分散処理環境におけるＩＰＣ支援システム

4U-4

矢吹道郎，稗田薫，Robert Deiters

上智大学

1．　はじめに

これまでのインター・プロセス・コミュニケーション（ＩＰＣ）はポイント・トゥ・ポイントの通信を基本としている。しかし、実際のプログラミング環境では複数のプロセス間の同時のＩＰＣが必要となり、マルチ・キャストのメッセージ通信も同様に重要である。

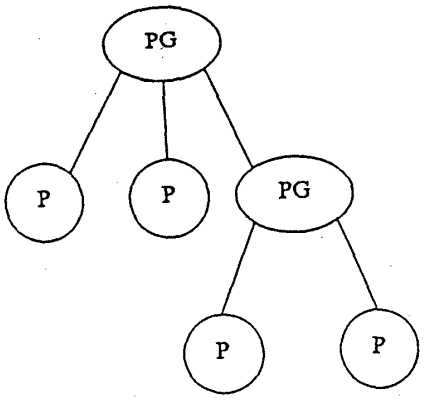
ここでは、分散環境における汎用のＩＰＣ支援システムの設計、並びに実装について報告する。このシステムでは、ネットワーク上における論理的なプロセスのグルーピングと階層的プロセス・グループのネイミングを導入しており、分散されたプロセスのグループ内においてマルチ・キャストを含むメッセージ通信を行ない、リソースを共有し、同期をとるプログラムを容易に作成することが可能である。実装はバークレイ版UNIX4.2BSD上で行なった。

2．　モデル

UNIX4.2BSDではＩＰＣの機能としてsocket[1]が実装されているが、ユーザにとって使いやすいとは言えず、他に、ネットワーク・ユーザのためのより高レベルのＩＰＣインターフェースとして、Courier[2]やＲＰＣ[3]などが実現されている。一般的なＩＰＣモデルとしては、複数のプロセスに対する通信が必要となり、マルチ・キャストのメッセージ通信が不可欠となる。マルチ・キャストを行なうためには、その通信相手の境界を指定する機構がなければならない。普通、ネットワークにおけるすべてのプロセスがお互いに通信を行なう必要は無く、同一の目的を持ったプロセス間でのみ通信を行なう。即ち、それはグループを形成していると考えることができる。そこで我々は、ネットワーク・プロセス・グループの概念を導入した。ネットワーク・プロセス・グループとは、インター・ネットワークにおいて協調して一つの仕事をなすもの、言い換えれば、何らかの通信を行なう可能性のあるプロセス群である。プロセス・グループは以下のような機能を備えていなければならない。

- ◆プロセスのグループへの参加と離脱のための機構
- ◆プロセスの実行
- ◆グループ内のプロセス（メンバ）へのメッセージの転送
- ◆グループのステイタス情報の維持とその通知
- ◆下位グループのアドレスの通知

プロセス・グループとプロセスの関係を[Figure 1]に示す。グループはグループあるいはプロセスをそのメンバとする。従って、グループは階層的構造をとる。そのため、ネットワーク・プロセス・グループの名前管理としては、UNIXのファイル・システムのような階層的な名前を用いた。



P : process
PG : process group

Figure 1
Process groups and processes

3．　実装

我々は、前述のＩＰＣモデルをUNIX4.2BSD上に実装した。プロセス・グループは前述のように単なる概念だけでなく、実際の機能を持たなければならない。それらの機能は、カーネルを変更しカーネルのサービスとすることも可能であるが、ここではポータビリティのためシステム管理者により実行されるUNIXのdaemonの形をとることとした。このdaemonはその機能から二つに分けられる。第一はグループ名をインター・ネットワークにおいてユニークに保ち、他のマシンのグループのアドレスを扱うするもので、アドレス・サーバ・プロセスと呼ぶ。第二は、実際のグループにおける処理を行なうもので、マネージメント・プロセスと呼ぶ。

3．1　アドレス・サーバ・プロセス

アドレス・サーバ・プロセスはこのシステムを用いるすべてのホストで予めdaemonとして実行されていなければならない。アドレス・サーバ・プロセスはすべての第一レベルのグループのアドレス（UNIX4.2BSDではポート）のデータを保持し、グループがネットワークにおいてユニークとなるように処理する。第二レベル以下のグループはそれぞれのマネージメント・プロセスにより取り扱われる。アドレス・サーバ・プロセスは以下の機能を持つ。

- ◆要求されたグループが存在するかどうか確かめる。
- ◆もし存在すればそのグループのマネージメント・プロセスのアドレスを要求者に返す。

IPC on Local Network Environment

Michirou YABUKI, Kaoru HIEDA, Robert DEITERS

Sophia University

◆存在しない場合、マネージメント・プロセスを実行しそのアドレスを要求者に返す。それと同時に、新たなグループがネットワークにおいてユニークとなるように、そのグループを他のホストのアドレス・サーバ・プロセスに登録する。

◆要求者のホストと異なるホストにおけるプロセスの発動を代行する。

◆要求者のホストと異なるホストにおけるプロセスに対するソフトウェア・インタラプトを代行する。

3．2　マネージメント・プロセス

マネージメント・プロセスは通信その他の実際の処理の要求を受け、それを処理する。マネージメント・プロセスは一つのグループに対しアドレス・サーバ・プロセスあるいはマネージメント・プロセスにより実行され、そのグループの存在中はdaemonとして働き、グループの終了により消滅する。従ってマネージメント・プロセスの実行は動的であり、また、そのロケーションも動的である。マネージメント・プロセスは以下の機能を持つ。

◆要求に従いプロセスをグループのメンバとして登録し、また削除する。

◆メッセージを指定されたプロセス（複数）に転送する。

◆要求に従い下位のプロセス・グループを作成する。

◆グループのステイタス情報を保持し、要求に従い通知する。

◆マネージメント・プロセス内のメモリを利用し、グループの共有データとする。

4．　通信と同期

通信方法としては、ポイント・トゥ・ポイント、マルチ・キャスト、ブロード・キャストをサポートしている。ブロード・キャストはグループのメンバすべてにメッセージを送信する。マルチ・キャストは、その"class"をプロセスがグループのメンバとして登録する時点で登録しておき、"class"を指定することにより実現する。また、通信において"key"の概念も導入した。受信プロセスはkeyを指定することにより、"key"が一致した場合のみメッセージを受けとる。"key"は順序制御あるいはオーセンティフィケーションに利用することができる。

同期は基本的に、必要なメッセージが到着するまで待機する同期的受信により実現される。送信はマルチ・キャストをサポートしているため非同期で行なわれる。また、非同期な受信あるいは共有データを用い、ビジーウェイティングにより同期を取ることも可能である。

5．　プログラム例

[Figure 2]に本システムを用いた、典型的なサーバ・クライアントのプログラム例を示す。ネットワーク・プロセス・グループに関するすべての要求は用意されたライブラリ・ルーチンを通じて行なわれる。

また、本システムを利用したアプリケーションとして、インター・ネットワークにおいて端末を通じ、３名以上で会話を行なうことのできるプログラムを試作した。

6．　結論

分散環境におけるＩＰＣ支援システムをUNIX4.2BSD上に試作し、テストを行なった。本システムを利用することにより、ユーザはネットワークにおいてメッセージ通信、リソースの共有、同期を行なうプログラムを用意されたライブラリ・ルーチンを用いて容易に作成することが可能である。現在のところ通信のスループット及びセキュリティに問題があるが、ネットワーク・アプリケーションのツールとして本システムは有効である。

```
/*
 * Test program : server
 */
#include "ipcslib/h"
main()
{
  IPCS *entry;
  char data[100];

  entry=begin("server","/gr1/gr2/gr3",0);
  strcpy(data,"This is test data¥n");
  sendtoclass(entry,"client","key",data,
    strlen(data));
  finish(entry);
}
----
/*
 * Test program : client
 */
#include "ipcslib.h"
main()
{
  IPCS *entry;
  HMEMBER fromprs;
  char data[100];
  int len;

  len=sizeof(data);
  entry=begin("client","/gr1/gr2/gr3",0);
  printf("Waiting data.¥n");
  waitrecv(entry,&fromprs,"key",data,&len);
  printf("from %s¥n",fromprs.name);
  printf("%s",data);
  finish(entry);
}
```

Figure 2. Example Programs

7．　参考文献

[1] Samuel J. Leffler, Robert S. Fabry, and William N. Joy, "A 4.2BSD Interprocess Communication Primer," Unix Programmer's Manual, CSRG University of California, Berkeley, July 1983.

[2] "Courier: The Remote Procedure Call Protocol," Xerox System Integration Standard, Xerox Corporation, December 1981.

[3] B. Lyon, "Sun Remote Procedure Call Specification," Technical Report, Sun Microsystems, Inc., 1984.