マルチランデブに基づくグループ通信機能を提供するJavaミドルウェアの提案

梅津高朗[†] 安本慶一[††]
中田明夫[†] 東野輝夫[†]

本論文においては，直接通信可能（1 ホップで通信可能）な端末が動的に変化する環境においてグループ通信機能を提供する Java ミドルウェアを提案する．提案ミドルウェアでは，各エージェント（移動端末上で動作するプロセス）が指定された条件に合致した場合にのみ他のエージェントとの間に動的に階層的な同期チャネルを生成する機能や，それらのチャネルを介した複数端末間のマルチランデブ（同期，データ交換，排他制御などを行う機構）に基づくグループ通信機能を提供する．実装実験を行い，チャネルの動的生成，および，グループ通信に要する時間を計測し，提案手法の実用性を確認した．

Middleware for Supporting Dynamic Establishment of Multi-way Synchronization Channels

Takaaki Umedu,[†] Keiichi Yasumoto,[††] Akio Nakata[†]
and Teruo Higashino[†]

In this paper, we propose a Java based middleware for group communication systems on ad-hoc environments, where pairs of nodes that can directly communicate each other may be dynamically changed. This middleware provides a facility of dynamic establishment of hirerarchical communication channels among agents (mobile hosts) where the channels can be established only if given conditions hold. It also provides a facility of powerful group communication based on multi-way synchronization (the mechanism that provides functions such as synchronization, data exchange, mutual exclusion and so on) over these channels. We have implemented this middleware using TCP and UDP and measured its performance.

1. はじめに

ネットワーク機能を有した機器の普及にともない，多人数参加型のアプリケーションに対する需要が高まっている．このようなアプリケーションにおいては，不特定多数の端末を効率的に結び付ける機構や，多数の端末間でのインタラクションを制御する機構などが必要となる．しかし，一般に多数の端末間のインタラクションを実現する通信は，プロトコルが複雑になり設計が難しい．そのため，多数端末間でのインタラクション機能が容易に利用できるミドルウェアが望まれている．

近年，Peer to Peer 環境において，コミュニケーション機能を提供するミドルウェア[1)]や，位置情報や嗜好に基づいてデータベースから選択的にデータを取得する Context-aware ミドルウェア[2)]，アプリケーションレベルでユーザ端末間に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[3)~5)]，無線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[6),7)]，モバイルエージェント技術を用いたミドルウェア[8),9)]などの通信ミドルウェアに関する研究がなされている．また，WCA [10)]のようなモバイルアドホックネットワークにおけるクラスタ作成，維持用のプロトコルも提案されている．しかし，多数のユーザが参加し，かつユーザ間で協調動作を行うようなアプリケーションを設計する場合には，グループ制御機能に加えて，マルチキャストのような 1 方向の非同期通信機能に限定されない，多数の端末による同期や，排他制御などの高度な

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†† 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

グループ通信機構が柔軟に設計できるミドルウェアが望ましい．

通信プロトコル向けに開発された仕様記述言語 LOTOS [11] では，複数並行プロセス間でのイベントの同期実行やデータ交換を効率的に記述できるマルチランデブと呼ばれる強力なグループ通信機構を用いたプロセス間通信が記述できる．また，プロセス代数 CSP [12] に基づく Java ミドルウェア JCSP [13] なども提案されている．しかし，これらのモデルでは複数のプロセスが自律的に通信チャネルを確立する機能を持たないため，多数の端末が互いに通信相手を動的に変更するようなアプリケーションの記述に適していない．

本論文では不特定多数の端末から構成されるグループの構築機能，および，構築したグループ内でマルチランデブに基づく通信機能を提供するミドルウェアを提案する．Java を用いて実現することで，携帯端末から，サーバまで幅広い環境で利用可能である．

提案ミドルウェアを用いてアプリケーションを設計する場合，システム全体を互いに独立に動作する**エージェント**の集合として記述する．提案ミドルウェアは，各エージェントが他のエージェントとの間に通信チャネルを動的に形成することで**エージェントグループ**を構築する機能を提供する．結合されたエージェント群全体を新たな 1 つのエージェントと見なし，さらに別のエージェントと段階的に結合することにより，多数のエージェントを含むエージェントグループを形成できる．通信チャネルが破棄された場合には，それらのエージェントはグループから離脱し，以後独立に動作する．グループ構築後は，複数プロセス間の同期機構（**マルチランデブ**）に基づく同期通信が利用でき，グループ内でのマルチキャストや排他制御などが簡潔に記述できる．直接通信できないエージェント間の同期制御も，他のエージェント群がメッセージを中継するよう同期判定プロトコルを設計しているため，特別な追加記述もなく実現できる．

いくつかの例題の記述，実装実験を通して，動的に生成された通信チャネル上でのマルチキャストや排他制御を含む典型的な分散協調アプリケーションが提案ミドルウェアにより簡潔に効率良く実装できることを確かめた．

2. 提案ミドルウェアの利用環境とその機能

2.1 提案ミドルウェアが想定する環境

本論文で提案するミドルウェアは以下のような環境を想定している．

- 複数のノードが並列動作し，互いに通信を行うことで協調動作する．
- 各ノードはアプリケーション全体が対象とするノードのうち，一部のノードとの間でのみ直接に通信が行える．
- 直接通信可能なノードの組合せは動的に変化しうる．

また，提案ミドルウェアは通信プリミティブとして環境に応じて抽象化した以下の機能のみを利用する．

- 各ノードは直接通信可能なすべてのノードに対してメッセージをブロードキャストできる．
- ブロードキャストされたメッセージを受信したノードは送信元ノードとの間に通信路を構築して通信を行うことができる（直接通信することが不可能になった場合には通信は切断される）．

提案ミドルウェアはこれらの単純な機能を用いてアプリケーション設計者に対して柔軟なグループ通信機構を提供する．起動時に各エージェントにはノード間の接続関係などネットワークに関する具体的な情報は与えられない．各エージェントは直接通信可能なエージェントに対してメッセージをブロードキャストし，それらのメッセージにより通信可能なエージェントを検出し，グループ構築を行う．

たとえば，実装環境をアドホックな無線環境と想定した場合，直接通信可能なエージェントは無線範囲内のエージェントとすればよい．IP ネットワーク環境に対して実装する場合には，(1) UDP のブロードキャストを用いることで，直接通信可能なエージェントを同一 LAN 内のすべてのエージェントとする実装や，(2) アドレスのリストを与えることで直接通信可能なエージェントの集合を明示する実装などが考えられる．また，他の環境として，携帯電話などが考えられるが，一般に現在利用可能な携帯端末の通信機能は特定サーバとの間の http のみに制限されている場合が多い．しかし，そういった場合でもサーバ側に用意した CGI や Servlet においてエージェントを実行させ，ユーザインタフェースのみを携帯端末で実行する，などの手法により提案ミドルウェアを実装することが可能である [14]．

2.2 グループ構築機能

エージェントグループはエージェント間の 1 対 1 の結合に基づいて構築される．各エージェントはエージェントグループ構築を目的として，**参加募集**と**参加要求**の 2 つの機能を利用できる．これらの機能の実行時にはエージェント間でデータの交換が可能で，交換されたデータに対する条件を指定できる．参加募集と参加要求を行った 2 つのエージェントが直接通信できる状

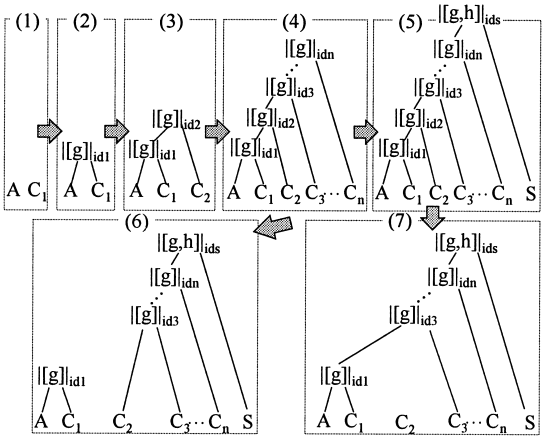


図 1 エージェントグループの構築とエージェントの離脱
Fig. 1 Construction of agent group and leaving of agents.

態にあり，交換したデータが条件を満たす場合にそれらのエージェントは結合し，エージェントグループとなる．結合してできたエージェントグループを単体のエージェントと見なすことで段階的にエージェントグループを拡大できる．

結合したエージェント間には結合時に指定されたチャネル（以下，**ゲート**と呼ぶ）における**同期関係**が設定され，エージェント間の通信制御に利用される．ゲートの集合 G における同期関係がエージェント $A1$，$A2$ の間に設定されていることを以下 $A1|[G]|_{id1}A2$ と表記する．この場合，$A1$，$A2$ で発生するイベントのうち，G に含まれるゲート上でのイベントは同期実行されることを意味する．ここで，$id1$ はミドルウェアがこの同期関係に与えた ID を表し，この $id1$ を用いてこの同期関係 $|[G]|_{id1}$ を解除できる（以下，ID は特に必要のない場合には省略する）．また，エージェントグループ $A1|[g]|A2$ を 1 つのエージェントと見なし，さらに別のエージェント $A3$ とゲート g，h で結合した場合，それら 3 つのエージェントの間の同期関係は $(A1|[g]|A2)|[g,h]|A3$ と表現できる．この場合，ゲート g でのイベントは $A1$，$A2$，$A3$ で同期実行され，ゲート h でのイベントは，$A1$ と $A3$ もしくは，$A2$ と $A3$ のいずれかの組で選択的に同期実行される（排他制御）．

図 1 は，4 章で例示するアプリケーションにより作成されるエージェントグループを表している．ここでは，エージェントグループを，同期関係を節，エージェントを葉とする 2 分木で表現している．以下，この木表現を**同期判定木**と呼ぶ．まず，エージェント A が参加募集を行い，エージェント C_1 が参加要求を出し，この 2 つのエージェントが結合する（図 1 (2)）．さらに参加募集を行い，エージェント C_1〜C_n と順に結合する（図 1 (3)，(4)）．その後，エージェント S と結合することで，g，h の 2 つのゲートにおける同期関係が設定されたエージェントグループ $((\cdots((A|[g]|_{id1}C_1)|[g]|_{id2}C_2)|[g]|_{id3}\cdots)|[g]|_{idn}\ \ C_n)|[g,h]|_{ids}S$ が構築される（図 1 (5)）．

結合の解除は，ID により指定された同期関係が解消された結果，解除を実行したエージェントがエージェントグループから離脱するよう実行される．たとえば，図 1 (5) において，$|[g]|_{id2}$ が解消される場合を考える．A もしくは，C_1 によりこの結合が破棄された場合には，エージェントグループ $A|[g]|_{id1}C_1$ が離脱する（図 1 (6)）．一方，C_2 によりこの結合が破棄された場合には，C_2 がエージェントグループから離脱する（図 1 (7)）．動作の詳細やグループ構築のフォーマルセマンティクスは文献 15) を参照されたい．

2.3 マルチランデブとデータの送受信

各エージェントではゲート上での**イベント**を定義できる．同期関係が設定されていないゲート上でのイベントは独立して実行され，同期関係が設定されているゲート上でのイベントはエージェント間で同期して実行される．複数エージェント間に同期関係を設定することで，3 個以上のエージェントを同期させることも可能であるため，このイベントの同期実行機構は**マルチランデブ**[11] と呼ばれている．

イベントには入出力パラメータを持たせることが可能であり，それらを用いてマルチランデブの際にデータの交換ができる．ここで，$g!E$ は，ゲート g における値 E の出力イベントを表し，$g?x : int[P(x)]$ は，ゲート g における int 型の変数 x に対する入力イベントを表し，条件 $P(x)$ が成立するときのみ x への入力イベントが実行されることを意味する．あるエージェントが実行するイベントに出力パラメータが指定された場合には，マルチランデブの結果，その出力値は他のエージェントが実行したイベントの対応する入力パラメータに代入される．1 つのイベントに入出力を混在させた複数のパラメータからなるリストを指定することも可能で，その場合には，指定された順序により入出力パラメータの対応が決定される．同期関係が設定されたエージェントの集合において，同期が指定されたイベント群が以下の条件を満たすときのみマルチランデブは実行される．

- すべてのイベントの入出力パラメータの個数と対応するパラメータの型がそれぞれ等しい．
- 各入力パラメータに対して少なくとも 1 個の対応する出力パラメータが指定されたイベントを含む．

表 1 提案ミドルウェアを構成する主要なクラス
Table 1 Main classes of our middleware.

クラス名	主要なメソッド	機能
Agent	Agent(CommunicationInterface[] i); Result Advertise(Gate[] s,Gate[] a, Object[] io,Guard g); Result Participate(Gate[] s,Gate[] a, Object[] io,Guard g); Event Synchronize(Event[] e); void Disc(ID id)	エージェントの基底クラス 他のエージェントへエージェントグループへの参加募集を行う エージェントグループへ参加要求を行う マルチランデブの実行 指定された id のマルチランデブチャネルでの結合を切断する
Gate	Gate(String n)	ゲートを定義するクラス
Event	Event(Gate g,Object[] io,Guard g);	イベントを定義するクラス
Guard	abstract boolean Evaluate(Object[] io);	Evaluate メソッドをオーバライドして io に関する条件を定義するクラス
Result	ID GetChannelID(); Object[] GetIO();	チャネル生成結果に関する情報を定義するためのクラス 生成されたチャネルの ID チャネル生成時に交換した入出力オブジェクトの配列
ID		グローバルな一意性が保証された汎用の ID
Communication-Interface		通信機能を抽象化したクラス
Communication-InterfaceUDP		CommunicationInterface の UDP/IP による実装
DiscException		チャネル切断を通知する例外クラス

- 対応する複数の出力パラメータが存在する場合それらの値がすべて等しい．
- パラメータに付与された条件が成立する．

マルチランデブが可能なイベントの組合せが複数存在する場合には任意の 1 つが非決定的に選択される．たとえば，$(A1|[g]|A2)|[g,h]|A3$ と同期関係が指定された 3 つのエージェントにおいて，$A1$ では，$h?x1 : int$ もしくは $h!$"$A1$ *hello*"，$A2$ では，$h?x2 : int$ もしくは $h!$"$A2$ *hello*"，$A3$ では，$h!1$ もしくは $h?x : String$ のイベントがそれぞれ定義されていたとする．この場合，実行可能なイベントの組合せとしては，$A1$, $A3$ による h での同期 (1) $(h?x1 : int, h!1)$，および，(2) $(h!$"$A1$ *hello*"$, h?x : String)$ と，$A2$, $A3$ による h での同期 (3) $(h?x2 : int, h!1)$，および，(4) $(h!$"$A2$ *hello*"$, h?x : String)$ の合計 4 通りが考えられ，そのいずれかが非決定的に選択される．他の $(h!$"$A1$ *hello*"$, h!1)$ のような組合せは型が一致しないため実行できない．

2.4 同期関係によるグループ通信の記述

マルチランデブを用いることで，マルチキャストを含む様々なグループ通信機構が簡潔に実現できる．たとえば，前述の図 1 (5) の例では，まず，すべてのエージェントの間にゲート g での同期関係が設定されているため，すべてのエージェントは g におけるイベントで同期する．よって，ゲート g でのイベントを用いてエージェントグループ内へのマルチキャスト通信を実装できる．次に，A と C_1〜C_n の間には h における同期関係は設定されておらず，これらのエージェントはゲート h におけるイベントでは同期しない．一方，S と S 以外のサブエージェントグループ $(\cdots((A|[g]|C_1)|[g]|C_2)|[g]|\cdots)|[g]|C_n$ との間には h における同期関係が設定されているため，S はこのサブエージェントグループと h におけるイベントを同期実行する．結果として，S とそれ以外のいずれか 1 つのエージェントが排他的に選択され同期が行われることとなり，h を介したイベントを用いて排他制御機構が実現できる．イベントの入出力パラメータに対する条件指定とともに用いることで，特定の条件を満たすユーザ 1 名を排他的に選択するといった仕様が簡潔に記述できる．

3. 提案ミドルウェアの仕様と実装

提案ミドルウェアを構成するクラスを**表 1** にあげる．提案ミドルウェアを用いる場合，Agent クラスのサブクラスとして定義したエージェント群によりアプリケーション全体を記述する．Agent クラスは，メンバメソッドとして 2 章で述べた参加募集/参加要求，および，マルチランデブ機能を提供する．以下，その仕様について述べる．

3.1 通信インタフェースの抽象化

ミドルウェアの移植性を高めるため，2.1 節で述べた提案ミドルウェアの実装環境を以下の 3 つのメッセージ交換機能を持つインタフェースとして抽象化した．

- メッセージのブロードキャスト

- メッセージの特定エージェントへの送信
- メッセージの受信

以上の機能を各環境で利用できる通信機能を用いて実装した CommunicationInterface オブジェクトを定義し，各エージェントの起動時に与える．

3.2 エージェントグループ構築メソッドの仕様

エージェント間の結合はエージェントグループへの参加募集/要求として以下に示す Advertise/Participate メソッドを用いて行う．

$$res_1 = \textbf{Advertise}(G_s, G_a, IO_1, Grd_1)$$
$$res_2 = \textbf{Participate}(H_s, H_a, IO_2, Grd_2)$$

各引数は，同期関係を設定するゲートの配列 (G_s)，エージェント間で共有するゲートの配列 (G_a)，それらを受け取る配列 (H_s，H_a)，入出力パラメータリスト (IO_1，IO_2)，および，結合条件 (Grd_1,Grd_2) を意味する．入出力パラメータリストと結合条件は 3.4 節で述べるイベントのパラメータと同様に与える．

結合が成功した場合には，共有が指定されたゲートの配列 G_s，G_a，全出力パラメータを格納した配列 IO，および，その同期関係を一意に示す ID を含む Result クラスのオブジェクトが返される．G_s，G_a に含まれるゲートが 2 つのエージェント間で共有され，G_s に含まれるゲートにおける同期関係が設定される．

3.3 エージェントグループ構築メソッドの実装

エージェントグループの構築は，参加募集メッセージのブロードキャストと，それに対する参加要求メッセージの返信により実装した．ここでは，Advertise を実行したエージェントを A，A と通信インタフェースを用いて直接通信が可能であり Participate を実行したエージェントを P_1，P_2 とする．まず，A は参加募集メッセージ M_A を直接通信可能なエージェント群に対してブロードキャストする．P_1，P_2 は Participate メソッドにより，参加募集待ち状態となっており，受信した M_A に対してそれぞれ参加要求メッセージ M_{P1}，M_{P2} を返信する．M_{P1}，M_{P2} はそれぞれ Participate の引数として与えられたゲートの配列，入出力パラメータ，結合条件を保持する．また，エージェントグループのツリー構造を維持するため，互いのエージェント群に重複が無いことを確認する必要がある．そのため，M_{P1}，M_{P2} は P_1，P_2 と同じエージェントグループに含まれるエージェントのリストも保持する．受信された参加要求メッセージは順にメッセージキューに保存される．A はメッセージキューの先頭のメッセージを取り出し，以下の条件の成否を確認する（ここでは M_{P1} が取り出されたものとする）．

- それぞれのエージェントグループが同じエージェントを含まない．
- G_s と H_s，G_a と H_a の配列長がそれぞれ等しい．
- 入出力パラメータと結合条件に関する条件が成立する．

条件が成立しない場合には，参加拒否メッセージを P_1 に返信して，メッセージキューの次の参加要求メッセージ（M_{P2}）に対して同様の処理を行う．条件が成立する場合には，同期判定木を更新し，同期許可メッセージを P_1 へ返信する．同期判定木の実装，エージェント間のマルチランデブは分散制御により処理する．各ノードは，それぞれの同期判定木上での親ノードと子ノードのアドレスを保持し，3.5 節で述べるアルゴリズムによりマルチランデブを実行する．その際，A を新たに設定された同期関係以下の部分木に対する同期判定を行うための節ノード（以下，**責任ノード**と呼ぶ）とし，A の子ノードとして，A の属していたエージェントグループの同期判定木の根ノード（同期判定木の根である責任ノード）R_A と，P_1 のエージェントグループの根ノード R_{P1} を登録する．また，R_A に対してこの責任ノードを新たな根ノードとするために根ノード変更メッセージを送信する．同期許可メッセージは，責任ノードを示す ID を保持しており，これを受信した P_1 は，R_{P1} に対して，同様に根ノード変更メッセージを送信し，処理が完了する．再度 A において Advertise メソッドが実行された場合には，まず，メッセージキューが検査され，参加要求メッセージが残っていた場合には，そのメッセージに対して処理が行われる．参加要求メッセージをキューに保存することで，繰り返し Advertise を行う際に，参加募集メッセージが頻繁にブロードキャストされることを回避できる．メッセージキューが空であった場合には，参加募集メッセージのブロードキャストから順に処理が行われる．また，参加要求メッセージが得られない場合にも一定時間 T でタイムアウトし，参加募集メッセージのブロードキャストから順に処理が行われる．T の値は，ブロードキャストの頻度と募集時の反応時間のトレードオフを考慮して自由に設定できる．

3.4 マルチランデブ通信メソッドの仕様

設定された同期関係に基づき，エージェントはマルチランデブに基づく同期通信を行うことができる．まず，各エージェントは実行したいイベント候補（**Event** クラス）の配列を作成する．各イベントには，イベントを実行するゲート名と入出力パラメータのリスト，および，同期を行うためのパラメータ条件を指定する．入出力パラメータは Object 型の配列で指定する．配列の各要素には，入力パラメータとする場合には受信

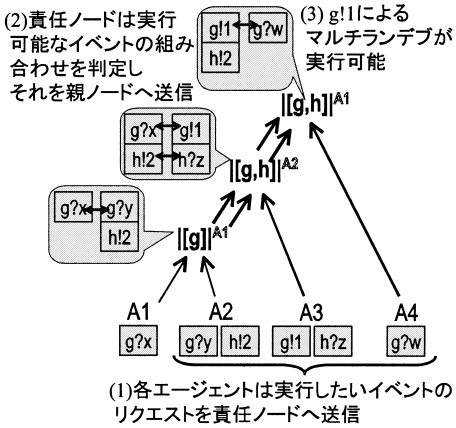


図 2 マルチランデブ実行可能性の判定
Fig. 2 How to evaluate executable events in multi-way synchronization.

するオブジェクトの型を表す Class 型のインスタンス（変数名は指定しない），出力パラメータとする場合には送信するオブジェクトそのものを指定する．また，同期を実行する条件を指定する際には，その条件を判定する **Guard** クラスのサブクラスを作成しパラメータとして与える．

作成したイベント候補の配列を引数として同期メソッド（**Synchronize**）を実行することで 2.3 節で述べたマルチランデブの定義に従い，エージェントグループ内で設定された同期関係に基づきマルチランデブの判定が実行されその結果がイベントクラスのインスタンスとして返される．返されたインスタンスは同期を行ったエージェントのリストと全出力パラメータの値を保持しており，それらを参照することで受信したデータなどを取得できる．

3.5 マルチランデブ通信メソッドの実装

図 2 は，同期判定がどのように行われるかを示した図である．図の同期判定木において $||[g,h]||^{A1}$ という記述は $A1$ がこの同期関係 $||[g,h]||$ の責任ノードであることを示している．同期判定木は LOTOS における動作式と等価であるため，文献 16) の手法に基づいて同期判定ができる．各責任ノードはそのノード以下の部分木における実行可能イベントの候補を列挙する．まず，子ノードから，実行可能なイベントのリストを含む**同期リクエストメッセージ**を受信する．責任ノードは指定された同期関係に基づき他の子ノードでの実行可能なイベントと組み合わせて実行可能な候補を列挙する．列挙されたイベントはさらに親ノードへと同期リクエストメッセージにより転送され，根ノードまでたどることで同期判定木全体において実行可能なすべてのイベント候補が列挙される．根ノードはそのうちのいずれかを選択し，選択されたイベントは木をたどってすべてのノードへ**同期通知メッセージ**により通知される．

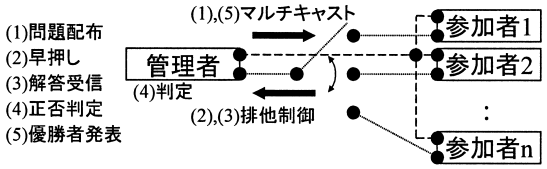


図 3 アプリケーション例（クイズゲームシステム）
Fig. 3 Example application (quiz game system).

4. アプリケーション例：クイズゲームシステム

ここでは，アプリケーション例としてマルチキャストと排他制御を用いたクイズゲームシステムをあげる．

- システム全体は，ゲーム進行を管理する参加募集エージェント A，ゲーム管理エージェント S と，ユーザが操作する n 個の参加者用エージェント $C_1, \ldots, C_n$ で構成．
- ゲーム開始時に参加者を募集し，一定人数もしくは一定時間までに参加表明を行ったユーザのみがゲームに参加．
- 早押し問題が出題され 10 問正解者が出た時点でその参加者の優勝としてゲームを終了．

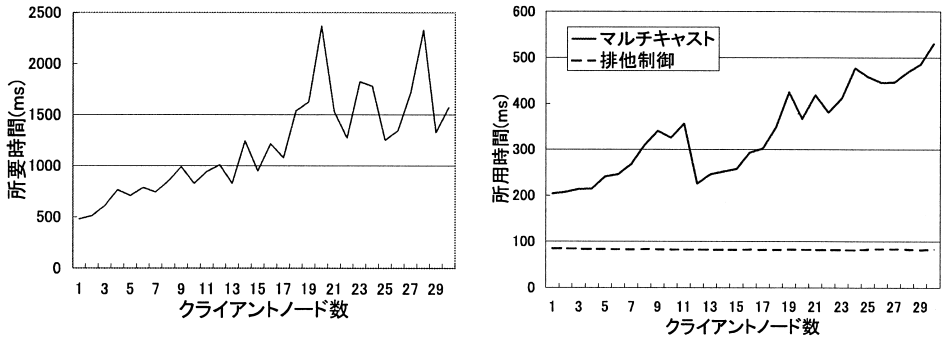
システムの概略を以下に示す（**図 3**）．まず，参加募集エージェント A が，参加者の募集を行う．その際に，ゲート g における同期関係を構築し，それとは異なるゲート h をすべてのノードに対し配布する．定員に達するか制限時間を過ぎたところでクライアントに対する同期募集を打ち切り，ゲート g，および，ゲート h における同期関係をゲーム管理エージェント S との間に構築し，図 1 のような同期関係で結合されたエージェントグループを構築する．

g でのイベントはすべてのクライアント，および，ゲーム管理エージェントで同期実行されるため問題の配付といった，データをマルチキャストする目的で利用できる．一方，h ではゲーム管理エージェントと参加者エージェントのいずれか 1 つが排他的に同期を行うため，早押しクイズの解答に利用できる．

このように提案ミドルウェアを用いることで典型的な分散協調アプリケーションを簡潔に記述できた．

5. 実験結果

提案ミドルウェアライブラリと通信インタフェースを UDP，TCP を用いて実装し，無線 LAN 環境にお



(1) エージェントグループ作成に要した時間　(2) マルチランデブ実行に要した時間

図 4 提案ミドルウェアの性能
Fig. 4 Performance of our middleware.

いて性能を計測した．4 章であげたクイズゲームシステムの例を用いて，図 1 に示した同期判定木で表現されるエージェントグループを作成し，クライアントノードの個数 n に対してグループ構築，および，マルチランデブに要する時間を計測した．実験は，802.11b 規格無線 LAN で接続された 2 台の PC を用いて，1 台目の PC☆で，参加募集エージェント A，および，ゲーム管理エージェント S，2 台目の PC☆☆で n 個の参加者用エージェント $C_1, \ldots, C_n$ を動作させる形で行った．エージェント間の通信はブロードキャストは UDP を介して，その他の通信は TCP を介して無線 LAN 経由で行うよう，実験環境を構築した．

まず，エージェントグループに対するエージェントの参加処理に要する時間を計測した（**図 4** (1)）．$n(1 \leq n \leq 30)$ 個の参加者用エージェントを起動させ，参加募集待ち状態にした後，参加募集エージェントとゲーム管理エージェントを起動した時刻からエージェントグループの構築が完了するまでに要した時間を計測した．メッセージを受信したエージェントから順にグループを構築し，実験中におけるノードの離脱は発生しないものとした．グラフは各ノード数に対して 1 回ずつ行った実験結果をプロットしたものであるため，メッセージの再送などに由来する性能のばらつきが発生しているが，エージェント数が 30 個程度のエージェントグループを 2.5 秒程度で構築でき，実用可能な性能が得られた．

次に，エージェントグループ内におけるマルチランデブの性能を計測した（図 4 (2)）．同様に，クライアント数を 1〜30 と変化させ，マルチキャスト（g での同期），および，排他制御（h での同期）を実現するマルチランデブに要する時間を計測した．ここでは，繰り返しマルチランデブを行った場合のマルチランデブ 1 回に要する平均時間を計測した．排他制御を行う場合はエージェント数によらず 1 秒間に約 10 回程度，マルチキャストでは 30 個のエージェントが結合した際でも 1 秒間に約 2 回程度のマルチランデブが可能であり，数十ノード程度の端末間の同期を実用的な時間で処理できることが分かった．

6. ま と め

通信相手が動的に変化するような環境においてグループ通信機能を提供する Java ミドルウェアを提案した．提案ミドルウェアは，相手を特定しない環境におけるグループ構築機能，および，グループ内でのマルチランデブに基づくグループ通信機能を提供する．提案ミドルウェアを用いることで，マルチキャストや排他制御を含むような分散協調アプリケーションにおける通信仕様を単純な構造で設計できた．

今後の課題としてはより実環境に近い形の実験，ミドルウェアが提供するマルチランデブ機能の高速化，同期関係を効率的に構築できる拡張などを予定している．本提案ミドルウェアでは，結合順序によりエージェント間の同期関係が変化するため，目的とする同期関係を構築・維持するためには，エージェントの結合順序の制御が必要になることもある．我々の研究グループでは文献 17) において，結合順序に依存せず，任意の同期関係を構築可能な形式モデルを提案しており，このモデルに基づいてミドルウェアを改良することで，問題を解決できると考えている．責任ノードの決定方法の最適化や，メッセージのキャッシュなどの

☆ CPU: Pentium III 1 GHz (Dual Processor), Memory: 1 GB, OS: Debian GNU/Linux 3.0, Java 2 SDK, Standard Edition, version 1.4

☆☆ CPU: Pentium III 1.13 GHz, Memory: 384 MB, OS: Windows XP Professional, Java 2 SDK, Standard Edition, version 1.4

手法により，高速な実行が行えるよう改良できると思われる．また，現状では，提案ミドルウェアは同期通信ベースの通信機能のみ提供しているため，マルチメディアデータなどのストリーミングの用途にはあまり適していない．今後，エージェント間の非同期通信機構を取り入れることで，マルチメディアストリーミングなども利用できるよう拡張したいと考えている．

参 考 文 献

1) Kortuem, G.: Proem: A Peer-to-Peer Computing Platform for Mobile Ad-hoc Networks, *Proc. Advanced Topic Workshop Middleware for Mobile Computing, In association with IFIP/ACM Middleware 2001 Conference* (2001).
2) Meier, R., Killijian, M.O., Cunningham, R. and Cahill, V.: Towards Proximity Group Communication, *Proc. Advanced Topic Workshop Middleware for Mobile Computing, In association with IFIP/ACM Middleware 2001 Conference* (2001).
3) 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).
4) Chu, Y.-H., Rao, S.G. and Zhang, H.: A Case for End System Multicast, *Proc. ACM SIGMETRICS*, pp.1–12 (2000).
5) 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).
6) Chen, K. and Nahrstedt, K.: Effective Location-Guided Tree Construction Algorithms for Small Multicast in MANET, *Proc. IEEE INFOCOM2002* (2002).
7) 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).
8) Bellavista, P., Corradi, A. and Stefanelli, C.: Protection and Interoperability for Mobile Agents: A Secure and Open Programming Environment, *IEEE Trans. Comm.*, pp.961–972 (2000).
9) Bellavista, P. and Stefanelli, C.: Mobile Agent Middleware for Mobile Computing, *IEEE Computer*, Vol.34, No.3, pp.73–81 (2001).
10) Chatterjee, M., Sas, S.K. and Turgut, D.: An On-Demand Weighted Clustering Algorithm (WCA) for Ad hoc Networks, *Proc. IEEE Globecom 2000*, pp.1697–1701 (2000).
11) ISO: Information Processing System, Open Systems Interconnection, LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807 (1989).
12) Hoare, C.A.R.: Communicating Sequential Processes, *Comm. ACM*, Vol.21, No.8, pp.666–677 (1978).
13) Welch, P.H., Aldous, J.R. and Foster, J.: CSP Networking for Java (JCSP.net), *Proc. Computational Science* (*ICCS 2002*), LNCS 2391, pp.695–708 (2002).
14) 西垣弘二，安本慶一，梅津高朗，東野輝夫，伊藤　実：マルチランデブによるグループ通信機能を提供する携帯電話アプリケーションのためのミドルウェアの実現，マルチメディア，分散，協調とモバイルシンポジウム論文集（DICOMO'2003），pp.757–760 (2003).
15) 安本慶一，中田明夫，寺島芳樹，梅津高朗，東野輝夫，谷口健一：マルチランデブチャネルの動的確立機構を持つモバイルアプリケーション記述言語の提案，コンピュータソフトウェア，Vol.19, No.2, pp.35–46 (2002).
16) Yasumoto, K., Higashino, T. and Taniguchi, K.: A Comipler to Implement LOTOS Specifications on Distributed Environments, *Computer Networks*, Vol.36, No.2–3, pp.291–310 (2001).
17) 梅津高朗，安本慶一，中田明夫，東野輝夫：ピアツーピアアプリケーション記述のための形式モデルの提案，マルチメディア，分散，協調とモバイルシンポジウム論文集（DICOMO'2004），pp.619–622 (2004).

(平成 15 年 8 月 14 日受付)
(平成 16 年 9 月 3 日採録)

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．

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 年同大学大学院基礎工学研究科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．

中田　明夫（正会員）

平成 4 年大阪大学基礎工学部情報工学科卒業．平成 9 年同大学大学院基礎工学研究科博士後期課程修了．博士（工学）．同年広島市立大学情報科学部助手．現在，大阪大学大学院情報科学研究科助教授．実時間システムや分散システムの仕様記述と検証法，プロセス代数，時相論理等の研究に従事．

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 年同大学大学院基礎工学研究科博士課程修了．同年同大学助手．現在，同大学大学院情報科学研究科教授，工学博士．分散システム，通信プロトコル，モバイルコンピューティング等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．