Vol. 34 No. 4 情報処理学会論文誌 Apr. 1993

トーラスネットワークにおける最適全対全通信方式

堀江健志† 林憲一†

並列計算機では，通信性能がシステム全体の性能に大きく影響を与える．そのなかで，全対全通信は，マトリクスの転置，FFT など多くの応用問題に繁頻に使用される通信パターンである．本論文では，トーラスネットワーク上での最適な全対全通信方式を提案する．本方式は，多次元トーラスや長方形に対しても適用することができる．本論文では，さらに，全対全通信性能に基づいたトポロジの比較を行うとともに，本方式を実際の並列システムに実現し本方式を検証する．

All-to-All Personalized Communication on a Wraparound Mesh

Takeshi Horie† and Kenichi Hayashi†

Some application programs on distributed memory parallel computers often require an all-to-all personalized communication pattern, where each processor must send a different message to each other. In this paper, we propose optimal data movement methods for all-to-all personalized communication on wrap-around (toroidal) meshes, assuming circuit-switched or wormhole routing and one-port communication model. Using this method, the number of phases on wrap-around meshes is shown to be half of that on meshes. This method can also apply to wrap-around multi-dimensional meshes. Experimental results on the AP1000 show that this method provides efficient all-to-all personalized communications.

1. はじめに

ハイパーキューブ，トーラス，メッシュなどのネットワークで接続された並列計算機では，通信の性能がアプリケーションに大きく影響する．そこで，ネットワークの性能を最大限に引き出す通信アルゴリズムを実現する必要がある．多くの応用で頻繁に使用される通信パターンとして，以下の4種類が知られている[1]．(1)一対全放送通信—一つのプロセッサが一つのメッセージをすべてのプロセッサに送信する．(2)一対全通信—一つのプロセッサが送信先プロセッサごとに異なるメッセージをすべてのプロセッサに送信する．(3)全対全放送通信—すべてのプロセッサがそれぞれのメッセージをすべてのプロセッサに送信する．(4)全対全通信—すべてのプロセッサが送信先プロセッサごとに異なるメッセージをすべてのプロセッサに送信する．これら4種類の通信の中では，全対全通信が最も多くの通信転送量を必要とする．この全対全通信は，マトリクスの転置，二次元 FFT, ADI など多くのアプリケーションで頻繁に使用される通信である．例えば，二次元配列が複数のプロセッサに分散されている場合，同一行のデータを一つのプロセッサにマッピングした状態から，同一列のデータを一つのプロセッサにマッピングする状態に分散しなおすとき全対全通信が発生する．

多くの研究者が，全対全通信をハイパーキューブネットワーク上で実現する最適な方法[1)-6)]を提案している．Johnsson らは，ハイパーキューブネットワーク上でのストアアンドフォワード（パケット交換）方式にもとづいた最適な全対全通信のアルゴリズムを提案している[1]．しかし，ストアアンドフォワード方式は，隣接プロセッサ間で行うストアアンドフォワードに必要な遅延時間が大きく，高い性能は得られない．また，すべてのチャネルで通信が行われているようにするために，プロセッサが隣接しているすべてのプロセッサからメッセージを送信あるいは受信しなければならない（マルチポート通信）．このようなプロセッサとネットワークとのインタフェースを実現することはハードウェア上困難である．したがって，プロセッサ

† (株)富士通研究所並列処理研究センター
Parallel Computing Research Center, FUJITSU LABORATORIES LTD.

が一つのメッセージの送信と一つのメッセージの受信しか行わない（一ポート通信）方式が現実的である．ただし，一ポート通信の全対全通信の時間は，マルチポート通信に比較して，ハイパーキューブの次元に比例して大きくなる[1]．

ストアアンドフォワード方式の通信性能の問題を解決するルーティング方法として，回線接続あるいはワームホールルーティング[7]がある．この方法では，通信の遅延時間がパスの長さにほぼ関係なくメッセージの長さだけに比例することになる．第二世代のメッセージパッシング並列計算機，iPSC/2[8]，nCUBE/2 では，回線接続[9]あるいはワームホールルーティングを使用したハイパーキューブネットワークを使用している．武と Boppana は，ハイパーキューブネットワーク上での回線接続方式にもとづいた最適な全対全通信のアルゴリズムを提案している[2),3)]．プロセッサとネットワーク間を一ポート通信としたとき，ハイパーキューブ上では，プロセッサ台数を P とすると，P 回のフェーズ（自プロセッサへの通信を含む）で実現できる．さらに，Ho と Bokhari は，通信性能と通信セットアップ時間の両方を考慮した効率のよい全対全通信のアルゴリズムを提案し，iPSC/2 あるいは iPSC/860 での性能を測定している[4)~6)]．

一方，最近では，トポロジとしてハイパーキューブではなくメッシュを使った並列計算機が登場してきている．Ametek[10]，インテル社の Touchstone プロジェクトは二次元メッシュ，J-Machine[11] は三次元メッシュのネットワークを持ち，ルーティングはワームホールルーティングを使用している．Dally は，ハードウェアのコストを考慮したときのネットワーク性能として二次元あるいは三次元のメッシュネットワークの優位性を指摘している[7]．このメッシュネットワーク上での最適な全対全通信アルゴリズムが Scott により提案されている[12]．

しかし，回線接続あるいはワームホールルーティングを使用したトーラスネットワーク上での，一ポート通信を仮定した最適な全対全通信方式はいままで提案されていなかった．

本論文は，メッシュの端と端が接続されているトーラス結合ネットワーク上での最適な全対全通信方式を提案する．トーラスネットワークでは，メッシュネットワークの2倍の性能を実現することができる．例えば，二次元トーラスの全対全通信は $P^{3/2}/8$ フェーズ，二次元メッシュの全対全通信は $P^{3/2}/4$ フェーズ[12]必要になる．本方式は，一次元トーラスから多次元トーラスのネットワークに適用することができる．また，正方形だけではなく長方形のネットワークにも適用することができる．さらに，本論文では，接続チャネルが単方向あるいは両方向のどちらに対しても最適な方式についても述べる．なお，本方式では，プロセッサとネットワークとのインタフェースは一ポートを仮定している．

Dally は，*k-ary n-cube* ネットワーク上でのランダムトラフィックにおける通信性能をハードウェア量を考慮して比較している[13]．全対全通信は，最も通信負荷の高い通信パターンである．そこで，本論文では，本方式を用いたときの全対全通信性能をもとに，トーラス，ハイパーキューブなどの *k-ary n-cube* ネットワークの性能比較を行う．Dally は，システム全体を二つに分けて，二つのシステムを横切るネットワークのワイヤ数をハードウェア量としている．これは，すべてのプロセッサを一つのボードに搭載することを仮定したネットワークコストである．ここでは，実際のハードウェアを実現するときに問題になると思われる，（1）1プロセッサに接続されるワイヤの数，（2）1ボード（複数のプロセッサが搭載されているとき）に接続されるワイヤの数をハードウェア数として考慮してトポロジの性能比較を行う．

現在，トーラスネットワークを持つ並列計算機としては，AP1000[14] と iWarp[15] がある．AP1000 は，二次元トーラス構造のネットワークを持ち，そのルーティング方式は，基本的にはワームホールルーティングを使用している．本論文では，本方式を分散メモリ型の並列計算機 AP1000 上に実現しその性能を測定し，本方式を検証する．

本論文では，まず，第2章で，トーラスネットワーク上での最適な全対全通信アルゴリズムを提案する．第3章では，全対全通信性能を対象に *k-ary n-cube* ネットワークの比較を行う．第4章では，全対全通信性能を並列計算機 AP1000 上に実現し，本方式を検証する．

2. 全対全通信アルゴリズム

2.1 モデル

本論文で扱う通信モデルについて述べる．プロセッサは，一メッセージの送信と一メッセージの受信とを同時に行うことができるものとする．通信チャネルは，両方向（同時に両方向転送可能）チャネルと単方

向チャネル（同時には両方向転送不可能）の二つのモデルを扱う．単方向チャネルを持つネットワーク上で両方向チャネルのアルゴリズムを実現するときは，半分の性能になる．ルーティングは，回線接続あるいはワームホールを仮定する．

全対全通信の時間は，転送スタートアップ時間を C_S，チャネルの転送レートを C_T，メッセージのサイズを m，転送フェーズの回数を L とすると次式で与えられる．

$$T_{aac}=L\left(C_S+\frac{m}{C_T}\right)$$

メッセージサイズが長いときは，m/C_T が主要因になるので，本章と第3章では，転送スタートアップ時間 C_S を無視して議論する．

2.2 下　　限

ここでは全対全通信の送受信フェーズ数の理論的下限を求める．隣合うプロセッサ間の距離をすべて1とすると，各プロセッサから送信したメッセージが目的のプロセッサに達するまでの移動距離はプロセッサの数で表現できる．この移動距離の平均 D にプロセッサの数 P，各プロセッサの送信するメッセージの数 S を掛け合わせれば，全対全通信に必要な通信路の数が求まる．この値を通信路の総数 N で割れば，各通信路での通信の負荷が均一とした場合の送受信フェーズ数 L が得られる．すなわち，

$$L=\frac{PDS}{N}$$

である．ただし，通信路は単方向チャネルのとき，隣合うプロセッサ間の通信路を1，両方向チャネルのとき2として計算する．

この値より少ないフェーズで送受信を行うプロセッサがあると，この値より大きいフェーズ数の送受信を必要とするプロセッサが現れるので，全プロセッサはこの値より少ないフェーズで全対全通信を終ることはできない．すなわちこの値が全対全通信に必要なフェーズの理論的下限となる．

2.3 一次元トーラス

2.3.1 理論的下限

上の議論に基づき一次元トーラスにおける全対全通信のフェーズの下限を求める．プロセッサの数を a $(=P)$ とするとメッセージの移動距離の平均は a が偶数のとき $\frac{a}{4}$，奇数のとき $\frac{a^2-1}{4a}$ となる．また通信路の総数は，単方向チャネルのとき a，両方向のとき $2a$ である．よって a が偶数のとき，単方向チャネルを用いた場合の送受信フェーズの下限 L_{1u} は，

$$L_{1u}=\frac{a\times\frac{a}{4}\times a}{a}=\frac{a^2}{4}$$

また両方向のときの下限 L_{1b} は，

$$L_{1b}=\frac{a\times\frac{a}{4}\times a}{4a}=\frac{a^2}{8}$$

となる．

2.3.2 下限を実現するアルゴリズム

次に下限を実現するアルゴリズムについて述べる．a を8以上の偶数とする．後で述べる特別な場合を除いて，各フェーズは四つのプロセッサ（A, B, C, D）を結ぶ一方方向のサイクルとなる．まず a 個のプロセッサを隣合う $a/2$ 個の2グループ，G_1, G_2 に分ける．G_1 の端のプロセッサをAとし，G_1 の残りのプロセッサの中からBを選ぶ．CとDは G_2 の中から，A，Bとそれぞれ $a/2$ 離れた位置に選ぶ．

(A→B→C→D→A) という一つのフェーズを表現するのに，選ばれた4点間の距離をそれぞれコロンで区切って，

$$\alpha:\beta:\alpha:\beta$$

のように表すことにする．ここで $2(\alpha+\beta)=a$ である．例えば，**図1**の場合，1:3:1:3 と表せる．

同じ表現で表されるフェーズに対して，Aを一つずつずらしていくと，この表現で表されるすべてのフェーズを重複なく選ぶことができる．

この中には $\beta:\alpha:\beta:\alpha$ で表現されるフェーズも含まれているので a が4の倍数のときは，

$$\begin{cases} 1:\frac{a}{2}-1:1:\frac{a}{2}-1 \\ 2:\frac{a}{2}-2:2:\frac{a}{2}-2 \\ 3:\frac{a}{2}-3:3:\frac{a}{2}-3 \\ \quad\cdots \\ \frac{a}{4}-1:\frac{a}{4}+1:\frac{a}{4}-1:\frac{a}{4}+1 \end{cases} \tag{1}$$

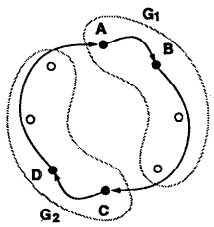


図1 4点サイクルの例（a=8 の場合）
Fig. 1 A cycle (a=8).

までの $\frac{a}{4}-1$ 通りを，4の倍数でないときは，

$$\begin{cases} 1:\frac{a}{2}-1:1:\frac{a}{2}-1 \\ 2:\frac{a}{2}-2:2:\frac{a}{2}-2 \\ 3:\frac{a}{2}-3:3:\frac{a}{2}-3 \\ \quad\cdots \\ \frac{a-2}{4}:\frac{a+2}{4}:\frac{a-2}{4}:\frac{a+2}{4} \end{cases} \qquad (2)$$

までの $\frac{a-2}{4}$ 通りを考えればよい．この場合一つの表現に対して，逆方向を含め a 通りのフェーズがある．

a が4の倍数のときはこのほかに，

$$\frac{a}{4}:\frac{a}{4}:\frac{a}{4}:\frac{a}{4} \qquad (3)$$

の場合があり，この表現で表されるフェーズは $a/2$ 通りある．

4点サイクルはこれですべて含まれているが，この中には図1のAとCのような $a/2$ 離れた点どうしの通信のフェーズが含まれていない．この2点サイクルは，

$$\frac{a}{2}:\frac{a}{2} \qquad (4)$$

と表現することとすると，$a/2$ 通りある．以上より，a が4の倍数の時は(1)，(3)，(4)の場合を合計して，

$$a\times\left(\frac{a}{4}-1\right)+\frac{a}{2}+\frac{a}{2}=\frac{a^2}{4}$$

4の倍数でない時は，(2)，(4)より，

$$a\times\left(\frac{a-2}{4}\right)+\frac{a}{2}=\frac{a^2}{4}$$

となり，いずれの場合も単方向チャネルを用いた場合の下限 L_{1u} が実現される．**図2**は，8プロセッサ構成の単方向一次元トーラスネットワークにおけるすべてのフェーズを示している．なお，黒点は通信を行っているプロセッサを示す．

両方向チャネルの場合は単方向チャネルの場合に得られたフェーズのうち，互いに向きの異なるサイクルどうしを重ね合わせることで実現できる．ただし各プロセッサは一ポート通信としているので，4点のうち，1点でも共有するものは選ぶことはできない．また二つのフェーズを4点とも重ならないように選ぶので，a は8以上でなければならない．

先に述べた(1)，(2)の場合とも，同じ表現で表されるフェーズのうち，適当にAがずれた互いに逆方向のサイクルを選ぶと，すべての表現について，**図3**のように二つのフェーズを4点とも共有することなく重ねることができて，各表現は $a/2$ 通りに半減する．a が4の倍数の場合は(3)の場合も2点サイクルの場合も二つの互いに逆方向のフェーズを重ねられる．しかし a が4の倍数でない場合は2点サイクルのフェーズの数 $a/2$ が奇数であり，すべてのフェーズを二つずつ重ねることはできず，1フェーズ余る．以上より a が4の倍数の場合は，$a^2/8$，4の倍数でない場合には，$\frac{a^2+4}{8}$ となり，a が8以上の4の倍数の場合に両方向チャネルを用いた場合の下限 L_{1b} が実現される．

2.4 二次元トーラス

2.4.1 理論的下限

$a\times a$ ($a=\sqrt{P}$) の二次元トーラスを考える．この二次元トーラスにおける全対全通信でメッセージの平

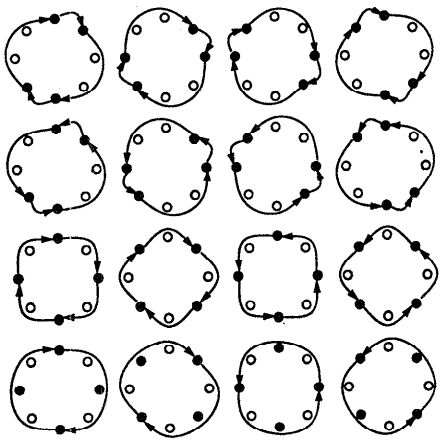


図2 単方向一次元トーラスネットワーク（8プロセッサ構成）におけるすべてのフェーズ
Fig. 2 All phases on a one-dimensional torus with eight processors.

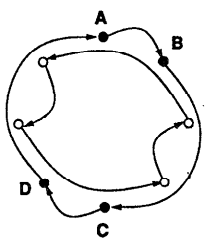


図3 二つのフェーズの重ね合わせ
Fig. 3 Two cycles.

均移動距離は，a が偶数のとき $a/2$ である．プロセッサの数は a^2，通信路の数は単方向チャネルのとき $2a^2$ 両方向のとき $4a^2$ なので，単方向チャネル時の送受信フェーズの理論的下限 L_{2u}，また両方向チャネル時の理論的下限 L_{2b} はそれぞれ，

$$L_{2u}=\frac{a^2\times\frac{a}{2}\times a^2}{2a^2}=\frac{a^3}{4}$$

$$L_{2b}=\frac{a^2\times\frac{a}{2}\times a^2}{4a^2}=\frac{a^3}{8}$$

となる．実際この下限は次に述べるアルゴリズムで実現される．

2.4.2 下限を実現するアルゴリズム

二次元トーラスにおける全対全通信アルゴリズムは，一次元トーラスの最適全対全通信アルゴリズムを二次元トーラスの水平方向と垂直方向に適用し，それらのクロス・プロダクト[12]で通信路を決定することを基本としている．**図4**に示すように，水平方向の通信路は水平方向に適用した一次元トーラスのアルゴリズムに従い，垂直方向の通信路は垂直方向に適用した一次元トーラスのアルゴリズムに従って決定する．この際通信路の決定は水平方向から行うことにする．

単純に水平方向と垂直方向のアルゴリズムを掛け合わせては，単方向チャネルの場合，全対全通信に必要なフェーズは $\left(\frac{a^2}{4}\right)^2=\frac{a^4}{16}$ となって下限より大きくなってしまう．しかし一つのフェーズでは水平方向，垂直方向とも四つのプロセッサしか使われていないことに注目すると，もし a が4の倍数なら，使われていないプロセッサの中から水平方向，垂直方向それぞれ四つずつプロセッサを選ぶことで独立な複数のフェーズを重ねることが可能である．**図5**に重ね合わせの例を示す．(a)と(b)，(c)と(d)が同じフェーズにチャネルが重なることなく通信を実現することができる．このように a が4の倍数の場合には $a/4$ のフェーズを重ね合わせることができるので，単方向チャネルの場合に必要なフェーズは，

$$\frac{a^4}{16}\div\frac{a}{4}=\frac{a^3}{4}$$

となり，理論下限 L_{2u} と一致する．

両方向チャネルの場合も同様に重ね合わせることが可能である．両方向の場合，水平方向，垂直方向とも八つのプロセッサを使っているので，a が8の倍数ならば複数のフェーズを重ねられる．これより両方向チャネルの場合の全対全通信に必要な送受信フェーズは，

$$\left(\frac{a^2}{8}\right)^2\div\frac{a}{8}=\frac{a^3}{8}$$

となり，理論的下限 L_{2b} と一致する．

これよりここで示した方法が，単方向チャネルを用いたときは a が4の倍数のとき，両方向チャネルを用いたときは a が8の倍数のとき，二次元トーラスにおける全対全通信の最適アルゴリズムであることがわかる．

2.5 長方形への拡張

二次元トーラスにはプロセッサ構成が正方形形状のものばかりでなく，128 (8×16)，あるいは 512 (16×32) プロセッサのような長方形形状の構成も考えられる．このような長方形形状の二次元トーラスに対しても前

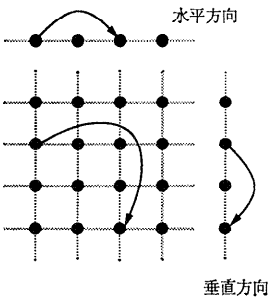


図4 クロス・プロダクトの例
Fig. 4 A cross product.

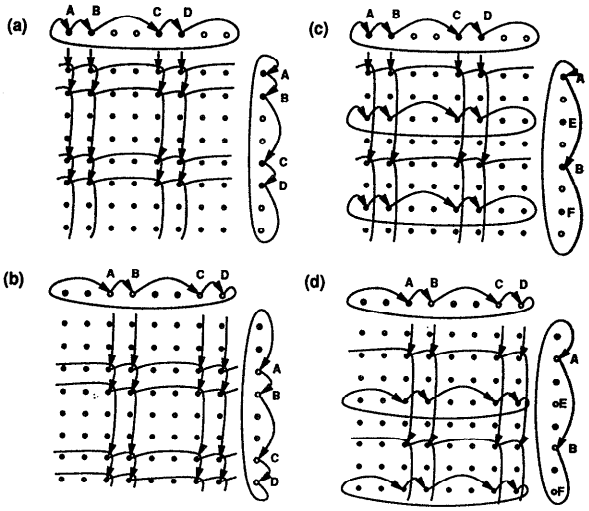


図5 重ね合わせの例
Fig. 5 Overlaps.

に述べたアルゴリズムが最適であることを示す．

2.5.1 理論的下限

まず長方形形状二次元トーラスにおける全対全通信の理論的下限を求める．長方形を $a\times b$ (a,b は偶数，$b\geq a$) とし，**図6**のようにプロセッサを半分ずつ二つのグループ G_1, G_2 に分ける．そして二つのグループ G_1, G_2 間で全対全通信をすることを考える．するとこのとき発生するメッセージの総数は $2\times\left(\frac{ab}{2}\right)^2$ であり，これが図6の実線で示される通信路を通ることになる．両方向チャネルとしたとき，実線で示される通信路の数は，図6の場合 $4a$ 本である．実線で示された通信路に注目すると，$2\times\left(\frac{ab}{2}\right)^2$ のメッセージが必ずここを通るから，図6の場合少なくとも，

$$2\times\left(\frac{ab}{2}\right)^2\div 4a=\frac{ab^2}{8}$$

フェーズ必要となる．プロセッサを二つのグループに分け，グループ間で全対全通信を行う場合，先に示した図6の場合のフェーズ数 $\frac{ab^2}{8}$ が最大フェーズ数になる．グループ分けは任意のプロセッサについて可能なので，ここで必要となるフェーズ数が，全対全通信に必要なフェーズ数の下限と一致する．長方形形状の二次元トーラスにおける，両方向チャネルを用いたときの全対全通信の理論的下限 L^r_{2b} は，$\frac{ab^2}{8}$ となる．単方向チャネルの理論的下限 L^r_{2u} は，$\frac{ab^2}{4}$ となる．

2.5.2 下限を実現するアルゴリズム

正方形形状の二次元トーラスの場合と同様に水平方向，垂直方向それぞれに一次元トーラスでの最適アルゴリズムを適用し，そのクロス・プロダクトで通信路を決定する．この際，重ねられる独立したフェーズ数は a と b のうち，小さい方に依存する．よって $b\geq a$ の場合，a が8の倍数から $a/8$ 個の独立したフェーズを重ねられるから必要なフェーズは，

$$\frac{a^2}{8}\times\frac{b^2}{8}\div\frac{a}{8}=\frac{ab^2}{8}$$

となり，a が8の倍数のとき，理論的下限 L^r_{2b} と一致する．単方向の場合も同様で，a が4の倍数のとき，$a/4$ 個の独立したフェーズを重ねられるから必要なフェーズは $\frac{ab^2}{4}$ となり，理論的下限 L^r_{2u} と一致する．

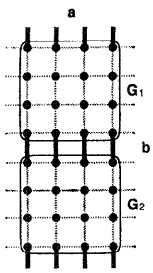


図 6 長方形形状二次元トーラス
Fig. 6 Divided rectangle.

2.6 高次元トーラス

一般に $a_1\times a_2\times\cdots\times a_n$ (a_m は，$a_1, a_2, \cdots, a_n$ の最大値) の直方体形状の n 次元トーラスにおける全対全通信の理論的下限 L^r_{nb} は，両方向チャネルのとき，$L^r_{nb}=\frac{a_mP}{8}$，また単方向チャネルのとき $L^r_{nu}=\frac{a_mP}{4}$ となる．ここで P はプロセッサ台数とする．

これに対し一次元トーラスでの最適アルゴリズムを n 個組み合わせ，そのクロス・プロダクトから通信路を決定し，独立なフェーズを重ね合わせることからなるアルゴリズムの必要フェーズ数も，両方向チャネルの場合，a_m 以外が8の倍数のとき，$\frac{a_mP}{8}$ となり，理論的下限 L^r_{nb} と一致し，最適なアルゴリズムであることがわかる．単方向チャネルの場合は，a_m 以外が4の倍数のとき，$\frac{a_mP}{4}$ となり，この場合もまた最適なアルゴリズムであることがわかる．

特に $a_1=a_2=\cdots=a_n=a$ の場合，n 次元トーラスにおける両方向チャネルを用いた全対全通信の理論的下限 L_{nb} は，a が8の倍数のとき，$\frac{P^{(n+1)/n}}{8}$ となる．同様に単方向チャネルの場合，n 次元トーラスにおける全対全通信の理論的下限 L_{nu} は，a が4の倍数のとき，$\frac{P^{(n+1)/n}}{4}$ となる．

2.7 メッシュとの比較

Scott は，メッシュネットワークにおける最適な全対全通信方式を提案している[12]．一般に n 次元メッシュネットワークの全対全通信性能は，両方向チャネルのとき，$\frac{P^{(n+1)/n}}{4}$，単方向チャネルのとき，$\frac{P^{(n+1)/n}}{2}$ となる．これより，メッシュネットワークの全対全通信性能は，トーラスネットワークの半分であることがわかる．

3. トポロジの比較

全対全通信は，最も通信転送量の多い通信パターンである．この章では，全対全通信性能をもとにネット

ワークのトポロジの性能比較を行う．対象とするネットワークは，実現が最も容易と考えられる *k-ary n-cube* を対象とする．比較においては，まず各チャネルの転送レートがすべて同じであるとしたとき，次に，ネットワークのコストを一定にして各チャネルの転送レートをトポロジにより変えたときの性能を比較する．なお，ここでは，通信セットアップ時間 C_T は無視し，通信時間のみを考慮する．

3.1 ネットワークのコストを考慮しないとき

一本のチャネルの転送レートがすべて同じであるとしたときの転送時間の比較を**図7**に示す．ハイパーキューブは，つねにプロセッサ台数回のフェーズで処理することができるので，全対全通信の下限を与えることになる．しかし，ハイパーキューブは，一ポート通信の条件のもとでは，全対全通信においてすべてのチャネルを使用しているわけではない．2.2 に従えば，

$$L_{Hypercube}=\frac{P\frac{\log P}{2}P}{P\log P}=\frac{P}{2}$$

となるので，半分のチャネルが遊んでいることになる．すべてのチャネルを使うようにすると，一ポート通信の条件を破ることになる．これは，他のトポロジにもあてはまり，例えば，64 (8×8) 台より少ない二次元トーラス構成，512 (8×8×8) 台より少ない三次元トーラス構成，4096 (8×8×8×8) 台より少ない四次元トーラス構成においても同じようにプロセッサとネットワークのインタフェースが転送の下限を与え，ネットワークのチャネルが遊ぶことになる．

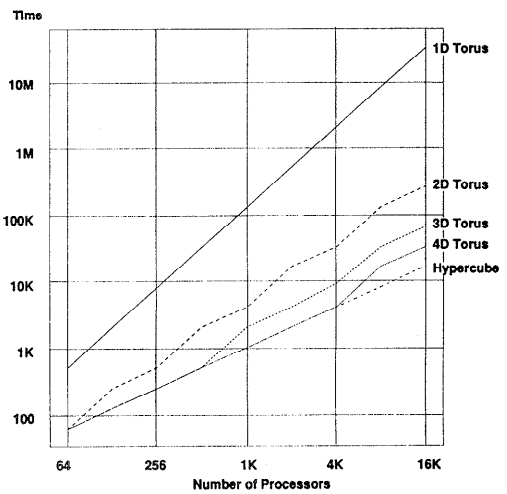


図 7 トポロジの比較
Fig. 7 Topology comparison.

3.2 ネットワークのコストを考慮するとき

実際のシステムを構築する上では，ネットワークのハードウェアの量を考慮しなければならない．そこで，ネットワーク構築のコストとして，Dally はネットワーク全体を半分に分けたとき二つのネットワークを横切るワイヤ数 (*Bisection Width*) を考慮している[13]．これは，すべてのプロセッサを1枚のボードに載せたときのネットワークのコストである．本論文では，*Bisection Width* 以外に1プロセッサに接続されるワイヤ数と複数プロセッサが搭載されたボードに接続されるワイヤの数をネットワークのコストとして考慮する．1プロセッサに接続されるワイヤ数を同じにした仮定は，システム全体のワイヤ数を同じにした仮定と同じである．なお，1ボードに 16 プロセッサが搭載されているものとする．全対全通信の性能と三種類のネットワークコストを**表1**に示す．

Bisection Width を一定としたときでは，一次元トーラスから三次元トーラスまですべて同じ性能になる．ハイパーキューブは，その半分の性能になるが，これはハイパーキューブのチャネルが常に半分遊んでいるからである．もし，マルチポート通信を仮定すれば，このネットワークコストの条件下では，すべて同じ性能を持つことになる．この事実はメッシュネットワークに対してもあてはまる．メッシュネットワークは端と端が接続されていないので，トーラスのときと比較して表1の *Bisection Width* は半分になるが，全対全通信の性能も半分になるので *Bisection Width* を一定としたときの性能はトーラスもメッシュも同じ性能になる．*Bisection Width* を一定としたとき同じ性能になるという事実は，*Bisection Width* 一定のもとではネットワーク容量は一定である[13]ということから考えれば当然のことがらである．なぜなら，全対全通信はネットワーク容量をすべて使うからである．

表 1 全対全通信とネットワークコスト
Table 1 All-to-all personalized communication and network cost.

ネットワークコスト	一次元	二次元	三次元	四次元	ハイパーキューブ
全対全通信	$P^2/8$	$P^{3/2}/8$	$P^{4/3}/8$	$P^{5/4}/8$	P
Bisection	4	$4P^{1/2}$	$4P^{2/3}$	$4P^{3/4}$	P
1プロセッサ	4	8	12	16	$2\log P$
16プロセッサ	4	32	80	128	$32(\log P-4)$

P: プロセッサ台数

次に，1プロセッサに接続されるワイヤ数を同じにしたときの性能を**図8**に，16プロセッサ搭載のボードに接続されるワイヤ数を同じにしたときの性能を**図9**に示す．これらの条件で16Kプロセッサ内のシステムを構築する場合には，三次元トーラス，四次元トーラス，ハイパーキューブの全対全通信の性能差はほとんどないということができる．ネットワークのトポロジは，適用する応用とコストを考慮して決定されるものである．全対全通信が頻繁に現れる応用に適用する並列システムでは，トポロジの次元は三次元トーラス以上のネットワークが必要であると考えられる．

4. 実　　験

ここでは，実際のシステムとして分散メモリ型並列計算機 AP1000 上に本論文の全対全通信方式を実現し，その性能を評価する．

4.1 AP1000 アーキテクチャ

AP1000[14]は，富士通研究所で開発された分散メモリ型の高並列計算機で，メッセージ通信を基本としたアーキテクチャ構成を持つ．AP1000はプロセッサを最小16台から最大1024台まで接続可能であり，各プロセッサはトーラスネット (T-Net)，ブロードキャストネット (B-Net)，同期ネット (S-Net) と呼ばれる三つの独立したネットワークで接続されている．

全対全通信のメッセージの転送には T-Net[16]を利用する．T-Net は二次元トーラスのトポロジを持ち，ワームホールルーティングを使用して転送する．T-Net を構成するルーティング・コントローラは，四つの単方向チャネルを持つ．全対全通信のアルゴリズムは，各フェーズですべてのプロセッサが送受信を行っているわけではないので，フェーズごとに同期をとることが必要である．そのためこの実験では同期専用の S-Net を利用した．

4.2 AP1000 での実測

実験は64(8×8)と256(16×16)台構成の計算機で行った．

AP1000 における全対全通信に関する時間を予想するために，S-Net による同期の時間と，T-Net 上での2プロセッサ間の通信の時間を測定した．S-Net による同期の時間はプロセッサ台数によらず 10.5 [μs] であった．一方，m [bytes] のメッセージを2プロセッサ間で通信する時間は単方向チャネルのとき $1.1+m/17.1$ [μs] であり，両方向チャネルのとき $1.1+m/7.1$ [μs] であった．両方向チャネルのときの転送レートが単方向チャネルのときの半分以下になっているが，これは実際には単方向のチャネルを両方向チャネルに使用するために，転送方向を反転するオーバヘッドがあるからである．ここでは単方向チャネルを利用したときのアルゴリズムで実験を行った．

測定結果から全対全通信に要する時間 T_{aac} は，

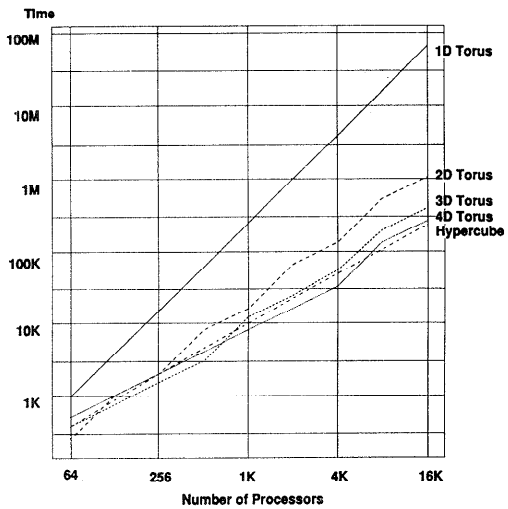


図 8 トポロジの比較（1プロセッサに接続されるワイヤ数で正規化）
Fig. 8 Topology comparison with a constant number of wires connected to each processor.

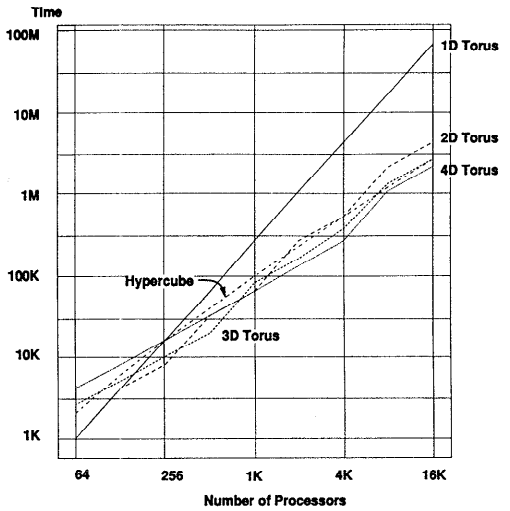


図 9 トポロジの比較（16プロセッサに接続されるワイヤ数で正規化）
Fig. 9 Topology comparison with a constant number of wires connected to one board.

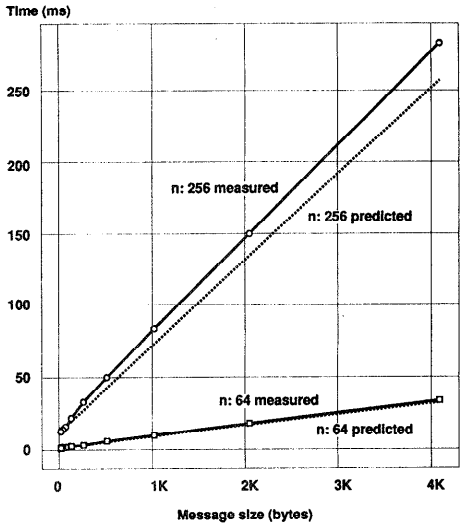


図 10 実験結果
Fig. 10 Experimental results.

$T_{aac}=L_{2u}(C_S+m/C_T)=L_{2u}(11.6+m/17.1)$ [μs]
と予想される．全対全通信に必要なフェーズ数 L_{2u} は64プロセッサのとき128で，256プロセッサのとき1024である．

図 10 に AP1000 における実験の結果と予測の値を示す．実線で実験の結果を，また点線で予測値を表している．いずれの場合も予測値とほぼ一致した結果が得られている．

実験結果より，本アルゴリズムは AP1000 の通信性能を最大限活かした全対全通信を実現することがわかる．

5. おわりに

本論文では，トーラスネットワーク上での最適な全対全通信アルゴリズムを提案した．本方式により，トーラスネットワークではメッシュネットワーク上での全対全通信に対して2倍の性能が得られる．また，本アルゴリズムが一次元から多次元のトーラスネットワーク，長方形形状にも適用できることを示した．

さらに，全対全通信の性能に基づいたトポロジの比較を行った．その結果，ネットワークのコストとして1プロセッサに接続されるワイヤの数，1ボードに接続されるワイヤの数を考慮すると，三次元，四次元，ハイパーキューブはほぼ同じ性能を与えることが示された．

並列計算機 AP1000 上に本アルゴリズムを実装し，性能評価を行った結果，本アルゴリズムはシステムが持つ通信性能を最大限活かした全対全通信を実現することがわかった．

本論文では，メッセージの転送時間が転送スタートアップ時間に対して十分長いと仮定して議論してきた．しかし，プロセッサの台数が増えてくると一般にメッセージの長さは短くなりこの仮定が成り立たなくなる可能性がある．メッセージが短い場合の検討も今後必要である．

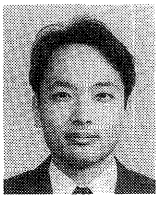
謝辞 日頃御指導，御助言いただく，並列処理研究センター石井センター長，白石部長，佐藤主任研究員，並列処理研究センター第二研究室池坂室長，研究室の同僚諸氏，ならびに，本研究の動機づけと有益な助言をいただいたソフトウェア研究部の武氏に感謝いたします．

参考文献

1) Johnsson, S.L. and Ho, C.T.: Optimum Broadcasting and Personalized Communication in Hypercubes, *IEEE Trans. Comput.*, Vol. 38, No. 9, pp. 1249-1268 (1989).
2) Take, R.: Optimal Routing Method of All to All Communication on Hypercube Netwoks, *Proc. 35th Annual Convention IPS Japan*, pp. 151-152 (1987).
3) Boppana, R.V. and Raghavendra, C.S.: All-to-All Personalized Communication on Circuit Switched Hypercubes, Technical Report, CENG 90-20 Dept. of EE-Systems (1990).
4) Ho, C.T. and Raghunath, M.T.: Efficient Communication Primitives on Circuit-switched Hypercubes, *6th Distributed Memory Computing Conference*, pp. 390-397 (1991).
5) Bokhari, S.H.: Complete Exchange on the iPSC-860, Technical Report 187498, NASA Contractor Report (1991).
6) Bokhari, S.H.: Multiphase Complete Exchange on a Circuit Switched Hypercube, Technical Report 187499, NASA Contractor Report (1991).
7) Dally, W.J.: *A VLSI Architecture for Concurrent Data Structures*, Kluwer Academic Publishers (1987).
8) Arlauskas, R.: iPSC/2 System: A Second Generation Hypercube, *Third Conference on Hypercube Concurrent Computers and Applications*, pp. 38-42 (Jan. 1988).
9) Nugent, S.F.: The iPSC/2 Direct-connect Communication Technology, *Third Conference on Hypercube Concurrent Computers and Applications*, pp. 610-619 (1988).
10) Seitz, C.L. et al.: The Architecture and

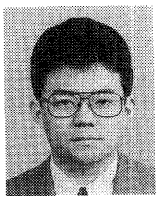
Programming of the Ametek Series 2010 Multicomputer, *Third Conference on Hypercube Concurrent Computers and Applications*, pp. 33-36 (Jan. 1988).
11) Dally, W. J. et al.: The J-Machine: A Fine-grain Concurrent Computer, *Inf. Process.*, pp. 1147-1153 (1989).
12) Scott, D. S.: Efficient All-to-All Communication Patterns in Hypercubes and Mesh Topologies, *6th Distributed Memory Computing Conference*, pp. 398-403 (1991).
13) Dally, W. J.: Performance Analysis of K-ary N-cube Interconnection Networks, *IEEE Trans. Comput.*, Vol. 39, No. 6, pp. 775-785 (June 1990).
14) 石畑宏明, 稲野 聡, 堀江健志, 清水俊幸, 池坂守夫: 高並列計算機 AP 1000 のアーキテクチャ, 信学論 (D-I), Vol. J 75-D-I, No. 8, pp. 637-645 (1992).
15) Borkar, S. et al.: Supporting Systolic and Memory Communication in iWarp, *The 17th Annual International Symposium on Computer Architecture*, pp. 70-81 (1990).
16) 堀江健志, 石畑宏明, 池坂守夫: 並列計算機 AP 1000 における相互結合網のルーチング方式, 信学論 (D-I), Vol. J 75-D-I, No. 8, pp. 600-606 (1992).

(平成 4 年 9 月 8 日受付)
(平成 4 年 12 月 10 日採録)



堀江 健志 (正会員)

1962 年生. 1984 年東京大学工学部電気工学科卒業. 1986 年同大学院修士課程修了. 同年(株)富士通研究所入社, 現在に至る. 並列計算機に関する研究開発に従事.



林 憲一 (正会員)

1967 年生. 1991 年東京大学工学部計数工学科卒業. 同年(株)富士通研究所入社. 現在並列処理研究センターにて, 並列計算機アーキテクチャの研究に従事.