DCA ルーティング機能の改善 ——Distance Vector ルーティング・アルゴリズムをブロードキャスト・ネットワークに適用した際の問題点とその解決策

Discussion on Enhancement of Unisys DCA Routing Functionality——Problems and Resolution in Applying Distance Vector Routing Algorithm to broadcast

小　西　　宏

要　約　大規模ネットワーク障害時に通信経路の復旧が遅延する現象を，ルーティング・アルゴリズムの性質とネットワークトポロジの関係から論ずる．FDDI 幹線などのバス・メディアで構成されたブロードキャスト・ネットワークにおいては，バースト的に増大する経路情報数が遅延の原因であるが，それはノードの密結合構造に関係している．この構造においてはいかなるルーティング手法も根本的な軽減策にならないため，構造認識の変革を必要とすることを導く．

そしてこの隣接性は実際のトポロジとは異なりマルチキャスト機能により与えられている仮想的なものであることから，ブロードキャスト・ネットワーク上の各中継ノードが隣接関係にある全てのノードの集合を一個の仮想ノードに対してスター状に配置されたものとして捉え直し，隣接ノードとの関係を仮想ノードとの関係に置換してルーティングを行なえば，大幅な経路情報と検索負荷の削減が可能になることを提示する．

Abstract　In case of failure of a large-scale network, sometimes telecommunication lines would be postponed to be fully recovered. This paper examines the postponement of network recovery at the viewpoint of the nature of the routing algorithm and its dependency on the network topology. In a broadcast network including numerous bus cables such as FDDI cables, the burst of routing information in a period, resulting from the fully connected topology (FCT) among neighbors, would cause the recovery of routes to take a lot of time. This paper also shows that any technology never helps to reduce the routing information across FCT so that the change of understanding of the network topology must be required.

The adjacency of every neighbor in FCT does not exist physically in the actual topology, but is logically recognized only by the multicasting functionality. In conclusion, this paper shows the possibility to reduce redundant routing information and load of searching for optimum routes on the assumption that an individual routing node in the broadcast network recognizes a set of its neighbors as nodes connected to a pseudo node in a star topology and performs the routing control based on the pseudo node relation instead of neighboring node one.

1. はじめに

最近ネットワークの基幹部分にはマルチキャスト型高性能メディア（FDDI/高速 Ethernet）の使用が一般化し，その帯域と接続能力の向上につれシステム規模も増加したことで，データの発生がバースト的な振舞いを示すようになった．それは時に通信制御機能が持つ処理能力を圧迫するまでに至ることがあるが，このピークは多くの

場合障害時，あるいは復旧時のルーティング情報再構築時において顕著に現れる．これはルーティングがいかにシステム形態と規模に依存するものであるかを示すと同時に，障害復旧の遅延及び二次障害誘発の危険性を暗示するものである．したがって今後通信インフラストラクチャの性能向上とともにシステムの巨大化が相伴って継続していく上で，今後想定されるシステム形態と規模に則した既存ルーティング機能の見直しが必要な段階にあると思われる．

本稿は米国ユニシスが開発した DCA ルーティング機能を題材に，マルチキャスト型通信インフラストラクチャの高性能化に対応した今後の DCA ルーティング機能のあり方について考察する．

2. DCA ルーティングの概要

2.1 DCA ルーティングの背景

本節では一般的なルーティングの種類とその歴史的背景を簡単に説明し，そこから DCA ルーティングを概観する．

2.1.1 標準ルーティング・プロトコル

TCP/IP プロトコル・スタックを使用したオープン・ネットワーク環境のルーティングには一般に RIP（Routing Information Protocol）が使用される．このアルゴリズムの原型はネットワーク最大フロー解法に用いられる最短経路選択手続きであり Distance Vector ルーティング・アルゴリズムと呼ばれる．RIP は Berkley UNIX，XEROX の XNS の一部機能として開発されたが，1969 年以降 ARPANET においても使用されるようになり，1988 年に IETF（Internet Engineering Task Force）により正式に標準化された（RFC 1058）．しかし 1980 年代の中頃から巨大化する Internet 環境に RIP が適応できなくなると予測されたため，IETF は RIP の機能を一部拡張したが(RIP-2)，その一方で RIP の後継プロトコルの標準化を 1988 年に開始した．ISO は，BBN 及び DEC が 1970 年代に開発したルーティング・アルゴリズムを基に IS-IS Intra-Domain routing protocol（ネットワーク・システムにおける中継ノード間のルーティング・プロトコル）の標準化作業を行った．そのアルゴリズムには Dijkstra による SPF（Shortest Path First）が採用されているが，それは有向グラフ表現されたトポロジにおける最短経路選択アルゴリズムである．このルーティング方式は 1991 年に OSI 標準（ISO 10589）として制定されたが，同時に IETF による RIP の後継（OSPF）としても標準化され（RFC 1247），これが現在最も体系化され利用されているルーティング方式である．今後のルーティングは ISO が検討している n 対 n ユーザ間を通信する Multi-peer-Multicast 転送構造のように，ATM などの広帯域網上に直接構成し既存ネットワーク層のボトルネックを回避する，即ち伝送メディアの機能に依存する傾向が強まり，ソフトウェア・アルゴリズムとしては縮退する方向にある．その他 ISO の Inter-Domain routing protocol（IETF の EGP に相当する）のように Domain（ネットワーク・システム）間のルーティングも存在するが，それは Domain の上位に管理者を設定して，各管理者間で情報交換する階層化ルーティング方式であるため，アルゴリズムはスタティック・ルーティングに近く規模の割に(情報量が多いがゆえに）単純な方式であるといえる．

2.1.2 DCA ルーティング・プロトコル

DCA は 1976 年に UNIVAC 固有の通信アーキテクチャとして開発されて以来，独自のネットワーク層プロトコル・モジュール（DUC/TSTN）とルーティング・モジュール（RTC）を使用してきたが，1989 年に OSI 製品の提供や LAN などの通信メディアで構成されたオープン・ネットワーク環境への対応を目的として，新規に OSI のネットワーク・プロトコル（ISO 8473）に準拠したネットワーク層プロトコル・モジュール（ルーティングを含む）が開発された（DNS：DCA Network Services）．しかしながら DNS の使用するルーティング・アルゴリズムは IS-IS Intra-Domain routing protocol に準拠しておらず，アルゴリズムは Distance Vector 型の独自なものを採用した．

2.2 ルーティングの概念

本節では DCA ルーティングを含めた Distance vector ルーティング・アルゴリズムを使用する標準的なルーティングを理解する上で必要となる基本概念を説明する．

2.2.1 ネットワークのグラフ表現

ネットワークの接続構造はグラフ G＝(N，A)

- N は通信制御装置に対応するノードの集合
- A は回線に対応する弧の集合

により表現される．つまり任意のノード v 1∈N に対して Adj(v 1)＝{w∈N｜(v 1，w)∈A} を定義すれば，Adj(v 1) は弧(v 1，w) により 1 対 1 に対応する．このような隣接構造において，ノード v 1∈N のすべての Adj(v 1) をリスト化したものを隣接リストという．ネットワークにおける経路とはノードの系列 (v 1，v 2，v 3，..，vn) において，任意の vi，vi＋1 に隣接関係 vi＋1＝Adj(vi) が成り立つものである．また通信回線は接続関係以外に伝送遅延等の物理的な振る舞いを持つ．この情報をコスト C(vi，vi＋1) として，弧 Ai＝(vi，vi＋1) に重み付けする．経路間の弧のコストの総和を距離と呼ぶ．コストを一律であるとすれば，距離は弧の数（ホップ・カウント）に比例する．ルーティングとは，発信ノード v 1 から目的ノード vn への経路において距離が最短になるもの，すなわち最短経路を決定する手続きである．経路(v 1，v 2，v 3，..，vn) は (v 1，vn)，または v 1 の視点から (Adj(v 1)，vn) で表す．

2.2.2 spanning tree

グラフ G＝(N，A) の各弧にコストが付加されており，その接続構造が各ノードの隣接リストによって表現されているものとする．この場合，G が任意の経路(v 1，..，vn)において，v 1≠vn となる構造を有するとき，G を木という．ノードの集合 N(G) において，s∈N(G) としたとき，順序対 T＝(G，s) を根つき木といい，s を T の根という．s から r∈T への経路が存在し，q がこの経路上のノードである場合，s から見て r は q に対し up tree 方向，q は r に対し down tree 方向にあるという．

特に q＝Adj(r) ならば，r は q の子，q は r の親といい，それぞれ r＝C(q)，q＝P(r) と表わす．また子を持たないノードを葉，葉でないノードを内点という．任意の s∈N(G) は隣接リストをもとに自身を根，他の x∈N(G) を葉または内点とし，且つ経路 (s，n) のコストが最小となる spanning tree(すべての n∈N(G) を接続する木) を生成することにより，最短経路を把握することができる．つまりルーティン

グ・アルゴリズムとはこの spanning tree の生成に他ならない.

2.3 SPF による spanning tree の生成

spanning tree を生成するアルゴリズムとしてまず SPF(以後手続き名称として Dijkstra と略記) を示す. これはコストが最小のノード v に接続する弧(v, w) を優先して探索し, 枝を生成する方法である.

```
procedure Dijkstra;
begin
{隣接情報検知処理}
 ノードの集合 V から始点となるノード s を除く;
経路の集合 T を空集合とする;
for s の隣接リストのノード x do                                    i1
    begin
    if 弧(s,x)∈弧の集合 A である then                              i2
        距離(s, x) := コスト(s, x)                                  i3
    else
        距離(s, x) := ∞;                                           i4
    end;

{経路情報更新処理}
whileVが空集合ではない do
    begin
    Vの要素のうち距離(s, v)が最小であるノード v を選択する;        1
    弧(P(v), v)を T の要素とする; {経路(s, v)が確定する}           2
    V からノード v を除く;                                         3


    for v の隣接リストに存在する各ノード y do                      4
        begin
        if 距離(s, y) > 距離(s, v) + コスト(v, y) then             5
            begin
            距離(s, y) := 距離(s, v) + コスト(v, y);               6
            P(y) := v;                                             7
            end;
        end;
    end;
end;                                          [注 右端のインデックスは step #]
```

図 1 に示すネットワークにおいて, s はまず隣接するノードのうちで距離(s, v) が最小のノード v を選択する. これが最短経路であることは自明である. その場合ノード v に隣接するノード y を隣接リストから検索し, 距離(s, v) とコスト(v, y) との和をその時点までに生成された距離情報と比較する. もし y が s の隣接ノードの集合の要素でなければ step 6 が実行され距離(s, y) が設定されるが, さらに s の隣接ノード q が y に隣接し, 距離(s, q)+コスト(q, y)<距離(s, y) となる可能性があるため最短経路としては確定されない. つまり経路(s, y) は暫定経路であり, 設定される T の枝(P(y), y) は暫定枝である. 他の隣接ノード q, x についても step 5 から step 7 の処理を終えた段階で, 終点が V に存在し, 終点の親が T の要素である暫定経路が生成される. したがって次に step 1 に回帰した時, V 中の最短の暫定経路が正規経路として確定する. もしさらに短い経路の終点 p∈V が存在すると仮定すれば P(p)∈T であるが, p≠s かつ p≠Adj(s) となり矛盾する.

これらの枝について次の命題が成立する．

- 生成される枝 l=(v，y)∈T に対して，最短距離(s，y)＝最短距離(s，v)＋コスト(v，y) である．
- 横断枝（始点，終点の間に親子の関係がない枝）l=(q，y) は T の要素ではない．つまり経路として生成されない．この場合最短距離(s, y)≦最短距離(s, q)＋コスト(q，y) である．
- したがって弧 l=(P(y)，y)∈T は発信ノード s を根とする spanning tree の P(y) から分岐した up tree 方向の枝となる．即ち s，y 間の最短経路は根 s から葉 y までの枝の連鎖であり，s において（Adj(s)=v，y）として一意に表現される．また距離が∞である経路は存在しない．

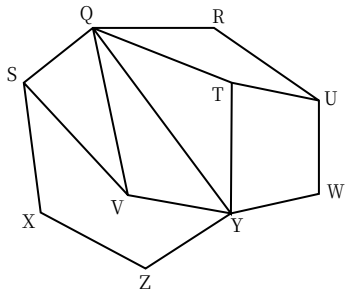

図 1　実際のネットワーク・トポロジ

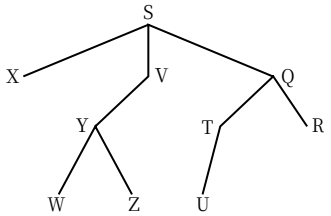

図 2　実際のネットワーク・トポロジ

2.4　DNS ルーティング・アルゴリズム（図 3）

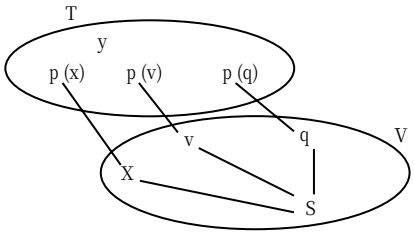

図 3　DNS ルーティング・アルゴリズム

経路情報更新処理において，任意のノード s∈V の隣接ノード x，v，q∈V から最短距離(x，y)，(v，y)，(q，y) が与えられれば，P(x)，P(v)，P(q)∈T であるから Dijkstra の step 4 から step 7，即ち

- 距離(s，y）＝MIN[距離(s，P(s)）＋距離(P(s)，y)] により経路（Adj(s)，y）が決定可能である．したがってある目的ノードに対する親までの最短距離が与えられたならば，最短経路決定処理の対象となるノードの集合 V は隣接ノードの集合に限定できる．これにより親から子へ引き継がれた経路情報を以下に示すように各ノードが処理し，その過程がネットワーク内のすべてのノードにより継承されれば Dijkstra が分散処理される．

① ノード s において P(s)から入力した目的ノードに対する経路情報に対して，隣接リストを基にその距離を更新する．

② 更新後の経路情報のうち，コストが最小となるものを選択し，正式経路とする．

③ C(s）にその経路情報を転送する．

この手続きが DNS で採用したルーティング・アルゴリズム（以後その手続き名称として DNS と略記）であるが，このように距離情報をノード間で更新しながら転送し広めていくルーティングは Distance Vector 型と呼ばれ，いろいろなバリエーションがある．RIP も Distance Vector 型であるが，コストとしてホップカウントを使用した情報を定期的に転送するため処理負荷が軽い．しかしホップカウントでは，経路の伝送遅延が考慮されず時間的に冗長な経路が確立される可能性が高い．s が必要とする親 P(s）の経路情報は，(P(s)，y）の最短経路情報であるにも拘わらず迂回路の冗長な経路情報が転送されれば，s はそれに基づいて経路を決定してしまうため，再度正規経路情報により算出をやり直す必要がある．したがってあらかじめ spanning tree に沿って転送することが最も望ましいが，ルーティング時には未だ得られていない．しかし入力した経路情報を更新後，その入力元以外の隣接ノードに対して出力すれば，転送される情報データのトレースは，s を根とし各弧の伝送遅延をコストとした場合 spaninng tree の非常に良好な近似となる（図 4）．この転送方法を Reverse-Path-Forwarding（RPF）と呼び，DNS はコストに伝送遅延を採用し RPF で転送することによりアルゴリズムの効率を向上させ，さらに経路情報の出力を障害検出時に限定し，障害への即対応を可能にすると同時に，ルーティングによる定常時の回線帯域の浪費を防止している．

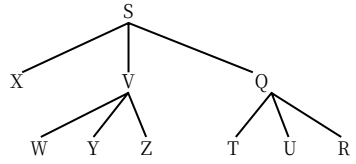

図 4　DNS による spanning tree の表現

Distance Vector 型アルゴリズムにより生成される spanning tree は，経路上の他のノードが有する隣接関係を把握できないため実際の接続構造とは異なった親ノードからの分枝にとどまり，s において最短経路はすべて（P(s)，y）と表現される．しかし Dijkstra により最短経路の一意性は保証されている．つまり s から P(s)，P(s)から最短経路（P(P(s))，y）の P(P(s)）というように各ノードが把握する y への最

短経路上の親を遡れば，それは Dijkstra による最短経路(s，y）に等しい．

3. Dijkstra および DNS による最短経路確立処理とノード間通信への影響

ノードはエンドシステム（ES）と中継システム（IS），及び両者を兼ねたシステムからなる．任意のエンドユーザ間の通信（ノード間通信）はルーティングが構成した 0 以上の中継システムを含む最短経路，即ち通信経路上にトランスポート・コネクションを確立して実行される．したがってルーティングによる通信経路の再構成はノード間通信を遅延もしくは中断する可能性がある．ここでは Dijkstra と DNS の各ルーティングとノード間通信への影響度の関係を考察するが，そのためまず初期段階と定常段階のルーティングを区別する．

- 初期段階：ネットワーク内の任意のノードが相互を認識してルーティングを行い，初期通信経路を確立する．
- 定常段階：既存通信経路に対して障害発生を検知し，通信経路の復旧を行う．

つまり初期段階はノード間通信の前段階であるため，通信への影響は定常段階のルーティングに限定される．さらに定常段階は次の二つの異なる過程からなるため，それぞれを別個に考察する必要がある．

① 障害検知，及び検知後の障害通知

② 障害通知受信による代替経路への変更

ルーティングがノード間通信に与える影響は，その処理負荷をデータ検索頻度の関数値と考えて検索数を算出すれば，処理負荷の度合い即ち所要時間を O 記法（オーダー）により表現することが可能である．検索数はデータ構造とアルゴリズムの関数に依存するが，アルゴリズムのみを検証するために，任意のノード s におけるノード t への通信経路のデータ構造を次のように一般化する（［ ］は record 及び record を構成する cell を，id は当該 cell 固有の識別子を示す）．

1. 網接続回線（LSA）：

   ［［回線タイプ］［RSA list］］

   回線（トランク）毎に生成され，リンク制御層とのインタフェースに使用される．

2. 論理リンク（RSA）：

   ［［リンクアドレス］［回線タイプ］［LSA id］［ADJ id］］

   隣接ノードとの論理リンク毎に生成され，LSA の子としてリンクされる．データの入出力制御及び隣接ノードの状態監視に使用される．

3. s 自身の隣接ノード（ADJ）：

   ［［ノードアドレス Adj(s)］［C(s，Adj(s))］［PATH list］］

   隣接ノード毎に生成される．通信経路の出発点として使用される．

4. t の隣接リスト（LSP）：

   ［［ノードアドレス t］［Adj（t)］［距離（t，Adj（t))］］

   受信した経路情報を保存したものであり，Dijkstra のみに使用される．任意のノード t が有する隣接リストである．

5. t への最短経路（SCA）：

［［ノードアドレス t］［距離(s，t)］［PATH list］［FCT list］

ルーティングにより経路情報から生成される任意の目的ノード t への最短経路である．

6. t への暫定経路（PATH）：

［［ノードアドレス t］［距離（s，t)］［Adj（s)］［SCA id]］

DNS は隣接ノード Adj(s)から受信したすべての経路情報から，PATH(Adj(s)，t)を作成する．これは Distance Vector により生成される spanning tree の性質，つまり経路はすべて親（隣接ノード）と目的ノードの関係であることから SCA(t）の代替経路に相当する．PATH 生成の目的は，障害発生後直ちに代替 PATH への切り替えを行い，ノード間通信への影響を低減することにある．

一方 Dijkstra はその実行時に LSP を枝として接ぎ穂しながら暫定経路として PATH を生成するが，その生成目的は SCA(t）の選定のみであり，それが確定した時点ですべての PATH を消去する．

なおこれらのデータ構造はいずれもアドレスによりハッシュソートされており，アドレスをキーとしたハッシングによる検索が可能であるものとする．また［データ list］型の cell は，該当するデータ構造が list 化されていてそれを用いた検索が可能であることを示す．

さらにルーティングはネットワーク・トポロジに依存するが，一般にネットワークを構成しているノードが互いに他の全ノードを隣接する，つまり任意の n∈V において，Adj(n)＝m{m は n を除く V のすべての要素} が成り立つ密結合構造を有する形態において，その処理負荷が最大になる．このようなネットワーク・トポロジは FCT（Fully Connected Topology）と呼ばれる．図 5 において任意のノード間に弧（回線）が存在するため，これは明らかに FCT である．ルーティング・アルゴリズムの性能差を明確にするため，この FCT を検査対象ネットワークのトポロジとする．

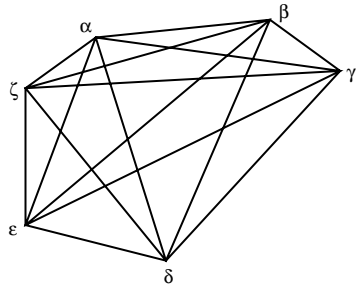

図 5　FCT の例（6 ノード構成時）

3.1　経路障害時のルーティング

経路障害は，コネクション（CO）型網ではリンク層からのリンク切断通知により，コネクションレス（CL）型網では ISH[*1] のタイムアウトによりそれぞれ論理リンクの切断として認識する．つまり障害リンク先のノードに対してその隣接性を失うことが，ルーティングの契機となる．

上記図 5 において，$\alpha\beta$ 間の経路障害時のルーティングを想定する．一般化のため図示されている以外にもネットワークにノードが存在し，総数 n ノードで構成されており，各ノードはいずれもエンドシステムと中継システムを兼ねたシステムであるものとする．

1)　Dijkstra の障害検知処理

この過程は Dijkstra の隣接情報検知処理 step i 1 から step i 4 に相当する．論理リンクの更新は，step i 2 において LSA に対応する RSA を検索して行う．RSA を一意に決定するため ISH に設定されたリンクアドレスをもとに検索するが，隣接ノードを結ぶ回線数は n−1 であるから，その検索数は log(n−1)①になる．RSA が決定すれば対応する隣接ノードの経路断（＝コスト無限大）を ADJ と経路情報に設定する．隣接ノードの検索は id により行われるため 1 回で済み n が十分に大きければ無視できる．step i 1 によりこの過程は隣接ノードの数だけ再帰処理されることから，

● 処理時間＝Σlog(n−1)＝O(n log n)(a)

この処理は障害を検知するノードのみが実行するため，経路情報受信後に障害を認知するノードと比べ通信への影響が次の理由により異なるように見える．

・障害を直接検知しないノード（ここでは α，β 以外のノード）は，α 及び β から出力された経路情報を受信してから経路障害を認識するが，この時点でデータの到達性は保証されておらず転送を継続するとネットワークの輻輳が発生し得るため，経路更新が完了するまでの間すべてのノードに対する通信を中断する

・障害を検知するノードは他のノード向けの通信を，障害検知直後から上記障害検知処理（経路情報の送信を含む）に加え，経路更新が完了して到達性が保証されるまで中断する

つまり直接障害を検知しないノードは，障害検知ノードが障害検知処理を実行している間もノード間通信を継続しているため，上記（a）の分だけ中断時間は短いようであるが，この間に転送されたデータは障害が検知され経路が変更されるまでの間ネットワークに滞留するため，実際のデータ転送所要時間を含めた中断時間には障害検知時間が含まれており，結果的に等しい．DNS についても同じことが成り立つ．なお経路更新処理については，3.2 節で説明する．

2)　DNS の障害検知処理

step 2 の RSA の決定，ADJ の検索と経路情報の設定処理は同じく log(n−1)②である．DNS は初期段階のルーティングにおいて全経路情報から，PATH（Adj(s)，t）{s，t は s＝t を除く任意のノード} を作成しており，距離の順にソートされてリスト化されている．したがってノード α におけるルーティングを想定すれば，障害経路上の隣接ノード β にリンクする PATH 数は n−1 となる．

まず β を経由する全経路の距離（コスト）を無限大にするため，step i 2 で特定された ADJ にリンクする PATH を検索する．これは ADJ の PATH リストを辿る処理となり，リスト内の PATH 数 n−1③に等しい．更新後の PATH は，対応する（目的ノードが等しい）SCA の PATH リストの最後尾（最下位）に付

加される．ここでSCA（β）が到達不能であるため，最短距離を有するリストの先頭のPATH（例えばPATH（γ，β））を代替経路として選択し，βへの最短経路（SCA）とする．SCAの検索は各PATH毎にidにより1回で求められ，リストの更新作業もたかだか2回で済むため無視される．DNSの場合は，障害ノード[*2]に対する更新処理だけを行えばよく，step i1による他の隣接ノードに関しての再帰処理が不要となる．以上より，

- 処理時間（②＋③）＝log(n－1)＋(n－1)＝O(n)

代替経路（γ，β）を初期経路情報としてαに通知した隣接ノードγにとってその経路は主経路（最短経路）であり，有効性が保証されている．しかしこの経路はαにとって最短経路とは限らないため，βへの通信を直ちに復旧させることはできない．これは次に説明するDistance Vectorの性質による．図1において初期段階のルーティング時，xは隣接ノードs，zからそれぞれtへの最短距離(s，q，t)，(z，y，t）が与えられ距離(x，s，q，t)＜(x，z，y，t）ならば，距離(x，t)＝MIN［距離(x，P(x))＋距離(P(x),t)］（Dijkstraのstep 4からstep 7）により最短経路(s，q，t）を決定する（2.4節参照）．この結果最短経路の親はsになるため，sにこの経路情報は通知されない．またxにおいて（z，y，t）は代替経路となるが，sにおいて代替経路(x，z，y，t）は存在しない．ここでqの障害が発生すれば，sはへの経路を代替経路(v，y，t）に切り替えてxに通知する．しかし距離(x，s，v，y，t)＞(x，z，y，t）である可能性があり，この場合，xはこの代替経路情報を肯定することはできない．つまりDistance VectorによるMIN［距離(x，P(x))＋距離(P(x)，t)］の経路の選定プロセスはトーナメント方式であり，最高位以外は必ずしも真の順位ではないため，最短経路が失われた場合は再度このプロセスをやり直す必要がある(Dijkstraにおいては，経路更新毎に本プロセス（step 4からstep 7）が行われる)．したがって経路障害を検知し，コストが劣る経路情報（Bad News）を送信した時は，次の手順によりその情報がネットワークに行き渡るまで障害ノードに対する通信は凍結される．

1） ノードαは障害ノードβを根とするspanning treeにおいて，αからup tree方向に分岐する隣接ノードの集合{γ，δ，ε，ζ…} すべてに対し，Bad Newsを送信する．
2） 経路情報受信処理（3.2節参照）においてBad Newsを入力した任意のノードζは，PATH（(Bad News送信ノード)，β）を更新する．もしコストの優る代替経路が存在したならばそれに切り替えてβへの最短経路を更新し，Bad News送信ノードに対しその旨の応答を送信する．
3） しかし代替経路が存在しなければζ自身もβ向けの通信を凍結しさらにBad News送信ノードを根とするspanning treeを想定し，ζからup tree方向に分岐する隣接ノードの集合のすべてに対してBad Newsを転送する．
4） 上記2）3）の処理をBad Newsを受信したノードが順次継承して実行する．そして任意のノードはBad News転送数を記憶し，その応答数が転送数に達した時点で障害ノードβに対する通信の凍結状態を解除する．この時点でβに対

する最善の代替経路が選択される．

したがって経路障害時のノード間通信の復旧には各隣接ノードの経路更新とその応答を要することになる．

3.2 経路情報受信時の処理

1) Dijkstra の経路更新処理

Dijkstra では初期段階/定常段階に拘わらず，常にネットワーク内のすべてのノード n∈V に対する経路情報の検索と経路の確立処理が必要であり，ノード数だけの再帰処理を要する．

したがって同一の障害情報を複数のノードから受信し重複したような場合に，不要なルーティングの実行を避けるため，既存 LSP と受信した経路情報の内容の比較を行う．これはノードアドレス $\alpha(\beta)$ により LSP を検索しコストを求めればよく，LSP は他のノード数 n−1 だけ存在するから，その検索数は $\log(n-1)$ ④となる．本ケースにおいては $\alpha(\beta)$ の新たな LSP では $\beta(\alpha)$ の隣接性が失われており，既存 LSP と内容が相違するため Dijkstra が行われる．

step 1 における最小コスト経路の選択即ち SCA の生成には，それまでに生成されたすべての暫定経路（PATH）の検索を要する．ここで Dijkstra の j 回目の再帰パスにおける PATH 数を nj ⑤とすると，それは step 6 毎に更新されるため各目的ノードに対し高々一つしか存在しない．つまり常に自身以外のノード総数を越えることはなく，nj≦n−1 である．step 5 から step 7 の隣接ノードによる PATH の生成には，LSP に対するノードアドレス $\alpha(\beta)$ による検索を要し，この検索数は先と同じく $\log(n-1)$ となる．次に暫定 PATH に対し目的ノードをキーとする検索回数は $\log nj$ となり，ここで step 6 により比較され，暫定経路の更新が行われる．step 4 により，この検索は $\alpha(\beta)$ に隣接するノードの数 n−1 だけ繰り返され，$(n-1)(\log(n-1)+\log nj)$ ⑥となる．

Dijkstra の再帰パスは処理ノード数 n−1 に等しいから，上記式⑤⑥は n−1 回づつ実行され，検索回数の上限値は，nj＝n−1 を式⑤⑥に代入して求められる．

● 処理時間 $\log n+\Sigma n+\Sigma n \log n=\log n+n^2+2\ n^2\log n=O(n^2\log n)$

以上ですべてのノードを対象とした再帰処理を終えて経路が確定するため，各ノード間の通信が再開する．

2) DNS の経路更新処理

DNS においては，定常段階の処理は Dijkstra と大きく異なり，隣接ノードから受信した経路情報（Bad News）に対してのみルーティングを実行する．したがって Dijkstra の step 4 から step 7 の処理で十分であり，再帰処理を必要としない．

Bad News を PATH と比較するために，まずその隣接ノードのアドレスにより ADJ を把握する．その検索回数は $\log n-1$ ⑦となる．次に ADJ の PATH リストを検索するが，それも同様に

$\log n-1$ ⑧となる．次に PATH に対応する SCA を求めるが，この検索及び PATH リストの更新は障害検知時同様に無視できる．

Bad　News 内の経路情報（コスト）によりそれまでの最小コスト PATH が更新されて，他の PATH（代替経路）に劣った場合には，代替経路に切り替えて SCA を更新し，その経路情報を応答として α に通知する．

また代替経路が存在しない場合は SCA を到達不能として β 向けの通信経路を凍結し，RPF によりさらに up　tree 方向の子ノードに転送する．本ケースにおいては，各隣接ノードは代替経路（γ，β）(δ，β)(ε，β)(ζ，β）を保有しており，それを応答として α に通知する．この時点で α の β に対する通信は復旧する．β においてもこの過程は同様に完結する．

step 4 により，この検索⑦⑧は障害ノード β を根とする spanning tree において，down　tree 方向に存在し，かつ SCA を更新する親ノードにおいて繰り返し実行されるため，処理時間はその累積値となる．しかしこのようなノードの数は n−2 を越えることはない．したがって⑦⑧より次に示す更新処理時間を要してネットワークの全ノードにおいて経路が確定し，ノード間通信が再開する．

● 処理時間 2(n−2)log(n−1)＝O(n log n)

以上よりいずれの場合も，経路更新処理時間が障害検知時間[*3] に対して突出しているため，そのオーダーに吸収される．しかし DNS の経路更新処理における負荷関数の増加率が O(n log n) に収まっているのに対し，Dijkstra の増加率は $O(n^2)$ を越えている．一般にプログラムの処理負荷関数の増加率は，ほぼ n とリニアな関係にある O(n log n) 以下が望ましいとされる．増加率が O(n log n) を越えていると，n が一定限度を越えた場合に処理速度が頭打ちとなり，処理が極端に遅延する状態に陥るからである．つまりネットワークの規模につれて Dijkstra によるルーティングの所要時間は，DNS と比べ著しく増大する．

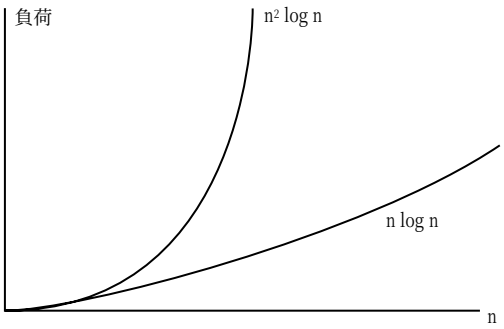

図 6　各負担関数の増加率

4.　バス型 FCT におけるルーティングの負荷

LAN や FDDI 等のバス幹線により構成されたブロードキャスト・ネットワークの内部構造（図 7）を考察する．このようなバス幹線に接続する任意のノード n∈V から出力された ISH はブロードキャストされて他のすべてのノード m{m は n を除くすべてのノード} と論理リンクを確立する．したがってノード n，m は互いに隣接ノードとして認識するため，ネットワーク内のノードが認識する構造は図 5 と同じく FCT

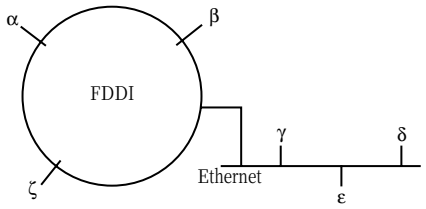

図 7　ブロードキャスト・ネットワーク

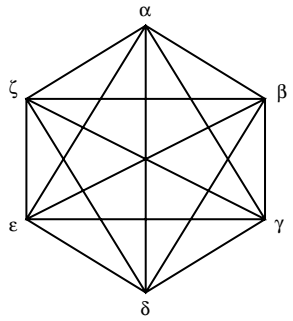

図 8　バス型 FCT の理論リンク

となるが，このような構造を実回線接続形態の FCT と区別してバス型 FCT と呼ぶ（図 8 参照）．

ここで DNS によるバス型 FCT に対する初期段階の処理を想定する．α∈FCT はノード Adj(α)∈FCT から受信する経路情報をすべて PATH(Adj(α), t) {t∈FCT} に変換する．このため PATH はネットワーク上すべてのノードの対となり，n^2 生成される．図 5 において最短経路(α，β）の代替経路は（α，γ，β)(α，δ，β)(α，ε，β)(α，ζ，β）である．これは α，β 間を接続する回線が切断した場合の迂回路としてそれぞれ実回線上に存在する．しかし図 8 のバス型 FCT においてはすべての代替経路は実回線ではなく，論理リンクであるためバス幹線への接続回線が切断した時点でノードが孤立し消滅する．つまり α，β 間の代替経路には意味がないことがわかる．

DNS は主経路（最短経路）に対応した代替経路が存在すれば，主経路障害を検知した場合，

速やかに通信経路を切り替えて継続させる．このためもし代替経路が誤ったものであればその無効性が検知されるまでの間転送したデータはネットワーク内に滞留し，かえってノード間通信の回復を遅延させてしまう．したがってバス型 FCT 上に代替経路が生成された時点で，その有効性を検証しなければならない．しかし図 9 のような FCT を経由する経路(α，β）の代替経路のうち有効なものとは，β のバス幹線への接続回線 l が切断しても，FCT とは異なる実回線 L により（γ，β）が接続している経路(α，γ，β）である．これに対し図 10 のように γ，β 間に FCT 以外存在しな

い形態においてFCTを経由する代替経路（α，γ，β）はlの障害時には無効である．すなわちαの経路生成の段階で，α自身に有効性が判断できるのは目的ノードβと直接接続する経路（α，β）の場合だけであり，間接経路（α，γ，β）の有効性についてはβ及びγにその検証を委ねるしかない．このためDNSはバス型FCTで任意のノードαが新たに図10のようにバス型FCTに加入しノードγ，βとの論理リンクを確立した時点でそれぞれに対し経路（α，β）（α，γ）の到達不能を通知する．この経路情報により無効な間接経路（γ，α，β）（β，　α，γ）を到達不能にすることができる．

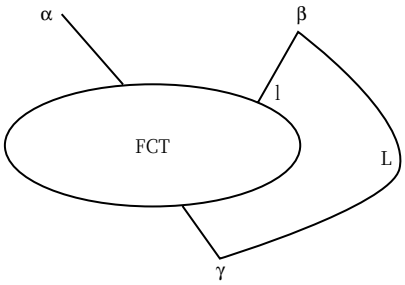

図 9　有効な代替経路

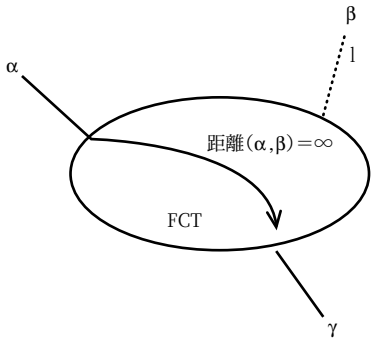

図 10　無効な代替経路とその除去

この処理が必要とする情報は目的ノード$\beta\gamma$への経路がFCT上を経由するか否かであるから，バス型FCTネットワークに対応してDNSは次のようなデータ構造を導入する．

- 目的ノードtへの経路上に存在するバス型FCT（FCT）：
  ［［ノードアドレスt］［RSA id］［SCA id］］

ここで図11のようなすべてのノードがFDDI網（バス型ネットワーク）に接続し，かつ経路障害に備えて$\alpha\beta$間，$\gamma\delta$間などのペアノード間に実回線による代替経路を設定したネットワーク・システムを想定する．ノードβのFDDI加入回線の障害に伴い，代替経路への切り替え処理で発生する経路情報数と必要となる検索数を3章同様に求め，ノード間通信回復までの時間を把握する．なお総ノード数は一般化のためnとする．

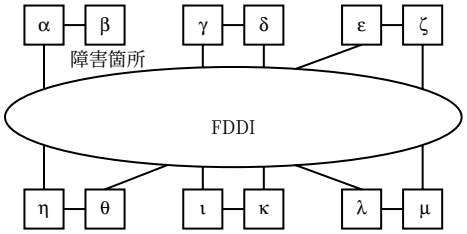

図 11　FDDI 上に構成されたネットワーク

1)　α，β 以外のノードの障害検知処理

β に対する論理リンクの切断を検知後，β への隣接性を失うため，その PATH を検索し β に対する主経路(SCA(β)) の到達性を凍結して，β を除く他のノードに Bad　News を送信する．この時点で β との通信は中断する．処理時間は PATH の検索負荷のオーダーに吸収され O(n) となる．

2)　α の障害検知処理

論理リンクの切断を検知するが，β への隣接性を実回線接続により失わない．したがって他のノードの β 向けの経路に対して α 自身を経由する代替経路に更新させるため，その到達性（コスト低下により Bad　News として）を他のノードに通知する．この時点で β 向けの通信は凍結する．処理時間は O(n) となる．

3)　β の障害検知処理

α を除くすべてのノードとの隣接性を失うため，それぞれの経路をすべて α を経由する代替経路に切り替え，到達性を α に通知する（コスト低下のため Bad News として）．この処理は，各ノードからの Bad News 受信時の DNS による経路更新処理に相当し，(n−2)回再帰するが，その度に各ノードのすべての PATH を検索し更新する必要があるため，処理時間は $O(n^2)$ となる．この時点で α を除くすべてのノードに対する通信を凍結する．

4)　α，β 以外のノードによる障害情報受信処理

Bad　News 受信時に DNS による経路更新が行われるが，特定の経路のみを更新対象とするため，処理時間は，O(n log n) となる．α の代替経路情報及び応答の交換により β への凍結を解除し β 向けの通信を再開し，β への到達可能性を通知する．したがって O(n log n)がその通信復旧までの最大所要時間である．

5)　α による障害情報受信処理

β からの Bad　News 数は，β 以外のノードからの Bad　News 数に等しく，計 2(n−1) 回の経路更新処理を行い，処理時間は O(n log n) となる．応答数を確認後に凍結を解除する．

6)　β による障害情報受信処理

Bad News は α によるものだけであり，処理時間は O(n log n) である．いずれも α を経由する代替経路情報であり，すでに PATH として存在しており更新不要であるため，応答数を確認後に各ノードへの通信の凍結を解除する．

次に加入回線の障害復旧を想定し，同様に通信経路切り替え処理によるノード

間通信回復までの時間を把握する．

7） β の障害復旧検知処理

α を除くすべてのノードとの隣接性を回復し，ADJ を生成し，β 自身の到達可能性を Good News で通知する．次に全 SCA（目的ノード）に対して無効な間接経路の生成を防ぐため，バス型 FCT 上の隣接性を検証し，それらの到達不能性を Bad News により通知する．この過程は（n−2）回の経路更新処理に相当し，かつ再帰毎にすべての SCA を検索するため，処理時間は $O(n^2)$ となり，この時点で再び α を除くすべてのノードに対する通信が中断する．

8） α の障害復旧検知処理

β に対する論理リンクの接続を検知後，障害検知時に行なった β への経路の α を経由する代替経路への変更を破棄させるため，β への経路すべてを到達不能とし，Bad News により通知する．この時点で β への通信を凍結する．処理時間は O(n）となる．

9） $\alpha\beta$ 以外の障害復旧検知と障害回復情報受信処理

β に対する論理リンクの接続を検知し ADJ を生成する．次に β からの Good News により最短経路を更新後，$\alpha\beta$ からの Bad News により無効なバス型 FCT 上の間接経路（$\alpha\beta$ を経由する PATH）が除去される．これらは既に最短経路ではないため応答のみを行う．各経路情報毎に DNS による経路更新が行われるが，特定の経路のみを更新の対象とするため，処理時間は，O(n log n）となる．この応答をすべて受信した時点で $\alpha\beta$ 間の通信が復旧する．

以上からバス型 FCT における DNS 実行時の通信の中断時間は最大 $O(n^2)$となり，負荷の高いネットワーク形態である FCT の実行時間に比較してさらにオーダーが 1 桁大きい値となる．この増加率では，n が十分に大きくなるとルーティングは非常に困難になる[*4]．したがってバス型 FCT 環境での DNS ルーティング負荷の根本的な削減策について考慮する必要がある．

5. バス型 FCT に対するルーティング処理の削減策

ルーティング負荷は内面的にはアルゴリズムとそのデータ構造に依存し，外面的にはネットワーク内の隣接ノード数と接続構造に依る．ゆえに対策としては，これらの各要素を変更することにより得られる削減効果を測定すると同時に，ノード間通信に与える影響を把握し遅延を生じない方法を講じることである．以下に各要素に対応する削減策の内容と効果及び通信への影響を示す．

5.1 アルゴリズムとデータ構造

ルーティング・アルゴリズムに関しては，経路がユーザにより与えられるスタティックなものを除けば，それは次の機能を満たしていることが必要十分条件となる．

① 経路情報の把握

② spanning tree の生成

③ 経路情報の転送

このうちアルゴリズムが主に関連するのは spanning tree の生成過程であるが，これを生成する最も効率の良いアルゴリズムとして縦形探索，横形探索，Dijkstra,

Floyd，Ford が確認されている．このうち Floyd は n 対 n の経路生成アルゴリズムであるため，負荷は Dijkstra よりも 1 オーダー大きくなる．また Dijkstra に比べてその分散処理形態（Distance　Vector 型）である DNS の方が処理効率が良いことは前節で検証済みである．同様に横形探索，縦形探索及び Ford によるルーティングに関しても，その分散処理形態である RIP を含めた Distance Vector 型ルーティングの方が効率が良い．したがってここでは Distance Vector 型ルーティング一般について検証する．

Distance Vector 型の基本アルゴリズムは，隣接ノード x，v，q から最短経路情報 (x, y)，(v, y)，(q, y) を受信し，距離(s, y)＝MIN[距離(s, P(s))＋距離(P(s), y)] により経路(Adj(s)，y) を決定する過程であるが，この部分に関しては効率のオーダーに影響する差はない．したがって経路情報の転送方法とデータ構造の差異に着目する．

経路情報の転送方法で最も負荷が軽減されるものは定期的な転送である．RIP もこの方法を採っている．この場合ジッタ（jitter）を行っている限りネットワーク障害が発生しても一斉に経路情報が発生することはなく，バス型 FCT においても最大ノード 1 台分の情報量に分散することから，その負荷は O(n log n) に抑えられる．しかしこの時間は同時に障害回復を遅らせる原因になる．RIP の場合通常経路情報転送間隔は 30 秒毎であるから，ノード数が十分に多い規模のネットワークにおいては障害の検出に最大 30 秒費やすことになる．このような遅延時間はリアルトランザクション処理中の端末セッションのリカバリには許容されない．逆にこの転送時間をこれより短くすれば障害検出時間の短縮が可能になる．しかしこの処置は障害がなくとも常時経路情報の転送とルーティング負荷を発生させ，特にネットワークの規模が大きければ，その全く無駄なトラヒックは多大な帯域の損失となる．

次に代替経路を持たないデータ構造を考える．この場合隣接ノードと主経路を直接結びつけるだけでよいから，バス型 FCT においても経路情報の処理負荷は O(n log n) に削減される．しかし代替経路がない場合は主経路が到達不能に陥った場合，改めて段階 1 のルーティングを実施し，各隣接ノードからすべてのノードに対する経路情報を処理しなければならず，その結果処理負荷は $O(n^2 \log n)$ になる．したがって逆に遅延を増大させてしまう．つまりこのデータ構造は Distance Vector 型での障害時のルーティング効率の向上には必要不可欠である．

5.2 ネットワークの階層化

ルーティングで発生する経路情報数の削減策としてネットワークの階層化について検証する．Dijkstra を使用する DEC ルーティング等はネットワークの大規模化とインターネット化に伴うルーティング負荷を経験し，その削減策としてネットワークの階層化の必要性と方式を講じてきた．ここでは IS-IS Intra-Domain routing protocol に規定されている階層化の概要を説明する．

5.2.1 代表ノード（designated）方式

これは各ノードが有する優先順位が最大のものをそのバス型 FCT の designated に選定し，それ以外のノードは，隣接情報を designated に対してのみ通知させることにより経路情報数の最小化を目的とするものである．この方式による処理負荷は次の

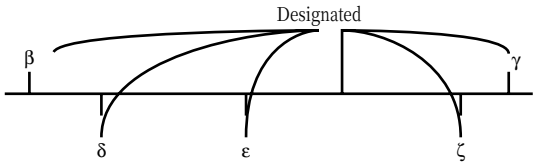

図 12　Designate ノード

ようになる．

- designated 以外のノード：経路変更検知処理の負荷は変わらないが，経路情報は designated から受信するため，経路情報受信処理における再帰パスは不要になり，負荷は O(n）まで軽減される．
- designated ノード（図 12)：一方 designated においては代表ノード方式を採らない時と同様，designated 以外のノードすべてを隣接ノードとして処理することから負荷は軽減されない．

つまりこのルーティング方式は負荷を唯一の designated ノードに限定する集中型ルーティングであるため，その性能が他のノードより飛躍的に優れていない限り，この処理がネックとなるためリカバリ時間の短縮は望めない．すべてのノードの負荷を軽減するには次に示す階層化ルーティングの適用が考えられる．

5.2.2　階層化ルーティング方式（図 12）

1）　Domain を複数の Area に細分化し各 Area 毎で Dijkstra ルーティングを行う．そのルーティングを level 1 と呼ぶ．

2）　実行結果は，各 Area の level 2 の IS に渡される．level 2 IS の集合は仮想の Domain を形成し，そこで再度 Dijkstra により Domain 内のすべての経路情報を把握する．

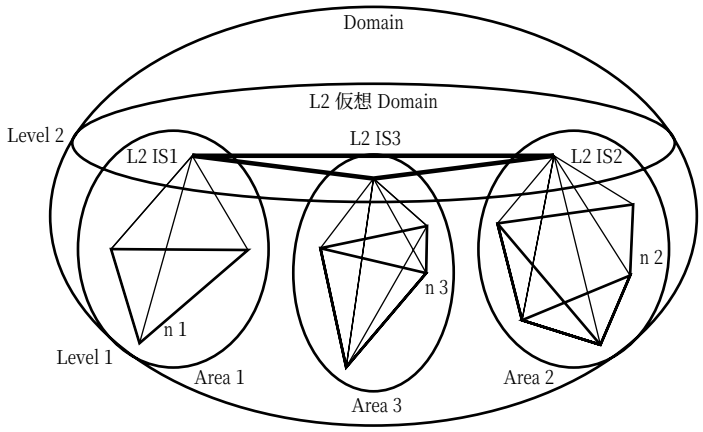

図 13　階層化ルーティング

ここで階層化ルーティングにおいて Dijkstra に代えて DNS を適用し，Domain を Area 数 k に細分化することでバス型 FCT における level 2　IS の処理負荷が n　log　n

の水準まで軽減したとすれば,

● 各 Area で実行される DNS の処理負荷＝$(n/k)^2$＝n log n

が成り立ち，k＝$\sqrt{n}$/log n となり，n＝10 で約 2 である．

しかし階層化ルーティングにより確立される通信経路は Area 内のノード間であれば最短経路となるが，Area が異なるノード n 1∈Area 1，n 2∈Area 2 間の場合，IS 1，IS 2 をそれぞれ Area 1，Area 2 の level 2　IS とすれば，設定される経路は，実際の最短経路(n 1，n 2）と異なり,

● 迂回路：最短経路(n 1，IS 1)＋最短経路(IS 1，IS 2)＋最短経路(IS 2，n 2)

が設定される．つまりネットワークを k の Area に細分化すると経路長は,

$(1+\sqrt{n}/\log n)/2$

つまり n＝10 の時 1.5 倍，n＝30 の時 2 倍になり，この結果トラヒックと伝送遅延も同じ倍率で増加することになる．

またこの環境におけるルーティングによる通信経路の復旧には，さらに level 2　IS によるルーティングが必要である（level 2 は level 1 の実行を前提とし，その完了後に行われる）が，そのルーティングの負荷は上式から,

● k^2＝n/log n

となり, 結局 n log n＋n/log n の処理時間がルーティング完了までに必要とされる．

したがって階層化により処理負荷を n log n の水準に軽減し得たとしても，10 ノードで 1.5 倍，30 ノードで 2 倍の冗長な通信経路が確立され，通信時のトラヒック及び伝送遅延も同じ比率で増大し，かつ通信経路の復旧時間は 30 ノードで一割遅延する．ルーティング本来の目的が最短経路確立によるノード間通信の時間短縮化であるにも拘わらず，階層化方式の適用は，ルーティングの負荷軽減のためにノード間通信に要する時間を犠牲にすることになる．インターネットのように複数の Domain を接続するような規模でない限り，負荷を階層化ルーティングにより n log n 並みに軽減したとしても，通信に与えるデメリット（トラヒック及び伝送遅延の増大）の方が多く改善策とはならない．

5.2.3　バス型 FCT の接続構造

本節ではバス型 FCT の論理リンク構造ではなく，そのグラフ構造に着目して仮想ノードの概念を導き，そのルーティング・アルゴリズムへの適用について考察する．グラフの定義より任意のノード n と Adj(n) は弧(n，Adj(n)) で 1 対 1 に結合されているはずである．しかし n の加入回線を切断すると，すべての隣接関係が失われ仮定に矛盾する．したがってバス型 FCT において任意のノード α が論理リンクにより Adj(α）として把握したノード（β など）はグラフ上は隣接関係にない．つまり論理リンクによる把握された隣接性は実トポロジとは無関係に，バス型幹線が有するマルチキャスト機能により実現するものである．ノード n の加入回線とバス型幹線の接点を mp(n) とすると，バス型 FCT のグラフ構造は，図 14 として表される．つまりノード α が実際に隣接しているのは mp(α）であり，FCT を形成しているのは mp(α)，mp(β)，…，mp(ζ）である．しかしこれらのマルチキャスト機能に対する接点はネットワークより下位のリンク層にあるから実体（自律的なルーティング機能）がなく，任意の接点間の経路には意味がない．したがって図 14 からすべての接

点と，それらを結ぶすべての弧を短絡除去すれば，マルチキャスト機能そのものを中心としてスター状に各ノードを配置した構造が現れる．これがバス型 FCT の真の構造である．さらにこのマルチキャスト機能の n 対 n の隣接性に実体を与える，すなわち自律的なルーティング機能を具備させ，それを便宜的に仮想ノードと名づければ，各ノードの隣接ノードは仮想ノードのみとなり（図 15），それぞれの処理負荷がかなり削減されるはずである．しかしこの仮想ノードに実際のノードを割り当てた場合，階層化ルーティングの designated 方式のように，負荷が中央のノードに集中してしまう．

それとは異なり，各ノードがバス型 FCT においてグラフ構造から仮想ノードの存在を想定し，隣接ノードとの関係を仮想ノードとの関係に置換後，仮想ノードに対するルーティングとともに，仮想ノードとして（仮想ノードのルーティングをシミュレートして）隣接ノードに対するルーティングを行なえば，経路情報と検索負荷が分散され多大な削減効果を見込むことができる．

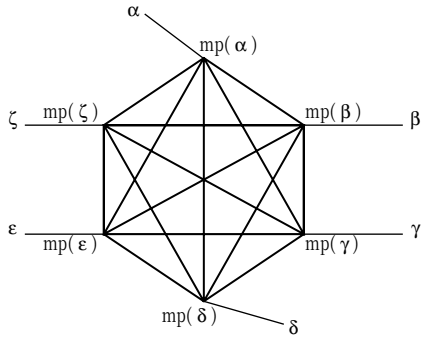

図 14　バス型 FCT のグラフ構造

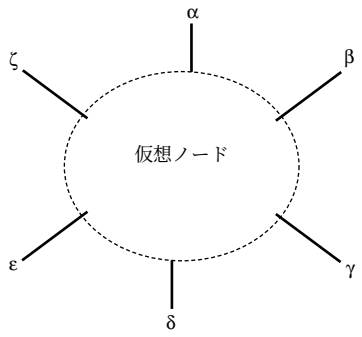

図 15　バス型 FCT の仮想ノード表現

5.3　仮想ノードのシミュレート

この仮想ノードをシミュレートして負荷を軽減する方式について考察する．ここでバス型 FCT 上の隣接関係を実リンクによる隣接性と区別するため，FCT 隣接と名付けノード n の FCT 隣接を Fadj(n）で表現する．

1）　初期段階のルーティングにおいて論理リンク確立によりバス型 FCT への加入

を認識すると，その論理リンクに対応した仮想ノードの存在を想定する．

2) 初期段階において FCT 隣接ノードに対して形成する spanning tree は仮想ノードから分岐する葉である（図 16）．したがって RPF 転送の定義より，FCT 隣接ノードに対して通知する経路情報は，他の枝から down trree 方向で与えられた経路情報に限定される．即ち，FCT 隣接ノードに関する情報は，他の同一 FCT 上の隣接ノードに与えない．

これにより無効な代替経路の形成が抑止されることになる．ただし図 17 のように $\alpha\zeta$，$\alpha\beta$ の間に代替経路が存在すれば，仮想ノードを媒介せず直接の隣接関係が成り立つため ζ を FCT 隣接ノードとは扱わない．

3) FCT 隣接ノードから与えられた経路情報は本来仮想ノードが受信する情報であるから，その処理をシミュレートするために必要なデータ構造 FCT を導入する（そのデータ構造には既存の FCT を転用する）．FCT は経路情報に対応する SCA と論理リンク情報（RSA）から生成する．FCT に対応する経路情報はすべて仮想ノードの子として扱われる．

4) 定常段階において FCT 隣接ノードに対する経路障害検出は本来仮想ノードにより行われる処理であるから，その情報は仮想ノードから与えられたものと判断する．したがって図 17 のネットワーク形態において，δ の障害時，α は bad news を図 18 のように up tree 方向 ζ，β に通知する．したがって FCT 隣接ノードである γ，ε には通知されない．

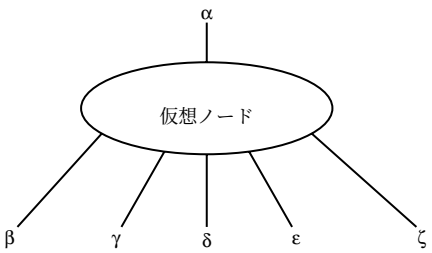

図 16　仮想ノードの spanning tree 表現（1）

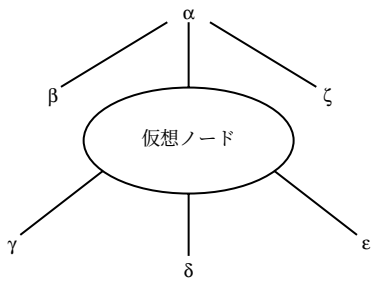

図 17　仮想ノードの spanning tree 表現（2）

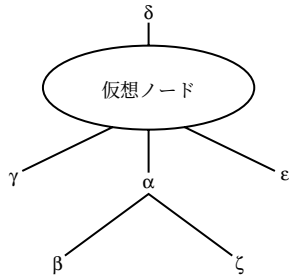

図 18　Bad News の転送

5.3.1　バス型 FCT への適用

仮想ノードをシミュレートしたルーティング方式を 4 章で想定したバス型 FCT ネットワーク環境に適用し，同様に図 11 においてノード β の FDDI 加入回線の障害に伴い，代替経路への切り替え処理で発生する経路情報数と必要となる検索数を求め，ノード間通信回復までの時間を既存の DNS と比較する．

1）　$\alpha\beta$ 以外の障害検知処理

β に対する隣接性を失い，PATH を検索し，β に対する主経路（SCA（β））を凍結する．ただし対応する FCT が存在するため，β の障害は仮想ノードより通知されたものと判断し up　tree 方向のノードを検証するが，すべて FCT 隣接ノードであるため存在せず，Bad　News は通知しない（図 19）．ここで新規仕様により，FCT 隣接ノードとの関係は仮想ノードとのスター型接続形態として把握されるため，初期段階において PATH は各ノードに対し一つしか生成されておらず，検索負荷は FCT 隣接ノードの検索のオーダーに吸収され O（n）となる．

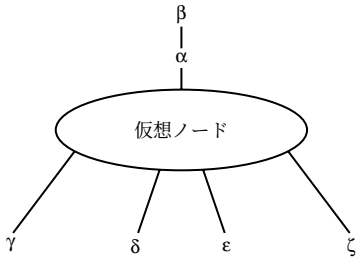

図 19　経路障害時の FCT 隣接性

2）　α の障害検知処理

隣接性を失わないため，β への経路を α を経由する唯一の代替経路に更新する．次に仮想ノードの処理としてその到達性を他のノードに通知する．β に対しては α を経由する代替経路情報を転送する．この時点で β への通信は凍結する．処理時間は O（n）となる．

3）　β の障害検知処理

α を除くすべての FCT 隣接ノードとの隣接性を失うため，代替経路を確認し

α に通知するが隣接ノードへの経路は1つづつしか存在しないため処理時間はO(n)となる．この時点で α を除くすべてのノードへの通信が凍結する．

4) α，β 以外のノードによる障害情報受信処理

α からの唯一の代替経路情報により更新が行われるが，β への経路は1つしか存在せず更新時に負荷はかからない．このため応答後 β への通信が再開するまでの所要時間はO(n)となる．

5) α による障害情報受信処理

β からの Bad　News は，n−1 の FCT 隣接ノードに対する仮想ノードによる経路更新処理として扱われ，処理時間はO(n)となる．応答後に凍結が解除する．

6) β による障害情報受信処理

α からの代替経路情報を受信後，到達確認応答を α に通知する．FCT 隣接ノードへの経路は1つづつしか存在しないため処理時間はO(n)となる．この時点で各ノードへの通信の凍結が解除する．

次に加入回線の障害復旧を想定し，同様に通信経路切り替えによりノード間通信が回復するまでの時間を把握する．

7) β の障害復旧検知処理

α を除くすべての FCT 隣接ノードとの隣接性を回復するため仮想ノード処理をシミュレートし，自身の到達可能を Good　News で各ノードに通知する．ノード間通信に影響は与えない．

8) β 以外の障害復旧検知と障害回復情報受信処理

β に対する論理リンクの接続を検知し ADJ を生成する．次に FCT 隣接性を確認後に仮想ノード処理をシミュレートし，自身の到達可能を Good　News により β のみに通知する．次に β からの Good　News により β に対する唯一の最短経路を更新し，FCT 隣接性のため FCT をリンクする．この処理もノード間通信に影響は与えない．

すなわち DNS が仮想ノードをシミュレートすることで，バス型 FCT において通信の中断は障害時に限定され，その時間はO(n)にまで抑制可能である．この値は既存負荷 $O(n^2)$ を一桁短縮し，目標値O(n log n)よりもさらに軽減することが可能である．

6. おわりに

昨今のネットワーク IT（情報技術）の変遷は，通信技術全般にわたる歴史的な発展を反映しているかのように見える．コンピュータ・システム内で文字データの転送に使用されたことに端を発するネットワークは，今や飛躍的に向上した帯域を持ち，あらゆる垣根を越えて音声・映像データといったマルチメディア情報を伝達することを可能としている．しかしながらこの進歩は通信インフラストラクチャの革新的な改良に依存したものであって，その著しい向上に対し，通信制御ソフトウェアによる障害検知や検出，障害箇所の迂回やリカバリ，または転送効率といった通信制御の本質ともいえる要素の重要性が低下した．もとより通信制御ソフトウェアは，通信インフ

ラストラクチャがその機構上持つ制約や発生しうる障害をいかに克服し，転送効率をいかに向上させるかに主眼をおきつつ発展してきたが，通信インフラストラクチャが高性能/高機能化すればするほど通信制御ソフトウェアの相対的な位置づけは低下しているように見える．本来通信インフラストラクチャと通信制御ソフトウェアとは互いの進歩を促しあうべきものであり，軽くスタンダードな通信制御ソフトウェアであればよしとするのではなく，本質的な性能向上への努力を怠ってはならないものと考える．

末尾ながら本考察が，通信制御ソフトウェアに関するソフトウェア技術の向上に微力ながら寄与することを願うものである．

＊1　ISH（IS Hello）：IS が隣接する IS 向けに定期的にマルチキャストするプリミティブであり，アドレス情報と生存の通知を目的とする
＊2　経路障害により隣接性を失ったノードを想定する．ノード自体の障害は通信不能になり，ここでは意味がない（復旧しない）ため除外する
＊3　ISH のタイムアウト時間など（論理）リンク切断検知にかかる時間を除く
＊4　実際に FDDI 幹線に多数のノードを接続した形態のネットワーク・システムにおいて，回線障害が生じた際に DNS による通信リカバリが著しく遅延し，結局上位層（トランスポート）のトランスポート・コネクションがタイムアウトにより切断する問題が発生することがある

参考文献

［1］ Princeton University,“ Computer Meghods in Problem Solving, Lab 8 : Shortest-Path Algorithm ; part 2 : Dijkstra's Algorithm（C 1 V 201）”
［2］ 伊理政夫, 他著“ Graph Thory with Excercises ”, CORONA Publishing, 1983.
［3］ San Diego University,“ Combinatorial Algorithms Grapths and Network Flows（CS 660）.
［4］ IETF,“ RFC 1058 Routing Information Protocol ”
［5］ Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ulman“ Data Stractures and Algorithms ” Addison-Wesley Publishing 1983.
［6］ Andrew S. Tanenbaum“ Computer Networks ”, Prentice Hall, 1996.
［7］ Vijay Ahuja “ Design and Analysis of Computer Communication Networks ”, McGraw-Hill, 1982
［8］ ISO/IEC/SC 6,“ IS 10589 IS-IS Intra-Domain Routing Protocol ”, 1996
［9］ IETF,“ RFC 1583 Open Shortest Path First ver. 2 ”

執筆者紹介

小　西　　宏（Hiroshi Konishi）
1960 年生まれ，1985 年大阪市立大学理学部地学科卒業，同年日本ユニバック(株)(現日本ユニシス(株))入社．UNISYS シリーズ 2200 システム専用通信制御ソフトウェアの開発，保守に従事．1991 年から ISO トランスポート担当（WG 4）及び ECFF 委員として標準化作業に携わる．現在，ネットワークシステム部 2200/IX ネットワーク室に所属．