4G-4

分散システム記述言語ＤＩＰＲＯＬにおけるプロセス間通信の実装法

赤田　正雄　　田中　英彦　　元岡　達

（東京大学　工学部）

１．はじめに

ＤＩＰＲＯＬ［１］は、メッセージ通信に基づく分散システム用の並行処理記述言語である。ＤＩＰＲＯＬの言語仕様は、ＮＯＳ（network operating system）記述用のＤＩＰＲＯＬ／Ｓと、ＮＯＳを環境として動作するＤＩＰＲＯＬ／Ｕの２階層になっている。ＤＩＰＲＯＬ／Ｓのコンパイラは、仮想スタック・マシン　ＤＩＰＲＯＬ　ＶＩＲＴＵＡＬ　ＭＡＣＨＩＮＥ（ＤＶＭ）のアセンブリ言語を出力する。

本論文は、ＤＩＰＲＯＬにおけるプロセス間通信のインタフェースであるポートと、ＤＶＭにおけるポートの実装に関する報告を行なう。

２．メッセージ通信のインタフェース　＜ポート＞

１）ＤＩＰＲＯＬ／Ｓのポート

プロセスは、外界とのインタフェースとして、

①　ポート　port

②　疑似ポート　pseudo port

を持つことができる。

①のポートは、他のプロセスとのメッセージ通信に使われ、相手ポートがローカルかリモートかはプロセスにとってtransparent である。

②の疑似ポートは、ローカルＩＯデバイスとのインタフェースである。ＤＩＰＲＯＬ／Ｓでは、この疑似ポートに対して、ＩＯデバイスを起動するＩＯ文と、その終了割込あるいはアテンション割込を待つＩＮＴＥＲＲＵＰＴ文を設けている。ＮＯＳの機能としてのＩＯデバイスの管理、すなわち分散環境における monitor　の役割と、入出力サービスの提供は、この疑似ポートをもったＩＯ管理プロセスがおこなう。

２）ＤＩＰＲＯＬ／Ｕのポート

ＤＩＰＲＯＬ／Ｕのプロセスは、①のポートを介してのみ外界とinteraction をもつ。

以下①のポートに関する説明を行なう。

３．ポートの生成・消滅

ポート制御ブロック（port control block　ＰＯＲＴＣＢ）がオブジェクト・コードのアセンブル時に設けられ、プロセス・ポート管理プロセスがcallするkernel function ＣＲＥＡＴＥ、ＤＥＬＥＴＥによってＰＯＲＴＣＢがプロセスにlink、unlinkされる。

４．ポートの接続

ポートがどのポートと通信ができるか、つまり、ポートの接続（connection）は、ＰＯＲＴＣＢの中に設けられたアクセス権テーブル（access right table　ＡＲＴ）に相手を登録することによっておこなわれる。この時、

ポート・ネーム：プログラマによってのソース・プログラム内で宣言され、参照されるもの

ポートＩＤ：ホスト番号、プロセス番号、ポート番号から構成され、ローディング時に設定されるもの

の二つの内の、ポートＩＤが用いられる。

Ａ）ポートのconnectionの指示

ポート・ネームによる接続の指示には、次の３レベルの形態がある。

①　source level connection

ＤＩＰＲＯＬのソース・プログラム内のＣＯＮＮＥＣＴ文による指定

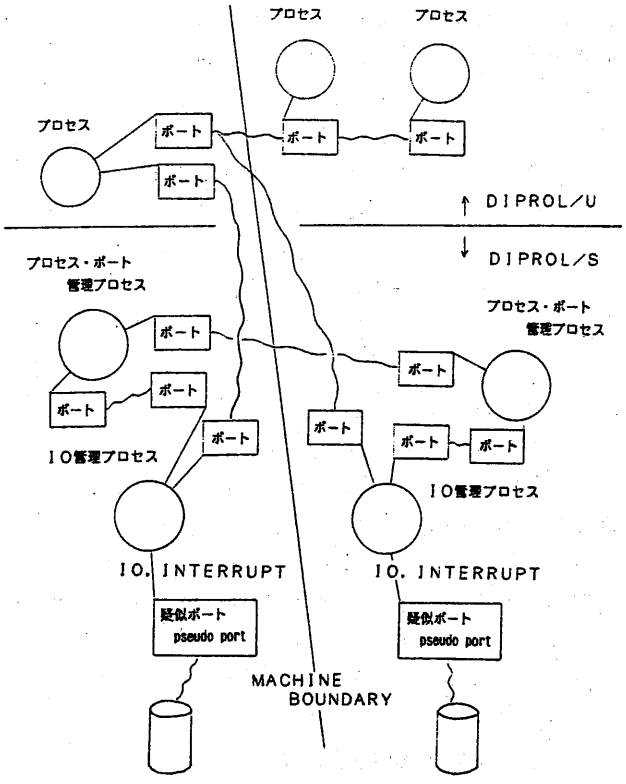

fig 1.　ＤＩＰＲＯＬのプロセスとポート

② port linker level connection
ポート・リンカで指定するもの
③ dynamic connection
プロセス・ポート管理プロセス（process port management process PPMP）に対して接続要求のメッセージを送信。

B）相手ポートIDのARTへの登録

ポートIDのARTへ登録は、次の2レベルで行なわれる。

① kernel registration
各ホスト上のPPMP間のconnectionの指示（A－①，②）に対しては、ポート・リンカを通す時点でARTにポートIDが登録される。つまり、ポート・リンカ、あるいはそのユーザがポート・ネーム→ポートIDのmapping に責任を持つ。

② normal registration
PPMP以外のプロセス間のポートのconnectionは、A－①，②に対してはプロセスのローデング時に、A－③に対してはプロセス走行時にdynamic に、PPMPが kernel function CONNECTをcallして行なわれる。従って、ポート・ネーム→ポートIDのmapping は、PPMPが責任を持つ。

5．ポートの実行

DVMのサポートしているプロセス間通信用のコマンド（ポートを起動するコマンド）には、PCL PSTの二つがある。ポートの属性・型（入出力指定 talk ／listen，能動性 active ／passive ，接続可能相手ポートの数 fanout ；…）は、ソース・プログラム内のポートの宣言で決定される。

DIPROLのポートには、①fanout＞1のときにどの相手ポートと通信がおこるか、②複数のポートに対してプロセスが事象待ちになる（NDCコマンド）、という2レベルの非決定性をもたせることができる。

fig ．2にポートの実行例を示す。

6．ポートの実装

現在、panafacom U300上にDVMの実装を行なっている。fig.3に示すのが、DVMにおけるポート実行部の構成である。図中のIF DTは、determinate にポートが起動された時の処理の流れを表わす。interpreter およびcupidintからポート・コマンドが発行され、illegal sequenceに対しては、例外が生じる。DIPROLでは、プログラマによる例外処理ルーチンのoverrideが可能である。

メッセージ転送の機構は、専用プロセッサCUP IDを用いている。また、プロセスとポートの同期はプロセス管理モジュール内の手続（RESUME，etc ．）のcallで行なっている。

ポート実行部はU300のアセンブラで書かれており、コードの大きさは2KB弱（PORTCBを除く）である。

＜参考文献＞

［1］ 赤田、他．「分散システム記述言語DIPROLの支援環境」、情処25全大、6G－6．

```
PROCESS SEND:                    PROCESS RECEIVE:
VAR MESS:CHAR:                   VAR BUFF:CHAR:
PASSIVE PORTA[3]                 ACTIVE PORTB>>CHAR:
            <<CHAR:
BEGIN                            BEGIN
.....                            .....
PORTA<<MESS:                     START PORTB>>CHAR:
.....                            .....
END:                             END:
```

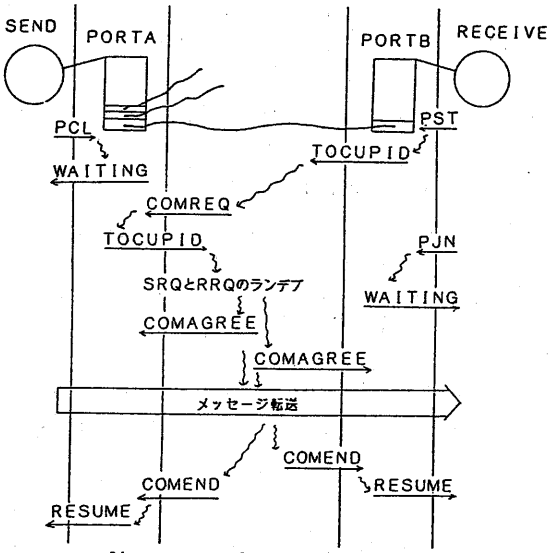

fig 2． ポートの実行の例

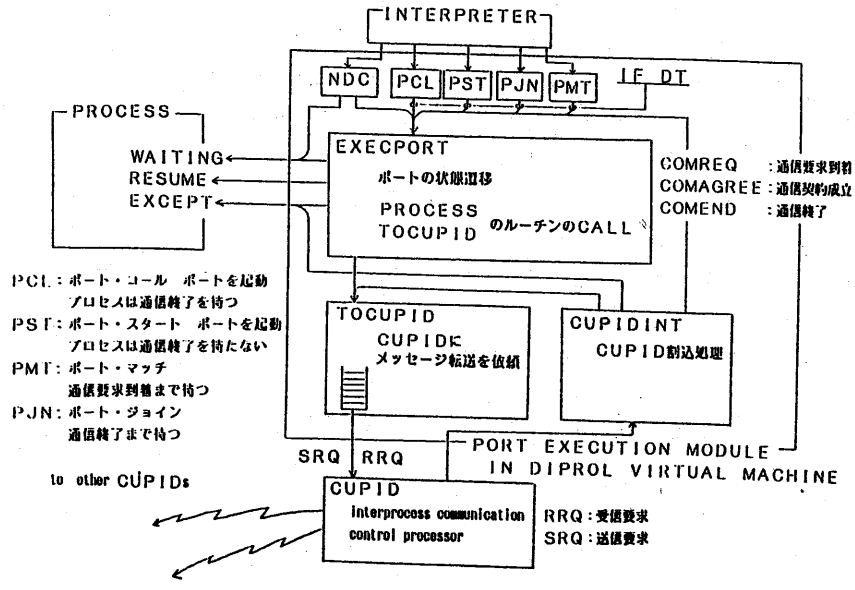

fig 3． DVMにおけるポート実行部