分散オブジェクト指向計算環境におけるグループ通信

和田智仁† 吉田隆一†

グループおよびグループ通信は，分散環境上でのプログラミングを支援する強力な抽象となりうる．本論文では，現在我々が開発中のオブジェクト指向計算環境DOOCEにおけるオブジェクトグループの枠組みの導入とグループ通信機構の実現について述べる．DOOCEは，グループを定義するためにオブジェクトリストとグループクラスの2つの記述法を用意し，これらによって定義されたオブジェクトグループに対するメッセージをグループ通信機構によって配送する．DOOCEグループ通信ではグループ通信メッセージの不可分性を保証し，さらに原子性を保証可能とし，メッセージの一貫性に関する問題をプログラマから隠蔽している．グループ通信機構では階層的な配送構造を採用し，グループメンバの局所性を利用した効率の良い配送を行い，また一貫性制御の負荷を各階層に分散している．これらによりDOOCEでは，記述言語レベルのオブジェクトによるグループを簡単かつ効果的に用いたプログラミングが可能である．

Group Communication in a Distributed Object-oriented Computing Environment

TOMOHITO WADA† and TAKAICHI YOSHIDA†

Groups and group communication can be powerful abstractions that offer many benefits to programmers in distributed computing environments. In this paper, we introduce a framework for object groups in our distributed object-oriented computing environment DOOCE and describe its implementation of group communication. DOOCE provides two methods of description for object groups, that is object list and Group class libraries, and transmits all messages to groups with group communication. DOOCE hides from programmers the problems which relate to consistency among group messages by delivering group messages in an indivisible manner, and providing the option to deliver the messages atomically. DOOCE runtime system employs a hierarchical delivery structure for group communication and delivers messages using locality of group members. Protocols to ensure indivisibility and atomicity of group messages are also realized hierarchically to distribute the corresponding load to each hierarchy. With DOOCE, programmers can describe an application using group abstraction in an easy and effective way.

1. はじめに

ネットワーク環境上での分散計算を支援する分散オブジェクト指向技術CORBA[11]や，オブジェクト指向言語においてネットワーク透過なリモートメソッド呼び出しを可能とするHORB[6]やJava RMI[14]，Voyager[10]などの手法が広まっている．これらを利用したアプリケーションでは，通常オブジェクト指向言語によって記述されたオブジェクト群をネットワーク上の複数の計算機に分散配置し，これらを協調動作させることによって全体の計算を行う．このような計算を行う場合に，複数のオブジェクトをひとまとめにし，オブジェクトグループとして取り扱うことを可能とすれば，オブジェクト群を利用するユーザの負担を軽減できると考えられる．しかし，現在利用されている計算環境や言語のほとんどは，言語レベルでのオブジェクトグループ，あるいはこれらのグループに対する通信（グループ通信）をサポートしていない．

オブジェクトグループの枠組みをプログラムから直接的に利用するには，オブジェクトの集合を定義する手段やグループに対するメッセージ送信の手段，さらに，グループからの返答の受信手段を記述言語においてサポートする必要がある．また，一般にグループに対するメッセージ送信は，受信者が複数存在し，これらの間のメッセージ受信に関する一貫性が問題になるなどpoint-to-point通信とは異なった性質を持つ．

† 九州工業大学情報工学部知能情報工学科
Department of Artificial Intelligence, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology

これらの問題をプログラマから隠蔽し，より使い勝手の良いグループ通信をプログラマに提供するには，グループ通信機構に何らかの工夫が必要になると考えられる．

グループのメンバ数あるいは分布はアプリケーションによって多様にわたる．特に，分散オブジェクト指向計算においては，OSで提供されるプロセスレベルのグループ通信やローカルなネットワーク内におけるマルチキャストと異なり，プログラムに現れる任意の(数/大きさの)オブジェクトがグループを構成することが可能で，これらのオブジェクトは広範囲のネットワーク上に広がる場合がある．したがって，言語レベルのオブジェクトにより構成されるグループを対象とするグループ通信の実現にあたっては，グループの規模に依存せず，かつ効率的なメッセージ配送の仕組みが要求される．

本論文では，現在我々が開発中の分散オブジェクト指向計算環境DOOCE[15]におけるオブジェクトグループの導入について述べる．まず，プログラム内でオブジェクトグループを定義する記述法を与え，オブジェクトグループに関するクラスライブラリを提供した．グループからの返答の受信に関しては，いくつかの選択ができるように記述法を与えた．さらに，実行時環境としてオブジェクトグループ通信を実現する機構を実現した．グループ通信機構は，メッセージ受信に関する一貫性を保つため，不可分性と原子性の2つの性質を保証している．オブジェクトグループへ送られるメッセージは，階層的なグループ通信機構によって効率良くメンバに配送される．

本論文は以後次のように構成されている．2章では，DOOCE記述言語におけるグループの定義および利用方法について説明し，3章でグループを用いた簡単なプログラム例を示す．4章では，階層的なDOOCEグループ通信機構の設計について述べ，5章では実装の詳細について述べる．最後に，6章でDOOCEグループ通信の記述/機構の評価を行う．

2. オブジェクトグループの記述

本章では，オブジェクトグループの枠組みを導入する土台となるDOOCEについて簡単に説明したのち，DOOCEオブジェクトグループの記述について解説する．

2.1 分散オブジェクト指向計算環境DOOCE

DOOCEは，オブジェクト指向言語と分散計算実行環境の自然な統合を目指したもので，プログラミング言語とその実行環境から構成されている．DOOCE

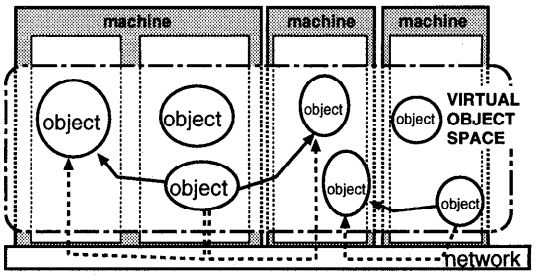


図1 DOOCE仮想オブジェクト空間
Fig. 1 DOOCE virtual object space.

はネットワークで接続された計算機群の上に，OSの提供するアドレス空間にまたがる仮想的なオブジェクト空間を提供し（図1），プログラマはDOOCE記述言語☆を用いてこの空間上のオブジェクトを記述する．OSの提供するアドレス空間は，プログラマが生成するオブジェクトの存在するユーザ空間と，システムが提供するオブジェクトの存在するシステム空間に分かれるが，システム空間はプログラマからは隠蔽されている．すべてのユーザオブジェクトはシステム空間に存在するシステムオブジェクトに自動的に登録/管理される．システムオブジェクトは，空間内ユーザオブジェクトのやり取りするメッセージの送受信やスレッドの生成/割当てなどを行っている．DOOCEプログラムは，トランスレータによってこれらのシステムを利用するコードへ変換された後，コンパイルされ，実行可能となる．このため，仮想空間上のすべてのオブジェクトに対してプログラム上から同一の方法でアクセス可能であり，プログラマはシステムやネットワークの介在を意識することなくアプリケーションの記述を行うことができる．

DOOCEではオブジェクト間のメッセージ交換によって計算が進行する．基本的なメッセージ送信の手段として，メッセージ送信後にその返答を待つ同期通信と，返答を待たずに次の動作を開始できる非同期通信の2種類を提供している．DOOCEではプログラミングを容易にするために，同一オブジェクトから非同期通信を用いて送信されたメッセージが，先に送信されたメッセージを追い越さないことを保証している．

2.2 DOOCEグループ通信

グループに対するメッセージの送信では，複数の受信者オブジェクト間のメッセージ受信に関する一貫性を保証するために，新たに次の2つの性質を提供する．

1つは，**不可分性（indivisibility）**の保証である．

☆ 現在のDOOCE記述言語はC++をベースとしている．

これはグループ通信が他のグループ通信によって分割されないことを保証するもので，DOOCE グループ通信はこの性質をつねに保証する．この性質によって，プログラマはグループに送信されるメッセージがグループ内メンバ間でつねに一貫した順序で受信されると仮定できる．

もう1つは，**原子性（atomicity）**の保証である．グループ通信に原子性が指定されると，そのグループ通信メッセージはすべてのメンバによって受信されるか，あるいは1つも受信されないかのどちらかとなることが保証される．原子性の保証はコストが大きくなると考えられ，また，グループの利用によっては原子性が必要でない場合もあるので，原子性の保証はプログラマにより選択的に行えるとした．

また，グループ通信においても point-to-point の通信と同様に非同期通信時のメッセージの追い越しがないことを保証する．これらの性質を保証することにより，プログラマを一貫性に関する煩わしさから解放できる．

2.3 オブジェクトグループの定義

プログラム中でオブジェクトグループを定義し，これを利用可能とするために DOOCE 記述言語に**オブジェクトリスト**と呼ばれる新たな記述を導入するとともに **Group クラスライブラリ**を提供する．これらによって定義されるグループに対するメッセージは，すべて後述のグループ通信機構によって配送される．さらに，返答の受信を柔軟に記述するための方法も与える．

2.3.1 オブジェクトリスト

オブジェクトリストは，オブジェクトの集合をグループとして定義するための記述法である．オブジェクトリストに対するメッセージは，リストに列挙されるすべてのオブジェクトに対して送信される．オブジェクトリストは，グループ通信を利用する際の最もプリミティブな記述法であり，これによってユーザオブジェクトから直接グループ通信を利用したメッセージ送信が可能となる．オブジェクトリストは変数として宣言し，生成することも可能であり，この変数中のオブジェクト参照は動的に変更できる．オブジェクトリストを用いることで，プログラマは簡単にグループを定義し，グループ通信を利用できる．

グループリストの利用例を**図 2** に示す．オブジェクトリストは，リスト記述子 “[ ]” によって生成され，要素となるオブジェクトはこの中に “,” で区切って列挙する．図 2 では **gr1** と **gr2** とがグループを構成している．原子性を保証する配送を行いたい場合には **atomic** 指定子を用いて，これを指定することができる．

```
Graphic *gr1, *gr2;
Graphic* [gr1,gr2]->redraw();
atomic Graphic* [gr1,gr2]->properties();
```

図 2 オブジェクトリスト
Fig. 2 Object list.

2.3.2 グループクラス

グループリストを用いた場合には，送信者自身がアクセスの対象となるグループのメンバ構成を把握する必要がある．グループの構成を利用者から隠蔽したい場合や，より抽象的にグループを扱いたい場合に前述の方法は向かない．そこでグループをより抽象的な方法で表現し利用するために，DOOCE では **Group** クラスを提供する．**Group** クラスのオブジェクトはグループのメンバシップ情報を管理し，受け付けたメッセージをすべてのメンバに対してグループ通信を用いて送信する．**Group** クラスのオブジェクトへのメッセージ送信の記述は，通常のオブジェクトと同様であるので，グループを用いることをプログラマに意識させずに，オブジェクトグループのサービスを提供することも可能である．

Group オブジェクトは，**図 3** で示されるメソッドを持つ．**grp_join()**，**grp_leave()** は，引数に示されるオブジェクトをグループに参加/脱退させる場合に利用される．グループに対して原子的なメッセージの配送を行うか否かは，**Group** オブジェクトの属性として指定する．この属性は，**grp_atomic_delivery()** / **grp_nonatomic_delivery()** メッセージによって変更できる（初期値は nonatomic）．現在の属性を問い合わせるには，**grp_is_atomic()** が利用できる．また，**Group** オブジェクトを生成する際には，型チェックのためにメンバとなるオブジェクトの型（クラス名）を，“< >” 内に指定する．グループオブジェクトは，グループのメンバシップを管理するためのメッセージのほかに，ここで指定されたクラスが受信可能なすべてのメッセージを受け取ることが可能となる．したがって，**Group<T>** 型のオブジェクトは **T** 型のオブジェクトと同様に扱うことが可能である．

実際には **Group** クラスは抽象クラスであり，グループを管理するためのインタフェースおよびグループオブジェクトの特別な初期化作業のみが定義されている．プログラム中で **Group** オブジェクトを生成し，利用するには，このクラスを継承した実装を用意する必要が

```
class Group<class T> {
  // membership data and attributes
public:
 Group() = 0;
 virtual int  grp_join(T* obj);
 virtual int  grp_leave(T* obj);
 virtual T*[] grp_member( );
 virtual int  grp_size( );
 virtual int  grp_atomic_delivery();
 virtual int  grp_nonatomic_delivery();
 virtual int  grp_is_atomic();
 ...
};
```

図3 Group クラス
Fig. 3 Group class.

ある．ただし，標準的な動作を行う Group オブジェクトの実装として，UGroup が提供されている．UGroup オブジェクトは，一般のオブジェクトと同様のユーザ空間に生成される．また，メンバシップといったグループの情報を1カ所で集中的に管理している．このため，オブジェクトを複製化し，耐故障のためにグループを利用する場合に，UGroup オブジェクト自身の信頼性が問題となる可能性がある．そこで，我々は Group クラスを継承する RGroup クラスを用意した．RGroup クラスを用いると，一般のオブジェクトと異なり，より頑強な（単点故障に耐えうる）オブジェクトを生成できる．RGroup オブジェクトの信頼性は，後述の実行時機構によって保証されており，そのメカニズムについてプログラマが考慮する必要はない．ただし，RGroup では信頼性を高めるために暗黙にいくつかの複製を生成するため，UGroup に比較して実行時の負荷が大きくなることにプログラマは注意する必要がある．

2.4 返答の受信

グループに対するメッセージには，通常複数のメンバからの複数の返答が返されることになる．DOOCE では，これらの返答を受信するために3つの手段を提供している．

1つ目は，最初に返された1つの返答だけを受信するという特例である（図4 行1）．これは，プログラム中に返答を代入する変数が1つだけ与えられた場合，暗黙に最初の1つだけが受信されるという仕組みである．この特例によってメッセージの送信者は，宛先がグループであるか単独のオブジェクトであるかを意識せずに，必要となる返答を1つだけ入手することが可能となる．また，この特例によって，過去に単独のオブジェクトによって提供していたサービスを，クライアント側のプログラムを変更することなく，グループによって提供することが可能となる．

```
1: ans = foo->bar();
2: Ans*[a1,a2] = [o1,o2]->bar();
3: Ans*[a1,a2] = firstn[o1,o2,o3]->bar();
4: RHandler<Ans> *rh;
5: rh = foo -> bar();
6: while ( rh -> remain() )
7:     a[i++] = rh -> receive();
```

図4 返答の受信例
Fig. 4 Example of replies acceptance.

2つ目は，複数の返答を受信するためにオブジェクトリストを用いる方法である（図4 行2，3）．グループ通信メッセージの返答受信にオブジェクトリストを用いた場合に限り，返答とその送り元オブジェクトとの対応付けが可能である．たとえば，図4の行2の場合，o1 からの返答は a1 に，o2 からの返答は a2 に割り当てられる．また，arrvlorder と firstn 指定子を使うことによって，返答を到着順に受理するように指定できる．arrvlorder は，到着順に変数に返答を割り当てることを指示するもので，また firstn は最初に到着した n 個を変数に割り当てることを指示するものである．firstn が指示された場合には，受理する返答の数はオブジェクトリストの要素数に依存し，残りの返答は無視される．

第3の方法として，リプライハンドラと呼ばれる特別なクラスを用意した（図4 行4~7）．複数の返答が必要となる場合に，オブジェクトリストを用いた返答の受信では，プログラマはあらかじめ送り先グループのメンバ数もしくは受信する返答の数を知っておく必要がある．このため，グループの構成が動的に変化する場合には，オブジェクトリストによる返答の受信の記述は不可能である．リプライハンドラは，グループからの複数の返答を送信者に代わり受信し，保持する．メッセージを送信したオブジェクトは，リプライハンドラに対して receive() メッセージを送信することによって，その返答を受け取ることができる．リプライハンドラは，receive() メッセージを受けると，受信した（する）返答を到着順に返す．リプライハンドラからは，到着した（する）すべての返答を読み出す必要はない．

3. 記 述 例

本章では，オブジェクトグループを使ったアプリケーションの記述例を示す．

グループを用いてオブジェクトを複製化し，これに

```
Server::Server(string g_name){
  RGroup<Server> *svg;
  svg = ns->lookup(g_name);
  if ( svg == 0 ) {
      svg = new RGroup<Server>;
      svg->grp_join(this);
      ns->entry(svg,g_name);
  }
  else {
      svg->grp_join(this);
      this->copy(svg);
  }
}
```

図5 複製グループの例
Fig. 5 Example of a replicated group.

```
Server *svr;
svr = ns -> lookup("aServer");
ans = svr->request();
```

図6 グループ通信利用例
Fig. 6 Example of describing group communication.

よって信頼性の高いサービスを提供するサーバクラスの例を図5に示す．Serverでは，コンストラクタを用いて複製グループを次のように構成する．Serverのコンストラクタでは，まず指定された複製グループ（g_name）がDOOCE名前サービス（ns）にすでに登録されているかどうかを調べる．指定された複製グループが見つからなかった場合には，そのオブジェクトが複製グループを構成する最初のオブジェクトであるので，そのオブジェクトはグループを作成し，そのグループに参加した後，そのグループを名前サービスに登録する．複製グループが見つかった場合には，そのグループに参加し，既存の複製オブジェクのコピーを行う．サービスを提供する側は，Serverオブジェクト生成時に同一の複製グループ名を指定するプログラムを，いくつかの計算機上で実行すればよい．

図6は上述の複製グループが提供するサービスを利用する例である．ただし，複製グループはGroupクラスを用いて構成されているため，サービスの利用者は，aServerがグループでサービスを提供しているのか，単独のオブジェクトによってサービスが提供されているのかを問題とする必要がない．

4. グループ通信の設計

DOOCEグループ通信の設計にあたっては，以下の要求事項およびシステムの性格に留意した．まず，要求事項として，次の2点があげられる．

不可分性の実現 不可分性を実現する方法として，メッセージ配送順序の一貫性制御を用いる方法がある．順序制御は1カ所で集中的に行えば容易に実現できるが，この方法では集中型システムの欠点を持つ．分散的な手法による順序制御は一般に高価であり，またメンバ数に比例して制御に必要となるコストが大きくなるため，DOOCEのようにメンバ数が多くなる場合に向かない．

原子性の実現 原子性は，2相コミット[5]を用いたメッセージの配送によって実現できる．2相コミットを用いたメッセージ送信の場合，コミット可否メッセージをすべての受信者から集める必要があり，メンバ数が多い場合にこれを収集するコストが問題となる．

システムの性格としては，次の2点があげられる．

オブジェクトの数および広がり プログラム中のオブジェクトは環境中の計算機やプロセスなどに比べ一般に数が非常に多く，グループ中のメンバ数も大きくなる可能性がある．また，DOOCE仮想オブジェクト空間は広範囲に広がることが可能で，空間中に存在するグループ数も大きくなる可能性がある．したがって，グループ通信機構はメンバ数やグループ数などの規模に依存しない方法によって実現される必要がある．

メンバの局所性 一方，一般の分散システムと同様に，相互に関連するオブジェクトは物理的に近い位置，すなわち，同一アドレス空間上，同一計算機上，同一サブネットワーク上に存在する可能性が高い．したがって，グループメンバの物理的な局所性を利用したメッセージの配送を行えば，通信コストを低減できる．

上記の留意点より，DOOCEではグループ通信メッセージの配送を行うシステムオブジェクト（**マルチキャストマネージャ**）をアドレス空間内のシステム空間およびDOOCEアプリケーションが動作する計算機上に階層的に配置し，階層構造を利用したグループ通信メッセージの配送を行うこととした（図7）．階層的な構造を利用することで，制御に必要となる負荷はそれぞれの階層に分散され，システム全体の拡張性を保存できる．また，この方式ではグループメンバが階層構造中に局所的に存在する場合に，その局所性を利用したメッセージの配送が可能となり（図7右下），通信コストを軽減できる．

マルチキャストマネージャの基本的な動作は次のとおりである．

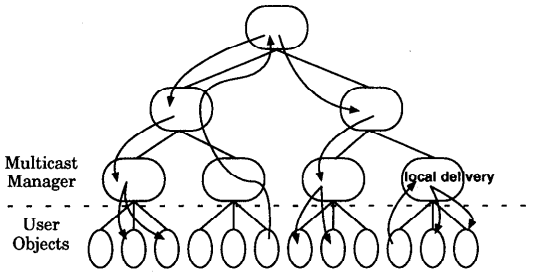


図7 階層的なメッセージ配送
Fig. 7 Message delivery hierarchically.

1. マルチキャストマネージャが下の階層のオブジェクトからグループ通信要求を受け付けたとき，

1-1 宛先となるグループメンバがすべて自己の管理下にある場合，これをグループのメンバオブジェクトに配送するか（最下層の場合），あるいは，宛先となるメンバが存在する下の階層のマルチキャストマネージャに配送依頼する．

1-2 自己の管理下のオブジェクト以外がこのグループのメンバに含まれている場合，そのメッセージを上の階層のオブジェクトにグループ通信要求としてそのまま転送する．

2. 上の階層のオブジェクトからグループ通信配送依頼が到着したとき，これをグループのメンバオブジェクトに配送するか（最下層の場合），あるいは，宛先となるメンバが存在する下の階層のマルチキャストマネージャに配送依頼する．

ここで，不可分性を保証するために，マルチキャストマネージャはすべての配送依頼を逐次的に行う．すなわち，1つのメッセージの配布が終了してから，他のメッセージの配布にとりかかる．これにより，あるグループに対するメッセージは，そのメッセージが到達する最上位の層で必ず順序付けされることになり，下位のシステムオブジェクトにおいても逐次的に配送を行う限り，グループ通信メッセージは不可分に配送されることになる．この方法は，グループメンバにオーバラップがある場合においても不可分な配送を保証できる．

グループ通信に原子性を保証することが指定された場合には，2相コミットを用いた配送を行う．ここでは階層構造を利用することで，コミット可否の確認を分散的に行うことができる．2相コミット操作は，グループ通信メッセージが到達した最上位の階層のマルチキャストマネージャをコーディネータとして，1つ下の階層のマルチキャストマネージャをサブオーディネータとして行えばよい．以下の階層でも同様に，サブオーディネータであるマルチキャストマネージャが新たにコーディネータとなり，配布対象となるマルチキャストマネージャに対して2相コミット操作を行う．これによって，グループメンバ数が増加した場合においても，コーディネータにかかる負荷は各階層に分散できる．

また，上述のグループ通信機構では，広範囲に存在するグループに対するメッセージを転送した後，より局所的に存在するグループに対してメッセージを配送した場合に，メッセージの追い越しが発生する可能性がある．すなわち，上の階層に要求したメッセージの宛先に自身が管理するオブジェクトが含まれていて，かつ，このメッセージの配送がいまだ終了していない場合に自身が局所的なメッセージの配送を行うと，メッセージの配送順序に追い越しが発生する可能性がある．point-to-point のメッセージ送信同様，メッセージに追い越しが発生しないことを保証するためには，上記のような場合に先に上位に要求したメッセージの配送が終了してから，後に受けた（局所的な）メッセージの配送を行えばよい．

5. グループ通信の実装

今回の実装では，前述の階層を DOOCE 環境の物理的な構成に対応させて，以下の4階層とした（**図8**）．

(1) OS の提供するアドレス空間
(2) 計算機内
(3) ブロードキャスト到達可能なサブネットワーク
(4) システム全体

物理的な構成と階層構造を対応させることで，局所的なメッセージの配布を行う際により効率的な配布が可能となる．また，ブロードキャスト到達可能なサブネットワークを1つの階層とすることで，この階層内でのメッセージ交換にブロードキャストを利用できるようになる．ブロードキャストのような低レベルのグループ通信プリミティブの利用は，メッセージ帯域の消費を低減する．

これらの階層に，下位の階層から順に，**ローカルマルチキャストマネージャ（LMM），サブマルチキャストマネージャ（SMM），およびマスタマルチキャストマネージャ（MMM）**の3種類のマルチキャストマネージャを置いた．

ただし，最上位の階層にはマルチキャストマネージャを置かないことにした．これは，最上位のマルチキャストマネージャの故障や，ネットワークの分断の際に，サブネットワークをまたがって存在するグループに対するグループ通信がいっさい行えなくなってしまうのを避けるためである．また，DOOCE 環境が非常に広

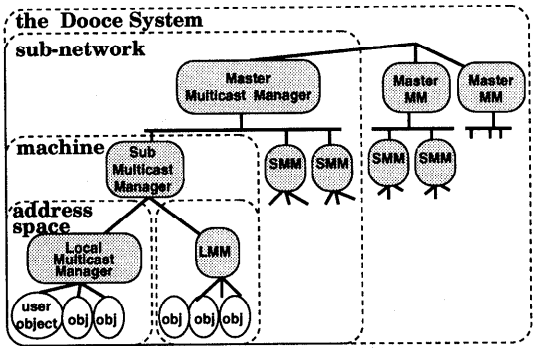


図8 Dooceグループ通信機構
Fig. 8 Dooce group communication structure.

域まで広がった場合におけるサブネットワーク数の増加とそれにともなう負荷の集中を考慮しても，この方式が優れると考えられる．この方式では最上位の階層でのみメッセージの一貫性制御を分散的手法によって実現する必要が生じるが，最上位では制御対象の数がメンバが存在するサブネットワークの数まで少なくなるために，このコストも大きくならないと予想できる．

各マルチキャストマネージャは以下の機能を持つ．

LMM OSの提供するアドレス空間に1つ存在するマルチキャストマネージャ．プログラム中のすべてのグループ通信の記述は，Dooceトランスレータによって LMM へのグループ通信要求に置き換えられる．LMM はアドレス空間内のすべてのユーザオブジェクトを担当しており，ユーザオブジェクトからのグループ通信要求を受け付け，これを処理するとともに，ユーザオブジェクトへの最終的なメッセージの配送を行う．

SMM Dooceオブジェクトが存在する計算機上に1つ配置されるマルチキャストマネージャ．同一計算機内のすべての LMM を担当し，メッセージの中継を行う．

MMM サブネットワーク内の SMM から1つだけが MMM として選出される．サブネットワーク内のすべての SMM を担当し，また，このネットワークの代表として他の MMM とメッセージの交換を行う．

すでに述べたとおり，階層構造の最上位にはマルチキャストマネージャは置かない．したがって，MMM は共同してシステム全体のマルチキャストマネージャの役割を果たす．そのため，MMM は，サブネットワーク内の SMM から受信したメッセージと外部のネットワークから受信したメッセージを一意に順序付け，この順序で SMM に配送する．MMM 間で交換されるグループ通信メッセージは total ordering[2),3)] の手法を用いて順序付けされる．また，MMM から SMM へのメッセージはネットワーク帯域の消費を抑制するために，ブロードキャストを使って伝えられる．ブロードキャストメッセージの紛失を防止するために，ここでは Amoeba[9),13)] で用いられている高信頼ブロードキャストプロトコル[7)] を採用している．

RGroup のオブジェクトは，信頼性を高めるために，それが生成されたサブネットワーク内の（MMM を含む）すべての SMM 上に複製を置くこととした．したがって，RGroup オブジェクトへのメッセージは，それが生成されたアドレス空間まで配送されずに，自計算機の SMM 中に存在する RGroup オブジェクトに渡される．ここで RGroup の状態は，SMM 間で一貫した状態でないと不都合が生じる．このため，RGroup へのメンバ変更等のメッセージは局所配送できないメッセージと同様に MMM に渡され，ブロードキャストにより全 SMM に一意な順序で配布される．

6. 評　価

6.1 グループの利用に関する評価

OS レベルのプロセス間グループ通信を提供する Amoeba や ISIS[1),2)] などの分散システムにおいては，グループ通信プリミティブを SendToGroup や ReceiveFromGroup，あるいは ABCAST，CBCAST といったグループ通信特有の関数で提供する場合が多い．これに対して Dooce ではオブジェクトリストあるいは Group クラスを用いてグループを定義すれば，これらに対するメッセージはすべて自動的にグループ通信を利用して配送される．このため point-to-point 通信とグループ通信とに別々の記述を用いる必要がなく，これら2種類の通信を同一のシンタックスから利用できる．さらに，返答の受理の記述では，最初の1つの返答だけを暗黙に受理するという特例を設けており，より透過的にグループ通信を利用することが可能である．

Emerald[12)] は，記述言語におけるオブジェクトのグループをサポートするオブジェクト指向言語とその実行環境の1つである．Emerald では，*channel* オブジェクトを用いたグループ通信を提供している．すなわち，ある channel に属するオブジェクト群を1つのグループとし，channel オブジェクトに対して broadcast 節を適用することでグループのメンバにメッセージが配送される．

Voyager[10)] は，100% Pure Java の分散オブジェクト技術（DOT）で，Java オブジェクトによるグループをサポートしている．Voyager では，*Space* と呼ばれるオブジェクトによってグループを定義できる．Space

に対してメッセージ送信を行うと，Spaceに属するオブジェクトにそのメッセージが配送される．

これらに対しDOOCEでは，すべてのオブジェクトが直接グループ通信を利用できるように，言語自体を拡張し，オブジェクトリストを提供している．すなわち，特殊な能力を持つシステムオブジェクト（channel, Space）を提供する代わりに，それぞれのオブジェクトにグループ通信を利用する能力を付加した，ということができる．これにより，DOOCEでは記述言語を用いてユーザ自身がカスタマイズしたGroupクラスを定義することが可能である．また，Groupやchannel, Spaceを使わずに，自らグループ通信を行えるため，それらを生成/設定する手間を省き，(グループを共有，繰り返し利用しない場合に）プログラムの記述を容易にすることもできる．

Voyagerでは，Spaceは実体のない論理的なグループで，実際にはSubspaceと呼ばれるより局所的なグループが互いにリンクを張り，論理的Spaceを構成する．Subspace間では，Subspace中に属するメンバの数にかかわらずメッセージの交換を1回だけしか行わない．このため，プログラマはSubspaceを用いて効果的にサブグループ化することで，ネットワークを流れるメッセージ数を低減し，配送作業を並列化できる．DOOCEでは，プログラムからこのような工夫をする手段を与えていない．しかし，グループ通信機構が局所性を利用した配送を自動的に行うため，プログラマがこれらについて考慮する必要性はまったくない．

DOOCEグループ通信が保証する不可分性は，グループ通信メッセージが他のグループ通信メッセージによって分割されないことを保証するものである．この性質は，グループ通信メッセージの全順序による配送を要求しない．したがって，シーケンサを利用するAmoebaのグループ通信や論理時刻を用いるISISのABCASTと異なり，通信機構では局所的かつ並列なメッセージの配送が可能となり，システム全体での通信コストを低減できる．問題は，全順序よりも弱い一貫性を保証する不可分性がプログラミングにどのように影響するかである．しかし，不可分性は関連するオブジェクト間におけるグループ通信メッセージの到着順序をつねに一貫させ，また，グループメンバにオーバラップがある場合でもその一貫性を保証する．一方で，DOOCEでは同一オブジェクトから送信されたメッセージの追い越しがないことを保証するため，グループ通信メッセージの因果順序は保たれる．このため，不可分性はグループを利用する際のメッセージ配送順序に関して必要十分な性質をプログラマに提供している，と考えられる．

また，今回Groupクラスでは，原子的な配送を行うか否かをグループの属性とした．このため，いったんグループに原子性を指定すると，これが必要でないメッセージについても原子的な配送を行わざるをえない．よりきめ細かい指定を可能とするには，メソッド単位でこれを指定できるような記述法が必要になると考えられる．

DOOCEでは，グループオブジェクト自体にアクセス不可能となった場合には，たとえグループ中のいくつかのメンバとアクセス可能であった場合でも，グループは利用できなくなる☆．したがって，より可用性の高いサービスを提供したい場合にこれが問題となる可能性がある．プロセス間グループ通信を提供するToransis[4]やTotem[8]では，ネットワークが分割した場合でもグループ活動を続ける手段（partitionable operation）と，さらに再接続した場合の結合手段（merging）を提供し，可用性の高いグループを実現できるとしている．Partitionable operationやmergingなどの手段を含め，ネットワーク分割に際して，どのような記述手段が要求されるのか，といった点についても今後検討する必要がある．

6.2 実装および性能に関する評価

DOOCEではグループ通信機構に階層構造を採用し，グループメンバの存在する場所を利用したメッセージの配送を行う．グループを構成するメンバの粒度が小さく，メンバ数が多くなる可能性のあるDOOCEでは，階層的な配送は交換するメッセージ数を低減するため，より効果的であると考えられる．さらに，局所的な配送のため，グループメンバが近い場所に存在する場合に短い時間で通信を終了することが期待できる．

今回の実装を用いて，実際のグループ通信にどの程度の時間が必要になるかを測定した．結果を**図9**～**図12**に示す．ここでは，DOOCE環境上に分散する複数のオブジェクトに対して，point-to-point通信（同期/非同期）とグループ通信（原子性無保証/原子性保証）を使ってメッセージを送信し，それらからの返答をすべて受信するまでの時間をユーザプログラム上で測定した．図で縦軸は通信時間を，横軸は通信対象となるオブジェクトの数を示している．それぞれの実験ではオブジェクトの分布をDOOCEグループ通信機構のある階層以下に制限し，測定を行っている．使用した計算機は，SunOS 4.1が動作しているSun

☆ RGroupによるグループでも，複製はそれが生成されたネットワークだけに存在するため，ネットワーク分割には対応できない．

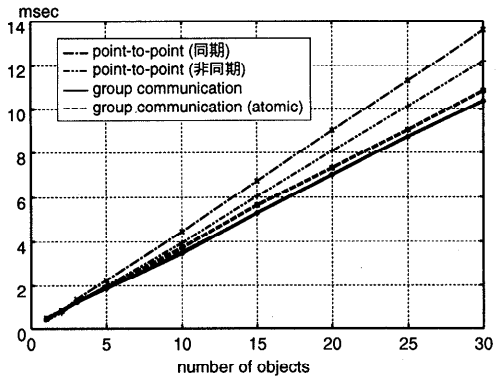


図 9 通信時間（同一アドレス空間内オブジェクト群）
Fig. 9 Communication time (in an address space).

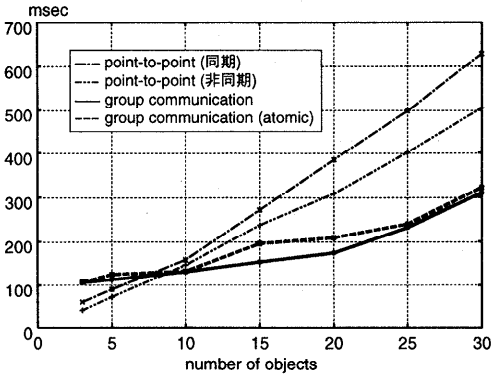


図 11 通信時間（同一サブネットワーク内オブジェクト群）
Fig. 11 Communication time (in a sub-network).

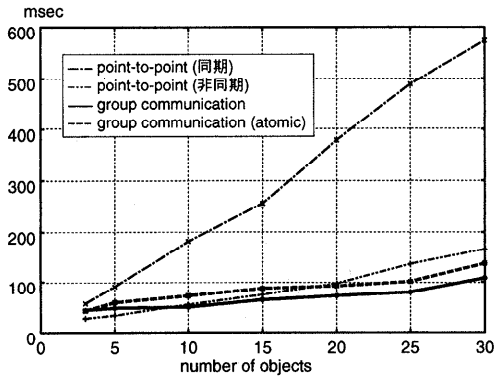


図 10 通信時間（同一計算機内オブジェクト群）
Fig. 10 Communication time (in a machine).

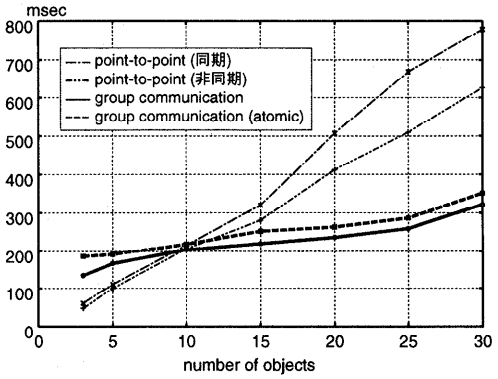


図 12 通信時間（複数サブネットワーク上オブジェクト群）
Fig. 12 Communication time (in DOOCE).

SparcStation 20（一部 SS10, SS5）を用いた．図 11 の実験で利用した計算機群は，スイッチングハブによる 10 Mbps Ethernet により接続されている．また，図 12 の実験では，ICMP による 400 Byte☆のパケット往復に 3 msec～5 msec 程度要する（比較的近距離にある）ネットワークを利用している．

図 9 は，単一アドレス空間内のオブジェクトのみを対象とし，これと同一のアドレス空間内から通信した結果である．メッセージはアドレス空間内のみで局所的に配送されるため，プロセス間の通信がまったく不要であり，非常に短い時間で通信を終了している．図 10 は，単一計算機上に最大 6 つのアドレス空間を生成し，これらの上のオブジェクトに対して，同一計算機の別のアドレス空間から通信した結果である．ここでも，point-to-point 通信を用いた場合と同等またはそれ以下の時間でグループ通信を用いたメッセージの送受が終了していることを確認できる．グループ通信では，メッセージがいっせいにメンバに配送されるため，point-to-point の非同期通信と同様に，メンバ間の並行性の効果が表れている．図 11 は，単一サブネットワーク内の最大 6 台の計算機上に，1 つずつのアドレス空間を生成し，それと同一サブネットワーク内の計算機から通信した結果である．ブロードキャストなどに要する時間が大きくなるため，メッセージを送付する対象数が少ない場合に point-to-point の通信より多くの時間が必要となるが，メンバ数が 10 を超える程度に大きくなるとグループ通信の方が短い時間で通信を終了している．これは，グループ通信では階層化とブロードキャストの利用によって，point-to-point の通信に比べてプロセス間通信の回数が少ないためと考えられる．図 12 では，最大 3 つのサブネットワーク上の最大 6 台の計算機にアドレス空間を 1 つずつ生成し，これの上に存在するオブジェクトに対して通信した結果である．ここでは，マスタマルチキャストマ

☆ 実験における，リクエストメッセージの平均パケットサイズ．

ネージャ間で分散的手法によるメッセージの順序付けが行われるが，図 11 の結果と比べて最大数十%程度しか増加しておらず，極端に大きな時間を必要としないことが分かる．

これらの実験から，グループ通信はメッセージの不可分性を保証するにもかかわらず，point-to-point 通信をメンバの数だけ繰り返し行うのと同程度の時間またはそれ以下の時間で終了することを確認できた．また，原子性を保証するように指定した場合でも，指定しなかった場合と比べて平均 10%，最大でも 40%程度のコスト増加となることも確認した．これらは，階層化による通信回数の減少，ブロードキャストの利用，並行に行われる配送/二相コミット作業などの結果である．さらに，メンバの存在が局所的な場合には，その局所性に応じて通信時間が短くなることも確認した．これらより，全メンバに対して単純に全順序保証アルゴリズムや 2 相コミットアルゴリズムを適用した場合に比べて，DOOCE では効果的にメッセージの配送を行っているといえる．

今回の実装では階層の最上位にマルチキャストマネージャを置かずに，サブネットワーク間では関連するネットワークの代表どうしで直接メッセージ交換と順序制御を行う方式を採用している．この方式では，グループのメンバが少数のネットワークに分散している場合には，グループメッセージの配送対象とならない他のサブネットワークへの影響がまったくなく，分散的手法による一貫性制御のコストも小さいため，有効であると考えられる．しかし，多数のサブネットワークにグループメンバが分散するような場合には，順序制御のためのコストが大きくなり問題が発生する可能性がある．一方で，単なる問合せのメッセージのようにメッセージの到着順序が問題とならないようなグループ通信の利用も考えられる．このため，不可分性の保証についても，原子性の保証と同様に選択的に行えるようにすることを現在検討している．

Totem もブロードキャスト可能なローカルネットワーク（ドメイン）を 1 つの階層とした階層的な配送構造を採用しているシステムの 1 つである．Totem では，階層内のメッセージ送信の際にトークンリング方式によるフロー制御を行い，オーバフローによるブロードキャストメッセージの紛失および再送を低減している．今回の我々の実験では，トラフィックがそれほど多くないためオーバフローはほとんど発生していない．DOOCE において，どの程度のトラフィックでオーバフローが発生し，それが通信コストにどのように影響してくるかを，今後調査する必要がある．

7. おわりに

DOOCE では，オブジェクトリストと `Group` クラスを用いてオブジェクトグループを定義できる．また，暗黙の返答受信，リストによる受信，リプライハンドラによる返答受信の 3 つの手段を用意している．プログラマはこれらを用いることで，柔軟なグループおよびグループ通信の記述/利用が可能である．また，グループ通信は不可分性と原子性を保証でき，メッセージの一貫性に関する問題を通信機構側で解決する．したがって，DOOCE グループ通信は，単にメッセージを複数に送信するような場合から，レプリカグループのように受信メッセージの系列が重要になるような場合まで，さまざまな場面において簡単に，かつ直接的に利用できると考えられる．

グループ通信機構では階層的な配送構造を採用し，拡張性が高く，通信時間の小さいグループ通信を実現した．特に，グループのメンバに局所性がある場合には，局所性に応じた短い時間でメッセージを配送できることを確認した．

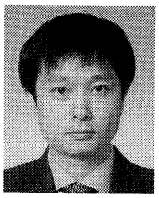
階層的な実装では，上位の階層の故障によって通信が行えなくなる可能性がある．今回の実装では最も上位には管理機構を設置せず，分散的な手法によってメッセージの伝達/順序制御を行い，単点故障によるグループ通信の停止を防いでいる．途中の階層についても同様のことがいえるが，現時点ではこれらへの対応は行っていない．耐故障性を高めるために，今後これらの故障への対応について考える必要がある．具体的には，マルチキャストマネージャの故障の検出，およびマスタマルチキャストマネージャの交替/選出についての対応が必要である．

参考文献

1) Birman, K.: The Process Group Approach to Reliable Distributed Computing, *Comm. ACM*, Vol.36, No.12, pp.37–53 (1993).
2) Birman, K. and Joseph, T.A.: Reliable Communication in the Presence of Failures, *ACM Trans. on Computer Sytems*, Vol.5, No.1, pp.47–76 (1987).
3) Dasser, M.: TOMP A Total Ordering Multicast Protocol, *Operating Systems Review*, Vol.26, No.1, pp.32–40 (1992).
4) Dolev, D. and Malki, D.: The Transis Approach to High Availability Cluster Communication, *Comm. ACM*, Vol.39, No.4, pp.64–70 (1996).
5) Gray, J.: Notes on Database Operating Sys-

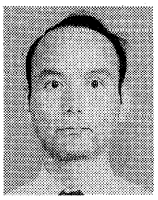
tems, *Operating Systems: An Advanced Course*, Bayer, R., Graham, R.M. and Seegmuller, G. (Eds.), pp.394–481, Springer-Verlag, Berlin (1978).
6) Hirano, S.: HORB Home Page, http://ring.etl.go.jp/openlab/horb/.
7) Kaashoek, M.F., Tanenbaum, A.S., Hummel, S.F. and Bal, H.E.: An Efficient Reliable Broadcast Protocol, *Operating Systems Review SIGOPS*, Vol.23, pp.5–19 (1989).
8) Moser, L., Melliar-Smith, P., Agarwal, D., Budhia, R. and Lingley-Papadopoulos, C.: Totem: A Fault-Tolerant Multicast Group Communication System, *Comm. ACM*, Vol.39, No.4, pp.54–63 (1996).
9) Mullender, S.J., Rossum, G.v., Tanenbaum, A.S., Renesse, R.v. and Staveren, H.v.: Amoeba: A Distributed Operating System for the 1990s, *Computer*, Vol.23, No.5, pp.44–53 (1990).
10) ObjectSpace: VOYAGER Core Technology User Guide 2.0 Beta 1 (1997).
11) OMG: The Common Object Request Broker Architecture Specification 2.2 (1998).
12) Pardyak, P.: Group Communication in an Object-Based Environment, *2nd International Workshop on Object Orientation in Operating Systems*, pp.106–116, IEEE Coumputer Society Press (1992).
13) Rossum, G.v. and Tanenbaum, A.S.: Short Overview of Amoeba, *the USENIX Workshop on Micro-Kernels and Other Kernel Architectures*, Berkeley, CA, USA, pp.1–10, USENIX Assoc (1992).
14) Sun Microsystems: Java Remote Method Invocation Specification 1.4 (1997).
15) 和田智仁ほか：分散環境におけるオブジェクト指向言語とその実行環境の開発，情報処理学会九州支部研究会報告，pp.288–296 (1996).

(平成10年 5 月8日受付)
(平成10年10月2日採録)



和田 智仁（学生会員）

1995年九州工業大学情報工学部知能情報工学科卒業．1997年同大学大学院情報工学研究科博士前期課程情報科学専攻修了．現在，同大学院博士後期課程在学中．分散オブジェクト技術，分散処理システム等に興味を持つ．



吉田 隆一（正会員）

1982年慶應義塾大学工学部電気工学科卒業．1987年同大学大学院工学研究科博士後期課程電気工学専攻修了．工学博士．同年九州工業大学情報工学部知能情報工学科助手．1990年同助教授．1993年から翌年にかけてオレゴン科学技術大学院大学において客員研究員．オブジェクト指向計算，分散計算，分散処理システム，オブジェクト指向データベースに興味を持つ．日本ソフトウェア科学会，人口知能学会，IEEE，ACM各会員．