網向きプロセス間通信制御プロセッサCUPIDの性能測定

和賀井　フミ子　　田中　英彦　　元岡　達
（東京大学　工学部）

第１章　はじめに

網向きプロセス間通信制御プロセッサCUPIDは、網向きＯＳの処理機能の向上を目指して、開発を行なった専用サブシステムである。

このシステムの背景には、以前開発を行なった研究用計算機網TECNETとその上のOS、NOS第I版がある。[1)~3)]

これらの使用経験から、網向きOSの核機能に注目し、効率向上の為にこのレベルで分散させることを提案した。[4)]

即ち、通信回線によって結合される計算機網を前提とした網向きＯＳの核機能について、考察・検討し、一部の機能を主計算機から分離することにより処理効率を上げることを試みた。

CUPIDは、核機能の中のプロセス間通信機能を、主計算機から分離し、専用ハードウェアを用いて試作したシステムである。その性能測定を行なったので、結果を述べる。

第２章　CUPIDの背景

２－１　網向きＯＳの基本的核機能と分割

網向きＯＳの基本的な機能としては
1)　プロセス間通信・同期
2)　マルチプログラミング
3)　プロセスの生成・消滅
4)　ケーパビリティ・チェック
等、があると考えられる。

NOS第I版の使用経験に基き、これらの機能分離をOSの処理効率の面から検討した。その結果、一部の機能については、主計算機内での処理が、他機能の処理に対してボトルネックになるものや、必ずしも主計算機内で処理する必要の無い機能があると考えられるので、この様な機能については、主計算機から分離することを考えた。

分離を行なう場合、必要な要素としては、
1)　主計算機内のＯＳの負担軽減に大いに貢献するもの
2)　機能を専用マシンとして取り出すことにより、主計算機とのインタフェースが明確になり、又、並列化を促すもの
3)　機能が、主計算機の機械語によって左右されないもの。

このような条件を満たす一例としてプロセス間通信機構をサブシステム化した。このことにより通信管理処理をプロセスの処理の流れから独立させることができたこと、通信に用いる膨大なバッファ空間が節約できること、等の利点が生ずる。又、同一構造にできるので開発の手間が省ける、と考えられる。

２－２　プロセス間通信の特徴と方式

ここで考えるプロセス間通信機構はNOS第I版の長所を受け継ぎ、改良を行なったものである。

主な特徴としては、
1)　プロセス間通信は、通信するプロセスの存在する場所に依らず、同一手法で行なえる。（transparency）

即ち、通信を行ないたいプロセスは相手プロセスを論理名で指定し、SEND RECEIVEなどの通信マクロを用いる。

２つのプロセスが通信マクロを発行し、合意が成立すると、メッセージ転送による通信が行なわれる。

2)　プロセスの指定する通信マクロの引数は、転送メッセージや相手プロセスの属性を論理名で記述する。プロセス自身の属性は、指定する必要はない。

3) メッセージは、必ず写しによって転送を行なう。通信回線を使用する場合は、パケットに分割する。

プロセス間通信処理機構では、以上の様な機能を実現するために、以下の手順で処理を行なう。

1) 通信マクロの引数をシステム内部表現に変換する。提出プロセスを同定し、通信の可否も調べる。

2) 相手のプロセスに通信要求を発行する。相手が同一計算機内にあれば、直接、無ければ、網通信コマンドを用いて回線経由で知らせる。

3) 相手プロセスからの通信要求が到着した時点で、通信契約が成立する。

この後、メッセージ転送が行なわれる。同一計算機内での転送は、一気に全部コピーする。他計算機への転送はメッセージをパケットに分割し、少しずつ送る。転送が終了すると、前者の場合は直接、後者の場合は網通信コマンドを用いて、終了通知をする。

2－3　プロセス間通信機構の分割

プロセス間通信機構は前節で見た様に、機能的に階層構造を成している。具体的には、ユーザレベル、通信マクロ処理レベル、通信管理レベル、回線制御レベル、の層である。

このうち、'純粋'に通信処理を行なう通信管理レベルと回線制御レベルとを主計算機のOSから、分離した。これに対応して、ハードウェアもCCP(Communication Control Processor)とLCP(Line Control Processor)との2層に分け、ソフトウェアの機能の纏りに対応した構成とした。

CCPは、主計算機毎に一台、LCPは回線毎に一台としたので、CUPIDの構成はCCP一台，LCP複数台，となる。

分割することによってインタフェスが生ずるが、機能単位で分けたので、必要最小限にまとめることができた。

主計算機とCCPとの間は、通信マクロ処理レベルと通信管理レベルのインタフェスそのものであり、通信マクロが発行した通信要求や、通信終了通知等を受け渡しする。

CCPとLCPとの間は、通信管理レベルで作成した網通信コマンドの受け渡しをする。

階層間のプロトコルとしては、プロセスどうしの通信マクロ、CCP間での網通信コマンド、LCP間には、HDLCによる伝送手順.がある。

メッセージの転送は、必ずコピーを行なう。機能を分割した為、実現法には工夫が必要になる。

転送先が同一計算機内の場合は、転送経路として考えられる方法は、主計算機内で直接コピーする方式や、一度主計算機から取り出してCCPヌはLCPを経由させる方法等があるが、処理効率等を比較して、主計算機からCCPに取り出し、再び主計算機に戻す方式を採用した。

転送先が異なる計算機の場合は、主計算機からCCPに取り込み、ここでパケット化してLCPに渡す方法や、主計算機からCCPに取り込み、更にLCPに渡してここでパケット化する方法が考えられるが、いずれもCCPの処理負荷を増し効率的でない。そこで、主計算機から直接LCPに取り込み、LCPでパケット化することにした。又、LCPでパケット化する為の契機は、CCPからの情報によって行なう方式にした。

表1

	I/Oポート制御語	転送内容	転送手段
HOST→CCP	IPL INIT		
	INFM	HOSTPARAM	DMA
CCP→HOST		CCP PARAM	PMM
CCP→LCP	LIPL INIT		
LCP→CCP	COM	LCP PARAM	SM

2－4

CUPIDのハードウェア

分割を効果的に実現する為のハードウェアとしては、

1) 論理階層との対応がよい
2) 転送ネックの回避
3) モジュール構成。及び、それらの間の接続が簡単になること
4) 回線規模は、全2重80kbpsで8組までサポートできること
5) 開発に要する期間や価格が低く済むこと

等を考慮してマルチ・マイクロプロセッサ構成を採った。

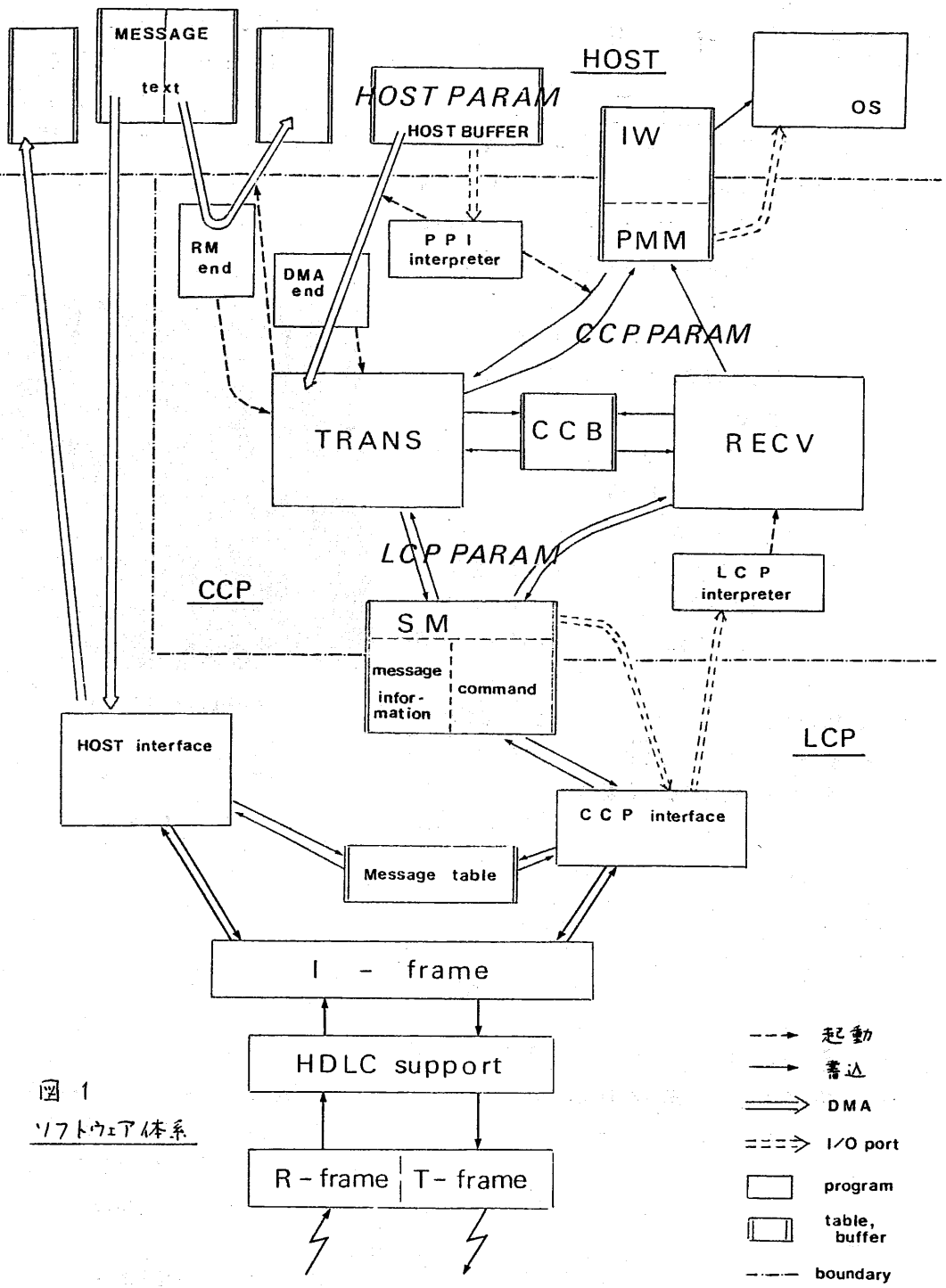


図1 ソフトウェア体系

インタフェス用のハードウエアとしては転送のオーバヘッドをできるだけ小さくする方式として、主計算機とCCPの間の制御用にI/Oポート LSI を、内容転送のためにDMAと擬似メモリ(PMM)を設けた。

CCPとLCPとの間には、制御用のI/Oポートと、柔軟性を考慮して、双方向から読み書き可能な共有メモリ(SM)を設けた。(表1)

一方、メッセージ転送用としては、反射機構(RM)と主計算機とLCP間にDMA機構を開発した。

RMは、主計算機からCCPに、データを1Byteずつ DMAで取り込み、主計算機内の宛先にDMAで書き込む。この為、RM転送中は、主計算機, CCPとも他の処理を実行できる。

第3章　実装

3－1　ソフトウェア

CCPのプログラムは、通信管理プログラム、割込処理プログラム、制御用のテーブルやバッファから成り、論理構造が明白になるように高級言語PL/Mで記述した。プログラムの大きさは、PL/Mで約3800行、コンパイルすると約20kB弱である。初期化プログラムの一部はROMに載せた。

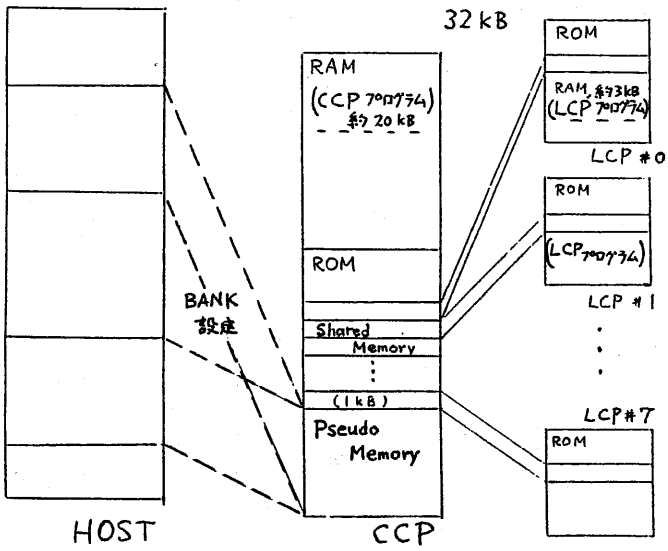
LCPのプログラムは、CCPや主計算機とのインタフェス処理、I-フレーム管理HDLCサポート、及びフレームの送信、受信プログラムから成り、処理速度を考慮して、(Z80の)アセンブリ言語で記述した。ステップ数で約2100行程の大きさである。

主な制御ブロックとしては、CCPに通信制御テーブル(CCB)、LCPにメッセージ管理テーブルがある。　(図1)

CCPとLCPのメモリ・マップ、及びPMMSMとの関係は、下図の様である。

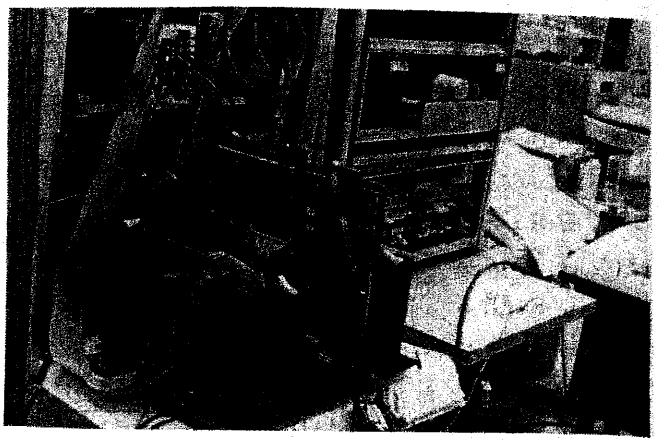
3－2　ハードウェア

現在までのところ、2組のCUPIDが実装されている。各々の主計算機は、OKITAC4300CとFACOM U-300であり、異なる建物にある。両者の間は、専用回線で接続され、回線距離は、実測によると、1.2kmであった。



CUPIDは、HOST固有モジュール、CCPモジュール、LCPモジュールの3部分と、これらを接続する2組のBusから構成され、2台とも構造が同一である。

主計算機とは、各々に適した固有インタフェスを用いて、差異を吸収している。



CUPIDとOKITAC 4300C

第4章　CUPIDの性能測定

4－1　測定の対象

前章までに述べた、プロセス間通信機構の処理時間を測定する。

対象としては、通信する両プロセスが同一計算機内にあるローカル通信と2つのプロセスが異なる計算機間にあるリモート通信について、最も基本的な場合を考える。即ち、一方のプロセスが通信マクロSENDを出し、他方がそれに応えてRECEIVEを出して正常にメッセージが転送された場合の時間を測った。

4－2　測定の方法

測定には、CCP内蔵のハードウェアタイマを利用し、カウンタとして使用した。

或るカウント値に達すると割込が生じ、その時の処理状況を調べて測る方法もあるが、割込禁止区間などの問題がある為、この方法は採らず、CCPプログラムの処理単位の始めと終わりにカウンタの値を読み出すルーチンを

埋め込む方法で測定した。

カウントの間隔は、100nsecのオーダから測れるが、あまり細かすぎても意味が無いので、40μsec毎にした。

4－3　測定の結果

1) OKITAC上でのローカル通信では、メッセージを512 Bytes 転送すると、全通信時間（プロセスが通信要求を受けて、対応する通信マクロを作成する時間は含まない）は、35.30 msであった。この内訳は、

通信管理プログラム	20.98 ms
512B転送　RM時間	8.08 ms
主計算機からDMAによる56Bの引数転送	0.92 ms
インタフェス用割込処理	5.32 ms

転送メッセージ長を変化させRM時間を測定すると、1 Byte当り15μsとなり、ほぼ比例していることがわかる。（グラフ1）　(RM時間)＝(15μs/B)×(転送長B)

2) リモート通信の場合は、

通信管理プログラム	SEND	22.30 ms
	RECEIVE	20.94 ms
主計算機からDMAによる28Bの引数転送		0.46 ms
インタフェス用割込処理		2.98 ms

メッセージ送信側でのLCPのメッセージパケット化処理を、転送量を変えて（LCPで）測定すると、処理時間は、パケット分割の個数に影響を受けることがわかった。（グラフ2）

3) 次に、主計算機のプロセスが次々に通信要求を発行した場合について見ると、通信管理プログラムの処理時間は、1次函数的に増えている。（グラフ3）

4) 回線上は、1 Byte当り104μsかかっていた。

第5章　考察

5－1　測定結果について

機能をハードウェアで分離すると、この間でのインタフェスのオーバヘッドが問題になることが多い。インタフェス処理の内訳を見ると、ローカル通信の場合は、主計算機内の2つのプロセスからの引数転送（1.88ms×2）とRM転送の後処理(1.36ms)があり、リモート通信の場合は、主計算機内のプロセスからの引数転送（1.88ms）とLCPとの引数処理（1.00ms）等がある。

CCPでの通信管理処理に対するインタフェスの処理時間の比は、

ローカル通信　$\frac{6.24}{20.98+6.24}=0.229$

リモート通信　送　$\frac{3.44}{22.30+3.44}=0.133$

リモート通信　受　$\frac{3.44}{20.94+3.44}=0.141$

となり、ローカル通信で23%，リモート通信で14%程度であった。

主計算機とCCP間のデータ転送でDMAを用いると、1B当り16μsであり、転送に必要な全時間は、

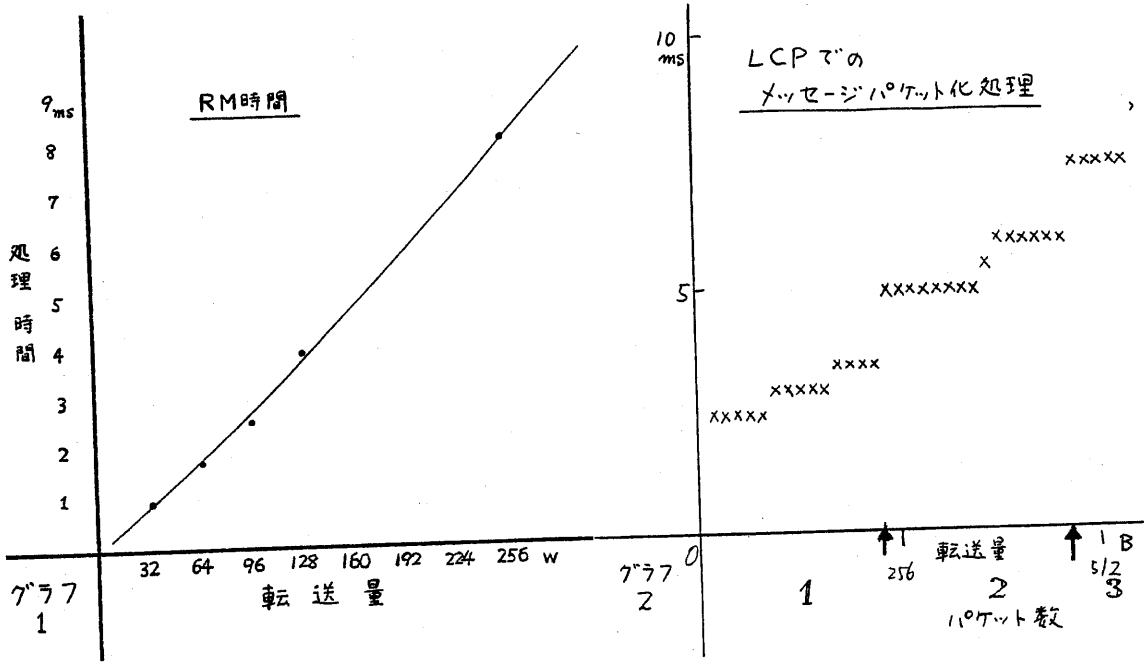
(16μs/B)×(転送数B)＋(DMA開始72.75μs)
＋(DMA終了割込処理 660μs)

擬似メモリを用いると、1B当り15.35μsであり、全転送時間は

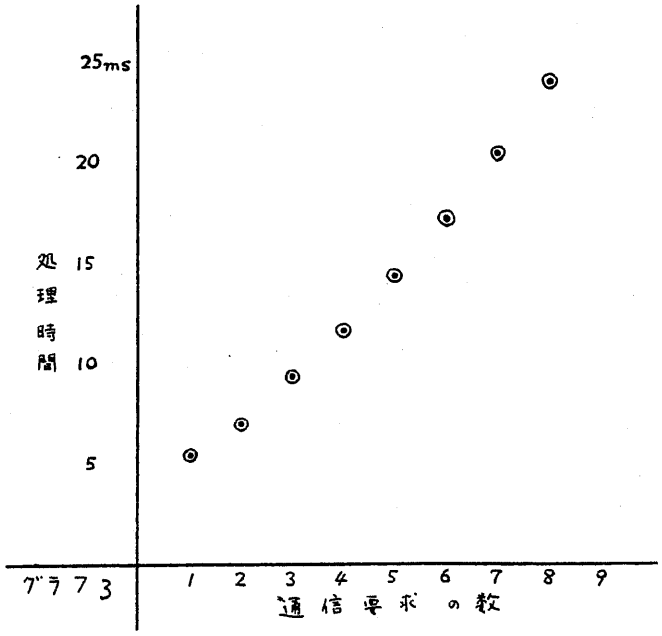
(5.35μs/B)×(転送数B)＋(PMM開始 4.5μs)

となる。DMAは転送中に他の処理ができるが、割込処理が必要なこと、擬似メモリは、転送のみしかできないが、割込処理が不要なこと、や上記の式とを考え合わせると、少量転送には擬似メモリ、大量転送にはDMAが向いている、と考えられる。

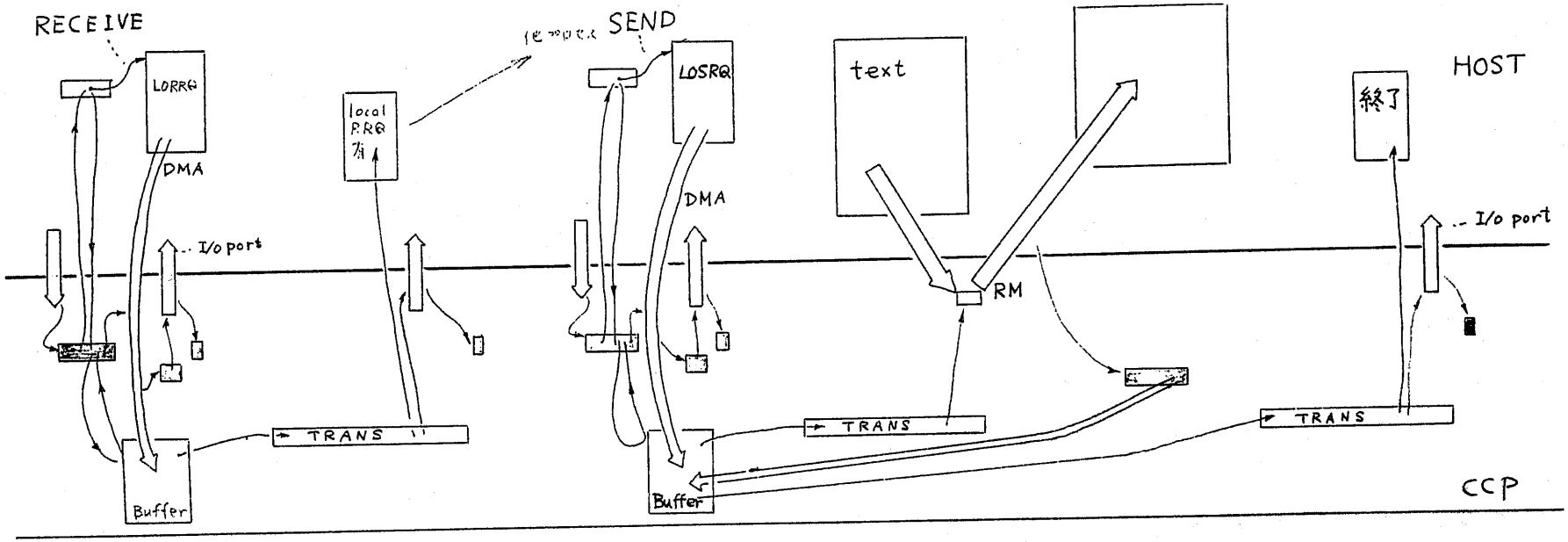
メッセージ転送は、ローカル通信の場合、RM機構を採用している。これを直接主計算機内で転送した場合と比較する。純粋な転送時間だけで比べると、OKITAC上では2B当り12.6μsでありRM機構は30μsと遅くなる。が、直接転送の場合は、OKIでは直接メモリからメモリへの転送ができないこと、主計算機内での転送前後の処理が必要なこと等のためOSの負担が増す。RM機構は、主計算機内の処理も、CCPの処


RM時間
9ms
8
7
6
5
4
3
2
1
処理時間
32 64 96 128 160 192 224 256 w
転送量
グラフ1

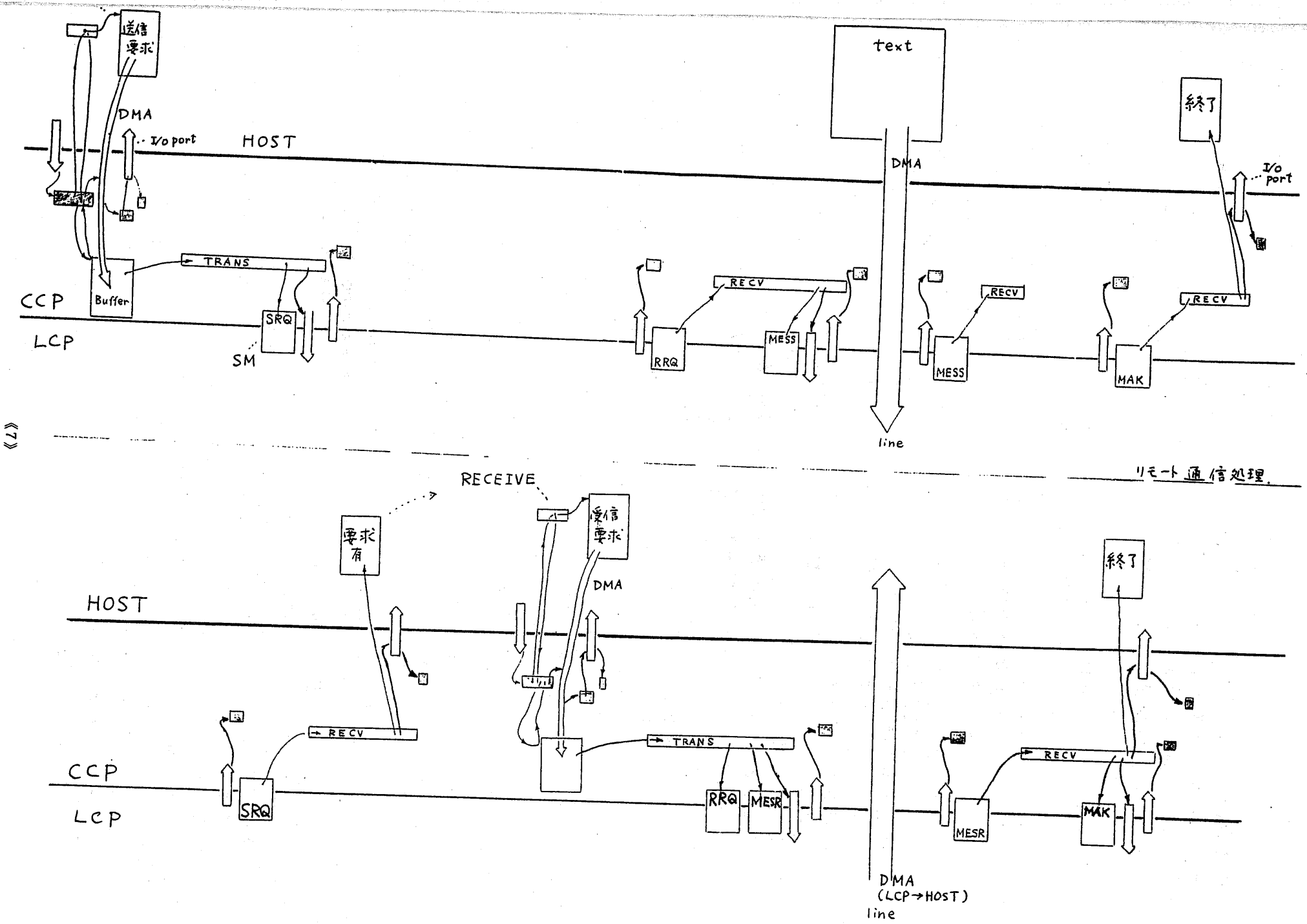
10ms
LCPでの
メッセージパケット化処理
5
0
256
転送量
1B
5/2
パケット数
グラフ2


25ms
20
15
10
5
処理時間
1 2 3 4 5 6 7 8 9
通信要求の数
グラフ3

《9》


RECEIVE
LORRQ
DMA
I/O port
local RRQ有
他プロセス
TRANS
Buffer
SEND
LOSRQ
DMA
text
RM
TRANS
Buffer
HOST
終了
I/O port
TRANS
CCP

ローカル通信処理


送信要求
DMA
I/O port
HOST
text
DMA
終了
I/O port
TRANS
Buffer
CCP
LCP
SRQ
SM
RRQ
RECV
MESS
RECV
RECV
MESS
MAK
line
RECEIVE
要求有
受信要求
DMA
HOST
RECV
CCP
LCP
SRQ
TRANS
RRQ
MESR
終了
RECV
MESR
MAK
DMA
(LCP→HOST)
line

《7》

リモート通信処理.

理も中断することなく転送ができるので、妥当な方法である、と言える。

リモート通信の場合、LCPで行なうパケット化処理時間は、主に分割するパケット数に依る。このことは、主計算機からLCPへのDMA転送は、処理ネックにはなっていないことを示している、と思われる。

LCPでのパケット受信処理は、直接測定できないが、送信処理と同様の傾向を示すと思われる。

5－2　分割に関して

特定機能をモジュール化して分割したことにより、インタフェースが明確になり、独立性が保てるようになった。主計算機内では、OSの負担が軽減し、効率向上に役立っている。

開発の面では、同一構造の構成なので、一組目が動くと二組目は容易に作ることができる。特に、ソフトウェアは殆んど変更することなく二台目に載せることができた。ハードウェアも、稼動している一組目を雛形として利用できるので、開発の手間が省け、デバグも容易であった。開発期間は、一台目のハードウェアは設計から実働まで2年余り、二台目は、一台目のデバグ中に作り始めて、9か月程であった。ソフトウェアはCCPプログラムでは、6か月人程度で、LCPのプログラムは、6か月人で書き上げ、少しずつデバグしている。

5－3　問題点と今後

今回、測定を行なったのは、プロセス間通信の最も基本的な場合であり、CUPIDや回線の負荷が問題にならない場合であった。CCPは、本当にLCPを8台までサポートできるのか、はまだ調べていない。そこでCCPやLCPの処理能力を調べることが、課題として残されている。調べ方としては、これまでに得られた測定結果を基にし、CUPIDでの処理の流れをモデル化してシミュレーションを行なうことを考えている。

核機能の分割の一例としてCUPIDを試作し、評価を行なってきたが、今後は、この結果を基にして、核機能の他の機能についても、分割を試みる予定である。具体的には、網向きOSの環境においては、必要であるが、従来あまり考慮が払われなかった保護機能について考察する。

この報告では、殆んど触れなかったが、ユーザレベルでは、分散処理向きOS記述言語DIPROLの開発が行なわれている。[5] DIPROLとの結合を急ぐ必要がある。

謝辞　CUPIDの性能測定の一部分は、当研究室の修士課程、服部泰明氏と赤田正雄氏にも分担していただいた。厚く感謝いたします。

参考

1) 田中・元岡：研究用電子計算機網TECNET
信学研資 EC 73－57

2) H. TANAKA and T. MOTO-OKA：
Distributed File Management and Job Management of Network-Oriented Operating System
Journal of Information Processing of JAPAN
Vol. 4　No 1　March 1981

3) 和賀井・田中・元岡：網向きオペレーティング・システムについての一考察
信学研資 EC 77－43

4) 和賀井・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の構成
情処・分散処理システム研究会 8-3 (1981)

5) 小森・田中・元岡：分散システム記述用言語DIPROL
情処・分散処理システム研究会 14-3 (1982)