コンテキストに基づいたグループ形成とグループ通信機能を提供する携帯電話向けミドルウェア

西垣弘二[†] 安本慶一[†] 梅津高朗[††]
東野輝夫[††] 伊藤実[†]

本論文では，不特定多数携帯電話ユーザ間の協調型アプリケーションを容易に開発することを目的に，地理的位置やコンテキスト（共通の話題など）に関する条件により動的にグループを形成する機能と，グループメンバ間で，マルチキャスト，同期，排他制御などのグループ通信機能を提供するミドルウェアの提案を行う．市販の携帯電話上の Java 実行系では，ユーザプログラム間の直接通信機能が利用できず，メモリ容量や通信プロトコルなども制限されているため，提案ミドルウェアでは，ユーザ端末側プログラムの大部分を，サーバ上のエージェントとして実行し，グループ形成やグループ通信を実現するための一連のメッセージ交換を，サーバ内のプロセス間通信で実現する方式を考案し，実装した．本ミドルウェアにより典型的な協調型アプリケーションを容易に設計でき，10,000 台規模のグループメンバ間の通信が実用的な速度で実行可能なことなどを確かめた．

Middleware for Cellular Phones Providing Group Formation Based on Context and Group Communication Facility

Kouji Nishigaki,[†] Keiichi Yasumoto,[†] Takaaki Umedu,[††]
Teruo Higashino[††] and Minoru Ito[†]

In this paper, we propose a middleware library for efficiently developing distributed cooperative applications consisting of a large number of cellular phone users. Our middleware provides (1) a dynamic group formation mechanism depending on users' locations and preferred subjects and (2) a group communication mechanism called *multi-way synchronization* for multicasting, synchronization and mutual exclusion. Most of Java executors on cellular phones do not support direct communication among user programs. Usable resources are also restricted. Therefore, in our middleware, most parts of user programs are executed on their servers as agents. Group formation and group communication mechanisms are implemented as inter-process communication on the server, and only the user-interface parts are executed on the cellular phones. From some experiments, we have confirmed that group applications consisting of ten thousands of cellular phones can be easily developed using the middleware, and that their group communication performance is reasonable for practical use.

1. まえがき

近年，携帯電話端末の高性能化が目覚ましい．Java などで記述されたゲームなどのアプリケーションの実行に加え，静止画やさらには動画の撮影を行い，GPS で得た現在位置を付加し，仲間に送付することも可能になってきている．これらの機能を用いて，旅先で，ある観光名所やレストランなどについての情報を持っているユーザと接続して生の情報（写真などを含む）を得たり，あるイベント会場で，場所を限定してユーザを募り，地域限定の情報を配布したりゲーム大会を行ったりするなどのアプリケーションが実現できる．

このような，携帯電話上でのユーザ協調型アプリケーションを容易に開発するには，あらかじめ指定した条件やユーザの位置情報をもとにしたグループ形成機構や，グループメンバ間での効率の良いデータ交換機構，さらには，メンバ間で同期や排他制御などの機構を提供するミドルウェアがあれば便利である．

本論文では，携帯電話ユーザ間の動的なグループ形成機構と，**マルチランデブ**（複数並行プロセス間で，

† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

指定された条件が成立するときに，同期通信によりデータ交換を行う機構)[1] によるグループメンバ間の高度なインタラクション機構を提供するミドルウェアの提案を行う．

上述したような高度なアプリケーションを実現するうえでは，現在市販されている携帯電話端末上の Java 実行系は，(1) ユーザプログラム間の直接通信機能が利用できない，(2) 利用できるリソース（CPU，メモリ，バッテリなど）が限られている，などの問題を有する．これらの問題を解決するため，提案ミドルウェアでは，ユーザ端末側プログラムの大部分を，サーバ上のエージェントとして実行する方式を採用する．マルチランデブを実現するための一連のメッセージ交換をサーバ内のプロセス間通信で実現し，入力インタフェースと表示部分（UI）のみを携帯端末上で実行する．エージェントと UI は，必要なときのみ，サーバ上で実行されるサーブレットを介して HTTP で通信させる．また，不特定多数携帯電話ユーザ間で効率良いグループ形成を行うため，地理的条件およびユーザコンテキスト（キーワード集合の間の関連度，日時，数値，ブーリアン値）をグループ形成時の条件に用いることとした．

提案ミドルウェアを，Java の Servlet および MIDlet からなるシステムとして実装した．実験の結果，提案手法により，グループ形成のためのコンテキスト距離の条件判定が十万台規模で，マルチランデブによるグループ通信が一万台規模の端末間で実用的な速度で動作可能なことを確かめた．また，本ミドルウェアを用いて，複数ユーザによるクイズゲームアプリケーションを実装した結果，実用上十分な性能が得られることを確かめた．

2. 関 連 研 究

グループ通信に関しては，近年，P2P 環境において，ユーザ端末間にオーバレイリンクからなるマルチキャスト配送木を構築し，効率の良いグループ通信を実現するための研究が多くなされている[2)~4)]．また，モバイルアドホックネットワーク上において，端末の位置に基づいて，マルチキャストによるグループ通信を実現するための研究もなされている[5),6)]．これらマルチキャストを利用したアプリケーションでは，あるコンテンツを受信するためには，既存のグループに参加する必要があり，ユーザ同士が，共通の話題などをもとに自律的にグループを形成し，頻繁にインタラクションを行うといったアプリケーションには向いていない．

一方，携帯端末向けのミドルウェアとして，FIPA[7)] と呼ばれるマルチエージェントプラットフォームをハンドヘルド PC や携帯電話などのリソースの限られた端末で実行できるよう軽量化した LEAP[8)] や Crumpet[9)] が提案されている．また，実際に Java 搭載携帯電話で動作する FIPA 準拠エージェントプラットフォームの試作[10)] や，e-jumon[11)]，picoPlangent[12)] といった携帯電話上で動作するエージェントプラットフォームが開発されている．これらの研究は，複雑な機能をサーバ側に肩代わりさせ実行するという点で提案手法と共通点がある．しかし，提案手法では，多数携帯電話ユーザ間で，話題や位置に基づいてグループを動的に形成し，形成したグループ内でマルチランデブによる密なグループ通信機能を提供するという点で大きく異なっている．

一方，文献 13) では，情報をオブジェクトとして扱い，場所と時間の有効範囲をタグとして付加することで，効率良く必要な情報を得るための枠組みおよびシステムとして，SpaceTag が提案されている．提案ミドルウェアでは，携帯電話ユーザとグループを形成し，情報交換することを目的としているため，SpaceTag とはアプリケーションの対象が異なる．提案ミドルウェアでは，通信相手（グループ形成対象）として携帯電話ユーザでなく，指定した場所および時間の条件を満たすときにのみ情報を発信するようなエージェントとしてサーバに登録・実行することで，携帯電話ユーザに対し，場所および時間的に限定した情報を効率良く配信できる．以上の点から，提案ミドルウェアは SpaceTag の実装環境としても適していると考えている．

さらに，グループを形成する手段として様々な方法が提案されている．文献 14) では，ある地理的範囲内で興味や知識が類似するユーザからなるコミュニティの実時間形成を可能にするモバイルコミュニティ形成方式を提案している．提案ミドルウェアでは，ある地理的範囲内に加えユーザ間の距離やその他の条件を扱える点，グループ内の高度なインタラクション機能を提供する点で異なる．また，文献 15) では，嗜好情報と意味情報を用い，P2P 技術を用いたブローカレスな情報検索手法を実現しているが，情報を検索することが主な目的であり，提案ミドルウェアが，グループを形成しグループメンバ間で高度なインタラクションを提供する点で異なる．

3. 提案する携帯電話用ミドルウェア

我々は，文献 16) において，無線環境で複数移動

表 1 ミドルウェア API の一覧表
Table 1 Middleware's API.

API	機能
advertise	条件を指定してグループメンバを募集する.
cancel	グループ ID を指定してグループメンバの募集を中止する.
participate	条件を指定してグループに参加を申し込む.
disconnect	通信関係を破棄しグループから離脱する.
DisconnectException	グループから離脱したことを通知する.
executeEvent	グループのメンバ間でデータの交換または同期を行う.

ノードがアドホックにグループを形成し，グループのメンバ間で通信を行うアプリケーションのためのミドルウェア（以下，既存ミドルウェア）を実現している．既存ミドルウェアは，無線範囲へのグループメンバの募集および参加応答によるグループ形成機構を提供し，グループ形成のための条件として端末位置と端末 ID を指定することができる．また，形成したグループ内において，同期，排他制御，マルチキャストなどの通信機構（マルチランデブ[1] のサブクラス）を提供している．本論文で提案するミドルウェアでは，携帯電話環境において既存ミドルウェアと同等の機能を提供するために，既存ミドルウェアのモデルを携帯電話環境向けに拡張する．既存ミドルウェアでは，グループメンバの募集および参加応答は無線範囲に限定される．しかし，本論文で提案するミドルウェアにおいて，携帯電話間の通信は基地局を介した通信であり，無線範囲の概念がなく，グループメンバの募集および参加応答はアプリケーションに参加するすべての端末が対象となる．そこで，効率良く参加メンバを選択することを目的に，グループ形成のための条件として指定可能な項目を拡張した．提案するミドルウェアの API 一覧を**表 1** に示す．

3.1 グループの動的形成機構

各携帯電話端末に対する動作プログラムのインスタンスを**エージェント**と呼ぶこととする．本ミドルウェアは，複数の携帯電話端末のそれぞれに対応するエージェント群が，グループメンバ募集メソッド $advertise(ch\_rel, val\_list, cond)$ と，グループへの参加要求メソッド $participate(val\_list, cond)$ を用いて動的にグループを形成する機構を提供する．ここで，ch_rel には，グループメンバ間の**通信関係**（マルチキャスト，排他制御あるいはそれらの組合せを仕様記述言語 LOTOS[1] 風の構文を用いて指定する）を指定し，val_list，$cond$ には，それぞれ，グループの形成を制約するための値のリストと条件式を指定する．

LOTOS の概要

ここで，LOTOS[1] は，ISO により開発された通信プロトコル向けの仕様記述言語であり，並列性を含むシステムの挙動を簡潔に表すための強力な構文を有する．LOTOS では，システムをいくつかの並列プロセスとして記述し，各プロセスの動作（**動作式**）は，**イベント**と呼ばれるゲート（環境との相互作用点，以下チャネルと呼ぶ）を介したプロセス外部との相互作用（値の入出力）の実行系列として定義される．LOTOS では，同期並列オペレータ $(A1|[g]|A2)$ を使用することにより，複数のプロセスが指定されたゲート上のイベントを同時に実行しデータ交換を行う，といった動作を記述することができる（**マルチランデブ**と呼ばれる）．LOTOS では，複数プロセスによるマルチランデブは，$A1|[g,h]|((A2|[g]|A3)|[g]|A4)$ のように 2 分木の構文で記述される．本論文では，同じチャネルに接続されたプロセスは，$|[g]|\{A1, ..., An\}$ のように集合を用いて表記することとする．また，集合で表記されたプロセス $|[g]|\{A1, ..., An\}$ を別のプロセス P と同期並列オペレータ $|[g,h]|$ で結合する際には，$P\ |[g,h]| - |[g]|\{A1, ..., An\}$ のように $-$ を用いて，オペレータ名とプロセス名の一部を区別できるよう表記する．

通信関係の指定

通信関係は，$\sharp 1|[g,h]| - |[g]|\{\sharp 2\}$ のように指定する（{} は集合を表す）．ここで，g，h はチャネルであり，$|[g,h]| - |[g]|$ は通信のタイプ（マルチキャスト，排他など）を表す（3.2 節で詳述）．$\sharp 1|[g]|\sharp 2$ は，チャネル g を介して，$\sharp 1$ と $\sharp 2$ が同期することを表している．ここで，非同期のチャネルを指定する場合は，g/as のように，チャネル名の後に $/as$ を付ける．また，$\sharp 1|[g,h]| - |[g]|\{\sharp 2\}$ において，$-$ は，チャネル間で同期関係が成立していることを表しており，$\{\sharp 2\}$ に含まれるエージェント間で g を介して同期し，$\sharp 1$ が g または h を介して $\{\sharp 2\}$ のエージェント群と同期することを表している．さらに，$\sharp 1$ は $advertise$ を実行したエージェントが接続される点，$\{\sharp 2\}$ は $participate$ を実行した複数のエージェントが接続される点である．

条件がマッチする $advertise$ と $participate$ の組が実行されることで，それらを実行したエージェント間に指定した通信方法による通信関係が確立され，**エージェントグループ**と呼ばれるグループが形成される．以降，他のエージェントが $participate$ を実行すると，同様に条件式を判定し，条件がマッチすることでエージェントグループに参加することができ，結果として動

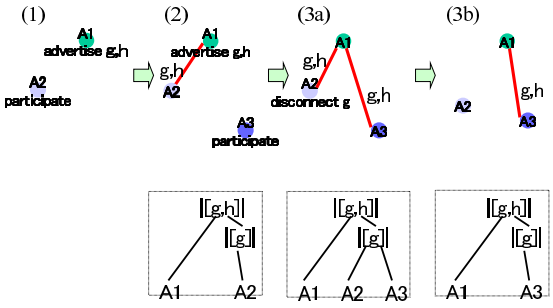

図 1 提案ミドルウェアにおける通信関係の確立・開放
Fig. 1 Establishment and disconnection of communication relation in proposed middleware.

的にグループを拡大していくことができる．*advertise* メソッドを実行することにより，アプリケーションを実行している間，グループメンバの募集を行い続けることができ，実行結果としてグループ ID が返される．ただし，条件に時間やメンバ数を指定することにより募集期間やメンバ数を制限することができる．ここで，グループメンバの募集を中止する場合には，グループ ID を指定して *cancel* メソッドを実行する．また，*participate* メソッドを実行することにより，グループメンバの募集を行っているグループに対して参加を試みることができ，実行結果としてグループへの参加の可否が返される．可の場合は返される値がグループ ID となる．

グループで共有される通信関係における各エージェントの接続点は，結合時に得られる ID によって区別できる．また，エージェントは，ID を指定して切断メソッド（*disconnect*）を呼び出すことで共有している通信関係を破棄し，グループを脱退できる．他方のエージェントの切断メソッド呼び出しによる通信関係の破棄は，切断例外（*DisconnectException*）がグループの残りのメンバに通知される．

メンバ募集の対象となるエージェントの集合を**アプリケーションドメイン**と呼ぶ．携帯電話ユーザが該当アプリケーションを実行することで，対応するエージェントがアプリケーションドメインに追加される．たとえば，**図 1** において，あるエージェント $A1$ および $A2$ がアプリケーションドメインに存在し，$A1$ が通信関係 $\sharp 1|[g,h]| - |[g]|\{\sharp 2\}$ と条件 $C1$ を指定して *advertise* を実行し，$A2$ が条件 $C2$ を指定して *participate* を実行したとする．この際，$A1$ と $A2$ の間で指定した条件（$C1$ と $C2$）が成立するとき，指定した通信関係が共有され，A1 と A2 からなるエージェントグループが形成される（図 1 (2)）（このとき通信関係は $A1|[g,h]| - |[g]|\{A2\}$ のように表される）．新たにアプリケーションドメインに入ったエージェント $A3$ が *participate* を実行した場合には，A1，A2，A3 の間に図 1 (3) に示すような通信関係を持つエージェントグループが形成される（このとき通信関係は $A1|[g,h]| - |[g]|\{A2, A3\}$ のように表される）．また，通信関係の共有の破棄は，*disconnect*（グループ ID）メソッドを実行することで行う（図 1 (3a)，(3b)）．図 1 下部の構文木は，エージェント間の通信関係を LOTOS 風の構文を用いて表したものである．

エージェントグループのメンバ間では，指定した通信関係に従い，マルチランデブ[1] による同期通信を行うことができる．詳しくは 3.2 節で述べる．

(1) グループ形成の条件 不特定多数の端末間でのグループ形成を効率良く行うため，提案するミドルウェアでは，(1) GPS などで取得される物理的な距離，のほかに，ユーザのコンテキストとして，(2) 与えられたキーワード集合の間の関連度，(3) 日時，(4) 数値，(5) ブーリアン値，を用いることとした．以上の (1)〜(5) の条件を適宜組み合わせて，グループ形成のための条件を指定する．たとえば，沖縄県内で琉球舞踊に興味のある 20 代の女性を条件に参加募集をかける場合，条件 (1) として位置が沖縄県内かを，条件 (2) としてキーワード “琉球舞踊” を，条件 (4) として年齢 “20〜29” を，条件 (5) として “女性か否か” を指定する．参加要求を実行する側は，値リストとして，位置，キーワード，年齢，性別を指定し，これらが上記の条件に一致する場合，グループに参加できる．

(2) のキーワードの間の関連度を数値化するため，与えられたキーワード集合 A，B に対する，**コンテキスト距離**を次式で定義する．

$$CDist(A,B) = 1 - \frac{|\,A \cap B\,|}{|\,A \cup B\,|}$$

ここで，$|X|$ は集合 X の要素数を表している．コンテキスト距離は，A，B の間にまったく関連がない場合に 1，関連が最も高い場合に 0 となる．たとえば，$A =$ { 子供服，ジーンズ }，$B =$ { ジーンズ } の場合のコンテキスト距離は 0.5 となる．ここでは扱わないが，テキストマイニング技術を用いてキーワードの関連度を数値化して利用することも考えられる．

3.2 マルチランデブによるグループ通信

以後，それぞれのチャネルへの入出力動作をイベントと呼ぶ．$g?x[p(x)]$ は，チャネル g 上の入力イベントであり，g から値を受け取り，式 $p(x)$ を満たす場合のみ変数 x に代入する（イベントは $p(x)$ を満たさないとき実行されない）．$g!f(x)$ は，チャネル g 上の出力イベントであり，式 $f(x)$ の値を g に出力する．

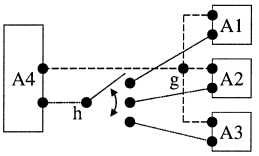

図 2 マルチキャストと排他制御
Fig. 2 Multi-cast communication and exclusive control.

本モデルでは，エージェントグループのメンバは同期通信を行う．

たとえば，4 つのエージェントが，通信関係が $A4|[g,h]| - |[g]|\{A1, A2, A3\}$ であるグループを形成したとする．この場合，チャネル g は，エージェント間の共有データバスとして振る舞う．すべてのメンバのエージェントに対するマルチキャストは，**図 2** のように実現される．A1，A2，A3 が入力イベントを実行し，A4 が出力イベントを実行したとすると，A4 が出力したデータは，A1，A2，A3 がイベントで定義した変数に代入される．

一方，チャネル h は，図 2 のように，サブエージェントグループ（$A1$，$A2$，$A3$ からなる）と $A4$ の間のスイッチングバスとして振る舞う．エージェント間で排他的に共有リソースにアクセスするメカニズムを容易に指定できる．A1，A2，A3 が出力イベントを実行し，A4 が入力イベントを実行したとすると，A1，A2，A3 が出力したデータのいずれかが，A4 がイベントで定義した変数に代入される．

さらに，もし $A1$，$A2$，$A3$，$A4$ の各々が，チャネル g と h 上で 2 つのイベントのどちらかを実行可能なら，g 上の組または h 上の組が非決定的に選択される．

グループ通信のためのプリミティブ エージェントが共有している通信関係を介してマルチランデブによる同期通信を行うためのメソッド（**executeEvent**）を定義した．各エージェントは，チャネル名を指定して **executeEvent** メソッドを実行することで，メンバ間の同期，排他制御，マルチキャストが可能となる．また，マルチキャストの場合には，通信関係におけるチャネル名を g/as のように指定することで，プロセス間で非同期通信を行うよう指定することが可能である．

3.3 携帯電話アプリケーション例

図 3 に示すような参加者を制限した早押しクイズゲームアプリケーションを，提案ミドルウェアを用いて実現する例を説明する．

システム全体は，主催者 1 人と任意の数（n 人とする）の参加者で構成され，主催者は，**図 4** に示す通信

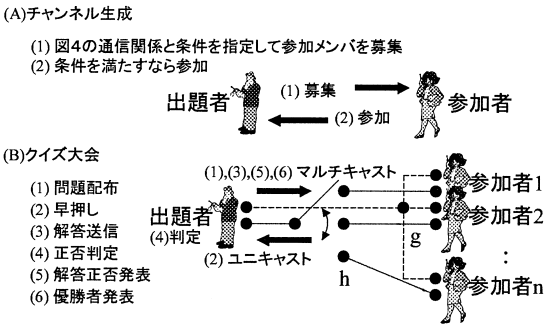

図 3 アプリケーション例（早押しクイズ）
Fig. 3 Example of application (quiz competition).

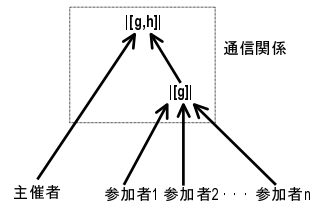

図 4 アプリケーション例における同期判定木
Fig. 4 Synchronization tree in example of application.

関係と，位置情報，およびクイズのテーマ（キーワード）をグループ形成の条件に指定して，クイズに参加するメンバの募集を行う．参加者も同様に，位置情報と参加したいクイズのテーマを指定してグループへの参加要求を行う．両者の指定した位置情報とテーマのコンテキスト距離が主催者の指定する値以下であるならば，参加者はクイズに参加できる．ここで，主催者は，ある一定の時間の間，メンバの募集を継続的に行い，参加者は，この間，クイズに参加できる．

参加の際には，データ共有用のチャネル g と排他制御用のチャネル h の 2 つのチャネルが図 4 の通信関係に従って共有される．問題の配布は，主催者からのチャネル g への出力イベントとして記述する（図 3 (B)-(1)）．また，各エージェントは一意な ID を持つものとし，出題に対する解答は，同時にはたかだか 1 人のみしか行えないようにしたい．そこで，出題に対する解答は，参加者からのチャネル h への自分の ID の出力イベントとして記述する（図 3 (B)-(2)）．複数の人が解答を実行しようとした場合には，通信関係（図 4）に従い，このうち 1 人の出力イベントが選択されて実行を許可されるため，排他的に 1 人のみ解答を許可できる．解答を許可された参加者の端末は，解答のデータをチャネル g を介して出題者，参加者を含む全端末に送信し，各端末はそれらのデータを受信して表示する（図 3 (B)-(3)）．

ミドルウェアを用いた実装は，**図 5** (a) および (b)

```
//グループメンバの募集
advertise(通信関係,位置情報,クイズのテーマ,募集人数);
//十問出題する
for(int i=0;i<10;i++){
    //問題の配付
    executeEvent(g!question);
    //解答の受信
    executeEvent(h?x);
    //正否の判定
    ……;
    //解答正否の発表
    executeEvent(g!correct);
}
//優勝者の発表
executeEvent(g!winner)
```

(a)

```
//グループへの参加応募
participate(位置情報,クイズのテーマ);
//十問出題される
for(int i=0;i<10;i++) {
    //問題の受信
    executeEvent(g?x);
    //問題の解答
    executeEvent(h!answer);
    //解答正否の受信
    executeEvent(g?y);
}
//優勝者の受信
executeEvent(g?z)
```

(b)

図 5 擬似コード
Fig. 5 Pseudo code.

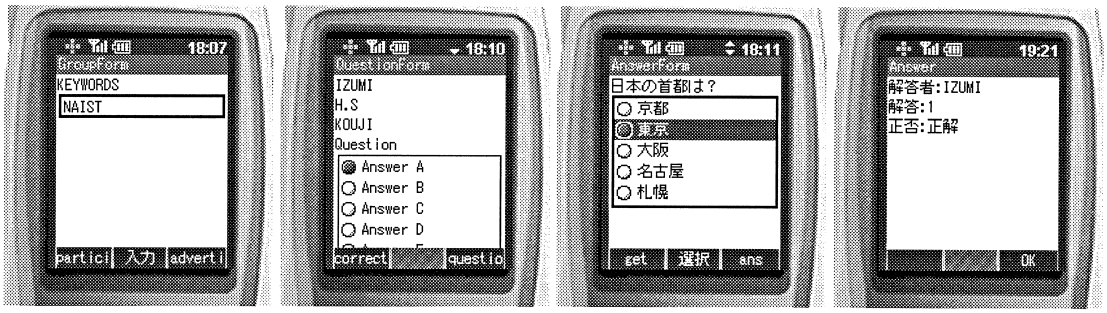

図 6 アプリケーション実行時のスナップショット
Fig. 6 Snapshots of application execution.

のようになる．

本アプリケーションを MIDlet として実装した．エミュレータによるアプリケーションの実行時のスナップショットを**図 6** に示す．左から，グループメンバ募集時，グループ形成時，問題解答時，解答結果表示時の様子である．

4. 提案ミドルウェアの実現方針

3 章のミドルウェアの各機能を携帯電話上で実現する際には，市販の携帯電話上の Java 実行系における，(1) 通信プロトコルとして http しか使えず，端末間の直接通信が行えない，(2) リソース（CPU，メモリ，バッテリなど）が貧弱である，といった問題点をいかに解決するかがポイントになる．

本論文では，端末側プログラムの大部分をサーバ上のエージェントとして実行する方法を採用する．また，上記 (1) の通信プロトコルの問題を解決するために，マルチランデブの実現に必要な端末間のメッセージ交換をサーバ上のエージェント間の通信により実現し，入力インタフェースと表示部分のみを携帯端末上で実行させる．また，携帯電話端末とサーバ上のエージェント間では，必要なときのみ通信を行わせることで，全体的な通信量を削減する．よって，マルチランデブによる同期実行を行った場合，エージェント間のインタラクションはサーバ上で完結するため，エージェント—携帯電話間の通信が切れていても処理が停滞することはない．

提案ミドルウェア上で動作させるアプリケーションでは，**図 7** のように，各端末で動作する Java プログラムを，サーバ上で動作するエージェントと携帯電話上で動作するユーザインタフェースとに分けて実現する．ユーザが携帯電話上のアプリケーションを起動すると，ユーザインタフェース部分が携帯電話に，対応するエージェントのプログラムがサーバにそれぞれロードされ実行される．エージェントは，サーブレットによって起動されるオブジェクトとして実行され，エージェントとユーザインタフェースは，サーブレットを介して，HTTP を使って RMI を模擬した通信を行う．エージェントは，グループ生成時に，後で述べる通信関係オブジェクトを共有し，マルチランデブによる同期通信を行う．

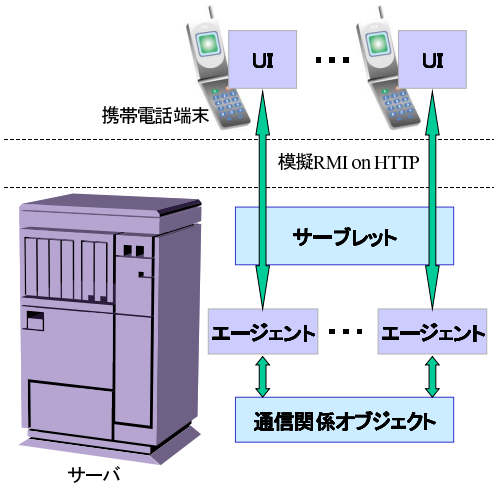

図 7 ミドルウェア実装のアーキテクチャ
Fig. 7 Outline of middleware architecture using servlet.

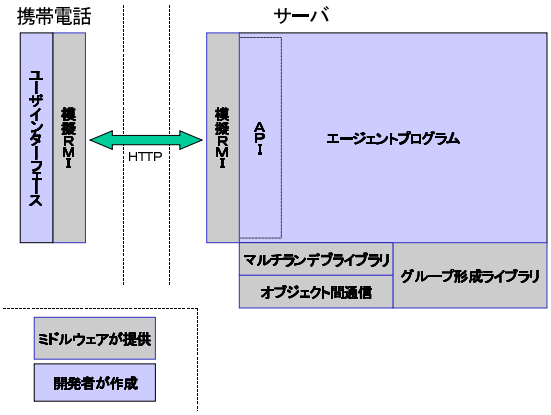

図 8 ミドルウェアの構成
Fig. 8 Block diagram of middleware.

5. ミドルウェアの実装

ミドルウェアの実装は，4 章で述べたように，大きく，サーバ側と端末側の実装に分けられる．本ミドルウェアを，**図 8** に示すように，携帯電話上のユーザインタフェースからサーバ上のエージェントプログラムを操作するための模擬 RMI，グループを形成するためのグループライブラリ，グループ通信を行うためのマルチランデブライブラリから構成した．以下に，詳細を記述する．

5.1 サーブレットの実装

エージェントを起動・管理する機能と，エージェントとユーザインタフェースの通信を媒介する機能を持つ，**LotosMServlet** クラスを実装した．**LotosMServlet** では，エージェントを起動するための **executeApp** メソッドと，エージェントにアクセスするための **executeMethod** メソッドを定義している．

executeApp/executeMethod メソッドの実装 LotosMServlet クラスにおいて，エージェントを起動するための executeApp メソッドを定義した．executeApp メソッドの引数として，エージェントの動作を定義したクラス名とユーザの ID を用いる．executeApp メソッドでは，クラス名から Java のリフレクション機能を用いてエージェントを起動する．また，起動されたエージェントは LotosMServlet クラスによって管理される．

エージェントにアクセスするためのメソッドとして executeMethod メソッドを定義した．executeMethod メソッドの引数は，ユーザの ID，アクセスしたいメソッド名，引数の数と引数を文字列にして “,” で区切ったものを用いる．executeMethod メソッドでは，ユーザの ID からエージェントのインスタンスを特定し，Java のリフレクション機能を用いてエージェントのメソッドを実行する．

5.2 エージェントの実装

エージェントの基底クラスとなる **LotosMApp** クラスを実装した．すべてのエージェントは，**LotosMApp** クラスの子クラスとして実装される．**LotosMApp** では，表 1 に示す API をメソッドおよびエクセプションとして定義・実装した．

グループ形成機構の実装 エージェントグループの形成のために，**GroupManager** クラスを実装した．提案するミドルウェアでは，executeApp メソッドにより起動されたエージェントをグループマネージャが管理し，エージェントは，グループマネージャに対してグループメンバ募集の登録やグループ参加の要求を行う．

図 9 のように，まず，(1) advertise を実行したあるエージェント A1 が，グループマネージャM にグループメンバ募集メッセージ m_1（キーワードや端末位置など含む）を登録する．このとき，通信関係オブジェクトを生成し，エージェント A1 にポインタを渡す．次に，(2) participate を実行したエージェント A2 が，M にグループ参加メッセージ m_2（キーワードや端末位置など含む）を送信する．(3) M は m_1 と m_2 に含まれる値，条件をチェックして，条件が満たされるならば通信関係オブジェクトへのポインタとともに Ack を渡す．すると，(4) 通信関係オブジェクトがエージェント間で共有され，グループが形成される．(5) (2) から (4) を繰り返す，という手順で行う．

advertise/participate により形成された**図 10** (a) のグループに対し，図 10 (b) のような通信関係オブジェ

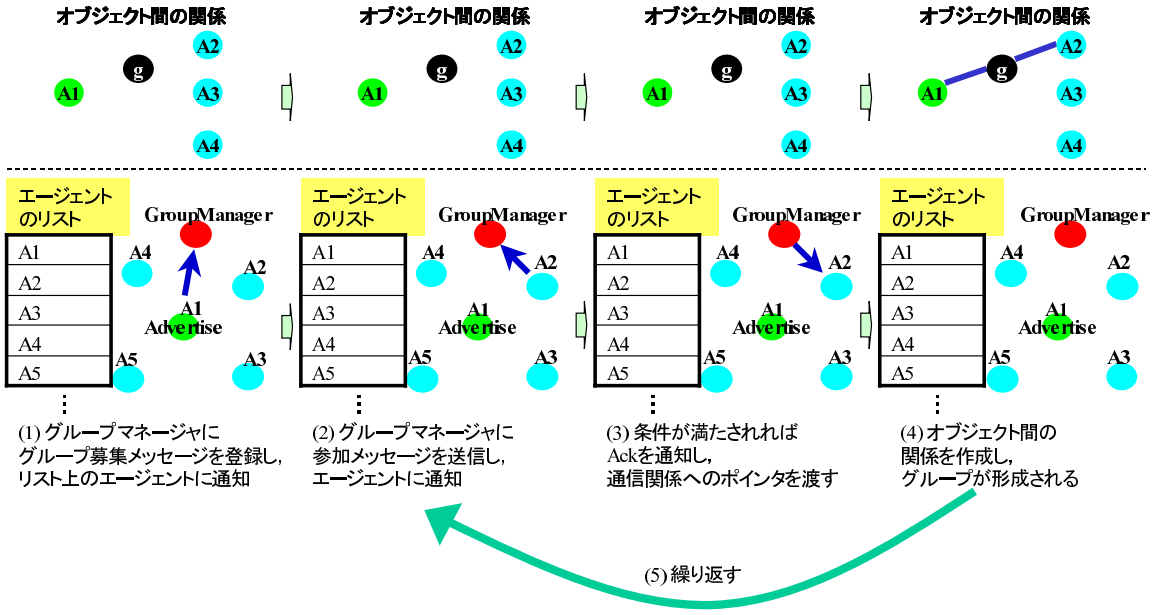

図 9 メンバ募集からグループ形成までの手順
Fig. 9 Advertising process.

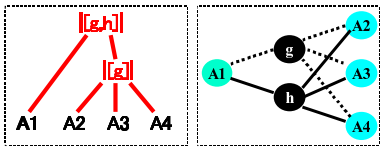

図 10 通信関係オブジェクト
Fig. 10 Channel relation.

クトが作成される．A1，A2，A3，A4 はエージェントで，g は共有のチャネル，h は排他のチャネルである．

マルチランデブの実装 実行の効率を考慮し，LOTOS におけるマルチランデブの次のサブクラスを実装の対象とした．

・通信関係が $\sharp 1|[G]|\{\sharp 2\}$ もしくは，

・通信関係が $\sharp 1|[G]|-|[H]|\{\sharp 2\}$ かつ $H \subseteq G$（ただし，G，H はそれぞれチャネルの集合）

提案ミドルウェアでは，executeEvent メソッドを用いてグループメンバ間でのマルチランデブによるイベントの同期実行を指定できる．マルチランデブによる通信の実装のために，同期，排他，マルチキャストの 3 つの通信関係クラス（Synchronization，Exclusion，Multicast）を定義し実装した．各エージェントは，それぞれの通信関係オブジェクトを共有することで，エージェント間でマルチランデブによる通信を行うことができる．各エージェントは，他のエージェントとのマルチランデブに基づく通信を行うために，通信関係オブジェクトにデータを書き込む（または通信関係オブジェクトからデータを読み込む）．

同期通信関係クラスにおいては，出力イベントを実行するエージェントがオブジェクトにデータを書き込み，入力イベントを実行するエージェントがオブジェクトからデータを読み込みイベントで定義した変数に代入する．この際，すべてのエージェントが上記の操作を行えるようになるまでエージェントの動作をブロックする．排他通信関係クラスにおいては，複数の出力イベントを実行するエージェントがいる場合，最も早くイベントを実行したエージェントのみがオブジェクトにデータを書き込むことができるよう制御する．入力イベントを実行するエージェントは，同期通信関係クラスと同様に，オブジェクトからデータを読み込みイベントで定義した変数に代入する．マルチキャスト通信関係クラスにおいては，同期通信関係クラスと同様であるが，データ交換が目的のため，イベントを非同期に実行できるようにした（エージェントの動作をブロックしない）．

たとえば，エージェント間の関係が図 10 のとおりであるとする．このとき，A1 に g（同期通信関係）に対する出力イベント，A2，A3，A4 に g からの入力イベントが定義されているとすると，A1 が g に出力を行い，A2，A3，A4 のそれぞれが g から入力する準備が整った時点でイベントが実行され，データが共有される．また，A1 に h（排他通信関係）からの入力

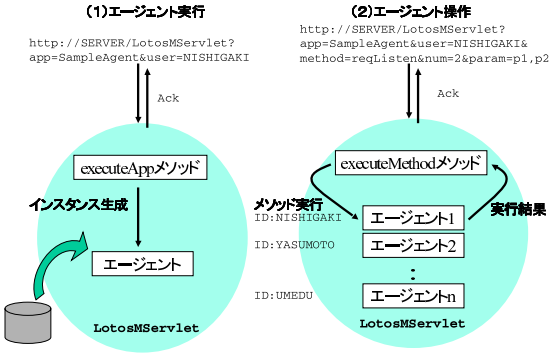

図 11 エージェントの起動・操作
Fig. 11 Starting and operation of an agent.

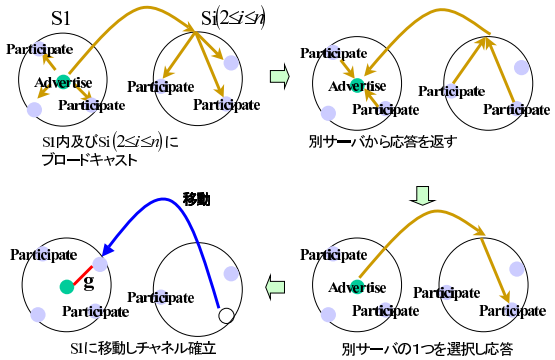

図 12 複数サーバを用いたグループ形成の手順
Fig. 12 Image of agent migration.

イベント，A2，A3，A4 に h に対する出力イベントが定義されているとすると，A1 が h から入力する準備が整い，A2，A3，A4 のいずれか 1 つ（A3 とする）が h に出力を行った時点でイベントが実行され，A1 と A3 の間でデータが共有される．このとき，A2 と A4 のイベントは実行されずに破棄される．

5.3 エージェントの起動・操作

HTTP によるエージェントの起動・操作 携帯電話端末上のユーザインタフェース（UI）とサーブレット上のエージェントの通信は HTTP プロトコルにより実装する．UI としては，HTTP による通信を実装した MIDlet を作成する．また，UI として，HTTP を実装した任意のプログラムが利用できる．UI からエージェントを起動する場合は，**図 11** (1) のように，サーブレットに対して HTTP のリクエストを行う．リクエストのパラメータには，起動したいエージェントのクラス名（SampleAgent）とユーザ ID（NISHIGAKI）を指定する．サーブレットは，リクエストパラメータを用いて，executeApp メソッドを実行する．実行結果は，HTTP のレスポンスとして返される．同様に，UI からエージェントを操作する場合は，図 11 (2) のように，サーブレットに対して HTTP のリクエストを行う．リクエストのパラメータには，操作したいエージェントのクラス名（SampleAgent），ユーザ ID（NISHIGAKI），メソッド名（reqListen），引数の数 (2) と引数を文字列にして「,」で区切ったもの (p1, p2) を指定する．サーブレットは，リクエストパラメータを用いて，executeMethod メソッドを実行する．実行結果は，HTTP のレスポンスとして UI に返される．ここで，UI とエージェント間の通信に **SOAP** を用いる方法も考えられるが，(1) 通信相手が決まっている，(2) 携帯電話のリソースの制限からプロトコルの処理はできる限り軽いことが望ましいなどの理由により，独自プロトコルを採用した．

エージェントから UI への（直接のレスポンスでない）情報送信については UI 側での受信タイミングを制御できないため，ポーリング（空のリクエストを定期的にエージェントに送る）が必要である．現段階では未実装であるが，読むべきデータがエージェント側にあることを何らかの手段（au の C メールサービスなど）で UI に知らせ，必要に応じて HTTP リクエストを送信することで，無駄な通信を削減することが考えられる．

5.4 複数サーバを用いた負荷分散

1 台のサーバ上にすべての携帯電話に対応するエージェントをロードし実行する方法ではスケーラビリティに欠ける．そこで，本節では，負荷分散のために複数のサーバを用いる場合の実現方針について述べる．基本方針として，グループ通信を高速に行うため，同じグループのエージェントは同じサーバで実行されるよう，グループ形成時にエージェントの再配置を行うこととする．このため，1 つのエージェントは 1 つのチャネルにしか参加できないという制約を設ける．この制約は，効率的な実現のために設置した制約として良い制約だと考える．

複数サーバ間におけるグループの形成は，**図 12** のように，(1) advertise を実行するあるエージェント（A1 とする）は，自身が存在するサーバ上のグループマネージャにグループメンバ募集メッセージ（キーワードや端末位置など含む）を登録し，同時に，別サーバ上のグループマネージャにもグループメンバ募集メッセージを登録する（図 12 (1)）．(2) 別サーバに存在する participate を実行するエージェント（A2 とする）が自身が存在するサーバ上のグループマネージャにグループ参加メッセージ（キーワードや端末位置など含む）を送信する（図 12 (2)）．(3) グループマネージャ

が条件を判定して満たされるならば，グループへの参加許可メッセージを A2 に返信する（図 12 (3)）．(4) グループへの参加許可メッセージを受け取った A2 は，A1 の存在するサーバに移動を行い，グループ通信用チャネルを確立する（図 12 (4)）．

6. 実験と評価

ミドルウェアを実装し，マルチランデブによるグループ通信の性能，グループ形成の性能，UI とエージェント間の通信性能，複数サーバを用いた場合のオーバヘッドについて調べた．また，本ミドルウェアを用いたアプリケーションを実装し，実行性能について調べた．アプリケーションとして，複数のキーワードを設定してグループを形成し，簡単なテキストを交換するアプリケーションを用いた．サーバとして Athlon2200+，メモリ 512M，Tomcat4.1.24，JDK1.4.1 を，UI として PC 上の携帯電話エミュレータおよび携帯電話（A5303H）上の MIDlet を用いた．

グループ通信の性能を測定するために，ノード数 n=10000 からなるエージェントに対して，同期，排他，マルチキャストの各通信関係に対するイベント実行にかかる時間を測定した．結果を**表 2** に示す．この結果から，ユーザ数が 10,000 規模の，頻繁なインタラクションを含むアプリケーションでも，サーバ内のエージェント間の通信がボトルネックになることはないことが分かる．ここで，エージェント間のデータの受け渡しは同一プロセス内のスレッド間でのポインタの受け渡しとして実現しているため，通信性能は交換されるデータサイズに依存しない（ただし，エージェントと UI の間の通信性能は，データサイズに比例する）．よって，エージェントが交換するデータの種類やサイズに違いによる，本ミドルウェアのグループ通信性能に差はないと考える．

また，グループ形成の性能を測定するために，ノード数 n=1000, n=10000, n=100000, n=1000000, キーワード集合の要素数 a=2，a=4 からなるエージェントに対して，コンテキスト距離の判定にかかる時間を計測した．計測結果を**表 3** に示す．この結果から，アプリケーションドメイン内のエージェント数が 100,000 規模であれば約 1.5 秒と実用上問題ない時間でコンテキスト距離の判定が可能であることが分かる．しかし，この数が 1,000,000 規模になる場合には，17 秒程度の処理時間を要しており，これを削減するためには，5.4 節で述べた負荷分散機構を用いて，サーバ 1 台あたりにマッチングすべきエージェント数を限定する必要がある．

表 2 イベント実行の性能（ミリ秒）
Table 2 Performance of event execution (msec).

通信関係	同期	排他	マルチキャスト
n=10000	11	40	11

表 3 コンテキスト距離判定の性能（ミリ秒）
Table 3 Performance of calculations and matching of context distances (msec).

	n=1000	n=10000	n=100000	n=1000000
a=2	60	202	1,557	17,712
a=4	70	214	1,557	15,182

表 4 複数サーバ使用時のサーバ選択の性能（ミリ秒）
Table 4 Performance of selection of the server (msec).

サーバ選択の順序	
Server1	8,825
Server1 to Server2	10,184
Server1 to Server2 to Server1	12,050

同様に，UI とエージェント間の通信性能を測定するために，UI として MIDlet を作成し，携帯電話（A5303H）からリモートメソッドの実行にかかる時間を測定した．計測したのは，HTTP でエージェント起動・メソッド実行のリクエストを行い，実行結果が返ってくるまでの時間であり，それぞれ 10 回測定した．UI からのエージェントの起動の時間は平均 8,825 ミリ秒で，UI からのメソッドの実行の時間は平均 1,280 ミリ秒であった．ここで，エージェントの起動に時間がかかっているのは，初回通信時に時間がかかることが原因である（携帯電話（A5303H）の仕様）．エージェント起動前に一度通信を行うと，エージェントの起動時間はメソッドの実行時間と変わらない．この結果は，単なる HTTP リクエストにかかる時間（1,175 ミリ秒）とほぼ同じことから，実用上十分な性能であると思われる．

最後に，複数サーバを用いた場合のオーバヘッドを測定するために，エージェント起動時にサーバの選択に要する時間と，エージェントの移動にかかる時間を測定した．サーバの選択に要する時間は，**表 4** のとおりであり，最大約 12 秒であった．また，エージェント（クラスサイズ：約 1 K バイト）1 個の移動に要する時間は 109 ミリ秒であった（100 個の場合，約 10 秒）．これらの結果から，複数サーバを用いたときのオーバヘッドは実用範囲内であると考える．

以上の実験結果から，本ミドルウェアを用いたアプリケーションの通信性能は，出力イベントの実行が UI からエージェントへのデータ送信時間（UI からのメソッドの実行の時間（1,280 ミリ秒））とイベント実行時間（表 2）の和となるために，約 1.3 秒と考えられ，

また，入力イベントの実行がイベント実行時間（表 2）とエージェントから UI への情報送信時間の和となるために，ポーリング周期（2〜3 秒）に依存すると考えられる（5.3 節で述べたように，エージェントから UI への情報送信にはポーリングが必要）．3.3 節で示したアプリケーションを実装してイベント実行（問題配布，早押し）に要した時間を測定した結果，想定したとおりの性能であることを確認した．

7. むすび

本論文では，携帯電話による協調型アプリケーション開発のためのミドルウェアの提案を行った．本ミドルウェアを用いることで，不特定多数の携帯電話ユーザ間での地理的位置およびコンテキスト距離に基づいたグループの動的形成機能と，グループメンバ間のマルチランデブによるイベントの同期実行の機能を使ったユーザ間の高度なインタラクションを使った協調型携帯電話アプリケーションの開発を容易に行える．

今後の課題として，BREW や新しいバージョンの Java 環境を使い，写真や動画などを送受信できる，より高機能で実用的なアプリケーションの実装などを予定している．さらに，セキュリティを考慮して，携帯電話—サーバ間の通信の暗号化および Java の標準 API を使ったユーザ認証の実装を予定している．また，テキストマイニング技術などを用いて，より効果的にグループを形成できるよう，コンテキスト距離を拡張したい．

参考文献

1) ISO: Information Processing System, Open Systems Interconnection, LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807 (1989).
2) Chu, Y.-H., Rao, S.G., Seshan, S. and Zhang, H.: Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture, *Proc. ACM SIGCOMM*, pp.55–67 (2001).
3) Chu, Y.-H., Rao, S.G. and Zhang, H.: A Case for End System Multicast, *Proc. ACM SIGMETRICS*, pp.1–12 (2000).
4) Pendarakis, D., Shi, S., Verma, D. and Waldvogel, M.: ALMI: An Application Level Multicast Infrastructure, *Proc. 3rd Usenix Symp. on Internet Technologies & Systems*, pp.49–60 (2001).
5) Chen, K. and Nahrstedt, K.: Effective Location-Guided Tree Construction Algorithms for Small Group Multicast in MANET, *Proc. IEEE INFOCOM2002*, pp.1180–1189 (2002).
6) Ko, Y.-B. and Vaidya, N.H.: Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms, *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications* (*WMCSA'99*), pp.101–110 (1999).
7) FIPA: FIPA2000 Specifications. http://www.fipa.org.
8) Bergenti, F. and Poggi, A.: LEAP: A FIPA Platform for Handheld and Mobile Devices, *Proc. 8th Int'l. Workshop on Agent Theories, Architectures, and Languages* (*ATAL-2001*), LNCS2333, pp.436–446 (2002).
9) Poslad, S., Laamanen, H., Malaka, R., Nick, A., Buckle, P. and Zipf, A.: CRUMPET: Creation of User-friendly Mobile Services Personalized for Tourism, *Proc. 2nd Int'l. Conf. on 3G Mobile Communication Technologies* (*3G 2001*), pp.26–29 (2001).
10) Nishiyama, S., Hattori, G., Ono, C. and Horiuchi, H.: Lightweight FIPA Compliant Agent Platform on Java-enabled Mobile Phone for Ubiquitous Services, 情報処理学会論文誌, Vol.45, No.2, pp.575–585 (2004).
11) http://www.e-jumon.com/
12) http://www2.toshiba.co.jp/plangent/
13) Tarumi, H., Morishita, K., Nakao, M. and Kambayashi, Y.: SpaceTag: An Overlaid Virtual System and its Applications, *Proc. 1999 Int'l. Conf. on Multimedia Computing and Systems* (*ICMCS'99*), Vol.1, pp.207–212 (1999).
14) 茂木信二，吉原貴仁，堀内浩規，小花貞夫：ITS におけるモバイルコミュニティ形成方式，情報処理学会論文誌，Vol.42, No.7, pp.1840–1846 (2001).
15) 中沢　実，服部進実：意味情報と嗜好情報に基づく P2P システムの提案と実装，情報処理学会論文誌，Vol.44, No.3, pp.826–834 (2003).
16) Umedu, T., Yasumoto, K., Nakata, A., Yamaguchi, H. and Higashino, T.: Middleware for Synchronous Group Communication in Wireless Ad Hoc Networks, *Proc. IASTED Intl. Conf. on Communications and Computer Networks* (*CCN2002*), pp.48–53 (2002).

(平成 16 年 4 月 6 日受付)
(平成 16 年 10 月 4 日採録)

西垣 弘二（学生会員）

昭和 45 年生．平成 6 年神戸大学理学部物理学科卒業．同年メルコ・パワー・システムズ（株）入社．プラント監視制御システム開発，産業用イントラネット応用開発に従事．平成 16 年奈良先端科学技術大学院大学情報科学研究科情報処理学専攻博士前期課程修了．現在同大学院情報科学研究科情報処理学専攻博士後期課程．モバイルコンピューティングシステム，コンテキストアウェアコンピューティングシステムに関する研究に従事．

安本 慶一（正会員）

平成 3 年大阪大学基礎工学部情報工学科卒業．平成 7 年同大学大学院博士後期課程退学後，滋賀大学経済学部助手．平成 9 年モントリオール大学客員研究員．平成 14 年より奈良先端科学技術大学院大学情報科学研究科助教授．博士（工学）．分散システム，マルチメディア通信システムに関する研究に従事．IEEE/CS 会員．

梅津 高朗（正会員）

平成 13 年大阪大学大学院基礎工学研究科情報数理系専攻博士前期課程修了．同年同大学院博士後期課程進学．平成 14 年同大学院博士後期課程退学後，同大学院情報科学研究科助手．プロトコル合成法の応用やアドホックネットワーク用ミドルウェアの研究に従事．

東野 輝夫（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院博士課程修了．工学博士．同年同大学助手．現在，同大学大学院情報科学研究科教授．分散システム，モバイルコンピューティング，通信プロトコル等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member.

伊藤 実（正会員）

昭和 52 年大阪大学基礎工学部情報工学科卒業．昭和 54 年同大学大学院基礎工学研究科博士後期課程退学後，同大学助手．昭和 58 年工学博士（大阪大学）．平成 3 年カナダウォータールー大学数学部客員準教授．平成 5 年奈良先端科学技術大学院大学情報科学研究科教授．データベース理論，分散システム等に関する研究に従事．電子情報通信学会，ACM，IEEE/CS 各会員．