4D-10　分散処理用記述言語 DIPROL の検討と実装

小森 斉　田中英彦　元岡 達
(東京大学 工学部)

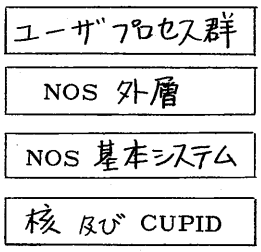
1. はじめに

DIPROLは、主記憶を共有しないような、複数のマシンから構成される分散処理システムのための記述言語で、前回の大会では、設計方針の概要を報告した。[1] 本稿では、その後の変更点や、特徴などについて述べる。

2. DIPROLの目的

DIPROL は、Pascal をもとにして、メッセージ通信型のプロセス間通信機構を中心に拡張，変更を行なった言語である。この言語は主にプロセス群を記述し、シンプルな核と、ハードウェア化されたプロセス間通信機構 CUPID [2][3][4] を土台として、NOS (Network Oriented - Operating System) を記述することを主目的としている。

DIPROL を使ってのNOSの記述構成は、右図のようになる。



3. プロセス間通信

CUPID が提供するプロセス間通信は、マシン間/マシン内の区別を意識しないもので、主に SRQ,RRQ という計算機間コマンドを用いて、メッセージごとに通信契約を行なう形態を持つ。

DIPROL の中では、Pascal の file 型の変数に対応する port というものを介して、プロセス間通信を行なう。port の宣言では、入出力の別，転送メッセージの型を指定する。

プロセス間通信用の標準手続きは、ポート名ほかの引数を持ち、通信が、終了するまで待つ種類のものと、通信の起動と終了を別の手続きに分けたものを用意している。後者により、おいてきぼりの通信をすることができる。

非決定性を含むメッセージ転送のための、条件付き複数受信待ちの構文[1]では、おいてきぼりにした port の終了待ちも含めることができるようにした。

port と port は、他方へのアクセス権を設定することにより、論理的に結ばれる。DIPROL には、システム記述用と、ユーザプロセス記述用の二つのレベルがある。前者の仕様では、ポートIDを引数としてアクセス権の付加/削除をマシン内で行なう手続き(核の機能の一つ)がある。アクセス権は通信先の port への記述子であり、これ無しには自分から通信を始めることはできない。

NOS の基本システム内のプロセス同士が、マシン間で通信する場合は、DIPROL で、あらかじめ宣言された型のメッセージ型を使う。このようなメッセージ通信については、CUPID が、毎回一致検査を行なう。NOS の基本システムは、その機能として、より上層で任意の型のメッセージをプロセス間通信に使うことを可能にする。すなわち、NOSの外層及びユーザのレベルでは、随意の型で(マシン間/マシン内を区別せずに)、port と port を結ぶことができる。

4. プログラムの構成と要素

システム記述用の DIPROL について、述べる。process の記述を並べたものが、program であり、コンパイルの単位となる。process の内の構文は、Pascal でいう program という単位とほとんど同じ形をしている。[1] DIPROL の program では、

これらのprocess間共通のデータ型や、定数の宣言に続き、processを複数記述できる。その後に、各プロセスのポートのアクセス権の初期設定があり、programが、その外に見せるport名と、内部のprocessのport名との結合が、最後に記述される。（右図）

NOSの基本システム（以下基本部という）は、マシンごとに作成する。コンパイル後のprogramを集め、各ポートをリンクして、基本部を作る。基本部より上のレベルを記述するときは、program内のprocessをどのマシンに置くかを指定できる。

programがその頭部で外に対して見せるように宣言したポートだけしか、外部からはアクセスできない。ユーザ用のコンパイラは、NOSがこのようにして外に見せたポートのうち一部だけをユーザに見せる。

processの記述の際、そこで使うポートを（手続きの記述の仮引数のように）宣言する。プロセス番号，ポート番号として、定数値を指定することもできる。これは、基本部のマシン間のプロセス通信の際には、網全体で統一された定数値を使うからである。

新しいデータ型として、アクセス権を扱うためにポートID型（portid），主記憶の管理のために、領域記述子型（memarea），プロセス管理のためにプロセスID型（processid）を設けた。これらには、それぞれ、特殊な標準手続きが用意される。

システム記述言語として要求される低水準の記述機能は、マシンごとに用意する登録ずみの手続きによる。

割り込み処理は、processとほとんど同じ構文で書き、その中では、割り込み待ちの手続きwaitioを通信用の手続きと同様に扱うことができる。

```
program ALPHA (xport,yport)
    const 宣言        外に見せるポート名
    type  〃
                  ポートの宣言
    process A( in u1: ～ ;out u2:～)
      -- プロセスの記述はPascalのプログラムの形式

    process B( out w3: ～ ; out wq: ～ )

    begin
          A( B.w3, C.zz ; C.xx ) ;   } アクセス権の初期化
          B(  ; C.xx ) ;
          ALPHA( B.wq , D.aa ) ;
                  ↑ xport, yportとの対応づけ
    end
```

5. 実装その他

ユーザ用のDIPROLではNOSの機能が使えるので、

- プロセスを動的に生成する（ファイル名やパラメータを指定して）ことができる。
- （ユーザに権限があれば）プロセスをどのマシンに置くかは、program内で指定できる。

NOSが基本部の下に仮定する核の機能は、プロセスのスケジューラ，ディスパッチャ、割り込みの初期段階と最終段階、主記憶の原始的な管理などである。

現在のところ実装は、Pascal-Pに手を加えて、カーネル/インタプリタ系でこの言語を実現する。対象マシンは、ミニコン二種と中型計算機一台の計三台に、CUPIDを装備したものを目標にしている。

[1] 大和、田中、元岡「分散処理システム記述言語の設計とそのカーネルの製作」第22回全国大会 pp557～558

[2] 和智井、田中、元岡「網向きプロセス間通信制御プロセッサCUPIDのソフトウェア」 同上 pp545～546

[3] 和田、小森、田中、元岡「網向きプロセス間通信制御プロセッサCUPIDのハードウェア」 同上 pp547～548

[4] 和智井、田中、元岡「網向きプロセス間通信制御プロセッサCUPIDの実装と評価」 今大会予稿集 5D-8