ハイパクロスバ・ネットワークにおける転送性能向上のための手法とその評価

朴 泰祐[†] 曽根 猛[†] 三島 健[†]
板倉憲一[†] 中澤喜三郎[†] 中村 宏[†]

ハイパクロスバ・ネットワークは，数千台以上のプロセッサを有する超並列計算機におけるプロセッサ間結合ネットワークとして，多くの優れた性質を持っている．これまで同ネットワークは，ネットワーク中でのデッドロックを回避するため，固定ルーティングを前提とした，store & forward あるいは wormhole 方式のルーティングのみを対象として研究されてきた．本論文では，同ネットワークにおけるデッドロック・フリーな適応ルーティングの手法を提案し，それによる性能向上について計算機シミュレーションにより評価する．また，単純な wormhole 方式の転送に代わり virtual-cut-through 方式のメッセージ転送を導入することにより，さらにその転送性能が向上することも評価する．これらの改良により，同ネットワークの転送性能が大幅に向上することが確認された．

Advanced Techniques for Performance Improvement of Hyper-Crossbar Network

Taisuke Boku,[†] Takeshi Sone,[†] Takeshi Mishima,[†] Ken'ichi Itakura,[†]
Kisaburo Nakazawa [†] and Hiroshi Nakamura [†]

Hyper-Crossbar Network has several excellent features as an inter-PU network of massively parallel processing systems. In researches on this network so far, the routing algorithm is based on store-and-forward or wormhole method with fixed routing to avoid deadlock on the network. In this paper, we propose a deadlock-free adaptive routing technique for Hyper-Crossbar Network, and evaluate the performance improvement with it. We also apply a virtual-cut-through technique to Hyper-Crossbar Network instead of simple wormhole routing, and evaluate its effect. We confirmed the network throughput is much improved introducing these techniques on Hyper-Crossbar Network.

1. はじめに

数千台以上の高性能RISCプロセッサをPU (Processing Unit) として，それらをネットワークによって相互結合し，非常に高い性能を持つ超並列計算機を実現しようという試みが現在盛んに行われている．特に，各PUがローカルなメモリを持ち，メッセージ・パッシングによって交信する，いわゆる分散メモリ型並列計算機では，超並列システムの実装は現実のものとなってきている．

このようなアーキテクチャにおいて，ネットワーク・トポロジとルーティング・アルゴリズムの選択は最も重要な問題の一つである．本論文では，hypercube系ネットワークの性質をうまく利用し，ランダム転送に対して高い柔軟性を持つ，ハイパクロスバ・ネットワーク(以下HXBと略)[1]を取り上げる．HXBは一種の閉鎖型多段クロスバ網で，PU間距離が非常に小さく，また通信チャネル数も多いため，大規模なシステムにおける複雑な転送パターンにおいても比較的高い交信性能を保つことができる．特に，転送パターンがランダムな場合，他の典型的な超並列向きネットワークに比べて高いスループットを実現することが可能である[3]．

HXBに対する過去の研究では，wormholeまたはstore&forward方式ルーティングにおける静的解析と，ランダム転送時の動的解析が行われてきた．これらの研究では静的な経路選択および単純なflit制御に基くwormholeルーティングが対象となってきたが，これらのルーティング手法を改良することにより，さ

† 筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

らにいろいろなパターンの転送においてその性能が向上すると考えられる．本研究では，このような性能向上の手法として以下の2通りを考える．

・virtual-channelを用いた適応型ルーティングの導入

・virtual-cut-throughの導入

これらの手法は基本的に独立なものであり，本論文ではこれらをそれぞれ独立に評価する．適応型ルーティングに関しては，ネットワーク中でのメッセージ間のデッドロック・フリーを保証するルーティング方式を提案し，それをHXBに導入した場合のスループットの向上について評価する．また，virtual-cut-throughの導入に関しては，HXBが得意とする比較的長いメッセージ・パケットの転送を行う場合に，従来のvirtual-cut-through方式をさらに改良したバッファリング方式を提案し，それをHXBに応用した場合について評価する．いずれの場合もランダムな転送を行った場合を対象とし，計算機シミュレーションによって評価する．

2. ハイパクロスバ・ネットワーク

HXBはn次元のbinary-hypercubeの各次元方向のPU数を2からm(>2)に拡張したものである．さらに，このmは，各次元ごとに任意のサイズをとることができる（このサイズが全次元方向で等しいものはbase-m n-cubeとして知られている[2]）．第i次元方向のPU数をm_iとすると，その次元方向のm_i台のPUは1つのクロスバ・スイッチ（$m_i \times m_i$）によって結合される．各PUはこのような各次元方向のクロスバ・スイッチに対するリンクを次元数nだけ持つ．このようにすると，総PU数は$\prod_{i=1}^{n} m_i$となる．また，クロスバ・スイッチで直接結合されていないPU間では最大nホップの転送を繰り返すことによってメッセージ転送を行うことができる．

図1に4×4×4構成の3次元HXBを示す．図中，PUとクロスバ・スイッチを結合している四角は，wormholeルーティングを行うためのルータで，エクスチェンジャ（EX）と呼ばれる．これは小規模なクロスバ・スイッチで，X，YまたはZ方向のクロスバ・スイッチ（XB）とPUの接続，あるいは各XB間の接続を行う．XB間のメッセージ転送を自動的に行うことにより，中継PUを介さない，wormholeルーティングが実現される．以下，X，Y，Z方向のXBをそれぞれX-XB，Y-XB，Z-XBと略す．

過去の研究の結果，HXBは数千台規模の並列システムにおけるネットワークとして，各種転送パターンおよびランダム転送における高い転送スループットを持つことが知られている[3,4]．しかし，これらの評価においてはメッセージ・ルーティングが単純であり，以下のような問題点があった．

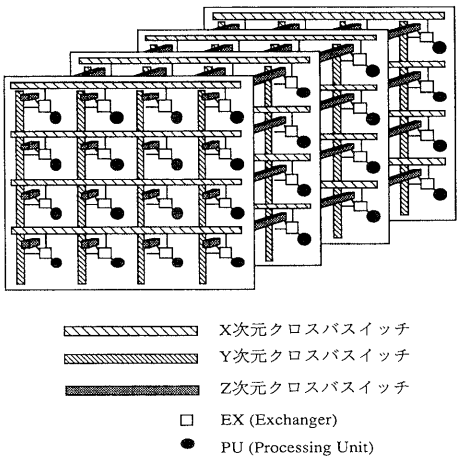


図1 3次元HXB（4×4×4）
Fig. 1 3 dimensional HXB (4×4×4).

・経路決定アルゴリズムが単純な固定ルーティング方式（転送すべき次元の順番が常に一定）である．このため，メッセージが衝突した場合，他の空いている経路を利用せず，その場でブロックしてしまい，結果的にネットワークのスループットが低下する．

・メッセージ転送方式として単純なwormholeまたはstore&forward方式が用いられている．前者ではメッセージがブロックされた時，その後続部分がネットワークの他のチャネルをブロックしてしまうため，余計な衝突が生じる．また後者では転送のホップ数が大きくなった場合，全体の転送レイテンシが大きくなってしまう．

本論文ではこれらの問題点を改善する2種類の手法を導入し，その評価を行う．

3. 性能向上の手法

前章で示した問題点の改善手法として以下の2種類の方法を導入する．

3.1 virtual-channelによる適応ルーティング

単純な固定ルーティングに代わる経路決定法として適応ルーティング（adaptive routing）を実現する方法を提案する．従来のHXBにおいて固定ルーティングが用いられていた最大の理由は，HXBにおいて単純に適応ルーティングを用いて経路を動的に決定すると，複数のメッセージ間のチャネルの連鎖に閉ループ

ができてしまい，そのままwormholeルーティングを行うとdeadlockを生じるためである．そこで，ここではvirtual-channelの導入によってdeadlock-freeを保証する適応ルーティングを提案する．なお，ここでは迂回ルーティングは行わないものとする．すなわち，転送においては無駄なXBを通過せず，かつ各次元のXBを通過する回数はたかだか1回とする．

まず，n次元HXBにおいて，各EXおよびXBの出力部にn枚のバッファを用意し，一つの物理チャネル（リンク）をn本のvirtual-channelとして利用する．virtual-channelの割り当ては，そのメッセージがこれまでいくつのXBを経由したか（いくつの次元を通過したか）によって決定する．n次元HXBではメッセージは最大n個のXBを経由して転送されるため，n本のvirtual-channelを用いれば，複数のメッセージのチャネル連鎖中に閉ループができることはなくなる．例えば，3次元HXBにおいてX→Y→Zという経路で転送されるメッセージaと，Y→Z→Xという経路で転送されるメッセージbは，中継点のEXの座標が一致した場合，単純な物理チャネル上では閉ループを構成する可能性がある．しかし，上述のvirtual-channelを用いるとメッセージaは$X_1 \rightarrow Y_2 \rightarrow Z_3$という経路で転送され，メッセージbは$Y_1 \rightarrow Z_2 \rightarrow X_3$という経路で転送されるため，閉ループはなくなる（X, Y, Zの添え字はvirtual-channel番号を表す）．よってdeadlock-freeな適応ルーティングが可能となる[6]．以下に証明を示す．

ネットワーク中の全virtual channelを$C_{v,d,i,s}$で表すとする．ここで，vはvirtual番号，dは次元番号（1, 2, …, n），iは次元dにおけるディスティネーションEXの番号，sはそのチャネルのソースEXの番号である（iおよびsはn次元空間のアドレスであるが，各次元を順序づけし，線形化されているものとする）．メッセージは各XBを通過する時，任意の次元（ただし過去通過していないもの）を選択でき，その際，v＝現在のステップ番号であるようなチャネルを使用するものとする．このためvは単調増加し，結果として上述のチャネル番号$C_{v,d,i,s}$は単調増加となるので，channel dependency graphにサイクルがなくなる．よってこのルーティングはdeadlock-freeである．（証明終わり）

以上の手法により，この適応ルーティングがdeadlockを生じないことは保証されるが，HXBは間接網であるため，経路がbusyであるかどうかをチェックする方法は直接網の場合より複雑になる．HXBにおける経路選択はEX上で行われることになるが，たとえここである次元方向へのチャネルがfreeだったとしても，その先のXBにおける出力チャネルがbusyであれば，メッセージは結局ブロックされてしまう．

そこで，EXはXBの全出力チャネルの状態を監視し，EXからそのチャネルまでのリンクがすべてfreeの場合のみメッセージを転送するように制御する必要がある．さらに，実際のwormhole転送ではメッセージの先頭がXBの出力チャネルに到達するまでに遅延が生じるため，それまでの間はそのチャネルを予め予約（reserve）するような制御も必要となる．HXBはPU台数の増加に応じて次元数と各次元のサイズを調整できるため，各次元のXBのサイズはそれほど大きくない（例えば16×16程度）．例えば，16 bitのデータ・パスに対し，1 bitのbusy/free状態を示す制御線を追加することは，それほど大きなコスト増にはつながらないと考えられる．

3.2 virtual-cut-throughによるブロック率の低減

単純なwormholeルーティングによるHXBの2つ目の問題点を解決するためにvirtual-cut-through方式による転送を考える．これはwormholeとstore&forwardの両者の利点を併せ持つ転送方式で，基本的にはwormhole転送を行い，もしメッセージの先頭で衝突が起こりブロックされた場合は，そのノード（EXまたはXB）中にメッセージを溜め込み，少なくともメッセージの後続部分が他のメッセージと衝突するのを防ぎ，全体的なブロック率を低減させる手法である[5]．

virtual-cut-throughによって転送効率を上げるためには，メッセージは適当なサイズのパケットに分割される必要がある．これは，ノードへのstoreが速やかに行われるようにするためと，ノードに用意すべきバッファがあまり大きくならないようにするためである．メッセージは次のノードに進む際，そのノードの出力チャネルがfreeであるか，またはそのノードのバッファがfreeであればそのノードに進むことができる．ここで，バッファがfreeであるというのは，そのノードのバッファに現在store中の他のメッセージが存在せず（一般に，複数の入力チャネルから1つのバッファへのstoreが行われることは困難である），かつそのメッセージを完全に収容できるだけの空きがある場合を指す．もし次のノードの出力チャネルがfreeであれば，メッセージはそこに進み，さらに次の段階でその先のノードを窺うことができる．もし経路上のすべての出力チャネルがfreeであれば，メッセージは結果的にwormhole方式で転送される．

以上が基本的な virtual-cut-through 方式の転送であるが，ここに1つの問題点がある．それは，もしメッセージが先に進めない場合，メッセージはそのノードに store されるが，store している最中に出力チャネルが free になった場合にどうするかである．そのような状況でも，そのメッセージの store が完了してから先に進むようにすると，制御方式は全体的に単純化されるが，store の回数が多くなり性能が低下する可能性がある．これに対し，たとえメッセージを store している最中でも，出力チャネルが空けば直ちに先頭からの出力を開始するようにする方法が考えられる．この場合，バッファは先頭から次々に出力されると同時に，後続部分が前段のノードから次々送られてくるため制御が複雑になる．しかし，1つのバッファにデータが保存される時間が最低限で済むため，全体的な性能は向上すると考えられる．そこで，今回はこの両者の方法を比較し，評価する．前者の方法を static buffering，後者の方法を dynamic buffering と呼ぶことにする．

dynamic buffering 方式を採用することにより，全体的なスループットはかなり向上すると思われるが，バッファリングを工夫することにより，さらに大きな性能向上が期待できる．static buffering および dynamic buffering の両方式では，あるノードに存在しているメッセージは，経路上の次のノードのバッファにそのメッセージ自体を完全に格納し得るだけのバッファの空きがある場合のみ，メッセージを転送する．しかし，この制約を排除して，たとえ1 flit 分であっても，とにかくバッファに空きがありさえすればメッセージの転送を開始するようにすると，メッセージがバッファ中に留まる時間がさらに短縮される．この方式では1つのメッセージのいろいろな部分が異なるノードのバッファに格納される可能性があり，制御が一層複雑になるが，メッセージは少しでも先に進める場合はどんどん進むため，dynamic buffering 方式よりさらにスループットが向上する．そこで，このバッファリング方式も前の2種類と同様に評価する．この方式を flexible buffering 方式と呼ぶ．

4. 評　　価

適応ルーティングと virtual-cut-through の導入は基本的には独立な概念であり，両者を組み合わせることによってさらなる性能向上が期待される．しかし，ここでは基本的な性能評価を行うため，それぞれを独立に評価する．

評価は計算機シミュレーションを用いて行う．一般的なメッセージ衝突状態を想定するため，ここではランダム転送を基本とした．モデルとしては，システム中の全 PU がランダムに選んだ相手 PU に対して一定長のメッセージを転送し，転送が終了したら再び相手をランダムに選んで転送を繰り返す，という状況を想定した．PU に到着したメッセージは割り込み処理によって適宜受信されるものとする．通常，メッセージの送受信は DMA コントローラによって行われるので，ここでは送信処理と受信処理が同時にできるものとした．また，1 PU から同時に送信されるメッセージの数に関しては，適応ルーティングを用いた場合と，virtual-cut-through を用いた場合では若干異なる．これは，それぞれのルーティング方式の特性の違いに依存するもので，詳細は各方式の評価の部分でそれぞれ述べる．

システムの規模としては 4096 PU または 1024 PU の3次元 HXB（16×16×16 または 8×8×16）を対象とした．XB および EX の各チャネルのバンド幅は1 flit/clock に正規化し，メッセージ長は flit 単位で表すものとする．3次元であるため，PU 間距離の最大値は3であるが，実際には途中に最大4つの EX と最大3つの XB が経由されるため，PU → PU の最大距離は8となる．

評価は1 PU 当りのメッセージ転送のスループットを，理想的な無衝突転送の場合を1として正規化して行った．ここで，理想的な無衝突転送とは，転送距離に依存するレイテンシと衝突による待ち時間がすべて0である場合を指す．この場合，L clock のシミュレーションにおいて1 PU から転送されるメッセージの総量は L flit となる．実際には 10000 clock のシミュレーションを行ったので，スループットはこの間に PU が転送した総メッセージ量を 10000 で割ったものである．評価結果の各グラフは横軸にメッセージ長を取り，各メッセージ長におけるスループットを縦軸に取っている．

4.1 適応ルーティングの評価

まず，適応ルーティングの転送性能の評価方法について述べる．ここではシステムの規模を 4096 PU とし，メッセージの転送方式は wormhole 方式とした（一般的に，HXB においては store&forward 方式より wormhole 方式の方が高いスループットを示す）．3.1 節に述べたように，全通信チャネル（PU ↔ EX，EX ↔ XB 間）は各々3枚のバッファを用いて3重に仮想化され，この上で迂回なしの適応ルーティングをシミュレートする．また，ある EX から複数方向の XB へのチャネルが free な場合は，X，Y，Z の優先順位で方向を決定することとする．

送信PUの振舞いについては以下のような仮定を置いた．適応ルーティングでは，PUから送出されたメッセージはX, Y, Zのいずれの次元方向にも移動することができるため，あるEXにおいてX次元方向に転送されているメッセージがブロックされたときに，そのEXを通過する後続のメッセージをvirtual-channelにより他の次元方向に転送することが可能となる．このことを考慮して，ここではPUからのメッセージ送出に関して以下のように仮定した．各PUはランダムに選んだ相手に一定長のメッセージを送る．1メッセージの転送要求をDMA回路に対して発行した後，一定時間の内部処理を行う(今回はこの長さを50クロックとする)．内部処理が終了して次のメッセージを送出する際，もしPUからネットワークへの出力チャネルがbusyのときは，それがfreeになるまで内部処理を継続し，freeになった時点で次のメッセージの転送を開始する．全PUは以上の動作を繰り返す．

適応ルーティングにおける評価結果を**図2**に示す．図2の点線は固定ルーティング(FIX)，実線は適応ルーティング（ADP）の場合をそれぞれ示している．またFIX 1とADP 1が，通常のHXBにおける，従来の固定ルーティングと今回提案した適応ルーティングの場合をそれぞれ表す．なお，固定ルーティングにおいては，経路決定はX, Y, Zの順に行われるものとした．

まず，固定ルーティング，適応ルーティングのどちらの場合もメッセージ長が長くなるにつれてスループットが増加していることがわかる．これは以下のような理由による．PUが1メッセージを送出するのに要する時間は，メッセージの先頭が相手PUに到着するまでにかかるレイテンシと，それに続くメッセージ本体が転送され終わるまでの時間（今はバンド幅が正規化されているため，これはメッセージ長そのものに等しい）の和となる．また，レイテンシはPU間の距離とネットワーク中で他のメッセージと衝突した際のブロック時間の和となる．HXBにおけるwormhole転送では，ブロック時間はメッセージ長に対してほぼ線形に増加するため[3]，メッセージ長が非常に短いときはブロック時間も小さくなり，1メッセージの送出に要する時間がPUの内部処理時間（50クロック）より小さくなる．この場合，メッセージの送出間隔がある程度広がり，ネットワーク中の総メッセージ転送量は少なくなる．それに対して，メッセージ長が長くなるとブロック時間も大きくなり，1メッセージの送出に要する時間がPUの内部処理時間より大きくなる．この結果，以前のメッセージを完全に送出し終らないうちに次のメッセージの送出要求が生じ，ネットワーク中にメッセージが満たされ，総メッセージ転送量が増加し，結果的にスループットが向上する．

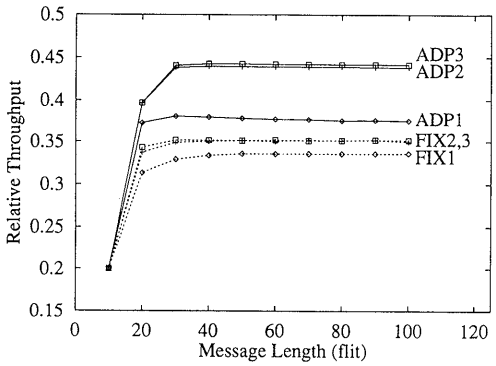


図2 適応ルーティングによる性能向上
Fig. 2 Performance improvement with adaptive routing.

次に，固定ルーティングと適応ルーティングを比較する(FIX 1とADP 1を参照)．メッセージ長が短いときはネットワークが空いているので，固定ルーティングと適応ルーティングで性能差は見られない．しかし，メッセージ長が長くなりネットワークが混雑してくると，適応ルーティングの場合は固定ルーティングに比べて最大で約20%程度性能が向上している．しかし，3次元の場合，固定ルーティングではX → Y, Y → Z, X → Zの3通りのチャネルしか利用していないが，適応ルーティングでは6通りすべてのチャネルを利用できる．このことを考えると，この程度の性能向上は十分とは言えない．

適応ルーティングにおいては，PUから送出されたメッセージはX, Y, Zのいずれの次元方向にも移動できることから，PUは最大で3つのメッセージを同時に送出することが可能となる．さらに，PUに到着するメッセージはX, Y, Zの各次元方向から同時に到着する可能性が高くなる．これらのことから，PUとEX間のバンド幅がボトルネックになっていると予想される．そこで，PUとEX間の物理的なバンド幅をネットワーク中のバンド幅の2倍，あるいは3倍に増加させることを考える．図2における凡例の数値（1, 2, 3）は，ネットワーク中のバンド幅に対する，PU ↔ EX間のバンド幅の比率を表す．

PU ↔ EX間のバンド幅を増加させると，適応ルーティングにおいてはかなりの性能向上が見られるが，固定ルーティングではほとんど性能が向上しない．固定ルーティングにおいては，X → Y → Zの順でルーティングが行われるため，PUから送出されたメッセージはいつでも最初にX方向に移動する．これにより，ほとんどのメッセージはEXにおけるX-XB方向へ

の出力において衝突し，そこでブロックされてしまう．逆に，受信 PU から見ると到着するメッセージはほとんどの場合 EX の Z-XB 方向から送られてくる．以上の理由により，固定ルーティングではネットワーク中の特定の次元方向の XB がボトルネックとなるため，PU ↔ EX 間のバンド幅の増強はスループットに影響しないものと考えられる．

適応ルーティングにおいて，PU ↔ EX 間のバンド幅をネットワーク中の 2 倍にしたとき，さらに性能は向上する（固定ルーティングに比べて約 30%の性能向上）．これは，先述したように，PU からネットワークに送出された複数のメッセージが異なる次元方向に移動することができるため，衝突する可能性が小さくなっているためである．また，受信 PU にとっても，メッセージはどの次元方向からも到着するため，PU と EX 間の 2 倍のバンド幅が有効に利用されている．それに対して，PU ↔ EX 間のバンド幅をネットワーク中のそれの 3 倍に増強した場合では，それほど大きな性能向上は見られない．これはランダム転送で 1 PU に同時に 3 方向からメッセージが到着する可能性が比較的低いことと，ネットワークが非常に混雑しネットワーク中に溜っているメッセージが多くなっていることの 2 つによるものと考えられる．

以上，適応ルーティングにおいて PU ↔ EX 間のバンド幅をネットワーク中のそれよりも増強することは大幅な転送性能の向上に寄与することがわかった．実装上は，EX における全入出力のうち PU に向いたリンクだけを増強すればよく，さらに，PU と EX は一対になっているため，実装上の物理的な距離が非常に近いことが予想されるため，その間のみ物理線の幅あるいは転送周波数を 2 倍に増強することは比較的容易であると考えられる．

4.2 virtual-cut-through 転送方式の評価

virtual-cut-through 方式の転送性能の評価を，ランダム転送におけるスループットによって示す．ここでは，メッセージの経路決定は固定ルーティング方式で行われている．PU からのメッセージ送出のタイミングは，4.2 節に述べた適応ルーティングにおける評価と若干異なっている．

適応ルーティングの場合は，ソース PU の EX からのメッセージ送出の方向が最大 3 通り存在するが，固定ルーティングの場合は，X 方向への転送が必要な場合，常に最初に X 方向への送出が行われる．3 次元空間でのランダム転送では，ほとんどのメッセージは X 方向への転送が必要となる（1024 PU の場合で全体の約 88%）．このことから，たとえ PU から複数のメッセージを EX に送出できたとしても，結局 EX → X-XB へのパスがボトルネックになると考えられるため，適応ルーティングの評価で行ったような，1 PU からの複数のメッセージの同時送出や，PU ↔ EX のバンド幅を増やすような工夫はスループットの向上に貢献しない．以上の理由により，virtual-cut-through 方式の評価においては，PU は 1 つのメッセージを送出した後，それが EX に完全に転送され終わるまでアイドル状態となり，PU → EX のパスが free になった段階で次のメッセージを送出するものとする．

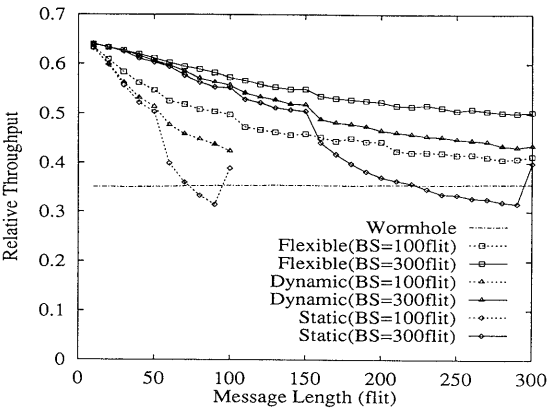


図 3 virtual-cut-through による性能向上
Fig. 3 Performance improvement with Virtual-Cut-Through.

シミュレーションによるスループットの評価を**図 3** に示す．ここではシミュレータの実行時間の都合上，1024 PU の場合を評価する．各ノードにおけるバッファ容量として，300 flit の場合と 100 flit の場合の 2 通りを評価した．

図中，一点鎖線は通常の wormhole 転送におけるスループットを示している．実線および点線が，それぞれバッファ容量（BS）が 300 flit の場合と 100 flit の場合の virtual-cut-through 方式を示している．また，プロットされた点の形が“□”のものが flexible，“△”のものが dynamic，“◇”のものが static の各 buffering 方式をそれぞれ示している．

全体的にメッセージ長が長くなるにつれスループットが低下することがわかる．予想されたように，基本的に flexible 方式が最も高いスループットを示し，次いで dynamic 方式，static 方式の順になっている．また，ノードにおけるバッファ容量が大きい程スループットが向上していることがわかる．

wormhole 方式のスループットがメッセージ長に依存せずほぼ一定である点が，適応ルーティングの評価において示した固定ルーティングの結果と異なってい

るが，これは前述したように，PUからのメッセージ送出のタイミングが異なっているためである．

以下に個々のbuffering方式の特徴を述べる．まずstatic bufferingの場合であるが，メッセージ長がバッファサイズに比べ十分短かい場合はある程度性能が向上するが，メッセージ長がバッファサイズの半分を越えたあたりから急激に下降する．これは，メッセージが次のノードのバッファに入り切らなければそのノードに進まないという条件によるもので，バッファサイズの半分を越えた段階では，格納すべきバッファがない状態でメッセージ長が大きくなっているため，ブロック率が高くなり性能が低下する．注目すべきは，メッセージ長がバッファサイズと等しい点では若干持ち直すという点である(V字特性)．この現象については以下のように説明できる．

今，長さM_aのメッセージaが次のノードを窺っている．次のノードのバッファには長さM_bのメッセージbが溜っているとする．バッファサイズをBとする($B>M_a$, M_b)と，メッセージbがさらに先のノードに転送される時，そのバッファ中の残りの容量が$B-M_a$となった時点で，メッセージaはそのノードに進むことができる．しかし，メッセージaの先頭がバッファに入った段階では，まだメッセージbの残りがそのノードに残っているため，結局メッセージaはそのノードで必ず待たされることになる．メッセージ長がバッファ容量の半分を越えると，この状況が発生し，それ以上メッセージ長が長くなるとメッセージのブロック時間が増えるだけなので，全体的なスループットは低下する．

しかし，もし$M_a=M_b=B$だとすると，メッセージaはメッセージbが完全にノードから出ていくまではそのノードに進めないが，そのノードに進んだ時はその前にメッセージbは存在しないため，その先でメッセージbがブロックしていなければ，メッセージaはそのままwormhole的に進むことができる可能性がある．このため，メッセージ長がバッファ容量に等しい場合は，上述のような無駄なstoreが減る可能性が高く，スループットが向上するものと思われる．

dynamic buffering方式ではこの問題が改善されている．メッセージは，バッファにstore中であってもその先のノードがfreeになれば，storeの完了を待つことなしに転送が開始されるため，上述の状況になってもスループットは低下しない．static bufferingの場合と同様に，メッセージ長がバッファサイズの半分を越えると性能は低下するが，上の理由により，低下はそれほど大きくなく，またV字型の特性も生じない．結果としてwormhole方式の性能を常に上回り，メッセージ長がバッファ容量に等しい場合でも約15%程度の性能向上が達成できる．

また，flexible buffering方式になると，スループットはさらに向上し，dynamic buffering方式に比べ約10%から20%の性能向上が見られる．これは，前方のバッファに少しでも空きがあればどんどんメッセージを送り込むという制御方式により，メッセージが先に進める可能性が高くなることでスループットが向上しているためである．さらに注目すべき点は，flexible buffering方式ではバッファ容量を越えるような長さのメッセージの転送が可能になっている点である(BS=100 flitの場合を参照)．これは，次のノードにそのメッセージが入り切るだけの余裕がなくてもメッセージをforwardしているため，メッセージ長とバッファ容量の間に何の制約もなくなっているためである．このことから，flexible buffering方式は，実装上の制約からバッファ容量がそれほど大きく取れない場合でも，HXBが本来得意とする比較的長いメッセージの転送において，その性能向上に十分貢献すると言える．

5. おわりに

本論文では，従来単純な固定ルーティングとstore&forgetあるいはwormhole方式の転送のみを対象として評価されてきたハイパクロスバ・ネットワークの転送性能に関し，virtual-channelを応用した動的な適応ルーティングと，virtual-cut-throughによる転送性能向上の手法を適用し，計算機シミュレーションを通じてその性能を評価した．

まずメッセージ転送における経路選択方式に関しては，n次元のハイパクロスバ・ネットワークにおける最大転送ホップ数がnであることを利用し，各通信チャネルをvirtual-channelを用いてn重に仮想多重化することにより，deadlock-freeな適応ルーティングを実現する方法を提案した．このルーティング手法を用いることにより，基本的なランダム転送において，ネットワークの転送スループットが固定ルーティングの場合に比べて約20%向上することが計算機シミュレーションにより確認された．さらに，送信PUから同時に多数のメッセージが送出される場合，PUとEX間のバンド幅がボトルネックになることがわかり，その点を改善した結果，固定ルーティング方式に比べ，最大で約30%の性能向上が確認された．

一方メッセージの転送方式に関しては，従来評価されてきたwormhole方式に対し，古典的なvirtual-cut-through方式を適用した場合，特にメッセージ長

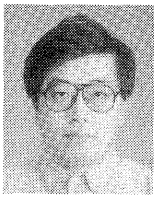
が短いランダム転送において大幅なスループットの向上が見られた．また，中継点のXBあるいはEXにおけるバッファへのflitの入出力を動的に処理するようbuffering方式を改良することにより，スループットはwormhole方式に比べ大幅に向上し，メッセージ長がバッファ容量に等しい場合でwormhole方式に比べ約40%の性能向上が達成できることがわかった．また，この場合メッセージ長がバッファ容量による制約を受けないため，任意の長さのメッセージを1パケットとして転送可能で，比較的長いメッセージの転送に向いているハイパクロスバ・ネットワークにおいて実用的であろう．

今回の評価ではこれら2つの方式を独立に扱ったが，これらの方式は互いに相反するものではない．今後はこれらの方式を融合した場合の性能評価と，ハードウェアコストを加味した場合のトレードオフについて評価していく予定である．また，適応ルーティングを実現する際，EXからその先のXBのチャネルの状況を調べ，さらに必要に応じてそれをreserveするための具体的な機構についても考察していく予定である．

謝辞　本研究を進めるに当り貴重な御意見を頂いた，筑波大学西川博昭助教授に深く感謝します．なお，本研究の一部は文部省科学研究費(奨励(A)05780225)および創成的基礎研究費(05 NP 0601)の補助による．

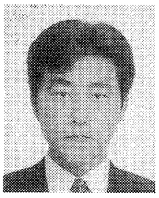
参考文献

1) 田中輝雄, 面田耕一郎：高並列計算機による空気力学シミュレーションの構想，第8回航空機計算空気力学シンポジウム論文集，pp. 99-108 (June 1990).
2) 鈴岡　節，田邉　昇，中村定雄，藤田純一，小柳滋：並列AIマシンProdigyの相互結合網の評価，信学論D，Vol. J 71-D, No. 8, pp. 1496-1501 (1988).
3) 朴　泰祐，斉藤哲也，板倉憲一，中澤喜三郎，中村宏：ハイパクロスバ・ネットワークの性能評価，信学技報CPSY 93-40 (1993).
4) 保田淑子，藤井啓明，田中輝雄，稲神泰弘：ハイパクロスバネットワークの通信性能評価，信学技報CPSY 93-25 (1993).
5) Kermani, P. and Kleinrock, L.: Virtual Cut-Through: A New Computer Communication Switching Technique, *Computer Networks*, Vol. 3, No. 4, pp. 267-286 (1979).
6) Dally, W.J. and Seitz, C.L.: Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, *IEEE Trans. Computer*, Vol. C-36, No. 5, pp. 547-553 (1987).



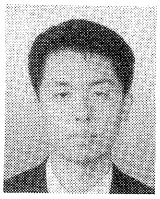
朴　泰祐（正会員）

昭和59年慶應義塾大学工学部電気工学科卒業．平成2年同大学大学院理工学研究科電気工学専攻後期博士課程修了．工学博士．昭和63年慶應義塾大学理工学部物理学科助手．平成4年筑波大学電子・情報工学系講師，現在に至る．超並列計算機アーキテクチャおよび並列処理言語・アルゴリズムの研究に従事．電子情報通信学会会員．



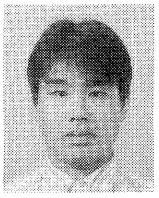
曽根　猛（学生会員）

1971年生．1994年筑波大学第三学群情報学類卒業．現在同大学院工学研究科電子・情報工学専攻に在籍中．超並列計算機アーキテクチャ，並列処理に関する研究に従事．



三島　健（学生会員）

平成6年筑波大学第三学群情報学類卒業．現在同大学院工学研究科電子・情報工学専攻修士課程に在籍中．並列計算機のネットワークの研究に従事．

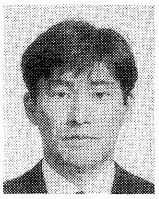


板倉　憲一（学生会員）

1969年生．1993年筑波大学第三学群情報学類卒業．1995年同大学院工学研究科前期博士課程を修了，現在同研究科後期博士課程に在籍中．並列計算機アーキテクチャの研究に従事し，性能評価方法などに興味を持つ．

中澤喜三郎（正会員）

昭和30年東京大学工学部応用物理卒業．昭和35年同大学院数物系博士課程応用物理修了．同年日立製作所入社．TAC，HITAC 5020，E/F，8800/8700，M-200 H/280 H，680 H，S-810等，超大型コンピュータ・スーパーコンピュータの開発に従事．平成元年より筑波大学電子・情報工学系教授，計算物理学センター向きの超並列処理システムCP-PACSの研究開発に従事．工学博士．IEEE，ACM各会員．平成5年度本学会論文賞受賞．



中村　宏（正会員）

昭和38年生．昭和60年東京大学工学部電子工学科卒業．平成2年同大学院工学系研究科電気工学専攻博士課程修了．工学博士．同年筑波大学電子・情報工学系助手．平成3年同講師，平成7年同助教授，現在に至る．計算機アーキテクチャ，並列処理，計算機の上位レベル設計支援に関する研究に従事．本学会平成5年度論文賞，本学会平成6年度山下記念研究賞各受賞．電子情報通信学会，IEEE各会員．