Hierarchical Dual-Net における素な経路および耐故障経路探索アルゴリズム

Arai, Jun / 荒井, 純

(出版者 / Publisher)
法政大学大学院情報科学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 情報科学研究科編 / 法政大学大学院紀要. 情報科学研究科編
(巻 / Volume)
10
(開始ページ / Start Page)
1
(終了ページ / End Page)
6
(発行年 / Year)
2015-03-24
(URL)
https://doi.org/10.15002/00011675

Hierarchical Dual-Netにおける素な経路および耐故障経路探索アルゴリズム
Disjoint-Paths and Fault-Tolerant Routing on Hierarchical Dual-Nets

荒井 純 *
Jun Arai

法政大学大学院情報科学研究科情報科学専攻
E-mail: jun.arai.4m@stu.hosei.ac.jp

Abstract

The hierarchical dual-net (HDN) was introduced as a topology of interconnection networks for ultra-scale parallel computers. The HDN is constructed by a symmetric product graph (called base network), such as three-dimensional torus and hypercube. A k-level hierarchical dual-net, HDN(B, k, S), is given by applying k-time dual constructions on the base network B. S defines a set of super-node that adjust the scale of the network. The node degree of HDN(B, k, S) is $d_0 + k$ where d_0 is the node degree of B. The HDN is node and edge symmetric and can contain huge number of nodes with small node-degree and short diameter. In this paper, we propose four efficient algorithms for finding disjoint-paths and a fault-free path on HDN. Both of the first and second algorithms find disjoint-paths on the HDN. The first algorithm has a limitation which the number of cluster in each class must equal to or greater than d_0+k, while the second algorithm eliminates this limitation. The rest of two algorithms find a fault-free path on the HDN. The third algorithm can always find a fault-free path in $O(2^k F(B))$ time with the number of faulty nodes in the HDN is less than $d_0 + k$, where $F(B)$ is the time complexity of fault-tolerant routing in B. On the other hand, the fourth algorithm can find a fault-free path on HDN with arbitrary number of faulty nodes. We performed two types of simulations for evaluating performance of our proposed algorithms.

1 まえがき

半導体の高集積化等によって，コンピュータの性能は常に著しい向上を見せてきた．2003 年に消費電力や発熱の問題から CPU の動作周波数の限界に直面してからは，コンピュータの並列処理化によって更なる性能向上がなされており，今日使われるコンピュータの大半は並列計算ができるものが一般的である．スーパーコンピュータはその時点での最先端技術を結集して開発される超高性能のコンピュータである．価格も性能も他のコンピュータとは比べものにならないほど高いが，近年では一般企業でも導入が進みつつある．その使用用途は主に核技術開発や自動車・航空機の設計やシミュレーション等大規模な科学技術計算であり，近年では分子設計や遺伝子解析等バイオ・化学分野での導入も活発化している．コンピュータの性能はいつの時代でも常に求められ続けており，その結果高性能なスーパーコンピュータの開発は国家間レベルでの競争が激化している．現在のトップクラスのスーパーコンピュータは日本の京や中国の天河二号，アメリカの Sequoia 等がある．

並列計算機の構成要素間を接続するネットワークを相互結合網と呼ぶ．現在 TOP500 のランキングトップである天河二号は，ノード数 1 万 6000 個，コア数に至っては 300 万を超える [1]．このような大規模なネットワークではノード間による通信性能がスーパーコンピュータの性能向上において重要な役割を持ち，相互結合網は活発な研究分野となっている．スーパーコンピュータのような大規模なネットワークでは各ノードの次数や直径を抑えることが性能向上のための大きな要素となる．今日のスーパーコンピュータではハイパーキューブや N 次元トーラスをベースとしたネットワークトポロジーが用いられている．これらのトポロジーは次数や直径が比較的少なく，対称性があり経路選択アルゴリズム等のアルゴリズムもシンプルに実装できる利点がある [2]．しかし，近い将来コア数が数千万〜数億に到達すると予想されており，これらのトポロジーでは要求する性能を満たせない可能性がある．

Hierarchical Dual-Net (HDN) はクラスタベースのトポロジーであり，ノード数の増加に対して次数や直径の増加がとても少ないという特徴があるため，大規模なネットワークでも次数と直径が抑えられる長所がある [3]．HDN を用いることによって，既存のスーパーコンピュータよりもケーブル数を減らした上で通信速度の向上ができる可能性がある．HDN は提案されてから間もないトポロジーであり，最短経路を探索する経路選択アルゴリズムは提案されているものの，その他の経路選択アルゴリズムは一切提案されていない．そこで本論では HDN における素な経路選択アルゴリズムや耐故障性経路選択アルゴリズムを提案しシミュレーションによる評価と考察を行った．

2 Recursive Dual-Net (RDN)

RDN とは図 1 のような再帰的な構造をしたネットワークトポロジーである [4]．RDN は 2 種類のパラメータを用いて RDN(B, k) と表記される．B はベースネットワー

*Supervisor: Prof. Yamin Li

クと呼ばれるネットワークであり B =RDN($B,0$) である．k は RDN の階層の数を示す．RDN(B,k) のノード数を N_k とすると，RDN(B,k) は $2N_{k-1}$ 個の RDN($B,k-1$) から構成されるネットワークであると表現できる．この時，それぞれの RDN($B,k-1$) のことをレベル k のクラスタと呼び，全てのクラスタは 2 つのクラス (クラス 0 とクラス 1) に分割される．図 1 の場合，上半分のクラスタがクラス 0 のクラスタで下半分のクラスタがクラス 1 のクラスタとなる．

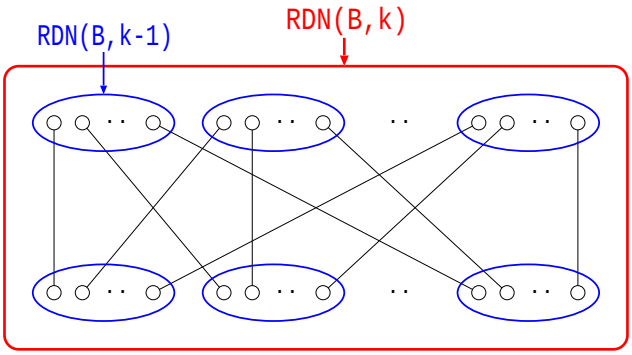

図 1: RDN の簡略図

RDN 内のノード n は 3 つの ID によって $n(n_0,n_1,n_2)$ と表記される．n_0，n_1，n_2 をそれぞれクラス ID，クラスタ ID，ノード ID と呼ばれる．クラス ID はノードがどちらのクラスに属するかを，クラスタ ID はノードがクラス中のどのクラスタに属するかを，ノード ID はノードがクラスタ中のどのノードであるかを示す．全てのノードは異なるクラスへのエッジを 1 本だけ持ち，そのエッジで繋がる異なるクラスに存在する 2 つのノード $u(u_0,u_1,u_2)$ と $v(v_0,v_1,v_2)$ の間には $u_0 \neq v_0$ かつ $u_1 = v_2$ かつ $u_2 = v_1$ という関係がある．RDN の特徴はノードの増加量に対して各ノードの次数や直径の増加量が少ないことや階層を増やしてもネットワークの対称性を維持できること，経路選択アルゴリズム等のアルゴリズムを再帰的に実装できること等がある．

3 Hierarchical Dual-Net (HDN)

RDN はレベルの増加によるノード数の増分が非常に大きいためノード数の調整が難しい．また，あるクラスタと異なるクラスにあるクラスタを繋ぐエッジが一本しかないため，そのエッジが故障すると迂回路の経路長が大きくなってしまう．HDN は RDN を改良したネットワークトポロジーであり，RDN の長所を残しつつこれらの欠点を解消することに成功している．

図 2 は HDN の一例である．レベル k の HDN は三種類のパラメータを用いて HDN(B,k,S) と表現される．B と k は RDN と同様にベースネットワークとレベルをそれぞれ意味する．$S=\{S_1,S_2,...,S_k\}$ は各レベルのスーパーノードである．スーパーノードはいくつかのノードからなる小規模なネットワークであり，スーパーノードの大きさが各レベルのクラスタやクラスタ内のノード数等に影響する．これらのパラメータを変更することによって，図 2 の HDN のように数十から数百ノードで構成される小規模なものから百万以上のノード数で構成される大規模なものまであらゆるネットワークを構成することができる．

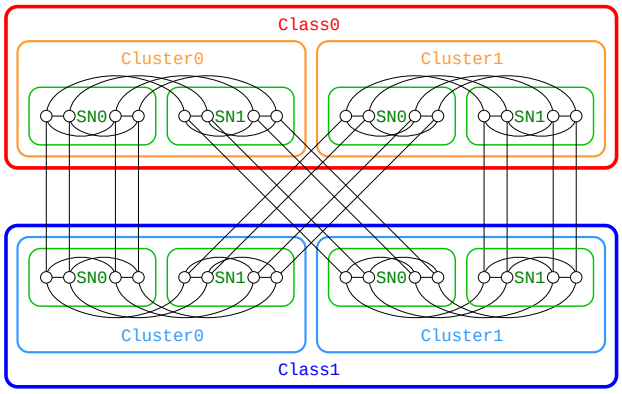

図 2: HDN(H_3,1,4) の例

HDN の全てのノードは RDN と同様に 2 つのクラスに分割され，各クラスは複数のクラスタで構成される．更に，各クラスタは複数のスーパーノードで構成される．よって，HDN のノードは 4 種類の ID を用いて $n(n_0,n_1,n_2,n_3)$ と表記できる．各 ID は順番にクラス ID，クラスタ ID，スーパーノード ID，ノード ID と呼ばれ，クラス ID は RDN と同様に 0 か 1 で定義される．クラスタやスーパーノードの数は HDN のパラメータによって変化するが，各クラス内のクラスタ数と各クラスタ内のスーパーノード数は必ず一致する．それらの数を仮に $N_{cluster}$ と置くと，クラスタ ID とスーパーノード ID は 0 から $N_{cluster}-1$ までの整数で定義される．全てのノードは異なるクラスと繋がるエッジをただひとつ持つ．そのエッジで繋がる異なるクラスに存在する 2 つのノード $u(u_0,u_1,u_2,u_3)$ と $v(v_0,v_1,v_2,v_3)$ の間には $u_0 \neq v_0$ かつ $u_1 = v_2$ かつ $u_2 = v_1$ かつ $u_3 = v_3$ という関係がある．つまり，同じスーパーノードにあるノード同士は必ず異なるクラスの同じスーパーノードへのエッジをそれぞれ持ち，各クラスタは異なるクラスの全てのクラスタへのエッジを少なくともひとつずつは持つ．

RDN と HDN には「全てのレベルでスーパーノードのノード数が 1 であるとき，HDN(B,k,S) = RDN(B,k) となる.」という関係性がある．また，HDN は複数の RDN をそれぞれ繋ぎ合わせて得られるネットワークトポロジーであるとみなすことができ，レベル k のスーパーノードのノード数と組み合わせる RDN の数が必ず一致する．図 2 の HDN の場合，スーパーノードのノード数が 4 であるためこの HDN は図 3 のように 4 つの RDN から構成されたネットワークトポロジーであると考えられる．

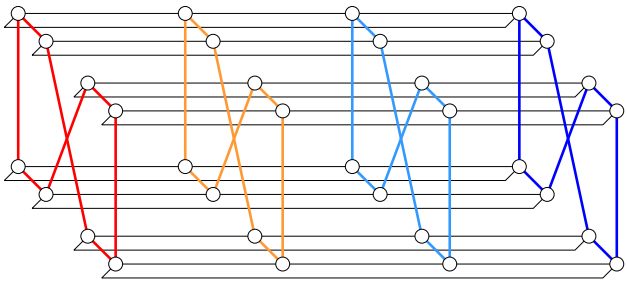

図 3: HDN は複数の RDN で構成される

4　素な経路探索アルゴリズム

素な経路とは互いに交わらない経路の事である．本論で提案する素な経路探索アルゴリズムは対称性を持つ HDN に対して有効であり，$d_0 + k$ 本の素な経路が探索される．素な経路は，一方の経路が故障等により使えなくても他方の経路に問題がなければ通信ができるため，耐障害性の面で強みを持つ．

HDN における素な経路探索アルゴリズムの入力は対称性を持つ HDN と HDN 内の 2 つのノード $u(u_0, u_1, u_2, u_3)$ と $v(v_0, v_1, v_2, v_3)$ の 3 種類である．これらの情報が入力された時，アルゴリズムは u と v を繋ぐ素な経路を探索し結果を出力する．本論では状況の違いに応じて使い分けることができる 2 種類の素な経路探索アルゴリズムを提案した．

4.1　クラスタ分散による素な経路探索アルゴリズム

このアルゴリズム基本的な考え方は，それぞれの経路を異なるクラスタに分散させることにより，分散後の経路同士で自由に経路を選択することができるというものである．ただし，ノードの次数が各クラスのクラスタ数よりも大きい場合，全ての経路を異なるクラスタに分散させることは不可能となる．よって，このアルゴリズムはノードの次数が各クラスのクラスタ数未満でなければならない．アルゴリズムの挙動は入力された 2 つのノードの位置関係によって大きく 3 通りに分かれる．2 つのノードが同じクラスタにある場合は Case A，異なるクラスにある場合は Case B，同じクラスの異なるクラスタにある場合は Case C としてそれぞれ扱う．

図 4 は Case A における素な経路の探索イメージ図である．探索イメージ図であるため，HDN(B, k, S) 内の各クラスタは円で簡略化されている．ノード u と v は図中の赤色で塗りつぶされたクラスタの内部に存在するものとする．Case A では赤色のクラスタ内部でアルゴリズムを再帰的に呼び出すことによって $d_0 + k - 1$ 本の素な経路の探索ができる．残り 1 本の経路は図中の太線の経路のように他のクラスタを経由して繋ぐ必要がある．この経路は他の経路との衝突を考慮する必要がないため，[3] のアルゴリズムで問題なく経路の探索ができる．

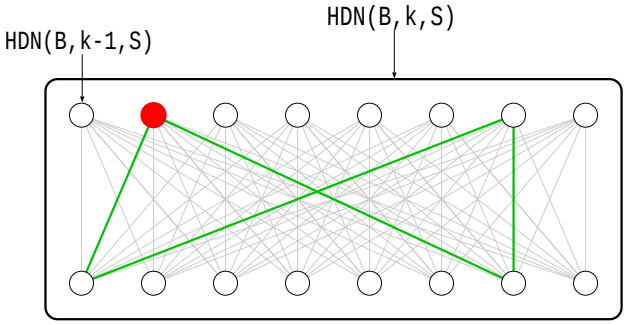

図 4: Case A における素な経路の探索イメージ

図 5 は Case B における素な経路の探索イメージ図である．ノード u と v はクラスタ R とクラスタ B の内部にそれぞれ存在するものとする．Case B ではそれぞれのノードから分散を行うと図 5 のようにオレンジの分散経路と水色の分散経路がそれぞれ作られる．各クラスタには異なるクラスにあるクラスタへのエッジが必ず存在するため，[3] のアルゴリズムでそれぞれの分散経路同士を繋ぐことによって素な経路の探索が完了する．

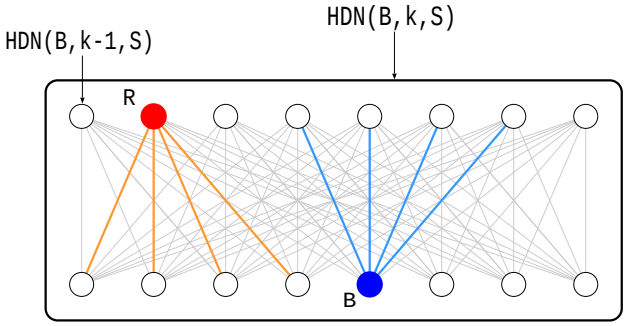

図 5: Case B における素な経路の探索イメージ

図 6 は Case C における素な経路の探索イメージ図である．図 5 と同様にノード u と v はクラスタ R とクラスタ B の内部にそれぞれ存在するものとする．Case C では各ノードから分散を行うと図 6 のように太線の分散経路がそれぞれ作られる．黒丸のクラスタのように両方の分散経路が含まれる場合は，クラスタの内部で [3] のアルゴリズムを使って経路同士を繋げればよい．最後に余った分散経路同士を極太線の経路のように繋ぎあわせると素な経路の探索が完了する．

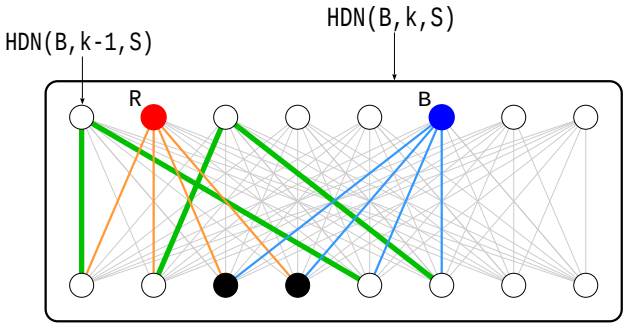

図 6: Case C における素な経路の探索イメージ

4.2　RDN 分散による素な経路探索アルゴリズム

このアルゴリズムは HDN が複数の RDN で構成されることに着目し，経路を異なる RDN に分散させることによって，経路の探索を行うアルゴリズムである．経路をそれぞれの RDN に分散させることによって，分散後の経路は RDN の経路選択アルゴリズム（[4] のアルゴリズム）および RDN の素な経路探索アルゴリズム（[5] のアルゴリズム）が利用できる．このアルゴリズムは先ほどのアルゴリズムにあった制限を取り除くことに成功している．アルゴリズムの挙動を図 7-8 を用いて説明する．図 7 のようにノード u と v が同じ RDN に存在する場合は比較的簡単に素な経路の探索が可能である．RDN の次数を r とする．このとき u，v を含む RDN で [5] のアルゴリズムを使うと真ん中の極太線の経路のように r 本の素な経路が探索できる．残りの $d_0 + k - r$ 本ずつあ

る太線の経路はいずれもノード u と v を含む RDN と隣接した RDN に繋がっているため，それぞれの隣接した RDN 内で [4] のアルゴリズムを用いることによって図中の左右の極太線の経路のように繋げることができる．アルゴリズムの詳細は [6] に載っている．

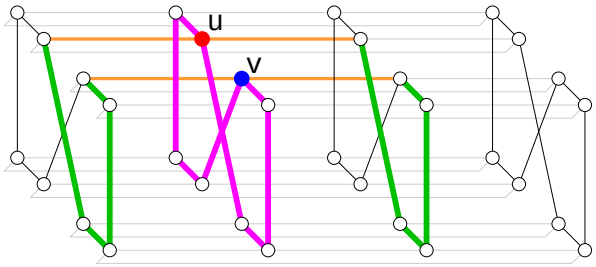

図 7: RDN 毎の分散による素な経路探索イメージ 1

一方，図 8 のようにノード u と v が異なる RDN に存在する場合は図 7 の場合に比べて処理が複雑になる．このような場合は最初にノード u からノード v を含む RDN への経路をそれぞれのスーパーノード内で合計 r 本作る．これによって得られたノード u を出発点とする太線の分散経路とノード v に対して [5] のアルゴリズムを使うと右側の極太線の経路のように r 本の互いに素な経路が得られる．残りの d_0+k-r 本の分散経路は，一方の分散経路をもう一方の分散経路を含む RDN へ移動させてから，移動後の RDN 内で [5] の耐故障経路探索アルゴリズムを使うことで左と真ん中の極太線の経路のように繋ぐことができる．

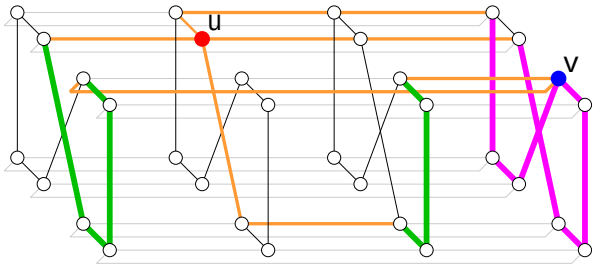

図 8: RDN 毎の分散による素な経路探索イメージ 2

5 耐故障経路探索アルゴリズム

耐故障経路探索アルゴリズムはネットワーク内の故障しているノードやエッジを使用しない迂回路を探索するアルゴリズムである．本論で提案する耐故障経路探索アルゴリズムはノードの故障のみを考慮したアルゴリズムである．一般的にノードやエッジの故障を一切考慮せずに最短経路を求めるアルゴリズムの場合は，ネットワーク全体の故障率の増加に対して通信成功率が大きく減少してしまう．これに対して，耐故障経路探索アルゴリズムは故障ノードを考慮したアルゴリズムであるため，ネットワーク全体の故障率の増加に対して通信成功率の減少は緩やかになる．本論文では，ネットワーク中のノード故障率に応じて使い分けることのできる 2 種類のアルゴリズムを提案し，それぞれのアルゴリズムの内容や特徴を説明する．

5.1 低故障率における耐故障経路探索アルゴリズム

このアルゴリズムは，ネットワーク全体のノード故障数が d_0+k よりも少ない場合にしか適用できない代わりに，確実に耐故障経路を見つけ出すことができるアルゴリズムである．このアルゴリズムの基本となる考え方は，ノード u と v を故障率が低い RDN に移動させ，RDN 内で [5] のアルゴリズムを使って探索するというものである．故障率が低い RDN を選ぶ理由は，[5] のアルゴリズムはネットワーク内の故障ノード数が r 以上の場合，耐故障経路が見つけ出せない可能性があるからである．

このアルゴリズムは 3 つのステップから構成される．ステップ 1 では，以下の 2 つの条件を全て満たすノード $u_s(u_{s_0},u_{s_1},u_{s_2},u_{s_3})$ と $v_s(v_{s_0},v_{s_1},v_{s_2},v_{s_3})$ を探し，ノード u と u_s を繋ぐ経路，ノード v と v_s を繋ぐ経路をそれぞれ探索する．

1. ノード u_s はノード u と同じノードか隣接ノードのいずれかである．
2. ノード u_s を含む RDN 内の故障ノード数が r 未満である．

ステップ 2 では，以下の 3 つの条件を全て満たすノード $u_r(u_{r_0},u_{r_1},u_{r_2},u_{r_3})$ と $v_r(v_{r_0},v_{r_1},v_{r_2},v_{r_3})$ を探し，ノード u_s と u_r を繋ぐ経路，ノード v_s と v_r を繋ぐ経路を [5] のアルゴリズムでそれぞれ探索する．ノード u_s と v_s の条件から [5] のアルゴリズムで必ず耐故障経路を探索できることが保証されている．

1. ノード u_s とノード u_r が同じ RDN に存在する．つまり $u_{s_3}=v_{s_3}$ が成り立つ．
2. ノード u_r と v_r が同じスーパーノードに存在する．つまり $u_{r_0}=v_{r_0}$ かつ $u_{r_1}=v_{r_1}$ かつ $u_{r_2}=v_{r_2}$ が成り立つ．
3. ノード u_r を含むスーパーノード内の故障ノード数が d_0+k-r 未満である．

ステップ 3 では，ノード u_r と v_r をスーパーノード内で繋ぎあわせる．以上のステップによって得られた経路を全て繋ぎあわせた経路 $[u\to u_s\to u_r\to v_r\to v_s\to v]$ がノード u と v を繋ぐ耐故障経路である．

図 9 はこのアルゴリズムの探索イメージ図である．図中の赤色のノードがノード u，青色のノードがノード v であり，故障ノードは黒で塗りつぶされている．最初にノード u_s と v_s までの経路を探索する．図 9 の場合，ノード u を含む RDN には故障ノードが存在しないため $u_s=u$ となる．一方，ノード v を含む RDN には故障ノードが 2 つ含まれているため $v_s\neq v$ となる．そのため，アルゴリズムは v と隣接したノードの中から適切なノードを選択し v_s を決定する．u_s と v_s までの経路の探索が終わると，アルゴリズムは u_r と v_r までの経路の探索を開始する．ノード v_s を含むスーパーノードには故障ノードが存在しないため，$v_r=v_s$ となる．一方，ノード u_s から直接ノード v_s を含む RDN へ移動することができないため，$u_r\neq u_s$ となる．このような場合，ノード u_s から v_r を含むスーパーノードへの経路を [5] のアルゴリズムで探索する．最後に u_r と v_r を結ぶ経路をスーパーノードの内部で探索し，得られた 5 種類の経路 $[u\to u_s]$，$[u_s\to u_r]$，$[u_r\to v_r]$，$[v_r\to v_s]$，$[v_s\to v]$ を繋ぎあわせると耐故障経路の探索が完了する．

このアルゴリズムは，ノード故障数が HDN のリンク数よりも少ない場合に必ず耐故障経路が見つかり，最大距離が $D(f(S_k))+2(2^{k-1}D(B)-\sum_{j=0}^{k-2}2^jD(SN^{k-1-j})-D(S_k))+2^{k+1}+3$ である耐故障経路を最大時間計算量 $O(2^kF(B))$ で探索できることが証明されている [7].

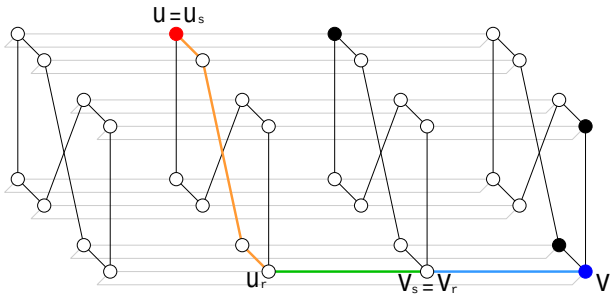

図 9: 故障ノード数が r より少ない HDN の耐故障経路探索イメージ

5.2 任意の故障率における耐故障経路探索アルゴリズム

このアルゴリズムはネットワークの任意の故障率に対して適用できるアルゴリズムである．先ほどのアルゴリズムと同様に，このアルゴリズムも 3 つのステップによって構成され，[5] のアルゴリズムを用いる．ステップ 1 では，隣接した RDN から最もノード故障率が少ない RDN を探し，その RDN に到達する経路を探索する．ステップ 2 では，RDN 内で [5] のアルゴリズムを使って 2 つの経路を同じスーパーノードに移動させる．故障ノードの兼ね合いによってはこの時点で耐故障経路が探索に失敗する可能性がある．ステップ 3 では，スーパーノード内で 2 つの経路を繋ぎあわせるが，これも故障ノードの兼ね合いによっては探索が失敗する可能性がある．このアルゴリズムの詳細は [7] に載っている．

6 シミュレーション・実験

提案したアルゴリズムの性能を評価するために，各アルゴリズムによる通信の 2 種類のシミュレーションを行った．片方は素な経路探索アルゴリズムのシミュレーションであり，本論で提案した素な経路探索アルゴリズムを使用して探索された経路の通信成功率や経路の長さ等を調べている．もう一方は耐故障経路探索アルゴリズムのシミュレーションであり，本論で提案した耐故障経路探索アルゴリズムを使用して探索された経路の通信成功率等を調べている．

6.1 素な経路探索アルゴリズムのシミュレーション結果

本シミュレーションは 3 次元のハイパーキューブをベースネットワークとして利用したレベルの HDN$(B,2,S)$ を使って行った．この HDN のノード数と次数はそれぞれ 1024，5 である．よって，本シミュレーションでは最大 5 つの素な経路が探索される．シミュレーションの具体的な方法は次の通りである．

1. ランダムに 2 つのノードを選択する．
2. （総ノード数 × 故障率）の分だけランダムに選んだノードを故障させる．ただし，1 で選んだノードは故障させないようにする．
3. 1 で選んだノードを繋ぐ素な経路を提案したアルゴリズムで探索する．
4. 結果を記録してから，故障したノードを全て修復する．
5. 試行回数に達するまで 1〜4 を繰り返し実行する．

図 10 は通信成功率に関して Case 毎にシミュレーションを結果をまとめたものである．ここでは得られた経路中に故障ノードが含まれていなければ通信成功とし，通信成功率は探索された素な経路のうち少なくとも 1 本以上の経路の通信が成功する確率でと定義した．Case 毎に比較してみると，Case A の成功率が他の Case に比べて大きく上回るという結果になった．

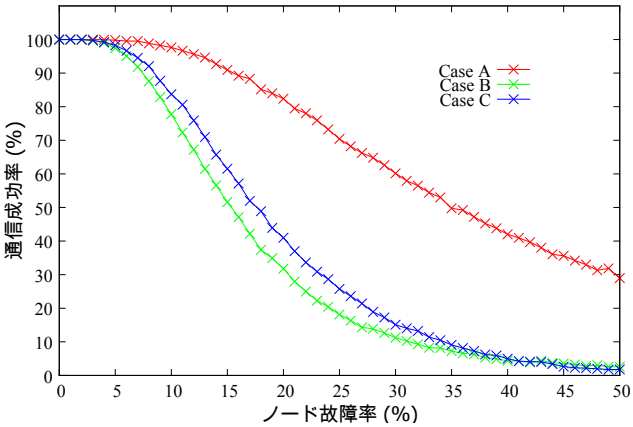

図 10: 通信成功率に関するシミュレーション結果

図 11 は通信が成功した経路の平均経路長を Case 毎にまとめた結果である．ここでいう経路長とは，経路中にあるエッジの本数を意味する．結果を見ると，故障率の増加に対して平均経路長が減少しているが，これは故障率が高いと経路長の長い経路の成功率が低くなるためと思われる．Case 毎に比較してみると，クラスタ内だけでほとんどの経路が作れる Case A の平均経路長が他のケースに比べて短くなっている．

6.2 耐故障経路探索アルゴリズムのシミュレーション結果

本シミュレーションは $3\times2\times5$ の 3 次元トーラスをベースネットワークとしたノード数が 5760 であるレベル 2 の HDN$(B,2,S)$ を使って行った．シミュレーションの具体的な方法は次の通りである．

1. ランダムに 2 つのノードを選択する．
2. （総ノード数 × 故障率）だけランダムに選んだノードを故障させる．ただし，1 で選んだノードは故障させないようにする．
3. 1 で選んだノードを繋ぐ耐故障経路を提案したアルゴリズムと [3] のアルゴリズムでそれぞれ探索する．
4. 結果を記録してから，故障したノードを全て修復する．
5. 試行回数に達するまで 1〜4 を繰り返し実行する．

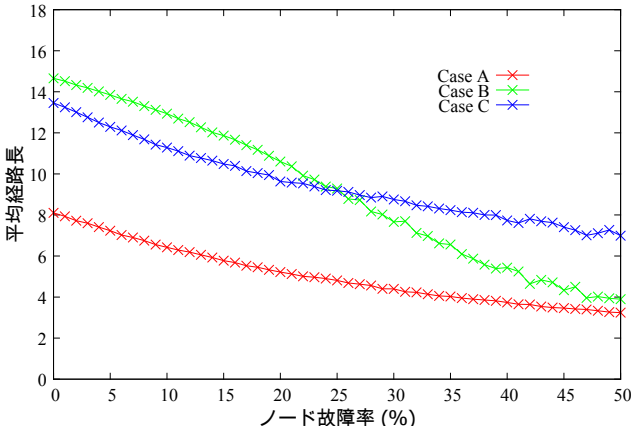

図 11: 平均経路長に関するシミュレーション結果

図 12 はそれぞれのアルゴリズムによる通信成功率をまとめたものである．図中の赤線が提案したアルゴリズムによる通信成功率，青線が [3] のアルゴリズムによる通信成功率である．[3] のアルゴリズムは故障ノードを考慮していないため，成功率が故障率の増加に対して大きく低下しており，故障率が 10%で成功率が 50%を下回っている．一方，提案したアルゴリズムの成功率は故障率の増加に対して緩やかに減少しており，故障率 20%でも成功率が 99.9%以上で，故障率 25%でも成功率は 98%を上回っている．

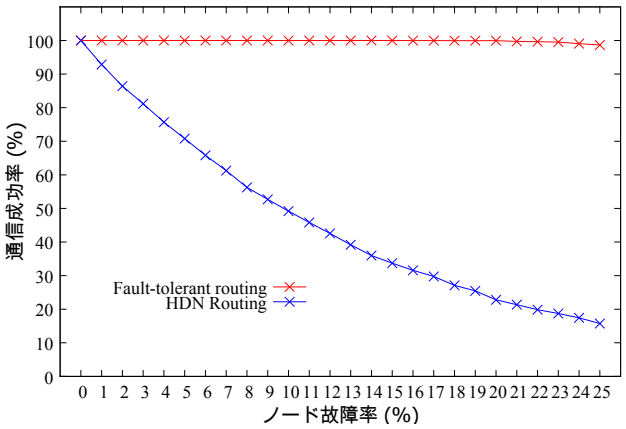

図 12: 通信成功率に関するシミュレーション結果

7 結論

本論では，HDN における素な経路探索アルゴリズムと耐故障経路探索アルゴリズムをそれぞれ 2 つずつ提案し，シミュレーションによる性能評価を行った．

本論で提案した素な経路探索アルゴリズムのうち，HDN におけるクラスタ分散による素な経路探索アルゴリズムには，HDN の各クラスのクラスタ数が $d_0 + k$ よりも少ないと使用することができないという制限があるが，HDN における RDN 分散による素な経路探索アルゴリズムでは，その制限を無くすことに成功し，任意の HDN に対して素な経路の探索が行えるようになった．

本論で提案した 2 種類の耐故障経路探索アルゴリズムは HDN のノード故障率に応じて使い分けることができる．低故障率の HDN における耐故障経路探索アルゴリズムは，HDN 内の故障ノード数が $d_0 + k$ よりも少ない時にしか使用することができないが，最大時間計算量 $O(2^k F(B))$ で必ず耐故障経路の探索ができることが保証されている．もう一方の HDN における耐故障経路探索アルゴリズムは，耐故障経路を確実に見つけ出すことはできないものの，任意のノード故障率に対して使用できる実用的なアルゴリズムである．

シミュレーションでは，探索された経路の通信成功率や平均経路長に着目し，各結果について考察を行った．シミュレーションの結果，それぞれのアルゴリズムによる通信成功率や各 Case の傾向が把握できた．更に，我々の提案したアルゴリズムが，故障ノードを考慮しない従来のアルゴリズムに比べて耐故障性の面で大きく優れたアルゴリズムであることを示すことが出来た．

今後の課題として，提案した各アルゴリズムの更なる性能向上や，HDN が既存のスーパーコンピュータに使われているトポロジーをはじめとした他のトポロジーとの性能比較が考えられる．スーパーコンピュータの実運用を踏まえた詳細な現実性のあるシミュレーションを行おうとすると，今までに我々が提案したアルゴリズムだけでは要求される性能を満たせない可能性がある．このような状況において，Node-to-Set の素な経路探索アルゴリズムや Set-to-Set の素な経路探索アルゴリズム等が，我々の提案したアルゴリズムよりも役に立つ可能性がある．よって，これらのアルゴリズムの提案が今後の特に重要な課題になると思われる．

参考文献

[1] TOP500. *Supercomputer Sites*. http://top500.org/, Nov. 2014.

[2] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, pages 867–872, 1988.

[3] Y. Li, S. Peng, and W. Chu. Hierarchical dual-net: A flexible interconnection network and its routing algorithm. In *Proceedings of the Second International Conference on Networking and Computing*, pages 58–67, Osaka, Japan, Nov. 2011.

[4] Y. Li, S. Peng, and W. Chu. Recursive dual-net: A new universal network for supercomputers of the next generation. In *Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'09)*, pages 809–820, Taipei, Taiwan, June 2009.

[5] Y. Li, S. Peng, and W. Chu. Disjoint-paths and fault-tolerant routing on recursive dual-net. In *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 48–56, Hiroshima, Japan, Dec. 2009.

[6] J. Arai and Y. Li. Disjoint-path routing on hierarchical dual-nets. *International Journal of Networking and Computing*, 4(2):260–278, July 2014.

[7] J. Arai and Y. Li. Fault-Tolerant Routing Algorithms for Hierarchical Dual-Nets with Limited and Arbitrary Number of Faulty Nodes. In *Proceedings of the Second International Symposium on Computing and Networking*, Shizuoka, Japan, Dec, 2014.