7T-8

通信系のシミュレーション簡易言語

井出政司　金田洋子　Terrence W. Holm　松下温
沖電気工業株式会社

1. はじめに

コンピュータネットワークの通信形態は通常の回線を利用した1対1のものの他に、LAN・衛星通信等に見られるような1対Nのものがある。近来、LAN等の発達によりこの1対Nの形態は数多く見られるようになった。この形態に対して通信方式を決定する上でシミュレーションは欠かせないものの1つである。周波数の分割や衝突発生時の動作等をきめこまかく明確にし、性能的・経済的により良いものを見出す手段となっている。同報機能をもった通信系の開発を行う時、そのシミュレーションにおいて、シミュレートされるモデルのプロセス数は大きなものとなり、より簡易かつ安価なシミュレーション方法が望まれる。我々は、このような通信系のシミュレーションを行うための簡易言語（OPUS）を開発し、その処理系をインプリメントした。本報告では、この言語の構造及び特徴について述べる。

2. シミュレーション簡易言語OPUS

OPUSは、通信系モデルのシミュレーションを目的とした簡易言語であり、その構造は、The Execution Capability Model（ECM）[HOLM85] に由来する。ECMは、プロセス通信をモデル化する方法であり、このサブセットをサポートするための簡単な仮想マシンの構築の研究がなされた。そして、ECMは、6種のコマンドのみを用いたactive objectsですべての計算がいかに実行されうるかを示した。この研究から、"procedures"と"processes" を1つの構文構造－"routine" －に一般化した。その解釈の違いは、routineがfunction call で呼ばれる(procedure) か、createで参照される(process)か、による。継続子については、2種(thenとelse)に簡略化した。OPUSは、この仮想マシンの中間コードにいくつかの構造を付加することにより得られた。

3. OPUSの構造

OPUSは、通信系のシミュレーション用としてできるだけ簡易であることをめざした。OPUSプログラムは、他の言語におけるサブルーチン、ファンクション、プログラム、タスクの概念を1つにまとめた単純な" routine" のみから構成される。

ステートメントは、次の3つより成る。

- routineコール(thenとelse継続子を持つ)
- true returnとfalse return
- loopとloopからのbreak

簡単な例を示す。

```
routine  Even(x)
    begin
        if eq(and(x,1),0) then
              return( );
        else
              return_false( );
              end_if
    end routine
```

4. 内部ルーチン及びライブラリルーチン

ビルトインの内部ルーチンがあり、それらは

- 算術演算
- 論理演算
- 比較演算
- ストレージ参照
- プロセス生成
- プロセス間通信
- キャラクタ入出力

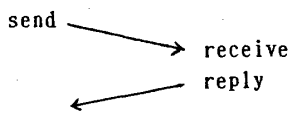
に対するものである。

プロセス生成ルーチンにより、容易にいくつものプロセスを発生させることができる。その形式は、

```
id:=create(routine,priority,name,
          privileges,environment,arguments)
```

routine：プロセスとして生成される手続き名
priority：スケジューリング用
name：ステータス表示用
privileges：送信とプロセス失敗時の消去保護
environment：nameをプロセス識別子にマッピングする共通プロセス
arguments：routineが使用するパラメータ
id：生成されるプロセスの識別子

プロセス間通信用に次の3種がある。




```
result,...:=send(who,label,values,...)
who,label,values,...
        :=receive(set_of_receive_labels)
reply(who,results,...)
```

who：プロセス識別子

A Small Language for Communicating Objects and a Simulaion Application
Masashi IDE, Yoko KINDA, Terrence W. HOLM, Yutaka MATSUSHITA
OKI Electric Industry Co.,Ltd.

label：ポート名
set_of_receive_labels：受信用ポート名
values,results：受け渡すデータのリスト

ライブラリルーチンとして次のものが用意されている。

・プロセスステータス表示
・ネームハンドラ
・ストリングルーチン
・入出力フォーマッティング
・伝送チャネル
・計測用ルーチン

伝送チャネルは、共通伝送媒体をシミュレートするためのものである。いったんプロセスとして生成された("channel:=create(Tx_Channel)")ならば、他のプロセスは、伝送を待っている媒体上でブロックされる。

```
message:=send(channel,LISTEN);
```

通信は、次のようにプログラムされる。

```
send(channel,TRANSMIT,message,time) then
        -- message was sent
else
        -- collision occured
        end if
```

計測用のルーチンとして次のものがある。

クロック:delay(x) - x単位時間待つ
time( ) - 現在の時刻を得る
乱数:random( ) - 一様乱数
Exp_dist( ) - 指数分布乱数
統計：通信時間
〃 の平均値
〃 の分散

例として、次の３つを示す。

allocate(v) - 統計用ストレージの生成
stat_time(v) - タイムインタバル事象の記録
stat_output(v) - 平均、分散のプリント

プロセス指向言語が、シミュレーションに使われるうるようにする有効なprocedureとして、

- シミュレーションタイムクロックルーチン
- 乱数発生器
- 統計用パッケージ

があり[SYDM85]、ＯＰＵＳはこれを備えている。この例となるインプリメンテーションが[FREL85]によりおこなわれた。

５．ＯＰＵＳの他の特徴

データタイプは、３２ビット整数型のみを持つ。
定数を使う各種の方法として、
数値演算（３＊Tx_Time）
文字（' Ａ'）
ストリング（" output＼m＼j"）
アスキー制御文字用のキャラクタエスケープ
がある。
単純なアレイ構造（実行時にダイナミックにリストを作る。）を持ち、そのリストの項目としては８，１６，３２ビット値としてアクセスされる。例として、

```
w:=list(87,203);
x:=get32(w,2);
```

プロセスと通信プリミティブはエラー発生時にエラーメッセージストリングをかえす。これは、デバッグの為にプリントされるることができる。

６．処理系について

ＯＰＵＳプログラムは、コンパイラによりＯＰＵＳ中間コードに翻訳され、インタプリタによって実行される。これらコンパイラ、インタプリタは、ＶＡＸ１１／７８５上にインプリメントされた。コンパイラとインタプリタは５５００行（ＬＥＸ，ＹＡＣＣ及びＣ）からなり、ライブラリは４０００行（ＯＰＵＳ）のサイズで、約４２％が実行コードである。ＯＰＵＳ実行時間を以下に示す。

１ＯＰＵＳコード　４０μｓ
routine call と return　２２０μｓ
send( )/receive( )/reply( )　１ｍｓ
create( )　１１ｍｓ

[TYNM86]では、8MHzの68000 プロセッサ上でのプロセス間通信send/receive/replyが１ｍｓと報告されている。

７．まとめ

以上をとりまとめると、
ープロセス指向シミュレーションは有用である
ー通信プリミティブの処理時間が簡単な算術演算の時間とくらべて多くを占めるのでsend処理時間を小さくすることは非常に有効である
ー最小の言語で十分である
今後このＯＰＵＳを使用してシミュレーションを続け、その有効性を確かめていきたい。

参考文献

[FREL85]P.Friel and S.Sheppard,
"Implication of the ADA Environment for Simulation Studies",
Simuletter,Vol.16(2),April 1985,pp.14-26.
[HOLM85]T.W.Holm,
"The Execution Capability Model",
MMath. thesis,University of Waterloo,1985.
[SYDM85]T.Saysam,
"Process-Oriented Simulation Languages",
Simuletter,Vol.16(2),April 1985,pp.8-13.
[TYNM86]T.Tuynman and L.O.Hertzberger,
"A Distributed Real-Time Operating System",
Software-Practice and Experience,Vol.16(5),
May 1986,pp.425-441.