式二つ目の欠点は、メッセージ用 Server がメッセージのやりとりを引き受けるため、すべての送信受信クラスはメッセージ用 Server 側で定義された抽象クラスなどを実装するためプロトコルの乱立が起こらずに済む。このようにメリットが多いことから通信プロトコルの設計では B 方式を採用した。

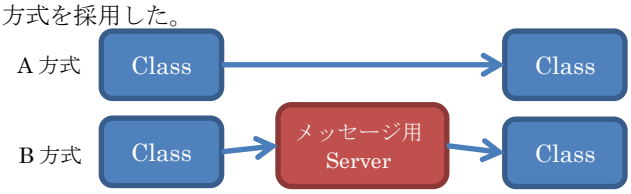

図 2-1 通信プロトコルの比較

3 エージェント間通信の設計

TCP/IP モデルにおけるネットワーク層・トランスポート層を参考にし、AgentSphere ネットワーク内の指定されたエージェントにデータを転送するエージェント間通信のためのプロトコルの実装を行った。TCP/IP モデルにおけるネットワーク層・トランスポート層を参考にした理由は、AgentSphere におけるエージェント間通信は、「目的の AgentSphere にメッセージを持った Messenger を移動する」「AgentSphere 内で動いている目的のエージェントにメッセージを届ける」という二つの要素を必要としており、前者はネットワーク層、後者はトランスポート層の設計に非常に似ているからである。通信プロトコルは、データ構造のみを定義しており、その定義を持つ Interface あるいは Abstract Class を継承することによって実装している。図 3-1 にエージェント間通信を構成する主要クラスの関係を示す。以下の節で、図 3-1 の各クラスについて述べる。

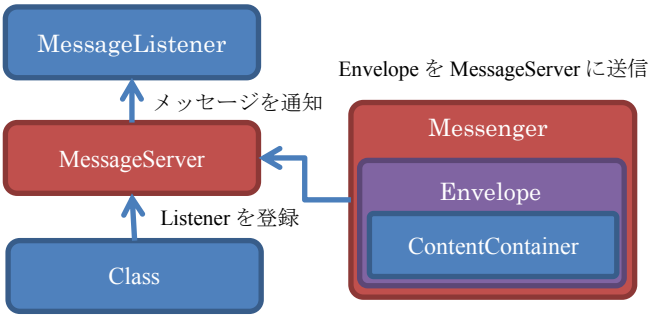

図 3-1 エージェント間通信を構成する主要クラスの関係

3.1 MessageServer

MessageServer はインスタンス間のデータの受け渡しの仲介役となる。MessageServer を使用するメリットは、MessageServer が仲介役となることにより、送信側と受信側は、相手を見つけインスタンスが受信対応しているかを調査するなどの処理を記述する必要なくなる。そして送信側

† 成蹊大学理工学部情報科学科 Department of Computer and Information Science,Seikei University

ǂ 成蹊大学理工学研究科理工学専攻 Graduate School of Science and Technology, Seikei University

と受信側は MessageServer が送受信に使用する後述する Envelope と ContentContainer のインスタンスを用意し API を使用するだけで、メッセージの送受信をすることが出来る。

3.2 MessageListener

MessageListener はインスタンスがメッセージを受け取りたい場合、MessageServer に登録するために実装しなければならない抽象クラスである。MessageSerever にメッセージが届き、登録されている MessageListener の中から対象となるエージェントが見つかると、登録した MessageListener のアクションが実行される。また、MessageListener を使っている性質上、エージェントだけではなく、メッセージ通信を利用できるのは AgentSphere を構成しているすべてのクラスとなる。すべてのクラスでメッセージ通信を利用できる利点は、例えば AgentSphere 開発者のモジュールにインタプリタがついていた場合に、そこにスクリプトを書いた Messsage を送ることにより、そのモジュールの制御が可能となることである。

3.3 Messenger

Messenger はメッセージを他の AgentSphere に届ける役割を持つ。Messenger は Agent クラスがスーパークラスになっているため、開発者は Messenger を拡張することによって、エージェント間通信における通信経路を自由に決定することができる。通信経路とは、目的地に到達するまでに通過する AgentSphere である。この通信経路を自由に決定できることにより、一部の AgentSphere の障害時に適切な経路を通って目的の AgentSphere に到達するようにプログラムすることが可能となる。

3.4 Envelope

IP データグラムのヘッダ部分に相当し、宛先情報と送信するデータの中身が格納される。Messenger は Envelope に書いてある情報を元に、対象となる AgentSphere 内のエージェントにメッセージを届ける。Envelope を拡張し、対応した Messenger を使うことによって、付加された情報を元に Messenger に対して細かい制御を行うことができる。例えば、Envelope に送信時間指定の拡張を施した場合は、Messenger が対応していた場合に、指定された時刻にメッセージを送信することが可能となる。対応していない場合は、その拡張が無視され Messenger がそれぞれの送信手続きを行う。

3.5 ContentContainer

IP データグラムのデータ部分に該当し、ContentContainer によってデータはカプセル化されている。Envelope が Messenger に対しての情報であるのに対し、ContentContainer は送信するデータに対しての情報である。

3.6 エージェントの宛先の指定方法

エージェントにメッセージを送るためには、送信先のエージェントを指定しなければならない。送信先のエージェントを指定するために、簡易的な指定方法と厳密な指定方法の二つを用意した。宛先の評価は StrictName,SimpleName の順で行われる。

3.6.1 StrictName

StrictName は宛先を指定するための厳密な指定方法である。StrictName は偶然の一致を避けるために、UUID を使用しなければいけない。Agent クラスは起動時に UUID を生成される。このため通常 Agent クラスが MessageListener に登録する際には、その値を使用する。たとえエージェントが他の AgentSphere に移動したとしても、その UUID は変わることがない。このため、エージェントの UUID を知っていれば移動したエージェントを特定することが可能である。

3.6.2 SimpleName

SimpleName は宛先を指定するための簡易的な指定方法である。ユーザーが決められる簡易的な名前のため、AgentSphere ネットワークにおいて一意に決められるものではない。SimpleName は、AgentSphere を構成しているモジュールに対して使われる。モジュールは AgentSphre を移動することはなく、AgentSphere 内において唯一であるように起動毎に UUID をつける必要がないためである。　同じ SimpleName の MessageListener が複数登録されている場合に SimpleName を指定してメッセージを送信した場合は、同じ SimpleName を持つ複数のエージェントのどれにメッセージを伝えるかは、MessegeServer の実装に任される。現在の実装では、最初に登録された SimpleName を持つ MessageLinstener に通知される。そのような場合に宛先エージェントを指定したい場合は、前述の StrictName を使用する。

3.7 エージェント間通信の具体例

メッセージ通信の例として、図 3-2 のように AgentSphere A において AgentSphere B の情報を定期的に収集するということを考える。もし情報を収集するエージェントが直接取りに行った場合、AgentSphere B から A に戻り B に再び行く間に変化した情報を取得することが出来ないが、メッセージ通信を使用した場合は、AgentSphere B で定期的に情報を収集している LoggingAgent が定期的に AgentSphere A の MasterLoggingAgent にメッセージを送信することができる。

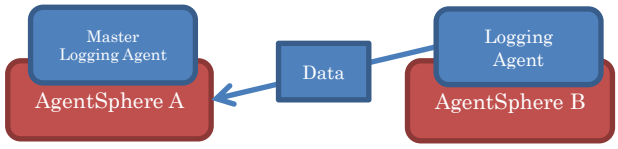

図 3-2 エージェント間通信の例

4　おわりに

本研究では、エージェントや AgentSphere を構成するモジュールなどすべての要素がメッセージ通信を行えるプロトコルの実装開発を行った。

今後の発展として、このプロトコルを様々なエージェントに利用してもらい、問題点をフィードバックしてもらい、より洗練されたプロトコルにしていきたいと考えている。

[参考文献]
[1] 赤井雄樹・横内　貴・若尾一晃・甲斐宗徳「強マイグレーションモバイルエージェントシステム AgentSphere の開発」,FIT2009, B-011,Sep.2009