エージェント基盤のための通信プロトコル：Agent Platform Protocol

高橋 健一　鍾 国強　雨宮 聡史　峯 恒憲　雨宮 真人

近年，ネットワークで接続された複数の機器が協調や交渉を行い仕事を達成するシステムを作る枠組みとして様々なエージェントが提案されている．これらのエージェントシステムでは，エージェントの活動を促進する環境（AP）が個々に準備されているが，それら異なる AP 上のエージェントシステムを相互運用する手段については提供されていなかった．そこで，本稿では，AP 間のプロトコルとして Agent Platform Protocol(APP) を提案する．APP を用いることで，１）サーバを介さない P2P な AP 間通信と２）物理的環境を意識する必要しないエージェントコミュニケーションを実現できる．

1　はじめに

インターネットの普及とともに，ネットワークで接続された複数の機器によって実現された多くの情報システムやサービスが提供されてきた．これらのシステムでは，分散された機器が互いの情報を交換することで協調や交渉を行い問題を解決する必要が出てきており，このようなシステムを作る枠組みとして様々なエージェントが提案されている．

エージェントシステムでは，エージェントの活動を促進するための環境（Agent Platform，以降 AP と記す）（例えば，KODAMA の ACZ [1] [2] や FRM の Field [3]，Hive の cell [4] など）が準備されている．エージェントは，AP 上で，仕事の達成に必要な情報を得るためのコミュニケーションや仕事の達成に必要な資源を得るための AP 間の移動を行う．これらの AP は，元々各エージェントシステム固有のものとして開発されているため，そのような異なる AP を持つエージェントシステムを相互運用することは難しい．エージェントの標準化促進団体 FIPA [5] では，Agent Communication Language（以降，FIPA ACL）や，Agent Platform などの仕様が定められているが，その FIPA ACL では，メッセージの送信側と受信側エージェントの物理的なアドレスを指定する，受信側エージェントの物理的位置を明確に意識した仕様となっている．しかし，これは，次のような弊害を引き起こす．

- モバイルエージェントとのコミュニケーションが困難．
  エージェントが移動すれば，エージェントが存在するアドレスも変わる．すなわち，受信側エージェントの物理アドレスを特定することは難しい．
- エージェントに，予め決定された連携を行わせることが難しい．
  例えば，２種類のエージェント，Ping と Pong が存在し，Ping は，適当なタイミングで Pong にメッセージを送り，Pong の存在を確認する仕事を持つとする．このとき，２種類のエージェント

Communication Protocol for Agent Platform : Agent Platform Protocol.

Kenichi TAKAHASHI, Zhong GUOQIANG, Satoshi AMAMIYA, 九州大学 大学院 システム情報科学府, Graduate School of Information Science and Electrical Kyushu University.

Tsunenori MINE, Makoto AMAMIYA, 九州大学 大学院 システム情報科学研究院, Graduate School of Information Science and Electrical Engineering, Kyushu University.

が別々にインストールされ，予めそれぞれのアドレスを特定できない場合，Ping は，Pong の物理的なアドレスを知ることができないため，送信先としては，”Pong”とだけしか指定できない．しかし，これでは，Pong が存在するアドレスが指定されていないため，Ping から Pong にメッセージを伝達するには，Pong の物理的位置を検索する機構が必要となる．

これらの問題を解決するためには，まず AP 間に共通のインタフェースを提供するためのプロトコルを定義しなければならない．そして，そのプロトコルでは，次のことを想定しなければならない．

- AP にエージェントの物理的アドレスを管理させ，エージェントに物理的な情報を意識させないようにする．
- 複数の種類の AP が存在しても，各 AP が一意にエージェントを識別できなければならない．
- AP によって，エージェントの管理方式が異なる．
- 要求した機能が，要求先 AP では実現されていない可能性がある．
- 一般に公開されていなければならない．

そこで我々は，AP 間でエージェントのコミュニケーションをサポートするためのプロトコルとして Agent Platform Protocol（以下，APP）を提案する．APP では，１３の method と header が定義されたリクエスト／リスポンス指向のプロトコルであり，AP はそれらに従ったメッセージを用いて他の AP と協調を行う．APP は，次の機能を提供する．

- サーバを介さない Peer to Peer（以降，P2P）な AP 間ネットワーク．
- エージェントメッセージの送受信機能．
- エージェントメッセージの伝達先の検索機能．
- エージェントの移動．

AP は，APP に従ってメッセージの送信先エージェントを検索し，メッセージの伝達を行うため，エージェントは，互いの物理的なアドレスを意識せず，コミュニケーションを行うことができる．エージェント開発者は，AP に APP を実装することで，他で開発された異種の AP との通信が行えるようになり，様々な場所で開発された様々な種類のエージェントを，互いに利用できるようになる．

以下，２節では APP について述べ，３節で APP を適用した AP の例として ACZ を紹介する．そして，４節で APP を適用した ACZ を用いて，APP の機能についての検証実験を行った結果について報告する．

2　Agent Platform Protocol（APP）

APP は，エージェントの活動をサポートする AP 間の通信プロトコルであり，図 1 のように位置づけられる．APP は，低レベルの通信プロトコルとして TCP [8] を持つ．TCP では，信頼性のあるコネクション指向型の通信を提供しており，データの欠損や重複，届けられる順序の違いなどを発生しない．

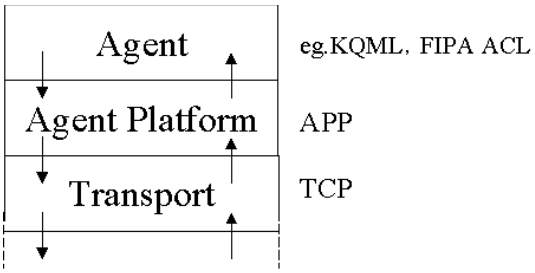

図 1　APP の位置づけ

APP の主な特徴として次の点が上げられる．

- サーバを介さない P2P な AP 間ネットワークの構築．

P2P なネットワークでは，各ネットワークの構成要素が置き換え可能なため，耐故障性が高く，一つの AP の故障が AP のネットワーク（以降，AP で構成されるネットワークを AP ネットワークと呼ぶ）に与える影響は少ない．また，サーバ／クライアントモデルで構成された AP ネットワークのように，サーバの役割を担う AP に負荷が集中するという問題はなく，更に，サーバの役割を担うような特定の AP への接続は不要で，AP ネットワークの任意な一つの AP への接続で AP ネットワークに加わることができる．

- エージェントメッセージの送受信機能の提供．

エージェントはコミュニケーション手段として，メッセージ（以降，APP によって用いられるメッセージ

表 1　APP で準備された method 一覧

分類	method	説明
Connect	LINK	AP 間の認証及び通信経路の作成
	DeLINK	AP 間の通信経路切断
	GET_LINK	要求先の AP が持つ通信経路情報の要求
	ALIVE	要求先が活動状態であるか否かの確認
Deliver	DELIVER	エージェントメッセージの伝達
	LOCATE_AGENT	エージェントの検索依頼
	GET_AGENT_LIST	要求先の AP が持つエージェント情報の要求
	PUT_AGENT_LIST	保持しているエージェントの情報を他の AP に通知
Mobile	GET_AGENT	エージェントを要求先の AP からローカルな AP に移動させる
	PUT_AGENT	ローカル AP 上のエージェントを要求先の AP に移動させる
Reply	ACCEPT	返答．要求を受理したときに用いる
	SUCCESS	返答．要求が成功したときに用いる
	FAILURE	返答．要求が失敗したときに用いる

と区別するために，エージェントメッセージと呼ぶ）を用いた情報交換を行う．

- エージェントメッセージの伝達先の検索機能の提供．

AP は，エージェントメッセージの送信先エージェントのアドレスが分からない場合，エージェントメッセージの伝達先を探さなければならない．エージェントメッセージの伝達先検索のため，APP では，（１）他の AP に検索を任せる方法と（２）他の AP から情報を集め自分自身で検索を進める方法とを提供する．

- モバイルエージェントの移動をサポート．

エージェントに，仕事の達成に必要な資源を求め AP 間を移動する手段を提供する．

2.1　AP メッセージ

APP で用いられるメッセージを AP メッセージと呼ぶ．AP メッセージは，method，header，body から構成され，この順に記述される．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，処理するのに必要な情報である．body は，各 method が対象とする任意のデータである．

AP メッセージ = {method, header, body}

2.1.1　method

AP は，エージェントのコミュニケーションや移動をサポートする環境を提供する．このため，APP では，１３の method を準備している（表 1）．

Connect，Deliver，Mobile に分類される method は，AP に対して要求を出し，その応答として，AP は Reply に分類される method を用いる．

Connect　サーバを介さないで P2P な AP ネットワークを構成するための method 群．LINK によって，AP 間での認証を行い，お互いの存在を知り，相互に信頼性のある通信を可能とする．GET_LINK は，要求先の AP が接続している AP の情報を要求する．AP は起動時に，まず LINK method を使って任意の AP に接続し，AP ネットワークに加わる．次に GET_LINK で AP ネットワークの情報を求める．新たに得た AP ネットワークの情報を利用して，複数の LINK を行うことで，特定の AP への依存度を下げることができる．DeLINK は，要求元の AP が通信を保証できなくなることを伝える．DeLINK を受け取った AP は，接続可能な他の AP との接続が分断されないように GET_LINK や LINK method を使って，ネットワークが分断されないように対処した後，DeLINK への応答を行わな

ければならない．ALIVE は，AP が実際に AP メッセージを受け付けることが可能であるかの確認を行う．

Deliver エージェントメッセージを送受信するための method 群．DELIVER は，エージェントメッセージを伝達する method では，body 部にエージェントメッセージを指定し，aimAt header(表 2 参照) でエージェントメッセージの受信側エージェントを指定する．DELIVER を受信した AP は，aimAt header が示すエージェントに body 部のエージェントメッセージを伝達する．受信側エージェントが存在する AP が分からない場合の検索手段として，LOCATE_AGENT と GET_AGENT_LIST を準備している．LOCATE_AGENT は，他の AP に検索を任せる方法であり（図 2(a)），LOCATE_AGENT を受信した AP は，aimAt で指定されたエージェントが存在する AP を探し出し，見つかれば，body 部をそのアドレス（IP アドレス/Port 番号）とした SUCCESS を返す．GET_AGENT _LIST は，要求先の AP に，それが持つエージェントの情報を要求する．これは，様々な AP からエージェントの情報を集め，その情報を用いて目的のエージェントが存在する AP を自分で見つける場合に利用する（図 2(b)）．また，PUT_AGENT_LIST は，保持しているエージェントの情報を他の AP に通知する場合に利用する．

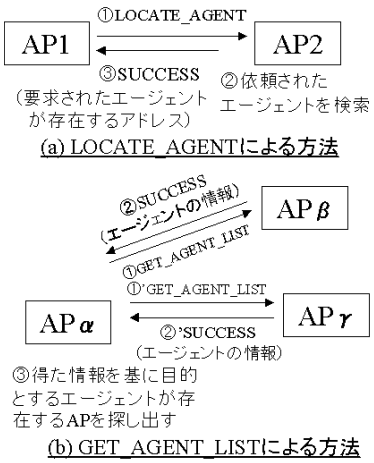

図 2　エージェントの検索方法

Mobile エージェントの移動を行うための method 群．POST_AGENT は，要求先の AP にエージェントを移動させ，GET_AGENT は，要求先の AP からエージェントを移動させる．

Reply Connect，Deliver，Mobile の各 method の応答として用いられる．ACCEPT は，各 method を受理したことを表し，FAILURE は，要求が失敗したことを，SUCCESS は，要求が成功したことを表す．

2.1.2　header

APP における header は，その役割によって，general-header，body-header，authentication-header に分けられる（表 2）．general-header は，すべての AP メッセージで必須であり，body-header は，body 部を持つ AP メッセージに必要とされる．authentication-header は，AP 間の認証が必要なときに用いられる．header は，

header = header-name ":" value

の形で表される．

2.1.3　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　APP の適用例

我々の研究室で研究・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）では，AP として ACZ （Agent Communication Zone）を利用する．本節では，まず，ACZ について述べ，次に ACZ への APP 適用例を示す．

3.1　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り，エージェント間のコミュニケーション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．通常，一つの計算機上に一つの

表 2　header 一覧

	header-name	value	説明
general-header	message-id	string	メッセージつけられた固有の ID
	from	AP-Address (IP-Address/port)	メッセージを送信した AP のアドレス
	via	*AP-Address	メッセージの経路
	refer	*message-id	メッセージの動機となった message-id
	AP-Type	string	AP のタイプ
authentication -header	authentication-type	string	認証方法
	authentication-value	string	認証キー
	authentication-expire	number	authentication-value の有効期限
	absolute	0 or 1	リンクの重要性
body-header	body-size	number	body 部のサイズ
	body-type	string	body 部のデータ形式
	aimAt	string	body 部のデータの対象

* は，0 回以上の繰り返しを表す．

ACZ が存在し，分散環境上では複数の ACZ が存在する．これらの ACZ は，ネットワークを構成し，分散した計算機の間での通信を制御する．この管理のために ACZ は，ローカル層とリモート層より構成される（図 3）．

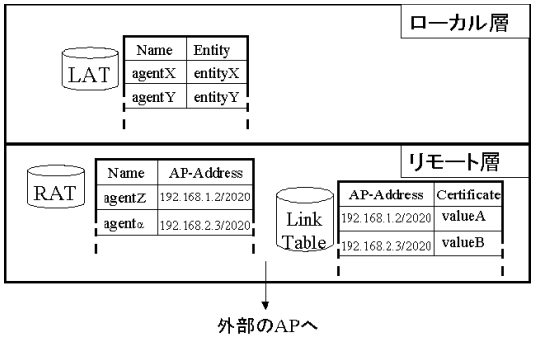

図 3　ACZ の構造

ローカル層は，ローカル ACZ 上のエージェントを管理するテーブル LAT（Local Agent address Table）を持ち，ローカル ACZ 上のエージェントへのエージェントメッセージの伝達を行う．ローカル ACZ 上のエージェントの生成または消滅時には，その名前が LAT に追加／削除される．

リモート層は，他の ACZ 上で活動するエージェントのアドレスを記したテーブル RAT（Remote Agent address Table）と，他の ACZ の認可情報を記したテーブル LT （Link Table）を持ち，他の ACZ 上で活動するエージェントへエージェントメッセージを伝達する際に利用される．

3.2　ACZ への APP の適用

ACZ のライフサイクルは，図 4 のように示される．

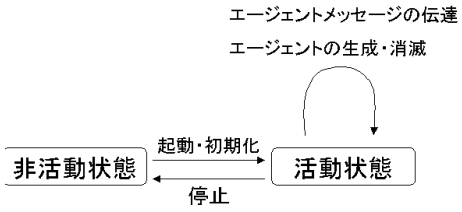

図 4　ACZ のライフサイクル

ACZ は生成されると直ぐに起動・初期化され，非活動状態から活動状態になる．活動状態中は，エージェントの生成や消滅，エージェントメッセージの伝達が行われる．そして，最後に停止され，活動状態から非活動状態に変化する．このように，ACZ の動作は，

起動・初期化，エージェントの生成・消滅，エージェントメッセージの伝達，停止の４つに分けられる．このうち，エージェントの生成・消滅はローカル層での動作であり，APP は関係しない．そこで，エージェントの生成・消滅以外の３つの ACZ の動作について APP がどのように適用されているかを示す．

3.2.1　ACZ の起動・初期化

KODAMA は，Java で開発されており，ACZ は次のコマンドで起動・初期化される．

java acz.ACZ (使用 port) (接続 IP:port)

図 5 のような状態において，

java acz.ACZ 2000 192.168.0.1:2020

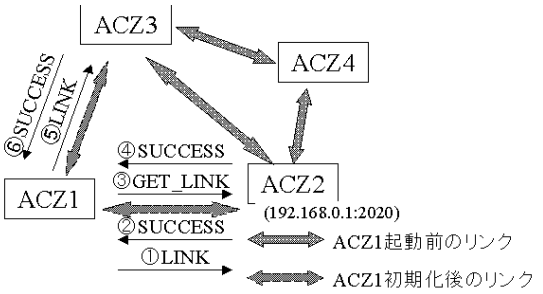

図 5　ACZ の初期動作

で ACZ1 を起動すれば，ACZ1 は，まず，LINK を使って，ACZ2 (192.168.0.1 /2020) との間で，相互通信のためのリンクを作る．これは，ACZ1 と ACZ2 が各自の LT に相互登録する（図 5 ①，②）ことによって実現する．次に，ACZ2 の持つ ACZ ネットワークの情報を集め（図 5 ③，④），集めた ACZ 中の任意の ACZ との間で相互の通信経路を作成する（図 5 ⑤，⑥）．図 5 では，ACZ3 とだけ通信経路を作成しているが，ACZ の設定[†1]によって，任意の数の通信経路を作成することができる．このことで，ACZ1 は特定の ACZ への依存度を減らすことができる．

3.2.2　ACZ の停止

図 6 のような状態において，ACZ1 が停止すれば，ACZ2 と ACZ3 の間での通信経路が無くなり，ネットワークが２つに分断される．このため，ACZ1 から DeLINK を受け取った ACZ2 は（図 6 ①），ACZ ネットワークの分断をさけるため，GET_LINK で停止する ACZ1 から ACZ ネットワークの情報を受け取り（図 6 ②，③），ACZ3 との間にリンクを構成する（図 6 ⑤）．

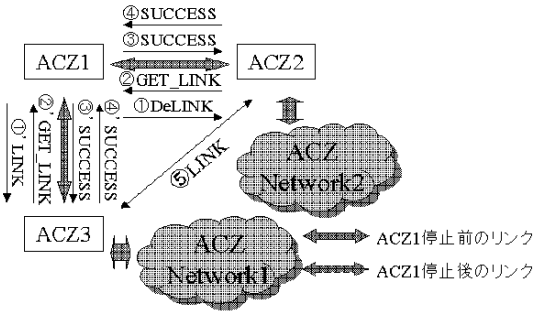

図 6　ACZ の停止動作

3.2.3　メッセージの伝達

エージェントメッセージの受信側エージェントが LAT のエージェントであれば，ローカル層でエージェントメッセージの伝達を行う．RAT に登録されているエージェントであれば，RAT からアドレスを求め，DELIVER を用い，エージェントメッセージを伝達する．LAT，RAT 双方に登録されていないエージェントであれば，そのエージェントの検索が行われる．

ACZ でのエージェントの検索は，KODAMA エージェントの論理的構造を参照し，LOCATE_AGENT を用いて行う．(図 7)．KODAMA では，エージェントは階層的に配置され，エージェントの名前が階層の場所を示す．例えば，”root・A”の名前を持つエージェントは，”root”の名前を持つエージェントを上位階層に持ち，”root・A・child”の名前を持つエージェントを下位階層に持つ．ACZ は，検索依頼を表す LOCATE_AGENT を用いて，次の方針でエージェントの検索を行う．

(a) 検索は RAT，LAT から refer header に含まれない ACZ で活動するエージェントで階層的に最も近いエージェントを見つけ，見つけたエージェントが目的のエージェントであれば，始めに LOCATE_ AGENT を出した ACZ に SUC-

†1　ユーザによって，ACZ 初期化時に作成される通信経路の枠組みが指定される．

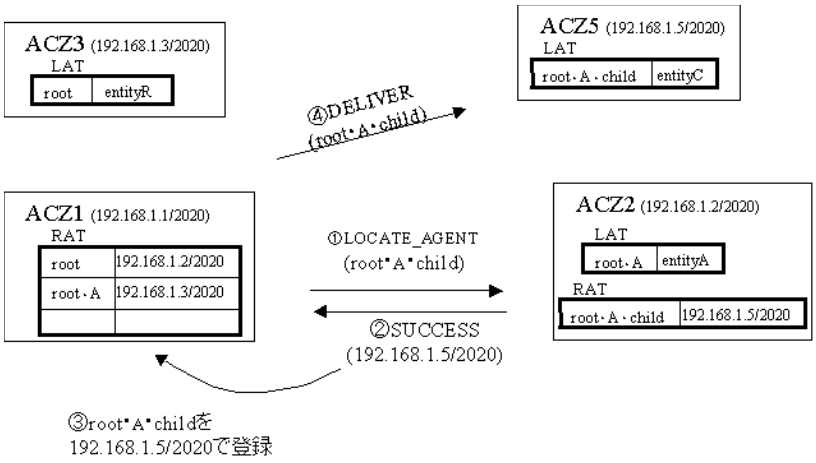

図 7　ACZ でのエージェント検索方法

CESS で応答し（図 7 ②），目的のエージェントでなければ，見つけたエージェントが活動する ACZ に LOCATE_ AGENT で検索を依頼する（図 7 ①）．LAT，RAT が共に要素を持たない場合は，最初に LT に登録された ACZ に LOCATE_ AGENT で検索依頼を出す．

(b) LOCATE_AGENT を受け取った AP は，via header を見て，すでに受け取っている LOCATE_ AGENT 要求であれば，始めに LOCATE_ AGENT を出した ACZ に FAILURE で応答する．まだ，受け取っていない LOCATE_ AGENT 要求であれば，(a) を行う．

(c) LOCATE_AGENT の応答として SUCCESS を受け取った ACZ は（図 7 ③），検索結果を RAT に登録する．応答として FAILURE を受け取れば，検索失敗である．

結果として，ACZ は，目的のエージェントを見つけた場合，DELIVER で（図 7 ④），エージェントメッセージを伝達する．

4　実験

APP の有効性を検証するために，APP を実装した ACZ で実験を行った．実験では，各計算機に一つの ACZ を準備し，合計 8 台の計算機を利用した．各 ACZ は，ランダムに停止・起動を繰り返すこととし，起動時には接続する ACZ を活動中の ACZ からランダムに選び，停止時には LAT，RAT，LT のテーブルを初期化した．また，各 ACZ には，１０のエージェントを準備し，それぞれに適当なエージェントとコミュニケーションを行わせることとした．この条件で 1 時間経過したあとにすべての ACZ を停止させた．このときの実験結果を表 3 に示す．

表 3　実験結果

総 AP メッセージ数	2831
総起動・停止回数	44
エージェントのコミュニケーション数	906

4.1　ネットワークに関する評価

実験では，ACZ の起動・停止が共に 44 回行われた．IP=192.168.35.76，port=2200 の ACZ 停止時に発生した AP メッセージを例 1 に示す．†2．

例 1 では，192.168.35.76 の停止によって AP ネットワークが分断されることを防ぐため，GET_LINK で AP ネットワークの情報を求め，192.168.35.64 との間に新しくリンクを作成した．192.168.35.73 との間には，すでにリンクを持っていたため，新しくリンクは構成しなかった．

結果，44 回の ACZ の起動・停止にも係わらず，AP

†2　すべての ACZ で 2200 番ポートを使用しているため，ポート番号は省略した．

1.DeLINK : From 192.168.35.76 to 192.168.35.65
2.ACCEPT : From 192.168.35.65 to 192.168.35.76 --(1の応答)
3.GET_LINK : From 192.168.35.65 to 192.168.35.76
4.SUCCESS : From 192.168.35.76 to 192.168.35.65,
[192.168.35.73, 192.168.35.64] --(3の応答)
5.SUCCESS : From 192.168.35.65 to 192.168.35.76 --(1の応答)
6.LINK : From 192.168.35.65 to 192.168.35.64
7.SUCCESS : From 192.168.35.64 to 192.168.35.65 --(6の応答)

例 1: ACZ 停止時の AP メッセージ

ネットワークから孤立した ACZ は，発生しなかった．すなわち，本稿で提案する APP を使用することで，サーバを介さない P2P な AP 間ネットワークを構成でき，AP ネットワーク接続の頑健性が確認できた．

4.2 エージェントのコミュニケーションに関する評価

実験では，906 回のエージェントのコミュニケーションが試みられた．このうち，903 回のコミュニケーションが正常に行われた．残り 3 回については，相手のエージェントが存在しないため，コミュニケーションを行わないものであった．この実験で，エージェントは，相手の物理的位置を意識することなく，コミュニケーションが正常に行えた．このように APP を使用することで，ACZ はエージェントの位置透過性を与えることができた．

5 おわりに

本稿では，エージェントの活動を促進する環境 AP を相互運用するためのプロトコルとして APP を提案した．APP は，サーバを介さない P2P な AP 間ネットワークで，エージェントが物理的環境を意識することなくコミュニケーションを行うための環境を与えることができる．

この検証のため，我々の研究室で開発している ACZ に，APP を実装した結果，APP を実装した ACZ は，動的な経路変更機能を持つ P2P な ACZ ネットワークを構成することができた．

今後の課題は，ACZ 以外の AP に APP を実装し，ACZ と相互運用が可能であるかを検証することと，APP で提供されるエージェント検索方法の効率を検証することである．

参考文献

[1] G. Zhong, K. Takahashi, T. Helmy, K. Takaki, T. Mine, S. Kusakabe and M. Amamiya, “KODAMA: As a Distributed Multi-agent System”, Proceedings of the 7th International Conference on Parallel and Distributed Systems: Workshops, IEEE Computer Society Press, pp.435–440, Iwate, Japan, July.2000.

[2] 高橋 健一, 高木 幸一郎, 鍾 国強, 雨宮 聡史, 峯 恒憲, 雨宮 真人,“分散マルチエージェントシステム KODAMA”, 「ソフトウェアエージェントとその応用」特集ワークショップ,SAA2000, 157–164, 2000

[3] T. Iwao, M. Takada, M. Amamiya, “Flexible Multi-Agent Collaboration using Pattern Directed Message Collaboration of Field Reactor Model”, Approaches to Intelligent Agents - Second Pacific Rim International Workshop on Multi-Agents, LNAI, vol.1733, Springer, pp.1–15, 1999.

[4] N. Minar, M. Gray, O. Roup, R. Krikorian and P. Maes, “Hive: Distributed Agents for Networking Things”, IEEE Concurrency, vol.8, no.2, 2000.

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

[6] T. Finin, Y. Labrou, and J. Mayfield, “KQML as an agent communication language”, In Software Agents, MIT Press, Cambridge, pp.291–316, 1997.

[7] E.D. Pietro, A.L. Corte, O. Tomarchio, A. Puliafito, “Extending the MASIF Location Service int the MAP Agent System”, IEEE Symposium on Computers and Communications, 2000.

[8] D. Comer, D. Stevens（村井 純, 楠本 博之（訳））, “TCP/IP によるネットワーク構築”, 共立出版, 2001.