計算機アーキテクチャ 119－6
（1996．8．27）

A-NETマルチコンピュータにおける適応型ルータの通信性能

廣田 守† 吉永 努† 馬場 敬信†

A-NETマルチコンピュータのルータは、トポロジ独立なメッセージ経路選択を行う適応型ルータである。そのA-NETマルチコンピュータにおいてメッセージ送受信を行うプログラムをファームウェアレベルで実行し、ルータの通信性能を評価した。実験は、メッセージ衝突がない場合とある場合の2つに分け、メッセージサイズ、転送距離、メッセージ転送間隔などを変えて実験した。その結果、メッセージ衝突がない場合の往復転送時間、マルチキャスト時間を得た。また、衝突が起こった場合、ある程度の遅延は認められたが、遅延時間が小さいことを確認した。この結果A-NETLのメッセージ送信においては、メッセージサイズや送信間隔が変化しても十分な性能を持つことが分かった。

Performance of Adaptive Router on the A-NET Multicomputer

Mamoru Hirota† , Tsutomu Yoshinaga† and Takanobu Baba†

The router of the A-NET multicomputer has been designed and developed to realize adaptive routing in a network-topology independent fashion. In order to evaluate the performance at the bare hardware level, we described some communication microprograms and executed them on the multicomputer. They were classified into two categories: ping-pong and message multicast with no collisions, and various patterns of message transmission with collisions. The latter transmission patterns include reverse flip, and uniform/random traffic. For each microprogram, we changed the message size, interval of message transmission, and distance between sender and recerver. Based on the results, we defined two equations to obtain ping-pong and multicast time on the A-NET multicomputer. We further confirmed that the communication latency, caused by the collisions, is quite small even for small interval time.

1. はじめに

近年MIMD型マルチコンピュータが注目を集めているが、その相互結合網の実現方法は2つに大きく分かれる。1つは、固定された結合網上でメッセージの経路選択に決定的アルゴリズムを用いる方法であり、2つめは、静的、あるいは動的にネットワークトポロジを可変とする方法である[1]。前者にはJ-Machine、AP-1000などがあり、後者にはPOOMA[2]などがある。

このような背景から我々は、並列オブジェクト指向トータルアーキテクチャA-NETプロジェクト[3]においてネットワークトポロジ独立なルータをもったマルチコンピュータを開発した。

このネットワークトポロジ独立という特徴により、種々の応用問題に適応すること、及び、オブジェクト割り付けが実行性能にどのように影響するかを調べられる。また、適応型ルーティングにより、局所的な混雑を回避し、空いている経路を有効に用いることで、結合網の性能を最大限に引き出せるようにも設計されている。現在、16ノードを実装し、ユーザアプリケーションプログラムを実行させ、動的評価が行えるようになっている。

ところで、実行評価を行う際、純粋なハードウェア性能を知っておくことが重要である。なぜなら、ハードウェア性能とソフトウェアオーバーヘッドを分離し、システムの性能を明らかにすることで、アプリケーションやランタイムシステムの挙動をより明確にすることができるからである。また、このことはシステム/アプリケーション双方の改善にも役立てられる。

本研究では、A-NETマルチコンピュータのルー

†宇都宮大学工学部情報工学科 Faculty of Engineering, Utsunomiya University

タについて、OS レイヤ等の上位層の影響を受けない、純粋なハードウェアレベルでの性能にできるだけ近い通信性能を評価することを目的とする。そのため、マイクロプログラムを用いたファームウェアレベルで、メッセージ送受信を行う最小の機能だけを実装して実験を行った。

以下本稿では、第 2 章にルータの構成とメッセージ送受信アルゴリズム、第 3 章にメッセージ衝突がない場合の性能評価、第 4 章にメッセージ衝突がある場合の性能評価、第 5 章にまとめと今後の課題を述べる。

2. ルータの構成とメッセージ送受信アルゴリズム

2.1 ルータのハードウェア構成

A-NET マルチコンピュータのノードプロセッサは、PE とルータで構成されている。図 1にルータと PE の一部を表す。

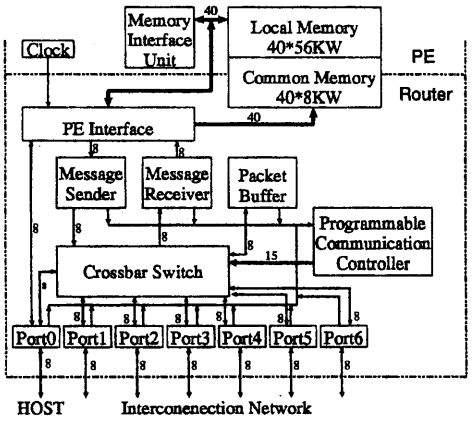


図 1: ルータのハードウェア構成

ルータは、ホスト用ポートと隣接ルータ間との結合用の 6 個のポートを持ち、これらとメッセージセンダ/レシーバ、パケットバッファ(PB) をクロスバスイッチ (CS) により結合する。メッセージの経路選択と CS の設定は、プログラマブル通信制御装置 (PCC) が行う。PCC は、CS につながったハードウェアブロックからラウンドロビン方式でメッセージの経路選択を受けはけ、レシーバや送出可能なポートをプログラマブルなロジックで決定することにより、トポロジ独立な制御を達成している。また、出力候補ポートから利用可能なものを選択して経路を適応制御する。CS につながったハードウェアブロック間の接続は、他のブロック間でのデータ転送中にも可能であり、ルータ内で同時に複数メッセージを送受信できる。PB はバーチャルカットスルールーティングを行うために設けてある。メッセージの最短経路上の出力候補ブロックが全て使用中の時、そのメッセージは 1 回のラウンドロビンの間はその場で待たされるが、2 回目の順番が回った時にも転送できなければ PB に一時退避される [4]。

プロトタイプは 30MHz クロックで動作し、ルータ内のピーク転送速度は 2 クロック/バイト (15MB/s) である。

2.2 メッセージ送受信アルゴリズム

ルータ-ルータ間のパケット交換は、受信ポート内の FIFO を緩衝バッファとして、ハンドシェークを行わず非同期に送信側主導で行う。

また、ルータ-ホスト間のパケット交換は、ホスト側とのデータ転送速度の違いを吸収するためハンドシェークによりデータ転送を行う [5]。

2.2.1 メッセージ送信

PE は、ルータとの共有メモリの特定のアドレスに、送信するメッセージの最終アドレスを書き込むことにより割り込みをかけ指定されているアドレス間のメッセージを送信する。

他ノード宛のパケットをポートから送出する場合、ポートは双方向半二重リンクであるため隣接ルータからのパケット入力と競合する可能性がある。そこで、ポートは PCC からの確保を指示する信号を検知してから PCC に確保が成功したか失敗したかを通知する。

確保が成功したことは、隣接ノードの受信ポートも確保したことに相当する。後は、送信ポートが CS からのデータ有効信号を見張ってデータ入力を検知し、隣接ノードに転送する。

2.2.2 メッセージ受信

ルータはメッセージを受信すると割り込みが発生し PE に通知する。PE はルータと共にバスリクエストを行う。バスが確保できるとルータ-PE 間共有メモリに受信したメッセージを DMA 書き込みする。その後バスを解放し終了する。図 2のようなアルゴリズムで受信処理を行う。

図 2中の OTHER_ NODE_ MESSAGE、MESSAGE_ SIZE、ROUTER_ ACCESS_ FLAG はルータ-PE 間共有メモリのアドレスの名称である。なお、図 2はメッセージ受信を機能させるための最低限の処理であり本稿における実験もこれに沿って行われている。

3. メッセージ衝突がない場合の性能評価

ルータのハードウェアにより近い性能を得るために、各処理が機能するための最小限のマイクロプログラムを PE に実装し実験を行った結果について述

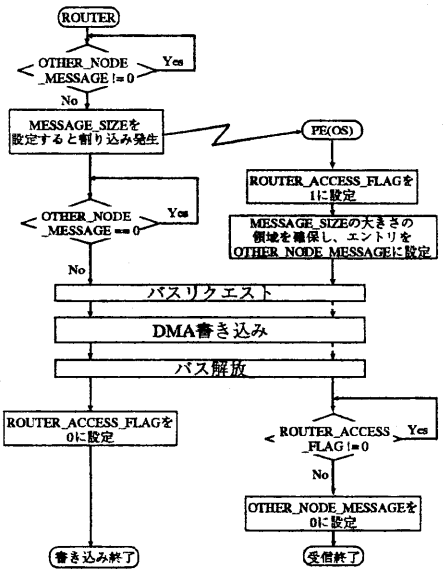


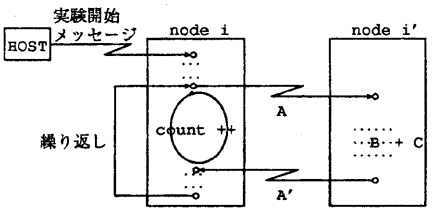
図 2: メッセージ受信アルゴリズム

べる。なお実験対象となるノード台数は現在実装されている16台とした。ネットワークトポロジは4次元のハイパーキューブを用いる。

なお、本章では、メッセージ衝突がない場合のメッセージ通信の性能を評価したことについて述べる。

3.1 ルータ-ルータ間の転送距離・転送メッセージサイズと遅延時間の関係

転送距離・メッセージサイズによって、どのように遅延時間に影響があるか実験した。



A ：node i から node i' までの転送時間
B ：node i'がメッセージを受信してからnode iに送信するまでの時間
C ：ルータのメモリへの書き込み時間など
A'：node i' から node i までの転送時間

図 3: メッセージピンポンの実験方法

実験方法として、図3のように2ノード間のメッセージピンポンを行い、往復転送時間を計測した。送信するノードが送信と同時にカウントを行い、ピンポンされたメッセージを受信した時にカウントするのをやめる。求められる往復転送時間 T_d は、図3中の A+B+C+A' である。

この方法で、転送距離・メッセージサイズを変化させて実験を行い、図4のような結果を得た。

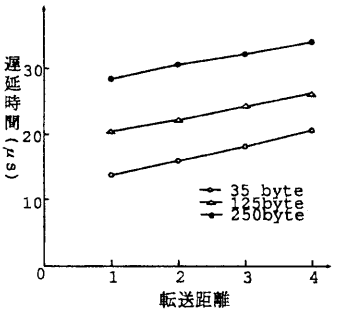


図 4: 転送距離・転送メッセージサイズと遅延時間の関係 (往復転送)

各メッセージサイズのグラフの傾きはほぼ一定しているので、距離 n と距離 n-1(n=2,3,4) の往復転送時間差は、約 2.1μs であることが分かる。この実験は、往復転送なので、片道転送を考えると、中継ノードで1ホップさせるための時間は、$2.1/2 = 1.05\mu s$ であると考えられる。また、距離1を例に挙げて遅延時間とメッセージサイズの関係を考える。35,125,250 バイトの時のそれぞれの転送時間は 14.16, 20.21, 28.49μs である。これより、転送するメッセージサイズが1バイト増すごとに 0.067μs 遅延することが分かる。

この実験で、B と C の時間が分かった。B はメッセージを受信して再び送信するまでの時間でマイクロプログラムのステップ数を数えることにより 4.84μs であることが分かる。また C は 0.067n μs(n はバイト数) である。

また A,A' について考えてみる。距離1の宛先ノードにメッセージの先頭が到着する時間を x とすると、1ホップ 1.05μs であるから、距離を D とすると $x + 1.05(D-1)\mu s$ であると考えられる。また、メッセージの先頭が到着する時間 x は転送するメッセージサイズによってほとんど変わらない。x については、この実験では測定することができないが、次節の実験により求めことができる。

なお、ピンポンによるメッセージ転送を複数回連続で繰り返しても、同様のことが言える。

文献 [4] で性能評価した時は、1ホップにかかる時間 1.0μs、1バイト増すごとに 0.067μs と本実験の結果とほぼ同じである。文献 [4] ではノードを2台つなげでロジックアナライザを用いて厳密に評価している。よって、実験項目によっては文献 [4] のような厳密な測定が不可能な場合でも、実験方法を工夫することにより、ファームウェアあるいはソフトウェアを活用することで、ある程度正確に測定でき

ると考えられる。

3.2 マルチキャスト性能

マルチキャストでは、複数個のメッセージを1度の送信命令で転送する。ルータはマルチキャストで送信したい複数個のメッセージの最終アドレスを特定のアドレスに書き込むことで、1個ずつ逐次に各ノードに転送していく。

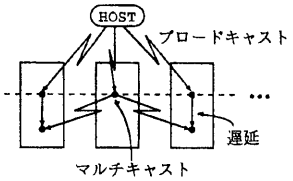


図 5: マルチキャスト性能評価の実験方法

実験は図5のように行った。ホストからのブロードキャストによる実験開始のメッセージは、同時に全ノードに到着する。マルチキャストで送信するメッセージ数は15個とし、それぞれ35バイトとした。また、メッセージを送る順番は、わかり易いように距離1となる宛先ノードから順に並べた。

この結果、図6のような結果が得られた。

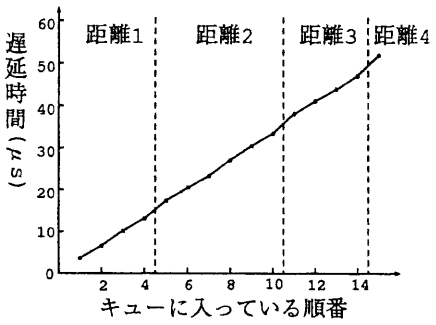


図 6: マルチキャストによるメッセージ送信と遅延時間の関係 (片道転送)

この実験により、i 番目のメッセージを送信してから、$i+1$ 番目のメッセージを送信するまでの時間は、約 $3.22\mu s$ であることが分かる。また、距離が変化するのは図6の破線のところであり、距離が増加する分、遅延時間が若干増加していることが分かる。

また、この方法で、距離1でのメッセージの先頭が到着する時間 (前節の x) が $x = 3.61\mu s$ であることが分かる。従って、前節のA,A'は $3.61+1.05(D-1)\mu s$ であることが分かる。これらの結果より、往復転送時間 T_d は次式で与えられる。

$$\begin{aligned} T_d &= A+B+C+A' = 2A+B+C \\ &= 2\{3.61+1.05(D-1)\}+4.84+0.067n \\ &= 9.96+2.1D \quad (\mu s) \end{aligned}$$

次は、マルチキャスト性能式を求める。キューの i 番目のメッセージを送信する場合、宛先ノードに届く時間を T_m、i 番目のメッセージを送信してから、$i+1$ 番目メッセージを送信するまでの時間を T_s とすると次式となる。

$$\begin{aligned} T_m &= 3.61+1.05(D-1)+T_s(i-1) \\ &= 3.61+1.05(D-1)+3.22(i-1) \\ &= -0.66+1.05D+3.22i \quad (\mu s) \end{aligned}$$

3.3 ノード-ホスト間の通信性能

ノード-ホスト間の往復転送時間をメッセージピンポンにより測定した。

その結果、35バイトのメッセージ往復転送時間は $2040.02\mu s$ となった。この結果はノード-ノード間の $14.16\mu s$ とは大きく違う。ホスト (NWS-1750) での通信処理はC言語プログラムで実現されている [6]。メッセージを受信するために、while文により絶えずデバイスファイルをreadしている。また、メッセージを送信する場合は、デバイスファイルにwriteする。これによりルータ- ホスト間通信はホストの処理の遅さがネックとなっている。

速度の改善としては、VME I/F のハードウェアやデバイスドライバの高速化が挙げられる。

4. メッセージ衝突がある場合での性能評価

本章では、メッセージ衝突が起こる場合を種々のパターンで実験し、メッセージ転送の遅延時間を調べた結果を述べる。

4.1 単純なメッセージ衝突

図3のような2ノード間のメッセージピンポンを2つ組み合わせてメッセージを衝突させる実験を行った。実験では、組み合わせを図7のように2つのパターンでそれぞれメッセージサイズは最小メッセージ構成の35バイトで行った。なお、ルータは最短経路の中から1つの経路を選択する設定で行った。

パターン1は、ノード0,3間とノード2,7間のピンポン転送を組み合わせたものであり、パターン2は、ノード0,5間とノード1,4間のピンポン転送を組み合わせたものである。以下は衝突がない場合にとる経路であるが衝突が生じた場合には、適応ルーティングにより冗長経路をとる場合もある。アンダーバーがついている数字は送信及び受信ノード番号を示し、そうでない数字は中継ノード番号を示す。

パターン1　$\underline{0} \to 1 \to \underline{3} \to 2 \to \underline{0}$
　　$\underline{2} \to 3 \to \underline{7} \to 6 \to \underline{2}$

パターン2　$\underline{0} \to 1 \to \underline{5} \to 4 \to \underline{0}$

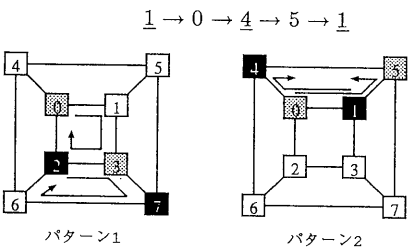


図 7: 単純なメッセージ衝突パターン

パターン1は、1つの経路でしか衝突しないパターンだが、パターン2は転送が行われる全ての4つの経路で衝突が起こる可能性がある。

実験結果は図8のようになった。パターン1では、ある程度メッセージ衝突による遅延は確認できるが、それほど大きくない。パターン2では、1度目の送信でメッセージが衝突し、大きく遅延するが、2度目からは、メッセージ送信のタイミングがずれてしまい、衝突することが少なくなることが分かる。

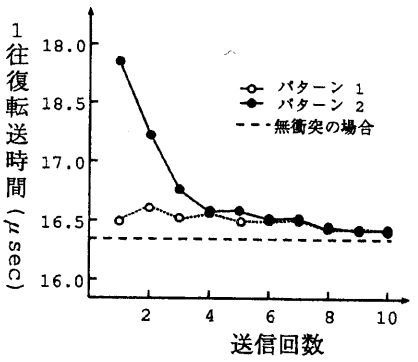


図 8: 単純なメッセージ衝突と転送時間の関係 (往復転送)

4.2 リバース・フリップ・パターン

前節の実験から、タイミングがずれるとメッセージ衝突による遅延は、あまり大きくないことが分かった。次は、より複雑な衝突が起こるようにメッセージ送信を行い、同時に送信間隔を変化させて計測した。

リバース・フリップ・パターンとは、2進数で表したノード番号 (x_1, x_2, x_3, x_4) がノード $(\bar{x_1}, \bar{x_2}, \bar{x_3}, \bar{x_4})$ にメッセージを転送するパターンである。このパターンのメッセージはネットワークの直径分転送されるため衝突が多く生じる。なお、16ノードの場合は各経路に2重衝突、あるいは冗長経路をとることにより、それ以上の衝突が起こる可能性がある。

実験方法として、図3のような2ノード間のメッセージピンポンを8組使い、メッセージサイズ35バイトでメッセージ送信間隔を変化させ実験した。図9に実験結果を示す。

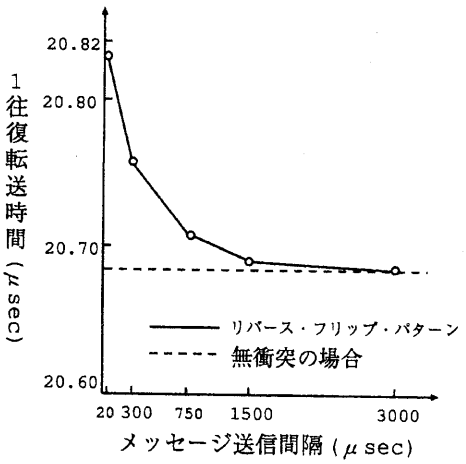


図 9: リバース・フリップ・パターンにおけるメッセージ送信間隔と往復転送時間の関係 (往復転送)

メッセージ送信間隔を変化させることにより、遅延時間が変化することが分かる。破線は無衝突の場合の転送時間を表し、送信間隔を縮めてネットワークの負荷を大きくしても、転送時間は変化しない。このことより、メッセージ送信間隔が大きくなるとネットワークの負荷は小さくなり、メッセージはほとんど衝突しないことが分かる。逆に、メッセージ送信間隔が小さくなって、ネットワーク負荷が大きくなるとメッセージ衝突が頻繁に起こり、遅延時間が増える。

4.3 ユニフォーム・トラフィック

本節と次節は、実アプリケーション・プログラムに近づけるために、メッセージの宛先を固定せず、非同期にメッセージを送信する実験を行った結果について述べる。

本節では各ノードが非同期に、ある一定の間隔で他ノードへ均等にメッセージを送るようにして実験を行った。評価結果を図10に示す。横軸がメッセージ送信間隔で、縦軸が1メッセージ当りの受信時間を示す。

メッセージ間隔が300μs以上は、メッセージサイズに関わらず、ほぼ同様な値となった。これは、ネットワーク負荷が小さいためメッセージ衝突が少ないと考えられる。また、メッセージサイズが大きい方が遅延しているが、これはサイズが大きい分だけ、ネットワーク中に残っている率が大きく、衝突が起きやすいと考えられる。

4.4 ランダム・トラフィック

本節では各々のノードが宛先をランダムに変えながら非同期にメッセージ送信する実験を行った結果

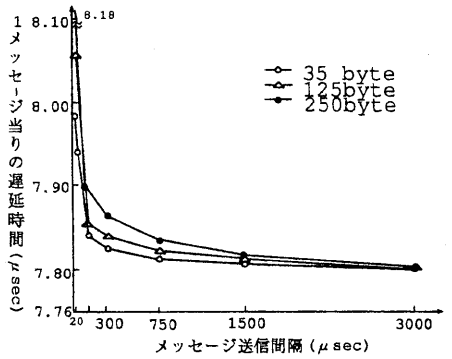


図 10: ユニフォーム・トラフィックにおけるメッセージ送信間隔と遅延時間との関係 (片道転送)

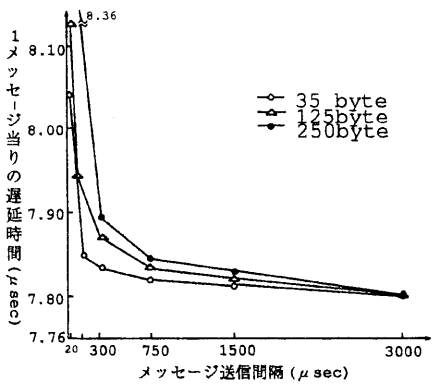


図 11: ランダム・トラフィックにおけるメッセージ送信間隔と遅延時間との関係 (片道転送)

について述べる。メッセージ送信間隔、メッセージサイズを変化させて行ったところ図11のような結果を得た。

ある程度送信間隔が大きくなるとグラフの傾きが緩やかになる形は、前節のユニフォーム・トラフィックと同じである。理由も同じであると考えられる。

ユニフォーム・トラフィックの場合、転送に規則性があり衝突によってタイミングがずれるとずれたまま規則的に送信するので、比較的少ない衝突で転送できると考えれる。しかし、ランダム転送では、そういった点で衝突が頻繁に起こる。このような理由でユニフォーム・トラフィックの場合と比較して遅延が顕著であると考えられる。グラフでは、急激な傾きを示しているが、縦軸は非常に小さいスケールであるので実際には、衝突による遅延時間は小さいと言える。このことは、4.2、及び、4.3 節の実験結果においても同様である。

5. まとめ

本稿では、A-NET マルチコンピュータにおいて通信性能を評価したことについて述べ、以下のような結果を得た。

- メッセージ衝突がない場合の往復転送時間 T_d を得た。
- マルチキャスト時間 T_m を得た。
- ノード-ホスト間の通信性能では、ホストの処理の遅さがネックとなっている。
- 単純なメッセージ衝突において 2 回目以降の送信ではタイミングがずれ、衝突の影響が殆どなくなる。
- 複雑なメッセージ衝突を起こすリバース・フリップ・パターンにおいて、ネットワークの負荷を変化させることにより、遅延時間が変化することを確認した。
- ユニフォームとランダム転送においては、送信間隔が 300μs 以下ではメッセージ衝突による遅延は認められたが、小さい時間であった。

A-NETL レベルの実際のアプリケーション・プログラムの通信は、本実験のようにハードウェア性能の限界で実行させている訳ではない。よって、以上の結果より、A-NETL のメッセージ送信においては、送信間隔やサイズが変化しても十分な性能を持つことが分かった。

今後の課題として、ネットワークトポロジを変化させた時の性能評価、及び、実アプリケーションの通信パターンによる評価が挙げられる。

謝辞 本研究は、一部文部省科学研究費 (基盤 (C) 課題番号 08680346)、電気通信普及財団、並列・分散処理研究推進機構の援助による。

参考文献

[1] 馬場敬信,“コンピュータアーキテクチャ”, オーム社,p.408(1994).

[2] Wiljo J. van Beek,Rob A.H van Twist,Marnix C. Vlot“Evaluation of the Communication Network of POOMA” ,International Conference on Parallel Processing,I-498(1990).

[3] 馬場敬信, 吉永努,“並列オブジェクト指向トータルアーキテクチャA-NET における言語とアーキテクチャの統合”, 信学会論文誌 Vol.J75-D-I No.8 pp.563-574(1992)

[4] 吉永努, 馬場敬信,“並列オブジェクト指向トータルアーキテクチャA-NET のノードプロセッサ”, 信学論,J79-D-I,2,pp.60-68(1996).

[5] 吉永努, 馬場敬信,“並列オブジェクト指向トータルアーキテクチャA-NET のためのトポロジ独立なルータの構成”, 情処学論,34,4,pp.648-657(1993).

[6] 廣田守, 吉永努, 馬場敬信,“並列オブジェクト指向トータルアーキテクチャA-NET – ホストマシンソフトウェアの実装 –”, 情報処理学会 第 52 回全国大会予稿集 2L-5 pp.6-101 - 6-102(1996)