アルゴリズム　15－2
（１９９０．５．１７）

# プロセッサとリンクの故障が混在する完全ネットワークにおける分散アルゴリズム

西川直樹†　　増澤利光††　　都倉信樹†

†大阪大学基礎工学部　　††大阪大学情報処理教育センター

プロセッサとリンクの故障が混在する完全ネットワークにおける放送問題とリーダー選択問題の通信計算量について考察し，リンクとプロセッサがそれぞれ高々$f_P$,$f_L$個故障する完全ネットワーク（nはプロセッサ数）で以下の結果を得た．
(1)方向感覚付き完全ネットワーク上の放送問題は$\Theta(n+f_L \cdot f_P+f_L \cdot \log f_L)$
(2)方向感覚のない完全ネットワーク上の放送問題は$\Theta(n+n \cdot f_L)$
(3)方向感覚付き完全ネットワーク上のリーダー選択問題は$\Theta(n+k(f_P+f_L)+f_L \cdot f_P+f_L \cdot \log f_L)$．ただし，kは起動プロセッサ数を示す．

# Distributed Algorithms in Complete Networks with Link and Processor Failures

Naoki Nishikawa†　　Toshimitsu Masuzawa††　　Nobuki Tokura†

†Faculy of Engineering Science
Osaka University

††Education Center for Information Processing
Osaka University

This paper considers fault-tolerant distributed algorithms in asynchronous complete networks with undetectable fail-stop link and processor failures. The problems studied in this paper are the broadcast problem and the leader election problem, and the followings are shown for complete networks with n processors and at most $f_L$ faulty links and at most $f_P$ faulty processors.
(1)The message complexity of broadcast is $\Theta(n+f_L \cdot f_P+f_L \cdot \log f_L)$,if global sence of direction is available,while
(2)the message complexity is $\Theta(n+n \cdot f_L)$,if no global sence of direction is available.
(3)The message complexity of leader election is $\Theta(n+k(f_P+f_L)+f_L \cdot f_P+f_L \cdot \log f_L)$ where k represents the number of initiator processors start the algorithm,if global sence of direction is available.

## 1．まえがき

通信用リンクで結合されている複数台のプロセッサ（すべてのプロセッサは対等であり，ホストプロセッサなるものはない．以下プロセッサをPEと略記）に，ある問題を解くために必要な情報が分散している状況で，それらの情報を交換しながらその問題を解くアルゴリズムは分散アルゴリズム(distributed algorithm)と呼ばれる．（但し，PE間には共有メモリはなく，PE間通信はリンクを通じたメッセージ交換によってのみ行なわれる．）分散アルゴリズムは非同期式ネットワーク（リンクを伝わるメッセージの伝送遅延，PE間の動作速度の比に関して，有限であること以外には何も仮定しないネットワーク）で考えられることが多い．これまでにリーダー選択問題や生成木構成問題などを解く多くの分散アルゴリズムが提案されている[5],[7],[9],[11]-[13]．

リンクやPEの故障が予想されるときに，それらの故障にもかかわらず種々の問題を解く分散アルゴリズムについても研究されている．特に非同期ネットワークではリンクやPEの故障を検知できないので，故障耐性のあるアルゴリズムを開発することは容易ではない．これまでに提案された故障耐性のあるアルゴリズムのほとんどは，リンクまたはPEどちらか一方のみの故障に限定して考えている[1]-[4],[6],[8],[10]．それらのアルゴリズムは，その解法をPEまたはリンクには故障がないという仮定に強く依存していることが多く，PEとリンクの故障が混在しているような状況では問題を解くことができない．そこで，本稿ではリンクとPEの故障が混在している場合の分散アルゴリズムについて考察する．

分散アルゴリズムに於いては通信計算量（最悪時評価）やその他の評価基準をより改善するために対象となるネットワークの形状にトポロジ制限を与えることがある．このような目的でリングや完全ネットワークが用いられるがここでは方向感覚付き完全ネットワーク[13]を用いる．

本稿では，放送問題（ある一台のPEのもつ情報を故障していない全PEに知らせる問題）について考察する．分散アルゴリズムにおいて，多くの場合ネットワーク中の全PEが結果あるいは終了の事実を認識してアルゴリズムを終了することから，他の問題の一部として用いることができる基本的な問題である．放送問題の通信計算量については，方向感覚付き完全ネットワークでPE故障のみの場合は$\Theta(n)$†，リンク故障(高々$f_L$個)のみの場合は$\Theta(n+f_L\cdot\log f_L)$であり，方向感覚のない完全ネットワークでは，PE故障のみの場合は$\Theta(n)$，リンク故障(高々$f_L$個)のみの場合は$\Theta(n+n\cdot f_L)$であることが知られている[10]．

本稿では，高々$f_P$個のPEと$f_L$個のリンクに故障がある（ただし$f_P+f_L\leq n-2$）場合に，方向感覚付き完全ネットワークにおける放送問題を解く通信計算量$O(n+f_P\cdot f_L+f_L\cdot\log f_L)$の分散アルゴリズムと，方向感覚のない完全ネットワーク上で，放送問題を解く通信計算量$O(n+n\cdot f_L)$の分散アルゴリズムを示す．そして，これらのアルゴリズムの通信計算量がどちらもオーダー的に最適であることを示す．

さらにこのアルゴリズムを用いて，方向感覚付き完全ネットワークでリーダー選択問題を解く通信計算量$O(n+k(f_P+f_L)+f_P\cdot f_L+f_L\cdot\log f_L)$（kは起動プロセッサ数）のアルゴリズムを示す．また，このアルゴリズムの通信計算量がオーダー的に最適であることを示す．（なお，文献(11)は方向感覚付き完全ネットワークにおけるリーダー選択問題の通信計算量についてPE故障(高々$f_P$個)のみの場合は$\Theta(n+k\cdot f_P)$，リンク故障(高々$f_L$個)のみの場合は$\Theta(n+k\cdot f_L+f_L\cdot\log f_L)$であることが示されているので本稿の結果はその拡張とみることができる．）

---

† 実数から実数への関数$f(n),g(n)$に対し，正定数$c$，$n_0$が存在し，すべての$n\geq n_0$ に対し，$f(n)\leq cg(n)$なるとき，$f(n)=O(g(n))$と書く．

同様に，$f(n)\geq cg(n)$なるとき$f(n)=\Omega(g(n))$と書く．

$f(n)=O(g(n))$ かつ，$f(n)=\Omega(g(n))$のとき，$f(n)=\Theta(g(n))$と書く．

## 2．諸定義

### 2．1 ネットワーク

［定義１］ネットワークNは二項組$N=(P,L)$で定義され，以下の性質を持つ．

(1)プロセッサ（以下では，PEと略記する）間の通信はリンクを介したメッセージ交換によってのみ行える．

(2)リンクは両方向の通信が独立に（反対方向の通信の影響を受けることなく）行える双方向リンクである．

(3)PEは互いに異なる識別子を持つ．PE uの識別子をid(u)と表す．

(4)ネットワークは非同期式である．つまり，メッセージがリンクを伝わる伝送遅延は有限だが可変であり前もって分からない．

(5)各PE（uとする）は，deg(u)個（deg(u)はuに接続するリンクの数を表す）のポート$\{a \mid 1\leq a\leq deg(u)\}$を持つ．各ポートはuに接続する各リンクと１対１に対応する．uのポートaがリンク(u,v)に対応するとき，vを$u[a]$と表す．ただし，$u[0]$はu自身を表す．

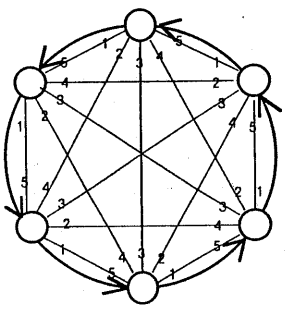
(6)リンクはFIFO (First-In First-Out)とする．つまりリンク(u,v)を用いてPEuがvに送信したメッセージはuが送信した順にvが受信する．□

以下，本稿ではネットワーク全体のPE数をnで表す．

### 2．2 完全ネットワークと方向感覚

［定義２］任意の相異なるPE間にリンクが存在するネットワークを完全ネットワーク（以後，CNと略記）という．CN Nが次の(条件ＳＤ)を満足するとき，Nは方向感覚付き完全ネットワーク(Complete Network with a Sense of Direction．以後，CN-SDと略記)である（図１）．

なお，任意の整数値x,yに対して式$\overline{x-y}$はx-yの法nの非負剰余を示す．



→:ハミルトン有向閉路

図1:方向感覚付き完全ネットワーク

（条件ＳＤ）uを任意のPEとし，a，bを（1≦a,b≦n-1）を満たす任意の正整数とする．v＝u[a],w＝u[b]とすると，w＝v[b-a]が成立する．（図２）□

つまり，CNの１つの有向ハミルトン閉路Ｈを考え，Ｈ上で距離がaのPEとはポートaを介して隣接しているCNがCN-SDである．

## ２．３　分散アルゴリズム

［定義３］分散アルゴリズム（以下では単にアルゴリズムという）は以下の性質を持つ．

(1)各PEが実行するプログラムは，受信したメッセージに対する処理（どのような内部計算をし，どのようなメッセージをどのポートから送信するか）を表す．

(2)すべてのPEのプログラムは同一である．但し，各PE uのプログラムでは，id(u)，ポート数deg(u)，$f_P$と$f_L$（ネットワーク全体での故障PEと故障リンクそれぞれの数の上限）を利用できる．

(3)いくつかのPEは，最初に他のPEからメッセージを受信しなくとも自主的にメッセージを送信する．このようなPEを起動PEと呼ぶ．起動PE以外のPEは，他のPEからメッセージを受信してからプログラムの実行を開始する．□

## ２．４　放送問題とリーダー選択問題

［定義４］N=(P,L)を任意のネットワークとする．Nにおける放送問題とは，あるPE u(∈P，放送PEと呼ぶ)の持つあるメッセージMをN中の全非故障PEに伝達する問題である．ただし，放送PEのみが起動PEになるとする．□

［定義５］N=(P,L)を任意のネットワークとする．Nにおけるリーダー選択問題とは，あるPE v(∈P)をリーダーとして選択する問題である．具体的にはvが「自分がリーダーである」ことを知り，v以外のPEが「リーダーの識別子はid(v)である」ことを知る問題である．ただし，１個以上の任意のPEが起動プロセッサになるとする□

［定義６］A を任意のアルゴリズム，N=(P,L)を任意のネットワーク，Qをある分散問題としたときに，リンクを伝搬中のメッセージが存在しない状況で，各起動PEが A の実行を開始したとする．このとき起動PEの起動のタイミングや各PEの動作速度，伝送遅延などの違いにより，様々な実行経過が考えられる．これらのどの実行経過においても，すべての非故障PEが有限時間内に停止し，すべての非故障PEが停止したとき問題 Qが解かれているとき，「A は N における問題 Q を解く」という．□

## ２．５　通信計算量

ここでは分散問題を解くために要するメッセージの個数に関する概念を定義する．

［定義７］アルゴリズムを実行したときにネットワーク全体で送信されるメッセージの総数（故障PEや故障リンクに対して送信されたメッセージを含む）を，そのアルゴリズムの「通信計算量」とよぶ．但し，起動PE，故障PEやリンクの遅延などの違いによりさまざまな実行経過があるが，その中で最も多くメッセージを用いる場合のメッセージ数で評価する（最悪時評価）．□

１つのメッセージで任意に多くのデータを通信できると考えるのは現実的ではないので，次の仮定をおく．

［仮定１］１つのメッセージ長はO(logn)ビットとする．また，各PEの識別子はO(logn)ビットで表せるとする．□

［仮定２］放送問題においてM（全PEに伝達しなければならないメッセージ）はO(logn)ビットで表せるとする．□

## ２．６　故障

［定義８］PEに故障があると，そのPEは送受信いずれも行えなくなる．また，そのPEが起動することはない．リンクに故障があると，そのリンクを用いた両方向の通信ができなくなる．□

以後，故障PEのことをFPE，故障していないPEをNPEと呼ぶ．

［仮定３］FPEおよび故障リンクを検知するような能力はネットワークにはない．□

［仮定４］アルゴリズム実行中はPEやリンクの故障が新たに生じたり，FPEや故障リンクが復旧したりしない．□

また，ネットワークにおいて，FPEと故障リンクの数には上限があり，それをそれぞれ$f_P$と$f_L$とする．各PEはこの値を利用してよいものとする．

［定義９］N=(P,L)に対してLive(N)=(P',L')は，全てのNPEとリンクよりなる部分グラフである．つまり$P_F$,$L_F$をそれぞれN中のFPEと故障リンクの集合とすると，$P'=P-P_F$，$L'=\{e\in P'\times P' \mid e\in L-L_F\}$とする．□

## ２．７　グラフ

［定義10］グラフＧは，頂点の集合VとVの相異なる二要素の二項組の集合Eの組(V,E)で定義される．

グラフG=(V,E)において，各i(1≦i＜m)に対し$(u_i,u_{i+1})\in E$を満たす相異なる頂点の系列$\langle u_1,u_2,\cdots,u_m\rangle$を$u_1$-$u_m$道と呼び，道に含まれる辺の数m-1をその道の長さと呼ぶ．２頂点u,vの間に長さ２以上のu-v道が存在し，$(u,v)\in E$であるとき，これを閉路という．任意の２頂点に対し，その長さが最小であるu-v道をu-v最短道と呼ぶ．このu-v最短道の長さをu-v距離と呼び，$d_G(u,v)$で表す．

グラフG=(V,E)が閉路を含まなければGを木と呼ぶ．木のある頂点uと任意の頂点vの距離がd以下であるとき，木Gはuを根とする高さdの木であるという．

グラフG=(V,E)，G'=(V',E')に対してV'⊆V,E'⊆Eがなりたつとき，G'をGの部分グラフという．特に,V=V'かつG'が木であるときG'をGの生成木と呼ぶ．□

## ３．CN-SDでのプロセッサとリンクの故障が混在する場合の放送問題

FPEが高々$f_P$個，故障リンクが高々$f_L$個存在するCN-SDで，放送問題を解くプロトコルdeforest（”森林開拓”の意）を示す．ただし，$f_P+f_L\leq n-2$とする．この条件はLive(N)が連結であることを保証する．

### ３．１　プロトコルdeforest

放送プロトコルdeforestは３つのステージからなる．

（ステージ１）最初に放送PE uから全てのPEに対して直接メッセージMを送信する．Mを受信したPEはMを受け取ったことをuに返信する．$f_P$個のFPEが存在し$f_L$個の故障リンク全てがuに接続している可能性があるので，uは$f_L+f_P$個のPEからの返信は期待できない．$n-f_L-f_P-1$個の返信を受信すればuはステージ２に移行する．

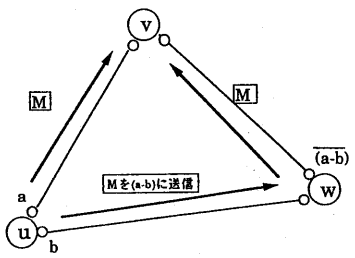



図２：方向感覚を利用した位置情報の伝達例

（ステージ２）放送PE uがMを（ステージ１でMを受信しなかった）PE vに伝達するには，リンク(u,v)が故障している場合，他のPEを介して中継する必要がある．もちろんこの中継PEとvとの間のリンクも故障している可能性があるので，複数の中継PEを作る必要がある（図３）．あるPE wを中継PEとして指定したとする．uはまだ返信を受け取っていないPEの位置を方向感覚を用いてwに連絡し，wはそれらのPEにMを送信する（図２）．wに対してMを受信したことに対する返信がきたならば，wはそのことをuに連絡する．各PEに対してuとwの両方からMを送信するので，故障リンクによってMを受信できないNPEの数は半分になる．従って，返信が期待できないPEの数は，$\lfloor f_L/2 \rfloor + f_P$個になる．Mは（ステージ１に受信したものも含めて）$n-\lfloor f_L/2 \rfloor - f_P - 1$個の返信を受信したならば，中継PEの数を１つ増やす．このようにして次々に中継PEを増やしていく．m個の中継PEを利用すれば，$n-\lfloor f_L/(m+1) \rfloor - f_P - 1$個のPEがMを受信することが保証される．従って$f_L$個の中継PEを用いれば$f_P$個以外のプロセッサにMを伝えることができる．

（ステージ３）高々$f_P$個のFPEが存在しうるので，$f_P$個のPEに対しては（故障していて永久に返信がこないかも知れないので，）Mの伝達の確認を待つことができない．そこでこれら$f_P$個のPEに対しては$f_L+1$個の隣接PEからM（とアルゴリズムの停止命令）を送信する．これにより，それらのPEがNPEならば故障リンクに妨げられることなく，有限時間以内にMを受信（してその後停止）する．

なお，各ステージにおける動作を森林開拓と見立てて（最初「woods(森林：Mを受信していないPE)」が生い茂っていると考え，「flat(平地:Mを受信したPE)」が広がっていく）アルゴリズムの各名称をつけた．

### ３．１．１　PEの状態

各NPE vのとりうる状態は以下の６つである．

woods：Mを受信していないPEの状態で，放送PE u以外のPEの初期状態でもある．

c_flat：uから直接Mを受信したPE（リンク(u,v)が故障していないことの判っている）の状態（close-flat:"近くの平地"の略）．

f_flat：中継PEからのみMを受信したPE（リンク(u,v)が故障しているかどうか不明）のPEの状態(far-flat:"遠くの平地"の略)．

colony：中継PEの状態("開拓村"の意味)．

col_bs：放送PEの状態(colony-base："開拓起点地"の略)．

halt：停止状態．

なお，以後”状態XXXのPE”を単に”XXX”と記述する．

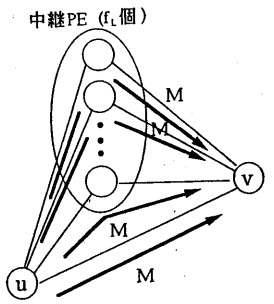



図３：リンク故障を回避するために中継PEを用いる

### ３．１．２　PEの局所変数

各プロセッサが用いる局所変数は以下の通り．

**・全NPEが用いる局所変数**

status：PEの状態を表す変数．

dir_cb(direction of colony-base)：放送PEに通じる（非故障リンクよりなる）道へのポート番号を記録する．

**・中継PE(colony)で用いる局所変数**

f-table[1..n-1]：自分（vとする）と非故障リンクでつながっていることの保証された，状態がf_flatのPEを記録する．f-table[t]＝trueならば，v[t]の状態がf_flatであることを示す．初期値はすべてfalse．

**・放送PE u(col_bs)で有効な局所変数**

netdata[1..n-1](network data)：他の全PEの状態を記録する．netdata[a]はu[a]の状態を記録する．初期値はすべてwoods．

#woods(number of woods)：netdata中で状態がwoodsのPEの総数．

#colony(number of colony)：netdata中で状態がcolonyのPEの数を示す．

#c_flat(number of c_flat)：netdata中で状態がc_flatのPEの数を示す．

netflat[1..n-1](network f_flat)：状態がf_flatのPE u[a]が，colony u[b]を介して放送PE uに連結しているとき，netflat[a]=bが成り立つ．初期値はすべて０．

### ３．１．３　メッセージ

以下の９種類のメッセージを用いる．

《fell1(M)》：最初に放送PE uから送信される(fellは"伐採"の意味)．

《clear1》：《fell1》に対する返信(clearは"伐採完了"の意味)．

《colonize(d)》：uからcolony，またはc_flat（vとする）に送信される．vはこのメッセージを受信すると，v[d]に《fell2》を送り，さらにvがc-flatならば，vはcolonyになる．(colonizeは"開拓村設立"の意味)

《fell2(M)》：colonyから送信される．

《clear2》：《fell2》に対する返信．

《clear3(d)》：colonyが，u[d]から《clear2》を受信したことをuに知らせるメッセージ．

《connect》：f_flat vが，リンク(u,v)が非故障であることをuに連絡するメッセージ(connectは"開拓起点地との連絡道完成"の意味)．

《order1(d)》：ステージ３に放送PE，colonyより送信する．受信したPE vはv[d]に《burn》を送信する(orderは"命令"の意味)．

《order2(d,d')》：ステージ３にuがcolony vに送信し，このメッセージを受信したvはv[d]に《order1(d')》を送信する．

《burn(M)》：(故障している可能性のある)まだuがメッセージMを受信したという返信を受信していないPEに対して送信され，かつそのPEを受信後に停止させる(burnは"(切り開いた土地を農地にするために)焼き払う行為"の意味)．

《halt》：停止命令．

### ３．１．４　PEの動作

簡単のため，アルゴリズムの記述に以下の送信命令を用いる．なお，これらの命令を実行するPEをvとする．

・send(i,(message))：v[i]に(message)を送信する．

・r-rec(t,message)：vはメッセージを受信するまで受信命令を繰り返す．最初に受信したメッセージはmessageに代入され，受信したポートはtに代入される．

col_bs uは#woodsの値によって，次のように３つのステージを実行する．（u以外のPEにはステージが関係なく，statusと受信メッセージによって動作する．）

ステージ１［deforestation（荒い伐採）］：
#woods＞$f_P+f_L$の間．

ステージ２［uproot　onebyone（一本ずつ引き抜く）］
$f_P+f_L \geqq$#woods＞$f_P$の間．

ステージ３［burn　up（残りを燃やしてしまう）］
$f_P \geqq$#woodsの間．

## 放送ＰＥの動作

（ステージ１：［deforestation］）

```
status:=col_bs;
#woods:=n-1; #colony:=0; #c_flat:=0;
for i=1 to n-1 do
        netdata[i]:=woods;
        netflat[i]:=0;
end for
for i=1 to n-1 do
        send(i,《fell1(M)》);
end for
while #woods＞fP＋fL do
        r-rec(t,message);
        if message=《clear1》 then
                #woods:=#woods-1;
                netdata[t]:=c_flat;
                #c_flat:=#c_flat+1;
        end if
end while
```

（ステージ２：［uproot　onebyone］）

```
while fP＜#woods do
        if  #woods≦fP+⌊fL/(#colony+1)⌋
         and #c_flat≧１) then
              netdata[t]=c_flatなるtを１つ選択;
              netdata[t]:=colony;
              #colony:=#colony+1;
              #c_flat:=#c_flat-1;
              for i=1 to n-1 do
                if netdata[i]=woods then
                  send(t,《colonize(i-t)》);
                end if
              end for
        end if
        r-rec(t,message);
        case message of
         《clear1》:
                #woods:=#woods-1;
                netdata[t]:=c_flat;
                #c_flat:=#c_flat+1;
         《clear3(d)》:
                if netdata[d]=woods then
                   #woods:=#woods-1;
                   netdata[d]:=f_flat;
                   netflat[d]:=t;
                end if
         《connect》:
                if netdata[t]:=woods then
                   #woods:=#woods-1;
                end if
                netdata[t]:=c_flat;
                #c_flat:=#c_flat+1;
        end case
end while
```

（ステージ３：［burn　up］）

netdataを参照して，colony,c_flat,f_flatの中から順に$f_L$個のPEを選ぶ．つまり，(1)#colony≧$f_L$ ならばcolonyだけから選び，(2)#colony＜$f_L$ かつ#c_flat＋#colony≧$f_L$ ならば，$f_L$-#colony個のc_flatとcolony全部を選び，(3)それ以外の場合には，$f_L$－#colony－#c_flat個のf_flatとcolony,c_flatのそれぞれ全部を選ぶ．

選んだ$f_L$個の各PE（u[d]とする）に対して次の動作を実行する．

```
for i=1 to n-1 do
        if netdata[i]=woods then
                if netdata[d]=c_flat
                  or netdata[d]=colony then
                   send(d,《order1(i-d)》);
                else
                   d':=netflat[d];
                   send(d',《order2(d-d',i-d)》)
                end if
        end if
end for
```

さらに以下の動作を行なう．

```
for i=1 to n-1 do
        if netdata[i]=c_flat
          or netdata[i]=colony then
                send(i,《halt》)
        end if
end for
status:=halt;
```

## 放送ＰＥ以外のＰＥの動作

イ）status＝woodsのPEの動作

```
while status=woods do
        r-rec(t,message);
        case message of
         《fell1(M)》:
                status:=c_flat;
                dir_cb:=t;
                send(t,《clear1》);
         《fell2(M)》:
                status:=f_flat;
                dir_cb:=t;
                send(t,《clear2》);
         《burn(M)》:
                dir_cb:=t;
                status:=halt;
        end case
end while
```

ロ）status＝c_flatのPEの動作

```
while status=c_flat do
        r-rec(t,message);
        case message of
         《colonize(d)》:
                status:=colony
                send(d,《fell2(M)》);
         《order1(d)》:
                send(d,《burn(M)》);
         《halt》:
                status:=halt;
```

```
        end case
end while
```

ハ) status＝f_flatのPEの動作

```
while status=f_flat do
        r-rec(t,message);
        case  message of
                《fell1(M)》:
                        status:=c_flat;
                        dir_cb:=t;
                        send(t,《connect》);
                《order1(d)》:
                        send(d,《burn(M)》);
                《halt》:
                        status:=halt;
        end case
end while
```

ニ) status＝colonyのPEの動作.

```
while status=colony do
        r-rec(t,message)
        case message of
         《colonize(d)》:
                send(d,《fell2(M)》);
         《clear2》:
              f-table[t]:=true;
              send(dir_cb,《clear3(t-dir_cb)》);
         《order1(d)》:
              send(d,《burn(M)》);
         《order2(d,d')》:
              send(d,《order1(d')》);
         《halt》:
              for i=1 to n-1 do
                        if f-table[i]=true then
                         send(i,《halt》);
                        end if
              end for
              status:=halt;
        end case
end while
```

### 3．2　アルゴリズムの正当性

［定理1］アルゴリズムdeforestは，有限時間内に放送問題を解く．

（証明）最初に放送PE uが有限時間内に停止することを示す．

ステージ1はuが《fell1》を他の全てのPEに送信し，$n-f_P-f_L-1$個の《clear1》が返信されるのを待つだけなので有限時間内に終了する．ステージ3もuが有限個数のメッセージを一方的に送信するだけなので，有限時間内に終了する．

ステージ2が有限時間内に停止しないと仮定する．uが送信する有限個の《fell1》と《colonize》に対する有限個のメッセージの送返信によってステージ2は動作しているので，いずれ以下のような状態になる．

「（放送PE uとcolonyからwoodsへのリンクがすべて故障していて）woodsからc_flatやf_flatになるものがなく，c_flatがいずれもcolonyにならず，どのf_flatもc_flatとならない」

上記の状態が発生したと仮定する．もしc_flatが一つでも存在すると，uとcolonyから残りのwoodsへのリンクが全て故障しているので，これらのリンク数は$f_L$を越えず，

$$\#woods \leqq f_P + \lfloor f_L/(\#colony+1) \rfloor$$

が成り立っている．すると，uはそのc_flatへ《colonize》を送信して有限時間内にc_flatはcolonyになる．よって，c_flatは一つもないことになる．

すると$f_P+f_L \leqq n-2$の条件からネットワークはNPEが非故障リンクで連結であるので，uから非故障リンクを少なくとも三つ以上介さないと残りのwoodsに対してつながらないことになる．しかし，これが成り立たないことを以下の補題1と補題2で示す．

［補題1］頂点数nの完全グラフG=(V,E)の任意の部分グラフG'=(V,E')について，$|E|-|E'| \leqq n-2$ならば，任意の頂点を根とする高さ2以下の生成木がG'に存在する．

（証明）$|E|=|E'|$のときはあきらか．$|E|>|E'|$のとき，E-E'=Fとしたときに(u,v)∈Fなるu,v∈Vに対し，$d_{G'}(u,v)>2$とする．このときu,v以外の任意の頂点w∈Vについて(u,w)∈Fまたは(v,w)∈Fとなる．このようなwのとりかたがn-2通りあり，(u,v)∈Fなので，$|E|-|E'| \geqq n-1$となって矛盾する．よって，任意のu,vについて，$d_{G'}(u,w) \leqq 2$が成り立つ．よって，任意の頂点を根とする高さ2以下のG'の生成木が存在する．□

［補題2］PE数n，故障リンク数が高々$f_L$，FPE数が高々$f_P$のCN（ただし$f_P+f_L \leqq n-2$）において，任意のNPE uに対して「uを根とする高さ2以下の生成木」が少なくとも一つは存在する．

（証明）FPEとそれに連結するリンクをすべて除外してできるネットワークは頂点数$n-f_P$のCNである．また$f_L \leqq n-f_P-2$である．よって，補題1より高さ2以下の生成木は存在する．□

よって先の仮定は否定されステージ2も有限時間内に終了する．

故に各ステージは有限時間内に終了し，uは有限時間以内に停止する．

次にu以外のNPEがMを受信し，有限時間内に停止することを示す．

ステージ3までに放送PEがメッセージMの受信を確認しているNPE(uのnetdataでの状態がcolony,c-flat,f-flatのいずれかのもの)は，途中で状態がf_flatからc_flatに変化したPEを除いて，Mを受信したのと同じ道筋で《halt》を受信する．途中で状態がf_flatからc_flatに変化したPEは，uがステージ3を開始する前に《connect》が受信されていればuから，そうでなければcolonyから《halt》を受信する．

ステージ3までに放送PEがメッセージMの受信を確認していなかったNPEに対して，ステージ3で$f_L+1$個の異なるプロセッサから《burn》を送信するので，少なくともそのうちの1個は受信される．

これらのことから全てのNPEは有限時間内にMを受け取って，停止する．

故にアルゴリズムdeforestは有限時間内に放送問題を解く．□

### 3．3　アルゴリズムの評価

［補題3］アルゴリズムdeforestの実行中にcolony となるPEは高々$f_L$個である．

（証明）col_bsはステージ2においてのみ新しいcolonyを作ろうと《colonize》を送信する．このとき，

$$\#woods \leqq f_P + \lfloor f_L/(\#colony+1) \rfloor$$

が成り立つ．また，まだステージ3に移行していないので　$f_P < \#woods$ が成立つ．従って，

$$\#colony < f_L$$

を導くことができる．□

［補題４］アルゴリズムdeforestの実行中に送信される《colonize》は高々$f_P \cdot f_L + O(f_L \log f_L)$個である．

（証明）

PEがcolonyとなったときに《colonize》はそのときの#woodsの値だけ送信される．PEがcolonyとなるとき

$\#woods \leqq f_P + \lfloor f_L/(\#colony+1) \rfloor$

が成り立つので，ｋ番目にcolonyとなるPEには高々$(f_P + (f_L/k))$個の《colonize》が送信される．また，補題４よりcolonyとなるPEの数は高々$f_L$個であるので，アルゴリズム全体の《colonize》の送信回数は高々

$$\sum_{k=1}^{f_L} (f_P + (f_L/k)) = f_P \cdot f_L + f_L \log f_L + O(f_L)$$

となる．□

［定理２］アルゴリズムdeforestの最悪時通信計算量は$O(n + f_P \cdot f_L + f_L \log f_L)$である．

（証明）《fell1》はステージ１に放送PE uが他の全てのPEに送信するだけなのでその総数はｎ-1個である．《clear1》はその返信なので高々ｎ-1個である．

ステージ２において，《colonize》が１つ送信されると，《fell2》が１つ送信され，その返信として《clear2》，さらにそれを受信したcolonyによって《clear3》が送信される．よって《fell2》，《clear2》《clear3》の送信総数は《colonize》の送信個数を越えない．補題４よりこれらのメッセージの送信個数はいずれも高々$f_P \cdot f_L + O(f_L \log f_L)$個である．

ステージ３の動作より，《order1》は選ばれた$f_L$個の中継PEへそれぞれ$f_P$個ずつ送信される．従って送信個数は高々$f_P \cdot f_L$個である．《order2》は《order1》の送信個数を越えることはなく，《burn》は《order1》の送信個数と同じで，どちらの送信個数も高々$f_P \cdot f_L$個である．また，《halt》の送信個数は高々2nである．

よってアルゴリズム全体での最悪時通信計算量は$O(n + f_P \cdot f_L + f_L \log f_L)$である．□

## ３．３　CN-SDでの放送問題の通信計算量の下界

FPEと故障リンクの混在するCN-SDでの放送問題を解くアルゴリズムdeforestの通信計算量はオーダー的に最適であることを示す．

［補題５］[10]PE数ｎ個のCN-SDに$f_L$個($\leqq$ n-2)のリンク故障のある場合の放送問題の通信計算量の下界は$\Omega(n + f_L \log f_L)$である．

［定理３］PE数ｎ個のCN-SDに$f_L$個のリンク故障と$f_P$個のPE故障がある場合の放送問題の通信計算量の下界は$\Omega(n + f_P \cdot f_L + f_L \log f_L)$である．ただし$f_P + f_L \leqq n-2$とする．

（証明）$f_P = 0$の場合を考えれば，補題５より，下界$\Omega(n + f_L \log f_L)$が証明できる．

放送問題を解く任意のアルゴリズムをＡとする．ネットワークN=(P,L)に対して，P'を$|P'| = f_P + 1$かつ$u \notin P'$を満たすPの任意の部分集合とし，$L' = \{(v,w) \mid v \in P', w \in P-P'\}$とする．Ａは，L'のうち少なくとも$(f_P+1)\cdot(f_L+1)$個のリンクにメッセージを送信するような実行経過があることを示す．

ＡがL'に$(f_P+1)\cdot(f_L+1)$個より少ないリンクにメッセージを送信して終了すると仮定すると，$|L''| \leqq f_L$（ただし，$L'' = \{(x,w) \in L' \mid (x,w)$はＡがメッセージを送信したリンク$\}$）が成り立つようなあるNPE $x(x \in P')$が存在する．すると，P'のx以外のPEが全てFPEで，L''が全て故障リンクである場合，xはMを受信することができない．よって先の仮定は誤っている．

故にＡは少なくとも$(f_P+1)\cdot(f_L+1)$個のメッセージを送信し，放送問題の通信計算量の下界は$\Omega(n + f_P \cdot f_L + f_L \log f_L)$である．□

## ４．方向感覚のないCNにおける放送問題

［補題６］[10]PE数ｎ個のCNに$f_L$個（$f_L \leqq n-2$）のリンク故障のある場合の放送問題の通信計算量の下界は$\Omega(n + n \cdot f_L)$である．

［定理４］PE数ｎ個のCNに$f_L$個のリンク故障と$f_P$個のPE故障がある場合（ただし$f_P + f_L \leqq n-2$）の放送問題の通信計算量は$\Theta(n + n \cdot f_L)$である．

（証明）３．で述べたdeforestアルゴリズムを次のように変更する．（変数などの細かい変更は省略）

(1)新たにcolonyとなったPEは放送PE uに通じる以外の全てのポートへ《fell2》を送信する．

(2)《order1》を受信したPEは《order1》を受信したポート以外の全てのポートへ《burn》を送信する．

(3)《order2》はidによって中継PEを指定する．

このような変更を加えてできるプロトコルは，《fell2》と《burn》の送信個数のみがdeforestよりも多くなる可能性がある．

《fell2》がcolonyごとに(n-2)個送信され，（依然としてcolonyの総数が高々$f_L$個なので）《fell2》はアルゴリズム中に高々$n \cdot f_L$個送信される．《burn》はステージ３で選ばれた$f_L$個のPEから高々(n-2)個ずつ送信されるので，全部で高々$n \cdot f_L$個送信される．よってこのアルゴリズム全体での通信計算量は$O(n+n \cdot f_L)$となる．

補題６より$\Omega(n+n \cdot f_L)$が成り立つので，CNにおける放送問題の通信計算量は$\Theta(n + n \cdot f_L)$である．□

定理２，４から方向感覚が放送問題の通信計算量を減らす上で有用であることが判る．

## ５．故障の混在するモデルにおけるリーダー選択問題

ここでは$f_P + f_L \leqq n-2$かつ$f_P < n/2$（$f_P \geqq n/2$では，故障リンクがなくてもリーダー選択問題を解くアルゴリズムは存在しない[8]）のときのリーダー選択問題を解くアルゴリズムdfLEAD（$f_P + f_L < n/4$のときに用いるdfLEAD1と，$n-2 \geqq f_P + f_L \geqq n/4$のときに用いるdfLEAD2よりなる）を示す．

## ５．１　アルゴリズムdfLEAD1

文献(10)のアルゴリズムLEAD(f)（$f_P=0$，$f_L < n/4$のときのリーダー選択アルゴリズム）は，$f_P + f_L < n/4$の場合でも，１つのPEをリーダーとして選択する部分に関しては正しく動作する．（この部分の通信計算量は$O(n+k\cdot(f_P+f_L))$である）リーダー選択後，リーダーの識別子を放送する通知フェーズをdeforestプロトコルで行なえば故障の混在するモデルのリーダー選択問題を解くことができる．これがdfLEAD1であり，全体の通信計算量は$O(n + k(f_P+f_L) + f_P \cdot f_L + f_L \cdot \log f_L)$になる．

## ５．２　アルゴリズムdfLEAD2

プロトコルdeforestでは，中継PEを最大$f_L$個まで作ることによって全NPEにメッセージを送信した．deforest実行時において，中継PEをｍ個用いるとMを受けとれないNPEの個数は高々$f_P + \lfloor f_L/(m+1) \rfloor$個である．従って中継PEを２つ作れば，$(n-f_P-\lfloor f_L/3 \rfloor - 1)$個のNPEには確実にメッセージを伝達できる．一方，$f_P + f_L \leqq n-2$と$f_P < n/2$より，$(n-f_P-\lfloor f_L/3 \rfloor) > n/3$である．そこで次のようなアルゴリズムdfLEAD2でリーダー選択問題が解ける．

**(アルゴリズムdfLEAD2)**

(1)起動プロセッサは”立候補者”となり，２個の

中継PEを作って自分の識別子を放送し，受信PEが
a)”立候補者”でも”支持者”でもないか，
b)より識別子の小さい”立候補者”または，その
”支持者”ならば，それを自分の”支持者”にする．
このとき各NPEは一時に２つ以上の”立候補者”の
”支持者”にならないようにする．このために，支持している”立候補者”を変えるには現在支持している”立候補者”に許可を求め，その返事がくることが条件になる．
（支持者の数＋中継PEの数＋１）がn/3より多くなった”立候補者”は”リーダー候補”になる．
”リーダー候補”となったPEは，支持者に別のPEの支持者となる許可を与えない．これにより”リーダー候補”になるPEは高々２個になる．
(2)”リーダー候補”となったPEは，deforestプロトコルをもちいて自分の識別子を放送し，n/2個より多くのPEを（先の”支持者”と同じような）”投票者”とする事に成功するとリーダーになる．
(3)リーダーの識別子の放送をdeforestプロトコルで行なう．
（注意：ネットワークのサイズｎが小さいときには中継PEを作る時点で問題が解けている場合があるので，ｎが４，５個のときには中継PEの個数を１個に，ｎが３個のときには中継PEをつくらないようにする．）
dfLEAD2は２つの中継PEによる部分的な放送と，deforestプロトコルによる放送，そしてその２つの放送にに対する返信によって成り立つ．「部分的な放送」に要するメッセージ数は各起動PEごとにO(n)で，deforestによる放送は高々３回しか行なわれない．そしてこれらの放送に対する返信は，これらの放送に用いられるメッセージの定数倍になるので，アルゴリズム全体での通信計算量は，$O(n+n\cdot k+f_P\cdot f_L+f_L\cdot \log f_L)$になる．
$f_P+f_L \geqq n/4$よりこれは
$O(n+k(f_P+f_L)+f_P\cdot f_L+f_L\cdot \log f_L)$と表せる．

### ５．３　アルゴリズムdfLEADの通信計算量

［定理５］dfLEAD1とdfLEAD2によって構成されたリーダー選択アルゴリズムdfLEADは，$n-2 \geqq f_P+f_L$かつ，$f_P<n/2$の条件下で，CN-SDにおけるリーダー選択問題を通信計算量
$O(n+k(f_P+f_L)+f_P\cdot f_L+f_L\cdot \log f_L)$で解く．□
一方，次の定理６により，この通信計算量は最適である．
［定理６］CN-SDにおけるリーダー選択問題の通信計算量の下界は，$\Omega(n+k(f_P+f_L)+f_P\cdot f_L+f_L\cdot \log f_L)$である．
（証明）各起動PEが$(f_P+f_L)$よりも少ないメッセージを送信するようなリーダー選択問題を解くアルゴリズムＡが存在すると仮定する．起動PEが１個だけのときに最初に出すメッセージがすべて故障リンクまたはFPEに送信される場合があり，このときＡはリーダー選択問題を解くことができない．従って，通信計算量の下界は
$\Omega(k(f_P+f_L))$である．一方，リーダー選択問題はリーダーの識別子を全てのNPEが知って終了しなければならないので，放送問題と同じ下界$\Omega(n+f_P\cdot f_L+f_L\cdot \log f_L)$が成り立つ．□

### ６．　あとがき

リンクとPEの故障が混在する完全ネットワークにおける放送問題について考察し，方向感覚の有無が通信計算量のオーダーに影響を与えることを示し，その放送プロトコルdeforestを用いてリーダー選択問題を解決するアルゴリズムを設計した．本稿で述べた放送プロトコルdeforestを利用して連結性関連問題など，さらに複雑な分散問題を解くことが今後の課題である．

**謝辞**

日頃から分散アルゴリズムについて有意義な御討論を頂く萩原兼一助教授，韓国広原大学朴政鎬博士に感謝致します．

**参考文献**

(1)H.H.Abu-Amara:"Fault-tolerant distributed algorithm for election in complete networks",Tech. Rep. RUU-CS-87-10, Dept. of Electrical and Computer Eng., University of Illinois(Apr. 1987).
(2)R.Bar-Yehuda and S.Kutten:"Fault tolerant distributed leader election with termination Detection in General Undirected Networks",Tech.Rep. #409,Department of Computer Sci.,Technion (Apr.1986, Revised August 1986).
(3)R.Bar-Yehuda,S.kutten,Y.Wolfstahl and S.Zaks : "Making Distributed Spanning Tree Algorithms Fault-Resilient (Extended Abstract)",Proc.of STACS 87 pp.432-444 (Feb. 1987).
(4)M.Y.Chan and F.Y.L.Chin : "Optimal Resilient Ring Election Algorithms", Proc. of 2nd International Workshop on Distributed Algorithms (LNCS 312),(Jul. 1987).
(5)R.G.Gallager, P.A.Humblet and P.M.Spira : "A distributed algorithm for minimum-weight spanning trees", ACM TOPLAS,5,1, pp.66-77(Jan.1983).
(6)O.Goldreich and L.Shrira:"Electing a Leader in a Ring with Link Failures", Acta Informatica,24, pp.79-91 (February 1987).
(7)E.Korach,S.Moran and S.Zaks : "Tight lower and upper bounds for some distributed algorithms for a complete network of processors", Prc. of 3rd PODC, pp.199-207(Aug. 1984).
(8)S.Kutten, Y.Wolfstahl and S.Zaks:"Optimal distributed t-resilient election in complete networks", Tech.Rep. #430,Dept. of Computer Sci., Technion(Aug. 1986).
(9)M.C.Loui, T.A.Matsushita and D.B.West : "Election in a complete network with a sense of direction", Information Processing Letters,22,4, pp.185-187 (Apr. 1986).
(10)増澤，西川，萩原，都倉，藤田："方向感覚付き完全ネットワークにおけるリンク故障を考慮したリーダー選択問題"，信学技報，COMP88-98(1989.3).
(11)D.Rotem, K.Korach and N.Santoro :"Analysis of a distributed algorithm for extrema finding in a ring", Journal of Parallel and Distributed Computing,4,6, pp.575-591 (Dec. 1987).
(12) N.Santoro : "On the message complexity of distributed problems",International Journal of Computer and Information Sciences,13,3, pp.131-147 (Jul. 1984).
(13) N.Santoro : "Sense of direction,topological awareness and communication complexity", SIGACT NEWS,16,2, pp.50-56 (1984).