Z-routing：イレギュラーネットワークにおける適応型ルーティングに関する研究

井川郁哉[†,☆] 舟橋啓[†]

近年，著しく性能が向上した商用の PC/WS を複数台接続したイレギュラーネットワークにおいて並列計算を行うことにより，全体として高い計算能力を提供する PC クラスタの研究がさかんに行われている．しかし，イレギュラーネットワークにおける既存のルーティングアルゴリズムは，デッドロックフリーを実現するため利用可能なパスやバーチャルチャネルの使用を制限しており，結果としてパフォーマンスの低下を招いている．本論文において，我々は Z-routing と呼ばれる新しいルーティングアルゴリズムを提案する．Z-routing は 2 本のバーチャルチャネルを使用することによって高い自由度を得ることができる．シミュレーションの結果から，Z-routing は既存のルーティングアルゴリズムと比べて，高い性能を示した．

Z-routing: Adaptive Routing on Irregular Networks

Yuki Igawa[†,☆] and Akira Funahashi[†]

Network-based parallel processing using commodity personal computers has been widely developed. Since such systems require high degree of flexibility and scalability of wiring, a high-speed network with an irregular topology is often needed. In traditional routing algorithms for irregular networks, available paths and virtual channels are considerably restricted in order to avoid deadlocks. In this paper, we propose a novel routing algorithm called Z-routing, which has an enough freedom in irregular networks by using two virtual channels. Simulation results show that Z-routing improves the performance compared with the traditional routing algorithms using two virtual channels.

1. はじめに

近年，パーソナルコンピュータ（PC）やワークステーション（WS）のような商用のコンピュータが複数台接続されたネットワークで並列処理を行う方法が，経済的で高性能な並列処理システムとして高い注目を浴びている[1)~4)]．

このようなシステムでは通常，高い柔軟性や接続のスケーラビリティの要求からトポロジフリーな高速ネットワーク（イレギュラーネットワーク）が用いられている．しかし，そのイレギュラーネットワークにおいて利用可能なリンクをすべて用いることは難しく，さらにその不規則性がデッドロックフリーなルーティングを困難なものとしており，結果としてイレギュラーネットワークのパフォーマンスはレギュラーネットワークと比較して低くなりがちになっている[5)]．

そこで本論文では新しいルーティングアルゴリズムである Z-routing を提案する．Z-routing はネットワークを独自の方法でグラフに見立てることによりルーティングを行う．その結果，Z-routing は既存のルーティングアルゴリズムに比べ高い自由度を達成し，さらにデッドロックの原因である環状構造の存在を許すことによって平均ホップ数を減らすことに成功した．

以降，2 章で様々なネットワーク形態を隠蔽するモデル，およびイレギュラーネットワークにおける既存のルーティングアルゴリズムについて述べ，3 章で Z-routing の提案を行う．そして 4 章でシミュレーション結果を提示し，5 章で今後の課題と結論を述べる．

2. イレギュラーネットワーク

イレギュラーネットワークとはノードが無規則に結合したネットワークであり，ネットワーク内に一定の規則性が存在しないという点が特徴としてあげられる．この特徴（irregularity）はネットワークの拡張を容易にする一方，ルーティングアルゴリズムの作成やデッ

† 三重大学工学部情報工学科
Department of Information Technology, Mie University
☆ 現在，神戸大学大学院自然科学研究科情報知能工学専攻
Presently with Department of Computer and Systems Engineering, Kobe University

ドロックの除去を困難なものとしている．

多数のPCやWSによって構成されるLAN（Local Area Network），WAN（Wide AreaNetwork）がイレギュラーネットワークの例としてあげられる．また，Network of workstations（NOWs）[1)~4)]などの高速なネットワークで接続されたPCやWSで並列処理を行う場合，そのネットワークには高い柔軟性や接続のスケーラビリティの要求からイレギュラーネットワークが用いられる．

2.1 ネットワークモデル

イレギュラーネットワークのルーティングはプロセッシングエレメントとスイッチの接続，スイッチどうしの接続など，様々な接続ケースを考えなければならない．しかし，接続ケースに応じてルーティングアルゴリズムを考えるのは得策ではない．そこで，それらの差を隠蔽するネットワークモデルへの抽象化の例としてNOWsを取り上げ簡単に説明する．

NOWsは一般的にswitch-based networkであるが，各スイッチはプロセッシングエレメントと個々に直結しているため，パケットを目的地のプロセッサのあるスイッチにルーティングできれば，短時間でデッドロックせずに各プロセッサに到着させることができる．よって，ルーティングアルゴリズムはスイッチ間のルーティングに焦点を当て，検討することができる．

スイッチ間の内部接続ネットワーク I は有向グラフ G で以下のように表せる．

$$I = G(N, C).$$

ここで，N はスイッチの集合，C はスイッチ間の双方向リンクを表している．

これにより，**図1**のようなイレギュラーネットワークは**図2**のように表すことができる．

図2のようにネットワークを一般化し，ルーティングアルゴリズムを検討していくことにより，ルーティングアルゴリズムはスイッチ間接続にとどまらず，直接網にも適用することができる．

2.2 既存のルーティングアルゴリズム

イレギュラーネットワークにおけるルーティングアルゴリズムはここ数年来，様々な提案がなされており，up/down routing [1)]などはすでに実装されている[3)]．

本章ではイレギュラーネットワークにおける代表的な4つのルーティングアルゴリズムについて説明する．

2.2.1 Primitive up/down routing

Primitive up/down routingはイレギュラーネットワークにおける様々なルーティングアルゴリズムの基礎となるもので，単純なルーティングアルゴリズムとして知られている．

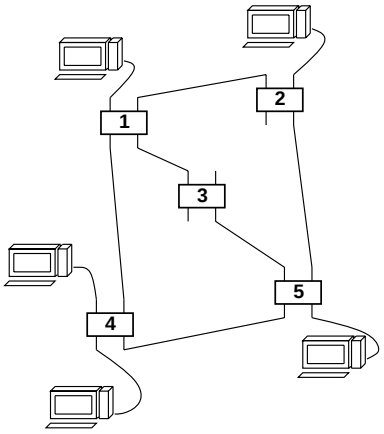

図1 PCクラスタにおける結合網の概略図
Fig. 1 Switch-based interconnection network.

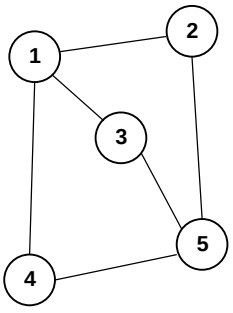

図2 有向グラフG
Fig. 2 Corresponding graph G.

まず初めに，ネットワークを以下の規則に基づきツリー（スパニングツリー）に見立て，その後ツリーの階層構造を利用したルーティングを行う．

1. ネットワークの中から任意にツリーのルートノードを選ぶ．
2. ルートノードと隣接するノードをツリーに付け加える．
3. ツリーの各ノードから隣接する，まだツリーに加えられていないノードをツリーに付け加えていく．
4. 全ノードがツリーに入るまで3.の作業を繰り返す．

上記のルールにより，ツリー上のルートノードを除くすべてのノードは親ノードを唯一持つようになる．このツリーが作成される際，親ノードと子ノード以外へ接続されているリンクは無視される．たとえば，**図3**における破線のリンクは実際には存在するが，ツリー上では無視される．

Primitive up/down routingのスパニングツリーには一般的に幅優先のスパニングツリー（Breadth-First Spanning Tree, BFST）が用いられている．よって，ツリーの生成の時間計算量は $O(n+m)$（n はグラフ

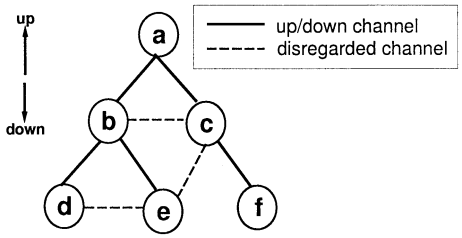

図 3 イレギュラーネットワークのツリー表現
Fig. 3 Tree expression of irregular network.

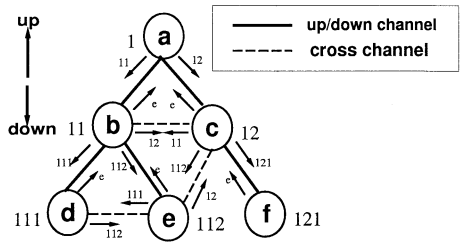

図 4 Prefix routing に用いられる directed-graph
Fig. 4 Directed-graph for prefix routing.

の頂点数，m は辺の本数）となり，低く抑えられている．

ルーティングアルゴリズムは単純である．目的地ノードが現ノードの子，孫ノードの場合，**down channel** を使用してそのノードの方向へパケットを転送する．目的地ノードが現ノードの子，孫ノードではない場合，**up channel** を使用して親ノードへパケットを転送する．

つまり，ルーティングは非最短型の固定ルーティングであり，一部の実在するリンクを利用することができない．

たとえば，図 3 においてノード **b** からノード **f** にパケットを転送する場合，パケットは **b**→**a**→**c**→**f** というルートに沿って転送されることになる．

デッドロックを防ぐために，パケットは up channel によってルートノード方向に必要数移動した後，down channel によってルートノードから遠ざかる方向へ必要数移動する．その結果，up channel と down channel の間に環状構造がなくなるため，デッドロックフリーが保証される．

2.2.2 Prefix routing

Wu ら[6]によって提案された Prefix routing は Primitive up/down routing で利用することができなかったリンクの一部を利用できるようにしたものである．

Prefix routing ではツリーを作成する際，各ノードとチャネルにラベルを付け，すべてのチャネルを有向グラフに組み込み，ルーティングに利用する．ノードとチャネルへのラベル付けの方法は以下のとおりである．

ルートノードのラベルを $L_n(r) = 1$ とし，ノード $L_n(v)$ の k 番目の子ノード u のラベルを $L_n(u) = L_n(v) \parallel k$ とする．なお，$\parallel$ は連結命令である．ツリーに含まれるチャネルに対しては，ノード v がノード u に比べてルートノードに近い場合，$L_c(v,u) = L_n(u)$，$L_c(u,v) = e$ とし，それ以外のチャネルに対しては，$L_c(v,u) = L_n(u)$，$L_c(u,v) = L_n(v)$ とする．e は空ラベルである（**図 4**）．

Prefix routing のスパニングツリーは Primitive up/down routing と同様に BFST が用いられているため，ツリーの生成の時間計算量は $O(n+m)$ である．また，ノードとチャネルへのラベル付けはツリーの頂点を 1 回ずつ割り振り，辺を 2 回ずつ割り振ることになるため，時間計算量は $O(n+m)$ である．よって全体の時間計算量も $O(n+m)$ となり，低く抑えられている．

ルーティングは，目的地ノードのラベルの “prefix” を持つチャネルがあればそのチャネルを選択し，なければ，ラベル ‘e’ のチャネル（up channel）を選択する．これにより，各パケットは Primitive up/down routing の経路をたかだか 1 回ショートカットすることが可能である．これは，パケットがショートカットを行うことが可能となる条件が，

1. 現在地ノードが目的地ノードと子孫関係にない，
2. ショートカット先のノードが目的地ノードまたは目的地ノードの先祖ノードとなる，

の 2 つであるため，1 度ショートカットを行うと 1. の条件が 2 度と成立しなくなるからである．

たとえば，図 4 においてノード **b**(11) からノード **f**(121) にパケットを転送する場合，Primitive up/down routing では **b**(11)→**a**(1)→**c**(12)→**f**(121) と 3 ホップ必要だが，Prefix routing では **b**(11)→**c**(12)→**f**(121) と 2 ホップで到着することができる．

Prefix routing はラベルを調べるだけでルーティングをできるので，他の up/down based routing に比べルーティングテーブルが小さいという特徴を持つ．また，Prefix routing は cross channel を使用した場合でも以後 down channel のみを用いてパケットを転送するため，up channel と down channel の間に循環構造をとることがなく，したがってデッドロックフリーが保証されている．

2.2.3 Minimal routing

Primitive up/down routing と Prefix routing は

down channelを使用した後，up channelの使用を禁止することによりcyclic dependencyを排除し，デッドロックフリーを実現した．しかし，Sillaらは既存のルーティングアルゴリズムをエスケープパスに使い，各リンクにfully adaptiveなバーチャルチャネルを追加するMinimal routingを提案し[3),5)] cyclic dependencyを除去することなしに，デッドロックフリーを実現した．以下にルーティングアルゴリズムを簡単に述べる．

まず初めに，各リンク間に2本のバーチャルチャネル（original channel，new channel）を用意する．そして，パケットは原則としてnew channelを用いて転送され，最短経路を選択する．しかし，選択可能なnew channelがすべて使用されている場合，パケットはoriginal channelに切り替えられて転送され，Primitive up/down routing，Prefix routingなどの既存のルーティングアルゴリズムを用いて経路を選択する．なお，1度original channelを使用したパケットは2度とnew channelを使用することはできない．

original channelは，既存のルーティングアルゴリズムを用いているためデッドロックフリーが保証されており，ネットワーク全体にわたるエスケープパスになるため，デッドロックフリーが保証される．

Minimal routingはパケットがnew channelを使用した場合，最短経路をとるため効率が良く，既存の非最短型のルーティングアルゴリズムの性能を大きく向上させることができる．

2.2.4 L-turn routing

Left-first turn routing（L-turn routing）は慶応義塾大学によって提案されたルーティングアルゴリズムである[7)]．L-turn routingはイレギュラーネットワークをL-R tree☆というスパニングツリーに見立て，ルーティングを行う．

L-turn routingを説明するに際し，ここで以下のように語を定義する．

- **left node**：L-R treeにおいて，各階層の一番左側のノード．**図5**においては**a**，**b**，**d**がこれにあたる．
- **right node**：L-R treeにおいて，left node以外のノードのこと．図5においては**c**，**e**，**f**，**g**，**h**，**i**がこれにあたる．
- **left path**：L-R treeにおいて，現在のノードよりも左側に存在するノードへのリンクのこと．図5において，ノード**e**からノード**b**へのリンクはleft pathである．
- **right path**：L-R treeにおいて，現在のノードよりも右側に存在するノードへのリンクのこと．図5において，ノード**e**からノード**c**へのリンクはright pathである．

☆ 実際はツリー構造ではなくグラフ構造であるためこの名称は適切ではないが，本論文では参考文献において定義されている単語をそのまま使用する．

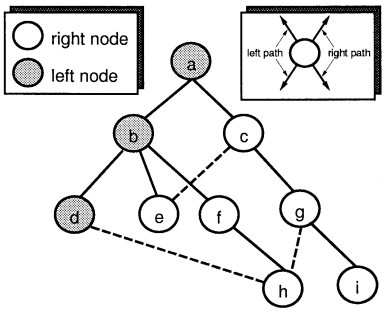

図5 L-R treeの例（ノード数9）
Fig. 5 An example of L-R tree (9 nodes).

以下に，L-R treeの構成法を述べる．

1. ネットワークの中から任意にL-R treeのルートノードを選ぶ．
2. ルートノードに隣接したノードの1つをleft node，他をright nodeとしてL-R treeに追加する．たとえば図5ではルートノードは**a**，left nodeは**b**，right nodeは**c**である．
3. L-R treeのleft node，right nodeについて以下の作業を行う．
   (a) right nodeにおいて，まだL-R treeに組み込まれていない隣接ノードをright nodeとしてL-R treeに追加する．
   (b) left nodeにおいて，まだL-R treeに組み込まれていない隣接ノードのうち，1つをleft node，他をright nodeとしてL-R treeに追加する．たとえば図5ではノード**b**の隣接ノードのうち，left nodeは**d**，right nodeは**e**，**f**である．
4. 3.の手順を全ノードがL-R treeに組み込まれるまで繰り返す．
5. L-R treeに漏れているイレギュラーネットワークのリンクを追加する．

図5における破線のリンクは，L-R treeの構成法の手順5.で追加したリンクである．

L-R treeの生成において，left node，right nodeの割振りは頂点をちょうど1度ずつ定義していくため，時間計算量は$O(n+m)$である．しかし，left path，right pathの割振りにはアルゴリズムが存在せず，最

適なパフォーマンスを得るため手作業で行われているため，現時点での全体の時間計算量は $O(n+m)$ より大きくなっている．

L-turn routing のルーティングは，

“right path を使用した後，left path を使用してはならない.”

という条件を守ればいかなる経路も選択可能である．よって，出発地ノードと目的地ノードの間には複数の経路が存在することになるため，L-turn routing は適応型ルーティングである．また，L-turn routing は Turn モデル[8]を用いることにより，上記の条件からデッドロックフリーが保証されている．

図 5 においてノード **d** からノード **i** にパケットを転送する場合，Primitive up/down routing や Prefix routing では **d→b→a→c→g→i** と 5 ホップ必要だが，L-turn routing では **d→h→g→i** と 3 ホップでパケットを転送することができる．さらに L-turn routing では **d→b→a→c→g→i** の経路も選択可能である．

L-turn routing は非最短型の適応型ルーティングアルゴリズムであるが，自由度が高いため全体のチャネル利用率が向上し，トラフィックが分散される．また，パケットの平均ホップ数を抑えることができる．

2.2.5 既存のルーティングアルゴリズムの問題点

Primitive up/down routing や Prefix routing ではスパニングツリーの生成の時間計算量は $O(n+m)$ と低く抑えられてはいるが，up channel と down channel の間に厳しい制限を設けデッドロックフリーを実現したためすべてのリンクを効果的に使うことが難しく，ルートノードにトラフィックが偏る，チャネルが空いていても最短経路をとりにくい，などの問題点が存在する．

Minimal routing は new channel より original channel の方が使用される割合が大きい傾向にあり，結局 original channel で用いられるルーティングアルゴリズムの性能がパフォーマンスを左右することになる．また，スパニングツリーの生成の時間計算量も original channel で用いられるルーティングアルゴリズムに依存する．

L-turn routing は L-R tree の作成方法が曖昧である．たとえば，図 5 においてノード **h** はノード **g** に対して位置的に左にも右にもなりうるため，パス **h→g** は left path にも right path にもなりうる．その結果，場合によっては使えない経路が存在することになり，そのことがパフォーマンスに影響を与える．また，left node，right node の割振りの時間計算量は $O(n+m)$ だが，left path，right path の割振りは手作業で最適な状態を求めているため，L-R tree の生成にかかる全体の時間計算量は $O(n+m)$ より大きくなる．

3. Z-routing

既存の up/down based routing algorithm においてあげられるルートノード付近へのトラフィックの偏り，および最短経路のとりにくさなどによって生じるパフォーマンスの低下は，イレギュラーネットワークにおいてデッドロックフリーを保証するためにルーティングアルゴリズムの自由度を低下させたことが原因である．

そこで我々は性能の向上を図るために，より多くの最短路と高い自由度を提供する新しいルーティングアルゴリズムである Z-routing を提案する．

以降，まずイレギュラーネットワークを red/blue graph に見立てる手順を説明し，その後 Z-routing のルーティングアルゴリズムを提案する．

3.1 red/blue graph の作成法

Primitive up/down routing や Prefix routing で使用されているツリーでは，ルートノードに向かうチャネルを up channel，ルートノードから遠ざかるチャネルを down channel と分類した．そして，Primitive up/down routing と Prefix routing は up/down という階層構造を下にデッドロックフリーを実現した．

しかし，我々はより自由度の高いルーティングを実現するために up/down の概念を捨て，red/blue という概念を導入した **red/blue graph** を提案する．以下に，red/blue graph の構成方法を述べる．

1. ネットワークの中から任意にノードを選択し，そのノードをルートノードとする深さ優先のスパニングツリー（Depth-First Spanning Tree，DFST）の構築を行う．なお，同時に関節点[9),10)]を求めておく．
2. 1. で構築した DFST に含まれる辺（木の辺，tree edge）を **blue edge** と定義する．また，ネットワークにおけるそれ以外の辺（逆辺，back edge）を **red edge** と定義して DFST に組み込む．この時点でネットワークに含まれるすべての辺が blue edge か red edge のどちらかに割り当てられたことになる．
3. 幅優先探索の応用で辺に番号付けをする．以下，blue edge と red edge の番号付けを，各々 n_b，n_r という表現法で定義する．
   (i) ルートノードに接続している blue edge の辺番号を 1（$n_b = 1$）とする．ここで，n_b と n_r との関係を，$n_r = n_b + 1$ と定義する．

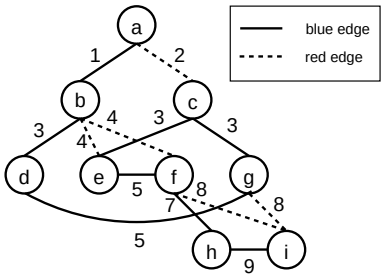

図 6 red/blue graph（9 ノード）の例
Fig. 6 An example of red/blue graph (9 nodes).

この定義により，ルートノードに接続されている red edge の辺番号は 2 となる．これらの辺番号の組みを，クラス $E(n_b, n_r)$（ここでは $E(1,2)$ となる）と定義する．さらに，$E(n_b, n_r)$ に接続しているノードのクラスを $N(n_b, n_r)$ と定義する．たとえば，$N(1,2)$ に含まれるノードは**図 6** よりノード **b** とノード **c** となる．

(ii) $N(n_b, n_r)$ に属するノードにつながっている blue edge と red edge を各々 n_r+1，n_r+2 と番号付けすることにより，残りの blue edge と red edge に帰納的に番号付けしていく．たとえば，図 6 において，ノード **b** とノード **c** につながっているまだ番号付けされていない blue edge と red edge は，各々3，4 と番号付けされ，$E(3,4)$ と定義される．

(iii) (ii) で $E(p_b, p_r)$，$E(q_b, q_r)$ $(p \neq q)$ のようにすでに定義された複数の辺のクラスが，あるノードに接続されている場合，そのノードに接続されているまだ番号付けされていない辺には次のように番号付けする．まず，番号 p_r と 番号 q_r を比較する．その結果，大きい方の番号のクラスを N として選択する．たとえば，図 6 におけるノード **f** がすでに $E(3,4)$ と $E(5,6)$ に接続されているとき，ノード **f**→**h** 間の blue edge の辺番号は 7 となり，ノード **f**→**i** 間の red edge は 8 となる．

(iv) (ii)，(iii) をすべてのノードがクラスに定義されるまで繰り返す．

図 6 は上記の方法により構成された 9 ノードの red/blue graph の例である．

辺を blue edge と red edge に分類することにより，グラフ内の環状構造をすべて検出することができる．red edge はグラフ内の環状構造の一辺を担う辺であり，もし，この red edge を取り除けばデッドロックフリーは保証される．しかし，red edge を取り除くことは自由度の低下を招いてしまうため，red/blue graph では辺に番号付けをし red edge も使用することによって自由度の高いルーティングを実現している．

3.2 Z-routing algorithm

本節では前節で述べた red/blue graph を利用する “Z-routing” と呼ばれる新しいルーティングアルゴリズムの提案を行う．

前章で述べたように，up/down based routing algorithm はツリーの階層間の移動に up/down などの厳しい制約があり，それが自由度を低下させる原因となっている．そこで Z-routing はグラフの階層間の移動の制限を緩くすることで性能向上を図る．

Z-routing は以下のように定義される．

(1) 2 本のバーチャルチャネル C_i（increment），C_d（decrement）を使用する．

(2) 出発地ノード S から目的地ノード D へのパスを検出する際，前節で定義した辺番号（n_b, n_r）を使用する．S → D 間の辺番号を並べたとき，以下の番号の並びのみを許すものとする．

(a) **昇順**：図 6 において **a**→**b**→**f**→**h** という経路の場合，辺番号の並びは 1，4，7．

(b) **降順**：図 6 において **h**→**f**→**b**→**a** という経路の場合，辺番号の並びは 7，4，1．

(c) **1 回のみの切替え**：バーチャルチャネル C_d から C_i への 1 回のみの切替えを許す．

(3) (2) により S → D 間の最短パスが検出された後，以下のようにしてパケットを転送する．もし辺番号の並びが昇順だった場合はバーチャルチャネル C_i を使用し，降順だった場合は C_d を使用する．もし “1 回のみの切替え” が可能なら，バーチャルチャネルを C_d から C_i へ切り替える．

上記の条件を守れば，Z-routing はいかなる経路も選択可能であり，自由度の高い非最短型ルーティングを実現する．また，red/blue graph の作成は，DFST の作成の時間計算量が $O(n+m)$（n はグラフの頂点数，m は辺の本数），辺の番号付けは幅優先探索（BFS）の応用でできるため時間計算量が $O(n+m)$，よって，red/blue graph の製作の時間計算量は $O(n+m)$ に抑えられる．よって，グラフの構築，および再構築にかかる時間を低く抑えることが可能である．

定理 1 Z-routing はデッドロックフリーである．

証明：red/blue graph は DFST を基に作られている．DFST は閉路を含まない木であり，ネットワークを DFST に含まれる辺（木の辺，tree edge，red/blue

graph における blue edge）とそれ以外の辺（逆辺，back edge，red/blue graph における red edge）[9),10)] に分けることによりネットワーク内に存在する全閉路を検出することができる．そして，ネットワークの同じ深さにおける blue edge と red edge に異なる番号付けをし，ネットワークの深さに応じて番号を増加させることにより，閉路をなす辺に異なる番号付けをすることができる．したがって，辺の番号の並びの昇順，および降順に応じてバーチャルチャネルを使い分けることにより循環構造を断ち切ることができるため，Z-routing はデッドロックフリーであることが証明される．∎

Z-routing における全ノードへの経路は

- 辺に付けられた番号を降順にたどることにより，どのノードからもルートノードにたどり着くことができる，
- ルートノードから辺に付けられた番号を昇順にたどることにより，どのノードへもたどり着くことができる，
- バーチャルチャネル C_d から C_i への 1 回のみの切替えを許すものとする，

ことにより，保証される．

Z-routing において全ノードへの経路を保証するために，以下のようにルーティングテーブルを作成する．**図 7** はネットワークを 2 連結成分[9),10)]の視点で表したものである．

- 出発地ノードと目的地ノードが同一の極大な 2 連結成分内に存在する場合は，出発地ノードから目的地ノードへの最短路を直接求める（例：図 7 において **v1** から **v2** へパケットを転送する場合）．
- 出発地ノードと目的地ノードが異なる極大な 2 連結成分に存在し，通過する間接点が 1 つのみの場合は，出発地ノードから間接点までの最短路と間接点から目的地ノードまでの最短路を求め，バーチャルチャネル C_d から C_i への切替えが 1 回以内になるようにそれらを組み合わせる（例：図 7 において **v1** から **v3** へパケットを転送する場合）．
- 出発地ノードと目的地ノードが異なる極大な 2 連結成分に存在し，通過する間接点が 2 つ以上の場合は，出発地ノードから間接点までの最短路と間接点から間接点までの最短路，そして間接点から目的地ノードまでの最短路を求め，バーチャルチャネル C_d から C_i への切替えが 1 回以内になるようにそれらを組み合わせる（例：図 7 において **v1** から **v4** へパケットを転送する場合）．

Z-routing は up channel，down channel の概念がないため，図 6 においてノード **b** からノード **i** へパケットを転送するような場合，Prefix routing や Primitive up/down routing では **b**→**a**→**c**→**g**→**i** と 4 ホップ必要だが，Z-routing では **b**→**f**→**i** と 2 ホップのみで到着させることができる．

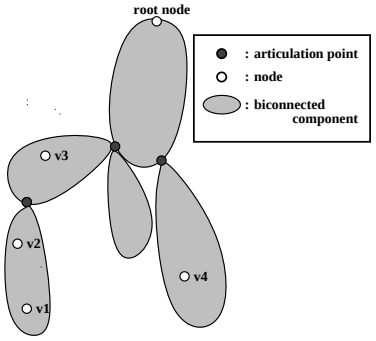

図 7 関節点と 2 連結成分
Fig. 7 Articulation point and biconnected component.

Z-routing は自由度が高いため全体のチャネル利用率が向上し，トラフィックが分散される．また，パケットの平均ホップ数を抑えることもできる．

4. 評　　価

今回提案した Z-routing，そして既存の Primitive up/down routing，Prefix routing，L-turn routing（Z-routing 以外はそれぞれ Minimal routing を併用）の各ルーティングアルゴリズムを C++で記述したフリットレベルシミュレータを用いて評価を行った．

今回のシミュレーションにおいて，ネットワークサイズ，バーチャルチャネルの本数，そしてパケット長はパラメータを変化させることによって選択した．各ノードはプロセッサ，リクエストキュー，そして双方向チャネルを提供するルータからなり，近隣ノードに接続した．ルータはチャネルバッファ，クロスバ，リンクコントローラ，バーチャルチャネルコントローラ，そしてコントロールサーキットからなる単純なルータモデルを使用した．

4.1 シミュレーション条件

今回の評価では，パケットの目的地を決定する際に，すべての目的地ノードがランダムに選択される uniform トラフィックを採用した．

また，以下の 2 つの指標により評価を行った．

4.1.1 ネットワークレイテンシ：

あるノード p がパケットの最初のフリットを入力バッファに挿入した時刻を t_0，目的地ノードである q がパケットの最後のフリットを受け取った時刻を t_1 とする．ここで $T_{lat}(p,q) = t_1 - t_0$ をネットワーク

表 1 シミュレーション条件
Table 1 Simulation parameters.

実行時間	50,000 clks
ネットワーク	Irregular or 2D torus
ネットワークサイズ	9 or 16 nodes
パケット長	128 flits（固定）
バーチャルチャネル数	2
ルーティング方法	Wormhole
トラフィックパターン	uniform

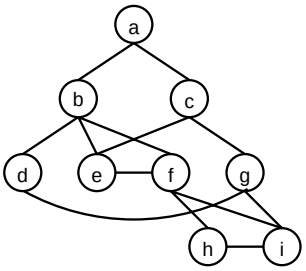

図 8 9 ノードイレギュラーネットワーク
Fig. 8 9 nodes irregular network.

レイテンシと呼び，ネットワークの性能を測る指標とする．

4.1.2 スループット：

ここでは，ネットワークのスループットを各ノードのルータがパケット中のフリットを次のノードのルータ，もしくは各ノードのプロセッサに転送する確率と定義する．すなわち，各ルータが毎クロック，フリットを転送するならば，スループットは 1.00 である．

シミュレーションに用いた条件を，**表 1** に示す．

シミュレーションで初めの 5,000 クロックはネットワークが安定せず，想定した負荷に達していないと考えられるため無視して評価を行った．1 ノードにつき 1 プロセッシングエレメントと仮定した．ネットワークのトポロジについては，イレギュラーネットワークでの性能評価としてランダムに 9 ノードを接続したもの（**図 8**）と 4×4 2D torus を用いた．Z-routing はバーチャルチャネルを 2 本必要とするので，シミュレーション条件におけるバーチャルチャネル数を 2 本（Z-routing 以外は Minimal routing を併用）として評価を行った．

図 9，**図 10** は 9 ノード，および 16 ノードにおける Z-routing の red/blue graph である．

4.2 9 ノード イレギュラーネットワーク

9 ノード イレギュラーネットワークにおける評価を，**図 11** に示す．これらの評価はリンク間の双方向チャネルが 2 本で，Z-routing 以外の各ルーティングアルゴリズムに Minimal routing を併用したときのものである．なお，グラフの x 軸はスループット，y 軸はレイテンシを表している．

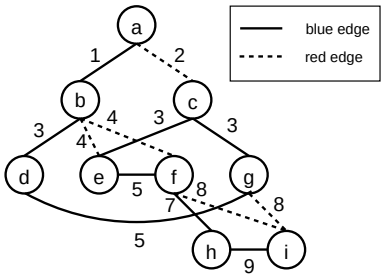

図 9 9 ノード red/blue graph
Fig. 9 9 nodes red/blue graph.

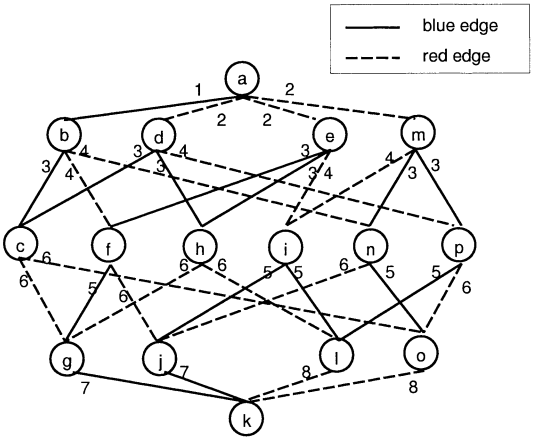

図 10 16 ノード red/blue graph
Fig. 10 16 nodes red/blue graph.

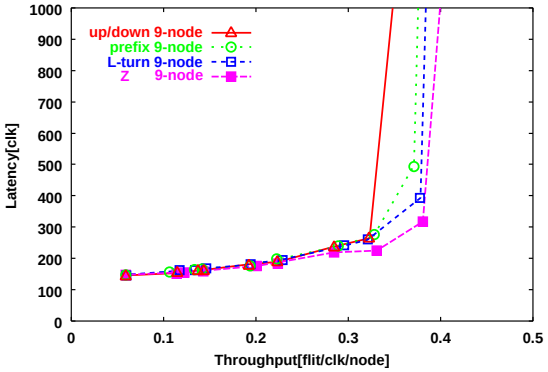

図 11 9 ノード イレギュラーネットワーク
Fig. 11 9 nodes irregular network.

図 11 より，Z-routing は Minimal routing を併用した既存のルーティングアルゴリズムに比べ，高い性能を示していることが分かる．ルーティングの際，Minimai routing を併用している既存のルーティングアルゴリズムは，Primitive up/down routing，Prefix routing，L-turn routing を original channel で使用

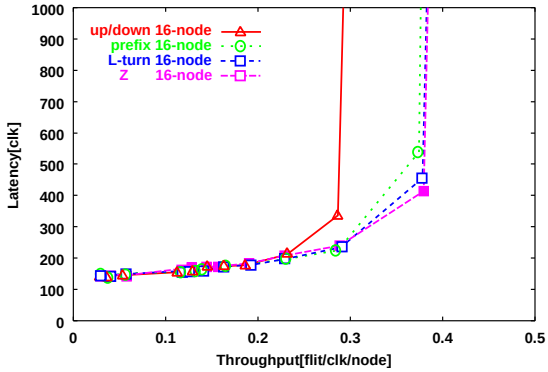

図 12 16 ノード 2 次元トーラス
Fig. 12 16 nodes 2D torus.

し，Minimal routing を new channel で使用する．new channel を使用することによってパケットは最短経路を選択することになり，自由度が増す．一方，original channel はエスケープパスとなっているため，デッドロックフリーを保証する．図 11 における，Primitive up/down routing，Prefix routing，L-turn routing の性能の差は，original channel での性能の差によって引き起こされたものである．しかし，Z-routing はチャネルに original，new の違いがなく，2 本を C_i，C_d という自由度の高いエスケープパスとして使用するため，最も高い性能を示したと考えられる．

4.3 16 ノード 2 次元トーラス

4×4 2 次元トーラスでの評価を**図 12** に示す．ネットワークサイズとトポロジを除くシミュレーション条件は，9 ノードイレギュラーネットワークと同様である．

図 12 から分かるように，Z-routing は Minimal routing を併用した既存のルーティングアルゴリズムに比べ，高い性能を示した．今回，16 ノードの評価はイレギュラーネットワークではなく，2 次元トーラスというレギュラーネットワークにおいて行った．しかし，図 12 より，Z-routing は高い性能を示し，レギュラーネットワークにおいても他のルーティングアルゴリズムに比べ高い性能を示すことが分かった．

最後に平均ホップ数の結果を**表 2** に示す．表 2 に示すとおり，Z-routing は既存のルーティングアルゴリズムと比べ，最も平均ホップ数が少ないという結果を得た．これは，Z-routing が既存のルーティングアルゴリズムに比べて自由度が高く，そのため選択可能な最短経路数がより多いためであると考えられる．

表 2 平均ホップ数
Table 2 Average hops of packets.

	9 nodes
up/down + minimal routing	1.94
prefix + minimal routing	1.85
L-turn + minimal routing	1.83
Z-routing	1.73
	16 nodes
up/down + minimal routing	2.28
prefix + minimal routing	2.18
L-turn + minimal routing	2.15
Z-routing	2.14

5. ま と め

“Z-routing” と呼ばれるイレギュラーネットワークにおける新しいルーティングアルゴリズム，および Z-routing に必要な “red/blue graph” を提案し，評価を行った．

red/blue graph の作成の時間計算量は $O(n+m)$ となり，既存のルーティングアルゴリズムの中でスパニングツリーの生成の時間計算量が最も低い up/down based routing と同じ時間計算量でグラフを生成することが可能となった．

Z-routing は辺番号を使用する際，3 つのルールのみが存在する単純なデッドロックフリーなルーティングアルゴリズムである．シミュレーション結果から，Z-routing は既存のルーティングアルゴリズム（Primitive up/down routing，Prefix routing，L-turn routing）に比べ，高い性能を示すことが分かった．

今後は，より自由度が高まるような辺の番号付けの方法の提案，Z-routing の出力チャネルの選択法（output selection function），Z-routing での最短路をより多く確保するアルゴリズムの考案，およびより多いノード数での評価を行う予定である．

謝辞 本研究の一部は，科学研究費補助金奨励研究（A）課題番号 13780226 の支援により行った．

参 考 文 献

1) Schroeder, et al.: Autonet: A high-speed, self-configuring local area network using point-to-point links, Technical Report SRC research report 59, DEC (1990).
2) Horst, R.W.: Tnet: A reliable system area network, *IEEE Micro* (1995).
3) Silla, F. and Duato, J.: Efficient Adaptive Routing in Networks of Workstations with Irregular Topology, *Proc. CANPC'97* (1997).
4) 西 宏章，多昌鷹治，工藤知宏，天野英晴：仮想チャネルキャッシュを持つネットワークルータの構成と性能，JSPP'99 論文集 (1999).
5) Silla, F. and Duato, J.: Is it worth the flexibility provided by irregular topologies in networks of workstations?, *Proc. CANPC'99* (1999).

6) Wu, J. and Sheng, L.: Deadlock-Free Routing in Irregular Networks Using Prefix Routing, DIMACS Technical Report 99-19 (1999).
7) 鯉渕道紘，舟橋 啓，上樂明也，天野英晴：Irregular Network における Adaptive Routing の提案，SWoPP2000 予稿集，pp.33–40 (2000).
8) Glass, C.J. and Ni, L.M.: Maximally Fully Adaptive Routing in 2D Meshes, *ISCA92*, pp.278–287 (1992).
9) Even, S.: *Graph Algorithms*, Computer Science Press (1979).
10) Tarjan, R.E.: *Data Structures and Network Algorithms*, SIAM (1983).

(平成 13 年 3 月 27 日受付)
(平成 13 年 12 月 18 日採録)

井川 郁哉

昭和 52 年生．平成 13 年三重大学工学部情報工学科卒業．現在，神戸大学大学院自然科学研究科情報知能工学専攻修士課程 1 年．並列計算機における相互結合網，ルーティングアルゴリズムに関する研究に従事．

舟橋 啓（正会員）

昭和 46 年生．平成 7 年慶應義塾大学理工学部電気工学科卒業．平成 12 年同大学大学院理工学研究科計算機科学専攻後期博士課程修了．工学博士．現在，三重大学工学部助手．並列計算機における相互結合網，ルーティングアルゴリズムに関する研究に従事．IEEE，IEEE-CS 各会員．