論　文

エージェント・プラットフォームを相互運用するための メッセージ通信プロトコル

学生員　高橋　健一*　非会員　雨宮　聡史*
非会員　鍾　　国強*　非会員　松野　大輔*
非会員　峯　　恒憲**　正　員　雨宮　真人**

Agent Platform Protocol: Bring heterogeneous Agent Platform Together

Kenichi Takahashi*, Student Member, Satoshi Amamiya*, Non-member, Zhong Guoqiang*, Non-member, Daisuke Matsuno*, Non-member, Tsunenori Mine**, Non-member, Makoto Amamiya**, Member

Due to the spread of distributed systems, a number of agent systems have been developed. Since most existing multiagent systems implement different agent platforms, it is extremely hard to get heterogeneous agents to work together. In this paper, we introduce a new network protocol called Agent Platform Protocol (APP), which is designed to meet the exact demands of agent interaction over world-wide networks. The most significant contribution of this protocol is that it supports Peer-to-Peer communication among agent platforms and hides the network-level implementation details from agents. To demonstrate its usefulness, APP has been used to realize the interaction between independent agent systems from two projects, JADE and KODAMA. JADE is an agent system in compliance with the FIPA specifications, and KODAMA is an agent system one being developed in our laboratory. The experimental results show that APP can help JADE and KODAMA agents to communicate with each other without getting the agents being aware of the difference in their agent platforms.

キーワード：相互運用，マルチエージェントシステム，Agent Platform，通信プロトコル
Keywords: Interoperability, Multiagent System, Agent Platform, Communication Protocol

1. はじめに

近年のインターネットの急速な普及から，相互に接続された計算機を利用した協調処理方式に基づいた問題解決の仕組みが求められている。このことを実現するための枠組みとして，様々なマルチエージェントシステム [(1) (2)] が提案されてきており，それらの間での相互運用の機運が高まっている。これらのマルチエージェントシステムは，図 1 に示すように，エージェント層，プラットフォーム層，ネットワーク層の 3 層構造としてモデル化できる。ここで，各層は次のように定義できる。

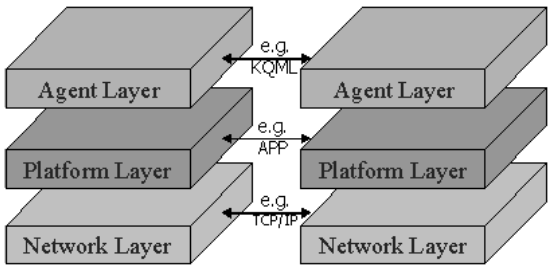


図 1　一般的なマルチエージェントモデル
Fig. 1.　A General Model of Agent System

エージェント層　論理的なエージェント間通信を実現する。この層において，エージェントは物理的なネットワークレベルの管理には係わらず，いつ，だれと通信を行うか自律的に決定し，Agent Communication Language（以下，ACL）を利用して協調する。

プラットフォーム層　エージェント間通信を物理的にサポートする。プラットフォーム層は，Agent Platform

* 九州大学大学院システム情報科学府
〒 816–8580　福岡県春日市春日公園 6–1
Department of Intelligent Systems, Kyushu University.
6–1 Kasuga-Koen, Kasuga-shi, Fukuoka 816–8580
** 九州大学大学院システム情報科学研究院
〒 816–8580　福岡県春日市春日公園 6–1
Department of Intelligent Systems, Kyushu University.
6–1 Kasuga-Koen, Kasuga-shi, Fukuoka 816–8580

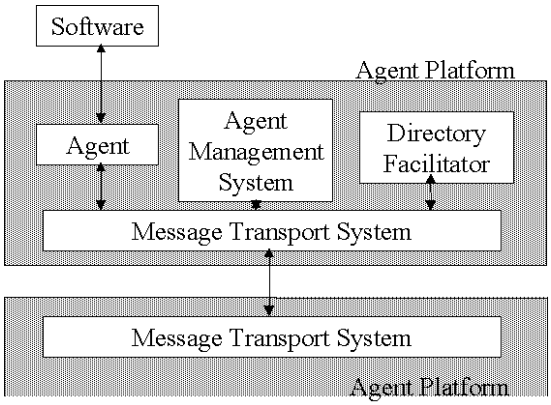


図 2　FIPA モデル
Fig. 2.　The FIPA Model

（以下，AP）から構成され，一般に，エージェントの ID 管理や物理的通信機能を提供することで，ネットワーク上でのエージェント間通信を支える。

ネットワーク層　ネットワーク上に分散する計算機間の通信を実現する。ネットワーク層では，計算機間通信のためのプロトコルとして TCP/IP などが利用され，信頼性のあるコネクション指向型の通信を提供する。

図 1 で示す各層がそれぞれの定義に従って個々に構築されたエージェントシステムの例として，KODAMA [3] や FRM [4] がある。

一方，エージェントシステムの標準化団体 FIPA [5] では，AP や FIPA ACL などの仕様を定めている。FIPA†の AP は，図 2 に示すように Directory Facilitator（以下，DF）や Agent Management System（以下，AMS），Message Transport System（以下，MTS）から構成される。DF は，他のエージェントにイエローページサービスを提供する。AMS は，エージェントの生成や消滅などの管理を行い，他のエージェントにホワイトページサービスを提供する。MTS は，他の AP 上のエージェントへエージェントメッセージ††を運ぶための通信路を提供する。FIPA ACL は，FIPA におけるエージェントが利用するエージェント間通信言語で，事実上，送受信者の物理アドレスを指定することとなっている。このように FIPA のモデルは，AP にエージェントの論理的なエージェント間通信を促進する DF を含むことや FIPA ACL の送受信者に物理アドレスを指定することから，エージェント層とプラットフォーム層を区別しないモデルとして見ることができる。

このような FIPA のモデルとして実現されたエージェントシステムと KODAMA や FRM のように各層がそれぞれの定義に従って個々に構築されたエージェントシステムとの間で相互運用を実現するには，1）FIPA のモデルに統一する方法と 2）各層でのインタフェースを規定する方法が考えられる。前者の方法では，物理アドレスの解決がエージェントによって行われるため，そのための仕組みがエージェントに必要になることや，AMS や DF といったコンポーネントを AP に準備しなければならないこと，また，各層がそれぞれの定義に従って個々に構築できなくなるなどの問題がある。一方，後者の方法では，各層のインタフェースを規定する必要がある。エージェント層レベルでは，お互いのエージェントが FIPA ACL や KQML [6] などをインタフェースとして利用することで相互運用は実現できる。また，ネットワーク層レベルでもすでに一般に使われている TCP/IP などの通信プロトコルが利用できる。しかし，プラットフォーム層間の相互運用を考えると，エージェント層で物理的なネットワークレベルの管理に係わらないですむような特別なインタフェースは定められていない。

そこで，我々は各々のモデルに影響を与えず相互運用を実現するためのプラットフォーム層のプロトコルとして Agent Platform Protocol（以下，APP）[7] [8] を提案している。APP は，1）エージェントメッセージの伝達機能，2）エージェント ID からのエージェントの位置を見つける機能，3）ネットワークの構成機能，を提供する。AP は APP に従って目的のエージェントを探し出し，エージェントメッセージを伝達するため，エージェントは互いの物理アドレスを意識することなく通信を行うことができる。

この APP を利用し，異なるエージェントシステム間での相互運用を実現する一例として，チャットシステムを作成した。エージェントシステムとして，我々が開発している KODAMA と FIPA 準拠のエージェントシステムである JADE††† [9] [10] を用いた。その結果，双方のエージェントが互いのシステムの違いを意識することなくチャットを実現することを確認した。

以下，2 章では APP について述べ，3 章で KODAMA と JADE への APP の実装について紹介する。そして，4 章で KODAMA と JADE によって実現されたチャットシステムについて報告する。

2. Agent Platform Protocol

APP は，プラットフォーム層のメッセージ通信プロトコルであり，method と header，body から構成される。APP を用いた AP 間のメッセージ（以下，AP メッセージ）を BNF 記法で示す。

```
<AP メッセージ> ::= <method> <header>* <body>?
```

ここで，method は AP メッセージが表す要求，または，応答を表す。header は，method を解釈，処理するために必要な情報で複数個ある。body は，各メソッドが対象とする任意のデータでオプションである。

〈2•1〉 Method　APP はプラットフォーム層の通信プロトコルであり，エージェントが物理的なネットワークレベルの管理に係わらないようにするために，次の機能が

† 2001 年 4 月時点での仕様を参照。
†† 本稿では，AP 間で交換されるメッセージと区別するために，エージェント間で交換されるメッセージをエージェントメッセージと呼ぶ。
††† JADE Version 2.5 を利用。

必要となる。

1） エージェントメッセージの伝達機能。エージェントは通信相手の物理アドレスやエージェントメッセージの伝達経路を意識しない。物理的な通信は，プラットフォーム層でサポートする。
2） エージェント ID からエージェントの位置を見つける機能。エージェントはいつも同じ場所に存在するとは限らない。あるエージェントは，条件によって，その存在場所を変えるかもしれない。エージェントの位置が変われば，再び，エージェント ID からエージェントの位置を見つける必要がある。
3） 任意の AP 間の接続でネットワークを構成する機能。エージェントによって，Pure Peer-to-Peer（以下，P2P）型（Gnutella[11] 型）や Hybrid P2P 型（Napster[12] 型），クライアント・サーバ型など，通信方法は異なる。プラットフォーム層では，これらのエージェントの通信方法に合わせ，物理的通信をサポートする必要がある。このためには，任意の AP 間の接続でネットワークを構成できる必要がある。また逆に，任意の AP 間の接続でネットワークを構成する機能を提供することで，個々の AP 内エージェントシステムにおける通信方法に基づいて，任意のエージェント間でメッセージを交信することができる。

上記 3 つの機能を用意することにより，AP は他の AP と共にネットワークを構成し，そのネットワークを利用してエージェントの位置をみつけ，エージェントメッセージを伝達することができる。これにより，エージェントは，通信相手のエージェントの物理アドレスを意識せずに通信できるようになる。APP では，このことを実現するために 12 種類の AP メッセージを準備する（表 1）。CONNECT，DELIVER に分類される AP メッセージは要求を表し，RESPONSE に分類される AP メッセージは，応答を表す。

CONNECT　任意の AP 間の接続でネットワークを構成する。情報を集中的に管理するサーバの存在を仮定せず，AP 間の接続を確立するための機能と AP ネットワークを構成するための機能を提供する。

DELIVER　確立された AP 間接続を通してエージェントメッセージを伝達する。エージェントは絶えず同じ場所にとどまるとは限らないため，エージェント ID からエージェントの位置（e.g. IP アドレス）を見つけるための機能を提供する。

REPLY　CONNECT，DELIVER の各メッセージに対する応答として用いる。

〈2･2〉 Header　APP における header は，

```
<header> ::= <header-name>":"<value>
```

のように表現され，その役割によって，general-header，body-header，authentication-header に分けられる。general-header は，to や from，message-id など必須の情報を表す。body-header は，body-size や body-type など body 部の情報を表す。authentication-header は，authentication-key や authentication-expired などの認証に必要な情報を表す。

〈2･3〉 Body　body は，method で必要とされる任意のデータである。例えば，DELIVER の場合，body は，エージェントメッセージとなる。

3. エージェントシステムの相互運用

APP を用いた異種のエージェントシステム間の相互運用例として，我々が研究・開発しているマルチエージェントシステム KODAMA（Kyushu university Open & Distributed Autonomous Multi-Agent）と FIPA 準拠のエージェントシステム JADE を利用した。

〈3･1〉 KODAMA　KODAMA は，図 1 で示された各層がそれぞれの定義に従って個々に構築されたソフトウェアエージェントシステムである。

KODAMA におけるシステムは，コミュニティと呼ばれるエージェント群の階層的な組み合わせによって形成される（図 3）。KODAMA におけるエージェントは，必ずいずれかのコミュニティに属し，そのコミュニティの範囲で個々のエージェントは自由に通信を行う。各コミュニティにはそれを代表するポータルエージェントが存在し，コミュニティの中と外とを結ぶ窓口の役割を演じる。エージェント間の通信は，コミュニティ構造によって制御される。具体的には，コミュニティ構造に従って，徐々に検索範囲を広げていくことで目的のエージェントを見つけ，通信を行う。従って，同じコミュニティに属するエージェント同士の通信は，コミュニティを隔てたエージェントとの通信より優先される。また，エージェントには，コミュニティの階層構造にしたがってユニークなエージェント ID が与えられる。

```
<エージェント ID> ::= <communityID>"."<uniqueID>
<communityID> ::= <エージェント ID>
<uniqueID> ::= [string]
```

ここで，communityID はエージェントが属するコミュニティの名前を示し，そのコミュニティのポータルエージェントのエージェント ID が用いられる。uniqueID は，エージェントが属するコミュニティのポータルエージェントから与えられるコミュニティ内でユニークな ID である。[string] は，1 文字以上の英数字の列である。このため，図 3 中，Y のエージェントに与えられるエージェント ID は”X.Y”となり，Z のエージェントに与えられるエージェント ID は”X.Y.Z”となる。この構造を利用し，2 エージェント間のコミュニティ構造上の距離を，「探索時に経由するポータルエージェント数+1」と定義する。例えば，2 つのエージェント，”X.W.V”と”X.Y.Z”のコミュニティ構造上の距離は，”X.W”，”X”，”X.Y”のポータルエージェントを経由しているため，その距離は 4 となる。

これらのエージェントは，AP として準備されている Agent Communication Zone（以下，ACZ）を利用する。

表 1　APP における Method
Table 1.　APP Message Methods

Types	Methods	Function
CONNECT	LINK	establish network connections among APs
	DeLINK	close established network connections
	GET_LINK	ask for link information
	ALIVE?	indicate that APs are active
DELIVER	DELIVER	deliver agent messages to specified agents
	LOCATE_AGENT	look for the network addresses of agents
	GET_AGENT_LIST	ask for information about agents and their host agent platforms
	PUT_AGENT_LIST	provide information about agents and their host agent platforms
RESPONSE	ACCEPT	requests are accepted
	REFUSE	requests are refused
	SUCCESS	requests are successfully accomplished
	FAILURE	requests fail

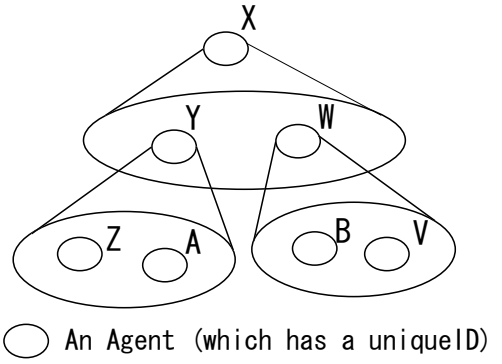


図 3　KODAMA のコミュニティ構造例
Fig. 3.　An Example of KODAMA Community Structure

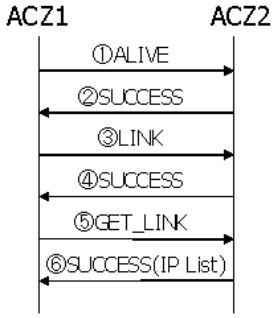


図 4　ACZ の初期化動作
Fig. 4.　Initialize Action of ACZ

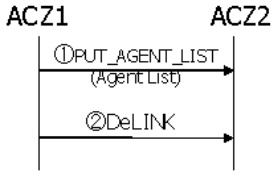


図 5　ACZ の停止動作
Fig. 5.　Stop Action of ACZ

ACZ は，論理的なエージェント空間を物理的な構造にマッピングする機構である。ACZ は，Local Agent address Table（以下，LAT）と Remote Agent address Table（以下，RAT），Link Table（以下，LT）を持つ。LAT は，その計算機上に存在するエージェントの ID を保持する。RAT は，異なる計算機上のエージェント ID をその物理アドレスと共に保持する。LT は，AP ネットワークの情報を保持する。これらのテーブルを用いて，ACZ はエージェント間の通信を実現するために必要となる物理的通信機能を提供する。この機能により，エージェントは通信相手の物理アドレスを気にすることなく，自由に通信を行うことができる。

KODAMA への APP の実装

ACZ が AP メッセージを受け取った時の，各 AP メッセージの解釈方法を表 2 に示す。以下，ACZ の起動，停止，エージェントメッセージの伝達時の動作について説明する。

1）ACZ の起動　KODAMA は，Java で開発されており，次のコマンドで ACZ が起動される。

```
java acz.ACZ <初期化 IP>
```

起動する ACZ を ACZ1，初期化 IP のアドレスに存在する ACZ を ACZ2 とすると，まず，ACZ1 は ACZ2 に対して ALIVE?メッセージ（図 4①）を送る。その返答として FAILURE メッセージを受け取った場合は時間を置いて再送する。SUCCESS メッセージ（図 4②）を受け取った場合は LINK メッセージによって ACZ2 との間に接続を確立する（図 4③，④）。そして，GET_LINK メッセージでネットワークの情報（図 4⑤，⑥）を集め，LT に登録する。

2）ACZ の停止　図 5 に ACZ の停止時の動作を示す。LT に保持されているすべてのアドレスに対して，PUT_AGENT_LIST メッセージ（図 5①）で RAT に登録されているエージェントの情報を知らせる。そして，DeLINK メッセージ（図 5②）で接続を閉じる（ACZ2 は LT から ACZ1 の情報を削除する）。

3）ACZ によるエージェントメッセージの伝達　エージェントメッセージの伝達は，LAT，RAT の状態について 3 つの手段が存在する。

Case 1　LAT に目的のエージェントが存在する場合。
LAT を参照し，直ちにエージェントメッセージを伝達

表 2　ACZ の各 AP メッセージに対する動作
Table 2.　Actions of ACZ for AP Message

Methods	Implementation
LINK	establish a new connection and save link information to LT
DeLINK	close an established connection and delete link from LT
GET_LINK	reply with the link information saved in LT
ALIVE?	reply with SUCCESS
DELIVER	if the specified agent is registered into LAT, deliver the message to that agent and reply with SUCCESS. Otherwise, reply with FAILURE
LOCATE_AGENT	reply with ACCEPT immediately and look up the network address of the specified agent. If it can be find, reply with SUCCESS
GET_AGENT_LIST	reply with the agent information in LAT and RAT
PUT_AGENT_LIST	register the agent information into RAT

する。

Case 2　LAT に目的のエージェントが存在しないが，RAT に目的のエージェントが存在する場合。RAT を参照し，目的のエージェントが存在するアドレスに対して DELIVER メッセージでエージェントメッセージの伝達要求を行う。その結果，SUCCESS メッセージを受け取った場合は，メッセージの伝達は成功である。FAILURE メッセージを受け取った場合（エージェントメッセージが伝達できなかった場合）は，RAT の情報を修正し，Case 3 の手段を利用する。

Case 3　LAT にも RAT にも目的のエージェントが存在しない場合。目的のエージェントのアドレスを，LOCATE_AGENT メッセージを用いて見つける。ACZ（この ACZ を ACZ1 とする）は，LAT と RAT に目的のエージェントが含まれていない場合，RAT に登録されているエージェントの中から，コミュニティ構造上で目的のエージェントに最も近いエージェントを探し出し，そのエージェントが存在する ACZ（この ACZ を ACZ2 とする）へ LOCATE_AGENT メッセージを送る。ACZ2 は，即座に ACZ1 へ ACCEPT メッセージを返し，LAT に目的のエージェントを含むか否かを調べる。LAT に目的のエージェントを含めば，ACZ1 へ SUCCESS メッセージを返し，含まなければ，ACZ2 が持つ RAT から目的のエージェントに最も近いエージェントを探し出し，そのエージェントが存在する ACZ へ LOCATE_AGENT メッセージを送る。これを繰り返すことで，目的のエージェントを見つける。

図 6 は，ACZ1 の RAT が”X.Y”と”X.W”の情報を含み，ACZ2 の RAT が”X.Y.Z”の情報を含み，ACZ3 に”X.Y.Z”のエージェントが存在するときの例を示している。ここで ACZ1 にいるエージェントが X.Y.Z にエージェントメッセージを送る状況を考える。まず，ACZ1 は”X.Y.Z”と RAT に含まれるエージェントとの距離を調べる。ここで，”X.Y.Z”と”X.Y”のコミュニティ構造上の距離は 1，”X.Y.Z”と”X.W”のコミュニティ構造上の距離は 3 として求まる。このため，ACZ1 は RAT に登録されているエージェントの中で，コミュニティ構造上で最も近い”X.Y”が存在する ACZ2 に LOCATE_AGENT

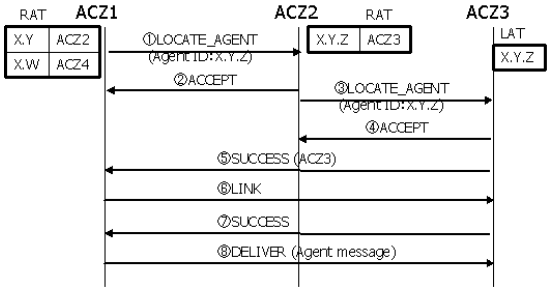


図 6　ACZ がエージェントの位置を探す時の動作例
Fig. 6.　The example for which ACZ searches an agent’s position

メッセージを送る。同様に，ACZ2 は”X.Y.Z”と RAT に含まれるエージェントとの距離を求め，結果，ACZ3 へ LOCATE_AGENT メッセージを送る。ACZ3 は，LAT に”X.Y.Z”のエージェントが存在するため，ACZ1 に SUCCESS メッセージを返す。このことで，ACZ1 は ACZ3 に”X.Y.Z”のエージェントが存在することを知り，DELIVER メッセージで ACZ3 にエージェントメッセージを届けることができる。

〈3･2〉 JADE　JADE は，エージェントシステムを効率的に組み立てるためのソフトウェアフレームワークである。我々が，JADE を KODAMA との相互運用を実現するエージェントシステムの一つとして選んだのは，1）ソースが公開されている，2）FIPA に準拠したエージェントシステムである，3）Java で開発されている，ためである。JADE は，FIPA 準拠のエージェントシステムであるため，AP 中に AMS や DF などを必要とする。AMS は，エージェントの生成や消滅など，AP の操作を管理する責任を持つ。DF は，エージェントにイエローページサービスを与える。エージェントは DF にサービスを登録し，また，DF からどのようなサービスを提供するエージェントが存在するかを知る。ACC（Agent Communication Channel）は，AP 内外のエージェント間通信支援を行う。

また，JADE における AP は，複数のエージェントコンテナから構成される。コンテナはエージェントの実行環境であり，各コンテナには，複数のエージェントが存在する。

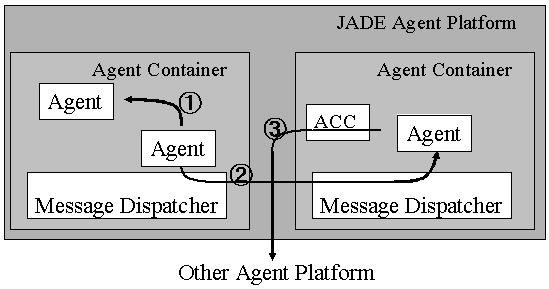


図 7　JADE エージェントプラットフォーム
Fig. 7.　JADE Agent Platform

このため，JADE エージェント間でメッセージ送受信を行うには，次の方法が利用される（図 7）。

Case 1　送信先が同一コンテナ上に存在する場合（図 7①）。イベントオブジェクトを使って，メッセージが送られる。

Case 2　同一の JADE プラットフォーム上であるが，異なるコンテナ上に存在する場合（図 7②）。Java RMI（Remote Method Invocation）を使って，メッセージが送られる。

Case 3　送信先が別の AP の場合（図 7③）。FIPA の基準に従って，IIOP（Internet Inter-ORB Protocol）を使ってメッセージが送られる。

また，JADE-AP は Local Cache（以下，LC）を持ち，エージェント情報をキャッシュする。

JADE への APP の実装

JADE では，前節で述べた Case 2，3 に関して APP を利用することとした。APP は，プラットフォーム層でのプロトコルであるため，エージェントの構造に制限を与えない。このため，APP を JADE に実装するには，JADE-AP のネットワーク接続に関する部分だけを AP メッセージに対応できるように変更するだけでよい。表 3 に各 AP メッセージ受信時の JADE-AP の動作を示す。JADE-AP では，AP 間で P2P ネットワークを構成するための機能について考慮されていないため，CONNECT に分類される AP 間で P2P ネットワークを構成するための AP メッセージについては特別な操作を行わなかった。

〈3・3〉 エージェント層の考慮　KODAMA では，すべてのエージェントはコミュニティ階層上のいずれかの位置に割り当てられる。そして，目的のサービスを提供するエージェントの探索は，このコミュニティ構造に従ったリンクの走査により行われる。このため，JADE-DF を，KODAMA の最上位コミュニティ（ルートコミュニティと呼ぶ）の一員として割り当てることで，KODAMA エージェントが目的のサービスを提供する JADE エージェントを見つけることができるようになる。また，JADE エージェントは，DF を利用して目的のサービスを提供するエージェントを見つける。そこで，目的のサービスを提供するエージェントが JADE 側で見つからなかった場合，KODAMA のエージェントが提供するサービスを見つけることができるように，DF は検索要求を KODAMA の最上位コミュニティのポータルエージェント（ルートエージェントと呼ぶ）へ送ることとした。

図 8　チャットインタフェース
Fig. 8.　Chat Interface

4. 実　　験

APP を利用した KODAMA と JADE の相互運用性を確かめ，その性能を評価するために，KODAMA と JADE 双方のエージェントを利用したチャットシステムを実現した。

〈4・1〉 チャットシステム　本実験で実装したチャットシステムは，一人のユーザに対して一つの ChatAgent（チャットのための GUI 機能を持つ）が割り当てられる（図 8）。各ユーザは，チャットを始めるときにチャットの相手を Message text ボックスで指定し，ADD ボタンを押す。このことで ChatAgent は，指定された相手を探し出し，相手からチャットの参加者一覧を受け取る。ChatAgent は，Other chatters ボックスに参加者一覧を表示し，それらの参加者に対してチャットへの参加を通知することでチャットへの参加が行われる。チャットへの書き込みは，Message text ボックスに書き込みたい内容を指定し，WRITE ボタンを押すことで Other chatters ボックスに表示されている相手に対して書き込み要求を送る。書き込まれた内容は，Received ボックスに表示される。チャットからの脱退は，REMOVE ボタンを押すことで Other chatters ボックスに表示されている相手に対して脱退通知が送られる。実験では，ユーザ A，B に対する ChatAgent を KODAMA で，ユーザ C，D に対する ChatAgent を JADE で実現し（図 9），各 ChatAgent は，それぞれ異なる JVM（Java Virtual Machine）上（KODAMA では異なる ACZ 上，JADE では異なるコンテナ上）で動作させることとした。

〈4・2〉 システムの動作　図 9 のような状態において KODAMA-JADE 間の通信は次のように行われる。エージェント層，プラットフォーム層での対応について，それぞれ示す。

表 3　JADE-AP の各 AP メッセージに対する動作
Table 3.　Actions of JADE-AP for AP Message

Methods	Implementation
LINK	establish a new connection immediately and reply with SUCCESS message
DeLINK	close an established connection and reply with SUCCESS
GET_LINK	reply with REFUSE (JADE does not have information for other APs.)
ALIVE?	reply with SUCCESS
DELIVER	deliver the agent message to a specified agent, if it exists
LOCATE_AGENT	reply with the network addresses of JADE agents
GET_AGENT_LIST	reply with the agent information in LC
PUT_AGENT_LIST	register the agent information into LC

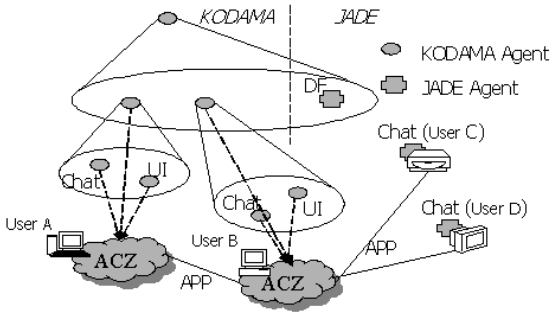


図 9　チャットシステム
Fig. 9.　Chat System

エージェント層での対応　KODAMA エージェント（ユーザ A）から目的のサービスを提供する JADE エージェント（ユーザ C）は，KODAMA のコミュニティ構造に従って見つけられる。JADE-DF が KODAMA のルートコミュニティの一員として割り当てられているため，サービス検索の範囲がルートコミュニティとなった時点で，JADE-DF にも検索メッセージが送られる。JADE-DF はユーザ C を表すエージェントを見つけて返す。この結果，KODAMA エージェントから目的のサービスを提供する JADE エージェントを見つけることができる。逆に JADE エージェント（ユーザ C）から目的のサービスを提供する KODAMA エージェント（ユーザ A）は，JADE 側でユーザ A を表すエージェントが見つからないときに，JADE-DF がルートエージェントに対して検索メッセージを送る。その結果，KODAMA 側でユーザ A を表すエージェントが見つけられ，JADE-DF に返される。JADE-DF はユーザ C のエージェントにその結果を知らせる。このように，JADE エージェントも目的のサービスを提供する KODAMA エージェントを見つけることができる。

プラットフォーム層での対応　エージェント層でユーザ A がユーザ C を見つけたとしても，ユーザ A のエージェントがいる ACZ が，ユーザ C のエージェントの物理アドレスを知らなければ，そのアドレス探索が必要になる。そこで，ACZ は，LOCATE_AGENT メッセージを JADE-AP に送る。JADE-AP は，指定されたエージェント ID から，JADE-AP のアルゴリズムに従って†ユーザ C のエージェントの物理アドレスを求めて返す。ACZ は，得られたアドレスに対して DELIVER メッセージを送ることでユーザ C のエージェントにエージェントメッセージを送る。逆に JADE-AP からユーザ A のエージェントの物理アドレスを見つけるときも，JADE-AP から LOCATE_AGENT メッセージが ACZ に送られる。ACZ は，ACZ のアルゴリズム††に従って，ユーザ A のエージェントのアドレスを見つけて返す。JADE-AP は，得られたアドレスに対して DELIVER メッセージを送ることでユーザ C のエージェントにエージェントメッセージを送る。

〈4･3〉 機能評価　実験によって以下のような機能が実現できていることを確認した。これらの機能は各層を独立に扱いながら相互運用を実現するという点から重要なものである。APP は，プラットフォーム層のプロトコルとして，物理的ネットワークレベルの問題解決機能を持つ。このため，物理アドレスの解決において，エージェント層におけるエージェントへの活動に対して影響を与えることなく，AP に APP を実装することができる。また，エージェント層で DF のサービス検索の仕組みに対応し，APP で AMS のホワイトページサービスの機能を吸収することで†††，FIPA における DF や AMS といった特別なコンポーネントを準備することなく，FIPA 準拠のエージェントシステムと各層がそれぞれの定義に従って個々に構築されてエージェントシステムの間での相互運用が実現できる。このように，エージェント層ではエージェント間の論理的通信を，プラットフォーム層では AP 間の物理的通信を，それぞれ独立に扱いながらエージェントシステム間の相互運用が実現できる。また，各層を独立に扱ったエージェントシステムでは，各層を独立に扱えないエージェントシステムより，構築容易性や保守性の面でも優れたものになる。

〈4･4〉 パフォーマンス評価　APP が実用上支障をきたさない速度で実現できるか否かを調べるために，エージェ

† JADE ではエージェント ID にアドレスを含むため，そこから簡単にアドレスを求めることができる。
†† 3.1 節 3）ACZ によるエージェントメッセージの伝達参照。
††† ライフサイクル管理機能はサービスとして他の AP に提供する必要がなく，それぞれの AP 内部で閉じた機能として提供できればよいと考える。

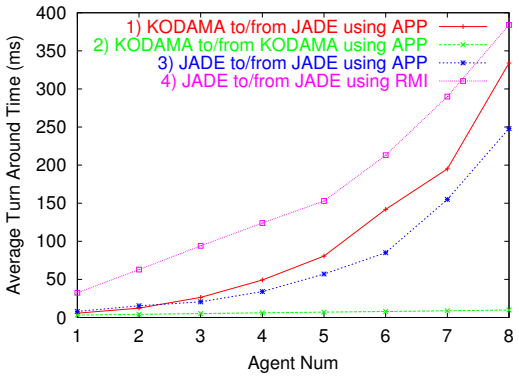


図 10　テスト結果

Fig. 10.　Performance results

ントメッセージのターンアラウンドタイム計測実験を行った。実験は，RedHat Linux 6.1，Java 2 SDK，Celeron 300MHz，256M memory の 2 台の計算機を 100 Base-T の LAN で接続して行った。実験では，それぞれのエージェントが 500 の ping メッセージを異なる計算機上の任意のエージェントに送り，かつ，ping メッセージを受け取ったエージェントはすぐに返答するようにした。この時，1 台の計算機上のエージェント数を徐々に変化させていき，1）APP を利用した KODAMA-JADE エージェント間通信，2）APP を利用した KODAMA エージェント間通信，3）APP を利用した JADE エージェント間通信，4）RMI を利用した JADE エージェント間通信，のそれぞれのケースで ping メッセージの平均ターンアラウンドタイムを計測した。この際，APP による物理アドレスの解決は，1），2）では始めの一回だけ行われ，それ以降は既知として扱われる。3），4）では JADE エージェントが FIPA ACL に含まれる送受信者の物理アドレスを利用するため，APP による物理アドレスの解決は行わない。図 10 に結果を示す。

図 10 で見られるように，平均ターンアラウンドタイムはエージェントの数が増えるに従って増大している。しかし，APP を利用したケース（1），2），3））は，RMI を利用したケース（4））に比べ，よい結果を示している。このことから，APP の実装が実用上支障をきたさないことが確認できた。

5.　終わりに

本稿では，エージェント間通信を物理的にサポートするプラットフォーム層のプロトコルとして APP を提案した。APP は，method，header，body から構成される AP メッセージによるリクエスト／レスポンス指向のプロトコルで，AP に物理的なネットワークレベルの解決を行わせることで，エージェントは物理的な環境を意識することなく通信することが可能となる。

APP を用いた異なるエージェントシステム間での相互運用の実現可能性を検証するために，我々が開発を行っている KODAMA と FIPA 準拠のエージェントシステム JADE の双方に APP を実装した。また，エージェント層では，JADE-DF に KODAMA エージェントと JADE エージェントがお互いのサービスを利用できるようにするための変更を加えた。KODAMA と JADE でチャットエージェントシステムを作成した結果，モデルの違いを吸収し，エージェントは物理的環境や KODAMA と JADE のエージェントの違いを意識することなく，チャットが行えることを確認した。また，パフォーマンス評価の結果，実用上支障をきたさない速度で APP を実現可能であることが確認できた。

今後の課題として，KODAMA や JADE 以外のエージェントシステムを加えた相互運用を実現することや，APP でエージェントのサービス要求から目的のエージェントを見つける仕組みのサポート，また，他のエージェントシステムが APP を容易に利用できるよう APP Plug-in を提供することなどがあげられる。

（平成 14 年 8 月 9 日受付，同 14 年 12 月 16 日再受付）

文　献

（1） S. Franklin and A. Graesser: ”Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents”, In Proc. of Third International Workshop on Agent Theories, Achitectures, and Languages, Springer-Verlag(1996)

（2） N. J. Jennings: ”Agent-based computing: Promise and Perils”, In Proc. of Sixteenth International Joint Conference on Artificial Intelligence, pp.1429-1436(1999)

（3） G. Zhong, S. Amamiya, K. Takahashi, T. Mine and M. Amamiya: ”The Design and Implementation of KODAMA System”, IEICE Transactions INF.& SYST., Vol.E85-D, No.2, pp.637–646(2002-4)

（4） T. Iwao, M. Okada and M. Amamiya: ”A Collaboration Platform for Multi-Agent Systems: Field Reactor Model”, The Transactions of IEICE, Vol.J85-A, No.8, pp.895–907(2002-8) (in Japanese)
岩尾忠重・岡田誠・雨宮真人: 「マルチエージェント連携機構：Field Reactor Model」, 電子情報通信学会論文誌, Vol.J85-A, No.8, pp.895–907(2002-8)

（5） FIPA - Foundation for Intelligent Physical Agent Specification, http://www.fipa.org

（6） T. Finin, Y. Labrou, and J. Mayfield: ”KQML as an agent communication language”, In Software Agents, MIT Press, pp.291–316(1997)

（7） K. Takahashi, G. Zhong, S. Amamiya, T. Mine and M. Amamiya: ”Communication Protocol for Agent Platform : Agent Platform Protocol”, In Proc. of MACC 2001, pp.30–37(2001-11) (in Japanese)
高橋健一・鍾国強・雨宮聡史・峯恒憲・雨宮真人: 「エージェント基盤のための協調プロトコル：Agent Platform Protocol」, マルチエージェントと協調計算ワークショップ論文集, pp.30–37(2001-11)

（8） K. Takahashi, G. Zhong, S. Amamiya, T. Mine and M. Amamiya: ”Interoperability between KODAMA and JADE using Agent Platform Protocol”, In Proc. of Agentcities: Challenges in Open Agent Environments, pp.90–94(2002-7)

（9） F. Bellifemine, A. Poggi, G. Rimassa: ”JADE - A FIPA-compliant agent framework”, In Proc. of PAAM’99, pp.97–108(1999-4)

（10） F. Bellifemine, A. Poggi, G. Rimassa, P. Turci: ”An Object-Oriented Framework to Realize Agent Systems”, In Proc. of WOA 2000, pp.52–57(2000-3)

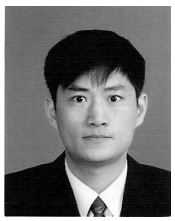
（11） The gnutella protocol specificatinon v0.4. http://www.clip2.com/gnutelaprotocol04.pdf

（12） Napster Home Page. http://www.napster.com

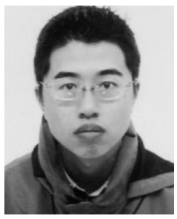
高 橋 健 一（学生員） 1976 年生。1999 年九州大学工学部情報工学科卒業。2001 年同大大学院システム情報科学研究科修士課程終了。現在，同大学院システム情報科学府博士課程在学中。マルチエージェントシステムの研究に従事。情報処理学会，電子情報通信学会，電気学会 各会員。

雨 宮 聡 史（非会員） 1973 年生。2000 年九州大学大学院システム情報科学府研究生。現在に至る。マルチエージェントシステム，分散情報検索システムの研究に従事。

鍾 国 強（非会員） 1971 年生。1993 年中国福州大学卒。1996 年中国四川大学修士課程了。2002 年九州大学院・情報・博士課程了。現在，九州大学ベンチャー ビジネスラボラトリー非常勤研究員。工博。IEEE，ACM 各会員。



松 野 大 輔（非会員） 1980 年生。2002 年九州大学工学部電気情報工学科卒業。現在同大学大学院システム情報科学府修士課程在学中。マルチエージェントシステム，知的情報検索に興味をもつ。



峯 恒 憲（非会員） 1963 年生。1987 年九大・工・情報工卒。1992 年同大大学院総合理工学研究科博士課程単位取得退学。同年同大学教養部講師。1996 年同大大学院システム情報科学研究科（現・研究院）助教授，現在に至る。博士 (工学)。自然言語処理，情報検索，エージェントシステム等の研究に従事。

雨 宮 真 人（正員） 1942 年生。1967 年九州大学工学部電子工学科卒業。1969 年同大学院電子工学専攻修士修了。工学博士。日本電信電話公社（現在 NTT）武蔵野電気通信研究所，同基礎研究所，同ソフトウェア研究所を経て，1989 年九州大学教授，現在に至る。専門は計算機科学，並列分散処理，人工知能，マルチエージェントシステム。