ハイパフォーマンス
コンピューティング 48－13
（１９９３．８．２０）

並列オブジェクト指向トータルアーキテクチャA-NET
— ルータハードウェアの簡単化とメッセージ交換の高速化 —

古谷 泰重　田口 弘史　吉永 努　馬場 敬信

宇都宮大学工学部

本稿では，トータルアーキテクチャA-NETにおいて，過去のシミュレーションにより判明した，通信オーバヘッドの削減を主な目的とする，メッセージ交換の高速化手法について述べる．また，メッセージ交換の改善に伴う，ルータハードウェアの簡単化について述べる．さらに，新方式に対応した実行モデルのシミュレータを用いて実験を行い，実機上での動的な性能を予測し，システムの妥当性を検証する．シミュレーションにより，次の2点の結果を得た．(1) 実行速度の向上率は0.9～3.2倍となった．(2) 通信量の削減率は32～58%となった．

A Parallel Object-Oriented Total Architecture A-NET
— Simplification of router hardware and high-speed message passing —

Yasushige Furuya　Hiroshi Taguchi　Tsutomu Yoshinaga　Takanobu Baba

Department of Information Science, Utsunomiya University
Utsunomiya-shi 321 Japan

We describe new mechanism for a high-speed message passing. Our major objective is to reduce the communication overhead. The mechanism also the router hardware makes simple. We have developed a simulator according to the mechanism. Using the simulator, we evaluated the effectiveness of the mechanism. The results show: (1)the execution time speeds up 0.9 ~ 3.2 times better than old mechanism. (2)the transmission late is reduced 32 ~ 58%.

1 はじめに

我々のプロジェクトでは，並列オブジェクト指向を核概念として，計算機，言語処理，応用の3つの立場から統合的に検討したトータルアーキテクチャA-NET[1, 2] の開発を行っている．これらの研究において，応用プログラムの論理チェックや，A-NET計算機を構成する各処理ユニットの性能バランスを決定するために，シミュレーションが必要であった．そこでこれまでに，UNIX ワークステーション上にシミュレータ [3] を構築し，実機上での動的な性能予測をするためのシミュレーションを行ってきた．

これまでの研究によって，メッセージ交換時のオーバヘッドが大きなものであることが確認された．特に，メッセージの受信が入力キューを介して行われることと，動的オブジェクト生成時に生成先ノードを探索する場合に顕著であることが判明している．

以上の問題点を元に，メッセージ交換の高速化を中心として，より効率的な実行モデルへの改善を行った．また，改善された実行モデルに対応したシミュレータにより，旧モデルとの比較実験を行った．

本稿では，まずA-NET 計算機および，ローカルOS の概要について述べる．次に，実行モデルの改善方針と実現方法について述べ，最後に旧モデルと新モデルに対応したシミュレータを使用しての比較実験と，その評価について報告する．

2 A-NET 計算機

2.1 A-NET 計算機の概要

A-NET 計算機は，並列オブジェクト指向言語 A-NETL の高速実行を行うためのノードプロセッサ [4]（以下単にノードと呼ぶ）を 1000 台規模のネットワークにより結合した MIMD 型高並列計算機である（図 1参照）．ネットワークはトポロジ独立であり，各ノードの接続形態は対象問題に合わせて選択可能となっている．

独立した処理要素であるノードは，オブジェクトのメソッド実行を行うプロセッシングエレメント（PE）[5] と，メッセージの経路選択および転送を行うルータ [6] から構成される．また各 PE には，図 2に示すような 320K バイトのローカルメモリが用意されている．

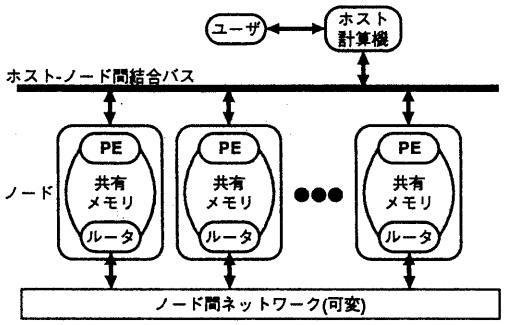


図 1: A-NET 計算機の概要

図中，古い領域・現在領域・未来領域は，世代別ガーベッジコレクション（GC）を行うために分割されている．古い領域は，ユーザの定義したオブジェクトと寿命の長いデータが占有する．この領域は，GC の対象とならない．現在領域・未来領域は実行時データが占有する．この 2 つの領域で，実際に利用されるのは現在領域のみである．現在領域を使い切ると GC を起こし，未来領域に必要なデータをコピーする．その後，未来領域と現在領域の立場を入れ換える．

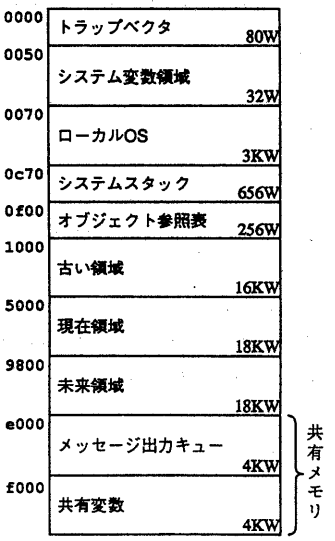


図 2: 実行モデル第 3 版のローカルメモリのメモリマップ

また，$(e000)_{16}$からの40Kバイトは，PE（OS）とルータの間で相互に情報を交換するための共有メモリとして用いられる．

2.2 ローカルOSの概要

A-NET計算機では，各ノード上でメッセージのディスパッチやコンテクスト管理を行うオペレーティングシステム（OS）が必要である．A-NETローカルOS[7]は，実行モデルを強く反映した専用OSである．ローカルOSが行う主な処理には以下のものがある．

- オブジェクトの初期化

ローカルOSはPEのローカルメモリにロードされた後，自身の状態変数およびオブジェクト参照表の初期化を行う．また，ユーザオブジェクトは再配置可能コードで転送されるため，ローカルOSでユーザオブジェクトの再配置を行う必要がある．

- メッセージ受理

メッセージ受理処理は，メッセージ割り込みによって行われる．ローカルOSは受理したメッセージがメソッド起動用メッセージであると，それをリスト（レディリストと呼ばれる）に追加する．

- メソッドの起動

ユーザメソッドは，メソッド起動用メッセージや起動条件の揃ったコンテクストから起動される．ローカルOSはレディリストの先頭要素を取り出してメソッドを起動する．

- 動的オブジェクトの生成

従来は，ファームウェアとルータが行っていた処理である．この動作は，ローカルOSが動的オブジェクト生成用メッセージを受理した際，レディリストに追加されることなく優先的に処理される．

- メソッド実行順序制御[8]

オブジェクトの自立性の向上を目的とし，より細かなメソッド起動を制御することが可能である．指定可能な制御には次の3つがある．(1) メソッドの優先順位を指定する機能．(2) 同一メソッドの二重起動を禁止する機能．(3) 条件が真であるときメソッドを起動する機能．

ローカルOSは，本稿で述べるメッセージ交換の高速化に大きな役割を果たしている．

3 メッセージ交換の高速化

ここでは，本研究の主目的であるメッセージ交換の高速化の手法を，旧仕様の実行モデルと対比させながら説明する．以下では，旧仕様の実行モデルを“第2版”，新仕様の実行モデルを“第3版”と呼ぶ．

3.1 メッセージ送受信方式

第2版では，PE・ルータ間共有メモリ上に，入出力用の2つのキューが存在していた．外部に対するメッセージ送受信は，それぞれのキューを経由して行われていた．しかしながら，この方式では以下の問題点が指摘された．

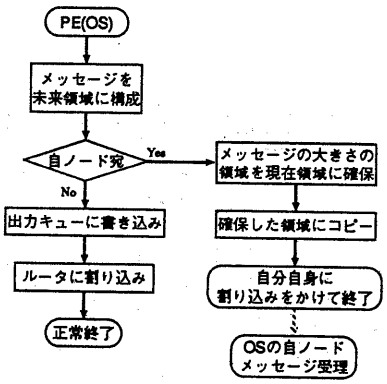


図3: メッセージ送信シーケンス

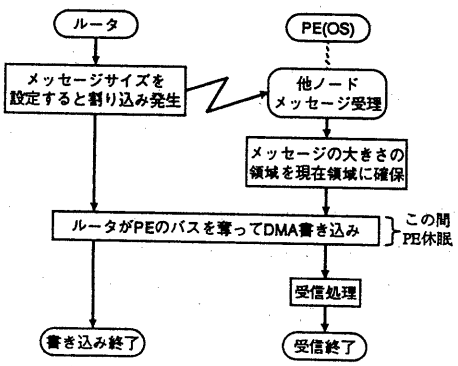


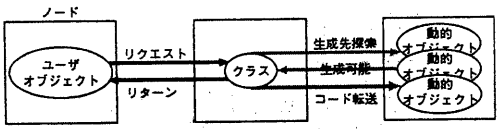
図4: メッセージ受理シーケンス

• メッセージ到着時にローカルOSが行う受理動作において，一旦入力キューに書き込まれたデータを，ローカルメモリ内の現在領域にコピーしなければならない．

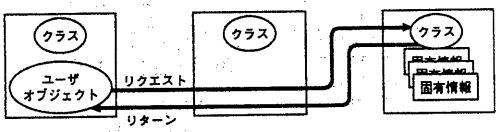
そこで第3版では，メッセージ入力キューを廃止し，代わりにルータがPE内のローカルメモリ上の現在領域に直接メッセージを書き込む機能を新設することとした．また，自ノード宛メッセージ出力の際は，出力キューにはメッセージを書き込まず，現在領域に直接書き込む機構を採った．改良された送受信シーケンスを図3，4に示す．

3.2 動的オブジェクトの生成方式

第2版では図5(a)に示すように，動的オブジェクトのひな型となるクラスオブジェクトは，全ネットワーク中で唯一のノードに配置されていた．生成依頼があるとルータが生成先ノードを探索し，回線交換方式により命令コードを転送していた．



(a) 第2版



(b) 第3版

図 5: 動的オブジェクト生成動作

この方式は，システム側が適当な生成先ノードを決定するのでユーザに負担をかけないという利点を持っているが，反面，以下のような問題点を含んでいた．

• 動的オブジェクトの生成先を探索する際のオーバヘッドが大きい．動的オブジェクトの生成時には，生成先探索・生成可能・生成失敗の各メッセージを対象ノード間で相互通信していた．

• 生成先ノードが確定するとオブジェクトコード自体を転送する必要があり，そのオーバヘッドが大きい．

• 動的オブジェクトを1個生成するたびに完全なコピーを作いるため，同一ノードに複数の動的オブジェクトを作る際，メモリ効率が良くない．

• ルータにおいて，2種類のデータ転送方式（パケット交換方式／回線交換方式）をサポートしなければならない．

そこで第3版においては，図5(b)に示すように，あらかじめクラスオブジェクトを全ノードに配置しておき，ユーザが生成先ノードを指定するようにした．また，命令コード部分を共有するため新たに作成するのは固有情報だけとなる．なお，生成処理はローカルOSが行うことになった．

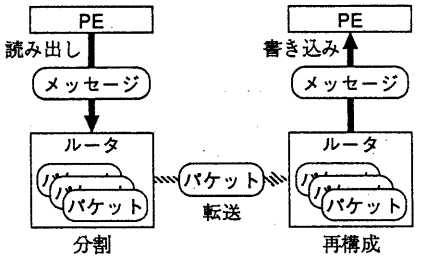
これにより，余分なメッセージ交換の必要がなくなり，メモリ効率が良くなっている．また，機能を簡略化したため，ルータにおいてノード間での回線交換方式による転送方式が不要となり，ハードウェアが簡単化された．

3.3 メッセージ分割／再構成

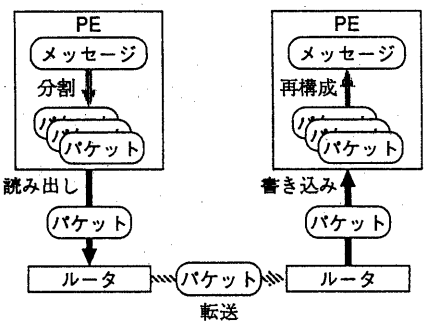
メッセージが一定長を越えると，メッセージは複数に分割されてから転送される．そして，受信ノードでは分割されたメッセージを再構成する処理を行う．

第2版において，この処理はルータが行っていた．しかし，過去のシミュレーションの結果から平均的なメッセージ長が16ワードを越えることは稀であることが判明しており，この機能をハードウェアで実現する必要はないと判断した．そこで，第3版では分割はファームウェア，再構成はローカルOSが担当することとし，ルータからこの機能を削除した（図6参照）．

この移行は，ハードウェアからソフトウェアへの移行であり，速度低下の要因となるため分割を最小限に押える必要がある．そこで，第2版では16ワードごとに行っていた分割を第3版では51ワードまで引き上げた．平均的なメッセージ長が16ワード以内であることを考えれば，この変更によってほとんど分割が起きないと考えられる．



(a) 第 2 版



(b) 第 3 版

図 6: メッセージ分割／再構成

3.4 ルータハードウェアの簡単化

前述してきたメッセージ交換の高速化手法の結果得られた，ルータハードウェアの簡単化についてまとめておく．

まず，動的オブジェクト生成方式の変更により，生成先探索時に交換するメッセージのための特別な機能が不要となった．また，オブジェクトコード自体の転送を行わないためノード間における回線交換方式が不要となった．

次に，分割／再構成処理が移行したため，ルータにおいてこの機能をサポートする必要がなくなった．

以上の簡単化の結果現在のルータの主な処理は，ホスト・ノード間におけるオブジェクトコードの転送とメッセージ通信，ノード・ノード間におけるメッセージ通信となっている．

4 実験

改善の効果を確認するため，シミュレータを用いて実験を行った．本実験は，今回の改善によって変更された部分に的を絞って行った．実験によって確認すべき点は以下のものである．

- メッセージ送受信方式の変更による実行速度の変化
- 動的オブジェクト生成方式の変更による実行速度と通信量の変化
- メッセージ分割／再構成処理の移行による実行速度と通信量の変化

4.1 シミュレータ

A-NET シミュレータは，ネットワークにより接続されている UNIX ワークステーション上に，PE・ルータ等の各処理ユニットを単位としてプロセスを構成している．各プロセスは，A-NET 計算機を忠実にシミュレートすることにより，ハードウェアからソフトウェアに至る評価を行うことを可能としている．また，プロセスを複数の計算機に分散させることにより，高速かつ大規模なシミュレーションを可能としている．

4.2 実験方法

今回の実験に使用したサンプルプログラムは，以下の 3 つである．

(1) 多体問題

物体の運動をシミュレートするプログラム．プログラムは，系が安定するまで続行する．また，プログラム規模が比較的大きく，多数の通信が行われている．

(2) N Queens

$N \times N$のチェス版上で，互いに取ることのできないN個のクイーンの配置パターンを数え上げるプログラム．$N = 4$の問題で，動的オブジェクトを次々に生成しながら処理を進める．

(3) 最短経路問題

nノードからなるグラフにおいて，任意のノード間の最短経路を求めるプログラム．$n = 15$のグラフにダイクストラのアルゴリズムを適用した．巨大なメッセージ（最大160 ワード）を送信するプログラムである．

シミュレーション時のネットワークトポロジは，全プログラムで4次元ハイパーキューブ（ノード数16個）を使用した．また，オブジェクトの配置情報は，第2版，第3版で同一とした．なお第3版において，動的オブジェクト生成に関するA-NETLの構文規則が大きく変更されている．そこで，アルゴリズムの変更とならないように留意してプログラムを再構成した．

評価において，実行時間の単位は，内部クロック30MHzとした場合での時間である．実行時間にはシステム起動時間，ユーザプログラムロード時間が含まれる．

4.3 結果と考察

表1，2に各プログラムの総実行時間と総通信量の比較を示す．なお，総実行時間の速度向上率と総通信量の通信削減率は以下の式(1)，(2)で算出した．

$$(\text{向上率}) = (\text{第2版}) \div (\text{第3版}) \quad (1)$$

$$(\text{削減率}) = \frac{(\text{第2版}) - (\text{第3版})}{(\text{第2版})} \times 100 \quad (2)$$

また，プログラム実行状況を確認するため，例として図7(a)(b)に“N Queens”の時間－並列度グラフを示す．両グラフで，並列度の最も高い最初の部分がシステムの起動とユーザプログラムのロードである．

結果から判明した事実と，その考察を述べる．

メッセージ送受信方式の変更による影響　本方式変更は実行速度に関して影響を与える．3.1項で述べたように，第3版では入力キューを廃止したため，第2版に存在した入力キューから現在領域にメッセージをコピーす時間がなくなっている．このため理論上，メッセージ受信回数が増加するほど，改善効果が著しく現れる．

サンプルプログラムの多体問題と最短経路問題を比較してみる．多体問題は，3つのプログラム中で最もメッセージ受信回数が多い．反対に，最短経路問題は，1ノードあたり数回しかメッセージを受信しない．表1の結果から，受信回数の多いプログラムにおける速度向上率の高さが確認できる．

よって，大規模なアプリケーションを構築する場合，第3版の方式を採用した利点は大きいと考えられる．

表1: 総実行時間

プログラム	実行時間		向上率
	第2版	第3版	
多体問題	261	163	1.6倍
N Queens	107	33	3.2倍
最短経路問題	20	22	0.9倍

（単位:*msec*）

表2: 総通信量

プログラム	通信量		削減率
	第2版	第3版	
多体問題	404	273	32%
N Queens	506	218	57%
最短経路問題	542	226	58%

（単位:Kbytes）

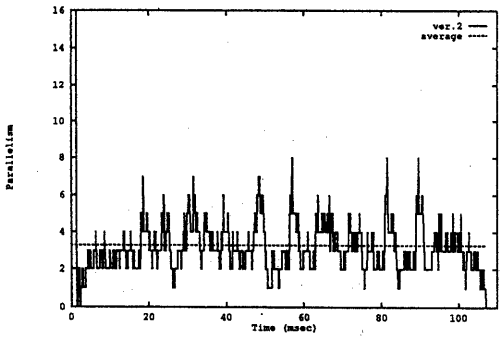
動的オブジェクト生成方式の変更による影響　まず，実行速度に関して考察する．第2版ではクラスオブジェクトが全ネットワーク中で唯一であったのに対して，第3版では各ノードに存在するため以下の2点の特長がある．

(1) 第2版では，動的オブジェクト生成のたびに，オブジェクトコードを目的ノードに転送し，再配置を行っていた．動的オブジェクト生成が繰り返される場合，再配置にかかる時間が積算され実行時間に大きく現れるようになる．第3版では，再配置の必要はなくなった．

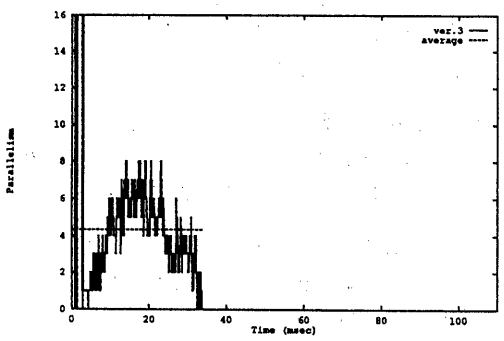
(2) 同時に多数の生成要求が起きたとき，第2版では生成動作が逐次的になっていた．一方，第3版では並列的に生成動作が行える．

サンプルプログラムでは，N Queensが動的オブジェクトを用いたプログラムである．しかも，このプログラムは1ノード中に複数個の動的オブジェクトを生成する．そのため，表1に示す通り速度向上率は非常に高いものとなっている．

次に通信量に関して考察する．特長として3.2で説明した，探索開始・探索成功・探索失敗の各メッセージと，オブジェクトコード自体の転送の廃止が挙げられる．特に，後者は生成回数が増えるほど，



(a) 第2版



(b) 第3版

図 7: "N Queens" の時間－並列度グラフ

またクラスオブジェクトが大きくなるほど影響が著しい.

N Queens は生成回数が多いため，削減率が大きくなっていることが確認できる（表2参照）.

よって，動的オブジェクトを用いたプログラムでは，実行速度と通信量の両面において，第3版の生成方式が優れていると結論付けられる.

分割／再構成処理の移行による影響　まず，実行速度に関して考察する. 3.3項で説明したように，この移行は実行速度を低下させる要因となる. しかし，分割が起きる可能性が低いことと，メッセージ受理の高速化がなされたことを合わせれば，目立った速度低下は見られないはずである.

サンプルプログラムの最短経路問題で評価する. このプログラムは，メッセージ受信回数が少ないためメッセージ受理の高速化の恩恵を受けにくい. また，このプログラムは交換されるほとんどのメッセージが分割されるという点で特殊である. そのため，表1に示すように第3版の方が遅くなっている. それでも，速度低下の割合は1割程度であり，分割／再構成処理の移行による影響は小さいと考えられる.

通常のプログラムでは，交換されるメッセージのうち分割されるものは少数である. 分割されないメッセージが多く交換されれば，メッセージ受理の高速化の恩恵を受け，全体の実行速度は向上する.

次に，通信量に関して考察する. 第2版では，ルータは受信したパケットが分割されたものであると一時保持しておき，再構成を行ってから PE に送信していた. 第3版では，ルータでの再構成処理が不要なため，バッファリングにかかる通信量がなくなり大幅な減少となっている. サンプルプログラムでは，最短経路問題の通信量が削減されている理由がこれにあてはまる.

3点目として，分割の発生率について検証する. サンプルプログラムの多体問題では，一部のメッセージが30ワード程度あり，第2版では分割を起こしていた. 第3版では，分割サイズの拡張により分割は発生しなくなっている. メッセージが巨大化するのは，一度に必要な引数を全て渡してしまうためである. 分割を押えるためには，ユーザが問題領域をできるだけ絞って少ない引数を渡すように工夫することも必要である.

その他の影響　ここでは，各プログラムで個別の現象について考察を行う.

まず，多体問題において，ローカル OS の改良によりメモリ効率が良くなっているため，第2版で発生していた GC が第3版では発生しなくなった. 次に，N Queens において，動的オブジェクトの生成動作が並列的に行えるようになったことから，並列度が上がっている（図7参照）.

5　おわりに

本稿では，A-NET アーキテクチャにおける実行モデルの改善とそのシミュレーションについて述べた. メッセージ交換の高速化を中心とした改善により，実行速度に関しては，おおむね向上している. 特

に，動的オブジェクト生成を用いたプログラムでの効果が著しい．また，通信量に関しては，ルータの簡単化に伴って余分なメッセージ（パケット）の転送がなくなり，大きく削減されている．

現段階の問題点として次の点が考えられる．

- 動的オブジェクトの生成方式の変更による問題．

第3版では，動的オブジェクトの生成先ノードの決定をユーザに任せている．このため，プログラムが大規模化した際に管理が難しくなる．この件に関しては，適当な生成先ノードを決定するライブラリを用意して対処する方針である．

- シミュレータの時間計測．

現シミュレータには，設計段階においてルータの性能予測がなされていなかったため，この部分の計測が組み込まれていない．この件に関しては，今後の予定で述べるように現在改良中である．

今後のプロジェクトの予定として以下のものが挙げられる．(1) 実機の完成．ノードプロセッサである，PE・ルータハードウェアともに設計が進んでおり，本年度中の実装を予定している．(2)A-NETL デバッガの開発．並列計算機専用のデバッギングシステムとして研究を進めている．現段階では，シミュレータ上での実装を予定している．(3) シミュレータにおけるルーティング時間の組み込み．現在ルータの性能予測が確定しており，シミュレータに組み込む段階である．これにより，ホップ数の違いからくる通信遅延等のネットワーク評価をシミュレータでより計測できるようになり，大規模プログラムの相互通信チェックやトポロジの変化とアロケーション処理による効果を確認することが可能となる．

謝辞

本研究は，一部，文部省科研費（試験研究 (B) 課題番号 04555077，重点領域研究（超並列）課題番号 05219203，奨励研究課題番号 05780228）の補助を受けている．

参考文献

[1] 馬場: “超並列マシンへの道”, 情報処理学会誌 Vol.32 No.4 pp.348-364(1991).

[2] 馬場, 吉永: “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”, 電子情報通信学会論文誌 Vol.J75-D-I pp.563-574(1992).

[3] 古谷, 町田: “A-NET 計算機におけるメッセージ交換の高速化とそのシミュレーション”, 宇都宮大学情報工学科 卒業論文 (1993).

[4] 吉永 他: “A-NET 計算機のノードプロセッサとその実行方式”, 並列処理シンポジウム JSPP'91(1991).

[5] 鈴木 他: “並列オブジェクト指向トータルアーキテクチャA-NET の要素プロセッサ”, 情報処理学会第 87 回計算機アーキテクチャ研究報告 91-ARC-87-4(1991).

[6] 吉永 他: “並列オブジェクト指向トータルアーキテクチャA-NET —ルータの構成—”, 並列処理シンポジウム JSPP'92(1992).

[7] 吉永, 馬場: “A-NET ローカル OS のメッセージ受理機構”, 情報処理学会研究報告 90-OS-48,pp.35-42(1990).

[8] 田口: “並列オブジェクト指向言語 A-NETL への実行順序制御機構の導入と実行性能の改善”, 情報処理学会第 46 回全国大会 講演論文集 (5) pp.33-34(1993).