アルゴリズム 13－3
（1990．1．31）

故障発生時における計算機網の 連結性を判定する分散アルゴリズム

和田 幸一*，守谷 幸男*，川口 喜三男*，森下 正浩**

*名古屋工業大学 電気情報工学科

**オムロン（株）

本稿では，非同期式計算機網において計算機やリンクに故障が生じたときの計算機網の連結性を判定する分散アルゴリズムの存在性や通信複雑度と理想時間複雑度の上界及び下界について考察する．故障計算機や故障リンクに隣接あるいは接続する計算機がその故障を検知する機構をもつと仮定する．また，計算機が故障する場合と故障をリンクのみに限定した場合に分けて考察する．特に，後者の場合は，対象とする計算機網を各計算機が保持する計算機網情報によって分類し，基本情報（自分の識別子と接続リンク数）のみの場合，基本情報と隣接計算機の識別子の場合，基本情報と計算機網のサイズの場合について考察する．

Distributed algorithms for connectivity of networks with faulty elements

Koichi Wada*, Yukio Moritani*, Kimio Kawaguchi-Izawa*
and
Masahiro Morishita**

*Department of Electrical and Computer Engineering,
Nagoya Institute of Technology
Gokiso-cho, Syowa-ku, Nagoya 466, Japan

**OMRON Corporation
Shimokaiinji, Nagaokakyou-shi, Kyoto 617, Japan

We consider asynchronous distributed algorithms for the connectivity of a network with faulty elements. It is assumed that every node can detect the faults of adjacent nodes and incident links automatically. We discuss such algorithms in the following cases: (1)faults occur at nodes, (2)faults occur only at links. In case (2), networks are classified three classes by the information about the network topology each node knows. (a)Each node knows its identity and the number of incident links (called basic information). (b)Each node knows the basic information and the identities of neighbor nodes. (c)Each node knows the basic information and the size of the network. We show that there do not exit such algorithms in case (1) and (2-a) and discuss the upper and lower bounds on the communication complexity and the ideal time complexity in case (2-b) and (2-c).

1 まえがき

計算機網上の複数台の計算機に分散している局所的な情報から，互いにメッセージを交換することにより，ある問題を解くアルゴリズムを分散アルゴリズムと呼ぶ．従来，種々の分散アルゴリズムが提案されているが [1, 2, 3, 4]，これらの多くは連結な計算機網を対象として考えられている．しかし，実際の計算機網では，リンクや計算機に故障が発生することが少なくなく，これらの故障により計算機網が非連結になることがある．計算機網が非連結であると，計算機網に通信不可能な計算機が存在することになり，従来の分散アルゴリズムがうまく動作しない場合がある．従って，計算機網においてリンクや計算機に故障が発生した場合に計算機網が連結であるかどうかを判定することは重要である．

本稿では，非同期式計算機網において計算機やリンクに故障が発生した場合に計算機網が連結であるかどうかを判定する問題 (故障時の連結性判定問題と呼ぶ) を解く分散アルゴリズムを考察する．始めに，計算機が故障する場合，各計算機がアルゴリズム実行前に如何なる計算機網情報 (計算機網に関する情報) を保持していても，連結性判定問題を解く分散アルゴリズムは存在しないことを示す．次に，故障をリンクのみに限定し，計算機網を各計算機が保持する計算機網情報によって，以下の3つのモデルに分類して考察する．

- 各計算機が基本情報（自分自身の識別子と接続しているリンク数）だけを保持している場合（基本情報モデル）．
- 各計算機が基本情報と隣接計算機の識別子を保持している場合（隣接計算機既知モデル）．
- 各計算機が基本情報と計算機網の計算機総数を保持している場合（サイズ既知モデル）．

分散アルゴリズムの効率の評価は通信複雑度 M と理想時間複雑度 T により行われる [5]．通信複雑度は，計算機網全体で分散アルゴリズムの実行中に交換されるメッセージ総数である．理想時間複雑度は，各計算機の処理時間を無視し，メッセージの伝播時間が単位時間であると考えた場合の分散アルゴリズムの実行時間である．本稿では，基本情報モデルにおいて，リンク故障時の連結性判定問題を解く分散アルゴリズムが存在しないことを示す．また，計算機網の計算機数，正常リンク数，故障リンク数をそれぞれ n，e_f，f と表すとき，隣接計算機既知モデルに対しては，$M = O(e_f + \min(fn, n^2))$，$T = O(n \log f)$ である分散アルゴリズムを示し，このモデルにおける各複雑度の下界は，$M = \Omega(\max(n, \min(fn\frac{\log f}{\log n}, n^2)))$，$T = \Omega(n)$ であることを示す．サイズ既知モデルに対しては，$M = O(e_f + n \log f)$，$T = O(n \log f)$ である分散アルゴリズムを示し，下界は $M = \Omega(e_f)$，$T = \Omega(n)$ であることを示す．

2 定義

2.1 諸定義

X，Y を集合とするとき，$X + Y = X \cup Y$，$X - Y = X \cap \bar{Y}$ と定義する．特に，$Y = \{y\}$ であるとき，$X + Y$，$X - Y$ を簡単に $X + y$，$X - y$ と表す．

関数 $f(n)$，$g(n)$ について，正定数 c，n_0 が存在して，任意の $n \geq n_0$ に対して，$f(n) \leq cg(n)$ となるとき，$f(n) = O(g(n))$ と表す．また，正定数 c，n_0 が存在して，任意の $n \geq n_0$ に対して，$f(n) \geq cg(n)$ となるとき，$f(n) = \Omega(g(n))$ と表す．

グラフ $G = (V, E)$ は，有限な空でない点の集合 V と，辺と呼ばれる V の相異なる要素の非順序対の集合によって定義される．

$e = (u, v)$ がグラフ G の辺であるとき，u と v は隣接するといい，u と e あるいは v と e は接続するという．e_1 と e_2 が G の異なる辺で同じ点に接続するとき，e_1 と e_2 は隣接するという．

グラフ G の頂点 v に接続する G の辺の数を，G における頂点 v の次数と呼び，$deg_G(v)$ あるいは簡単に $deg(v)$ と表す．

2つのグラフ $G = (V, E)$ と $G' = (V', E')$ が，$V' \subseteq V$ かつ $E' \subseteq E$ を満たすとき，グラフ G' をグラフ G の部分グラフという．特に，$V = V'$ であるとき，グラフ G' はグラフ G の生成部分グラフという．また，$G = (V, E)$ と $U \subseteq V (U \neq \phi)$ に関して，U を点集合とする G の極大な部分グラフを G の U による誘導部分グラフと呼び，$G(U)$ と表す．

グラフ $G = (V, E)$ の互いに異なる頂点の有限列 $P : u_0, u_1, \cdots, u_{n-1}, u_n$ が，各 $i (1 \leq i \leq n)$ に対して，$(u_{i-1}, u_i) \in E$ を満たすとき，P を u_0-u_n 通路と呼ぶ．このときの n を P の長さという．

グラフ G のある隣接している2つの点 u，v に対して，長さ2以上の u-v 通路が存在するとき，G には閉路が存在するという．

G において，u-v 通路が存在するとき，u と v は連結しているという．G において，相異なるどの2つの点も連結しているとき，グラフ G は連結であるという．連結でないグラフを非連結であるという．G の極大な連結部分グラフを G の連結成分という．連結グラフ $G = (V, E)$ と $A (\subset V)$ に対して $G(V - A)$ が非連結であるとき，A を G の分離集合と呼ぶ．

閉路の存在しないグラフを林といい，連結な林を木という．また，グラフ G の生成部分グラフで林であるものを生成林といい，グラフ G の生成部分グラフで木であるものを生成木という．

ある1点を根と呼ぶ木を根付き木という．点 r を根とする根付き木 T において，$P : u_0 (= r), u_1, \cdots, u_{n-1}, u_n$ が通路であるとき，各 $i (1 \leq i \leq n)$ に対して u_{i-1} を u_i の親といい，u_i を u_{i-1} の子という．また，各 i，$k (0 \leq i \leq k \leq n)$ に対して，u_i を u_k の先祖，u_k を u_i の子孫という．特に，各 u_i に対して，u_i 以外の u_i の先祖を u_i の真の先祖，u_i 以外の u_i の子孫を u_i の真の子孫という．子を持たない点を葉と呼ぶ．T の各点 u に対して，u- r 通路の長さを u の深さといい，任意の点の深さの最大値を根付き木 T の深さという．以下，本稿では根付き木を単に木と呼ぶ．

2.2 計算機網と分散アルゴリズム

計算機網 $N = (P, L)$ は，互いに異なる整数の識別子をもつ計算機の空でない有限集合 P と，リンクと呼ばれる P の相異なる要素の非順序対の集合によって定義される．計算機 u，$v (\in P)$ に対して，$(u, v) \in L$ であるとき，計算機 u，v 間に全2重通信路が存在し，u から v へあるいは v から u へ互いに独立に通信ができるものとする．即ち，リンク (u, v) は，u から v への通信路と v から u への通信路の2つの独立な通信路から構成される．このとき，u は v の (v は u の) 隣接計算機という．また，計算機網 N の計算機総数 $|P|$ を N のサイズという．

本稿では，計算機網をグラフとみなし，グラフに関する用語や記法を計算機網に関しても用いる．

各計算機 u には，u が識別できる自然数の番号のついたポートが存在し，各ポートは送信用と受信用の2つの待行列 (queue) から構成されている．u に接続する $deg(u)$ 本のリンクは，1番から $deg(u)$ 番までのポートにそれぞれ1本ずつ接続されているものとする．p 番のポートをポート p と呼ぶ．計算機網 N において，u のポート p にリンク (u, v) が接続しているとき，この接続関係を $\rho_N(u, p) = (u, v)$ と表す．但し，計算機網が明らかなときには，$\rho_N(u, p)$ を単に $\rho(u, p)$ と表す．

各計算機 u はプログラムにおいて次の2つの通信命令を実行できるものとする．

- $send(\ll \mathrm{M}(s) \gg, p)$

  M：メッセージの種類

  s：メッセージ M と共に送信する情報

  p：ポート番号 $(1 \leq p \leq deg(u))$

u のポート p の送信用待行列の最後尾に $\ll \mathrm{M}(s) \gg$ を入れる．

- *receive*

すべてのポートからランダムにポートを1つ選び，そのポートの受信用待行列の先頭にメッセージがあれば，そのメッセージを取り出し，そのメッセージの種類とそのメッセージと共に受信した情報及び受信したポート番号をバッファに入れる．選んだポートの受信用待行列の先頭にメッセージがない場合は，もう1度ポートを選び直し，メッセージをバッファに入れるまで続ける．

$\rho(u,p) = (u,v)$ であり u が "$send(\ll \mathrm{M}(s) \gg, p)$" を実行したとき，$u$ は v に $\ll \mathrm{M}(s) \gg$ を送信したという．また，u が "*receive*" の実行によってポート p の受信用待行列から $\ll \mathrm{M}(s) \gg$ が取り出されたとき，u は v から $\ll \mathrm{M}(s) \gg$ を受信したという．

本稿では，計算機網に関して次のことを仮定する．

［仮定1］計算機網は計算機やリンクに故障が発生しない限り連結である．

［仮定2］計算機網には共有メモリは存在せず，計算機間の通信はリンクを介するメッセージの送受信のみで行われる．

［仮定3］ポートを構成する待行列はサイズに制限のない FIFO (First-In First-Out) 待行列とする．

［仮定4］計算機 u とその隣接計算機 v において，それぞれのポート p，q に対して $\rho(u,p) = \rho(v,q)$ であるとき，u のポート p の送信用待行列の先頭にあるメッセージは，リンク (u,v) を伝って有限時間内に v のポート q の受信用待行列の最後尾に入る．但し，この時間の上限は仮定しない．

［仮定5］計算機網を構成する各計算機の処理速度は一般に異なる．

［仮定4］の後半及び［仮定5］は計算機網が非同期であることを意味する．

ある問題を解くのに必要な情報が，計算機網上の複数の計算機に分散しているとき，この問題を分散問題と呼ぶ．各計算機がメッセージの送受信によりお互いの情報を交換しながら，分散問題を解くアルゴリズムを分散アルゴリズムと呼ぶ (以下，単にアルゴリズムと呼ぶ)．アルゴリズムは，各計算機が実行するプログラムによって表される．但し，すべての計算機は同一のプログラムを実行するが，初期状態 (プログラム実行前の状態) で各計算機が保持している計算機網に関する情報はそれぞれ異なる．各計算機 u が初期状態で保持している計算機網に関する情報を u の計算機網情報と呼ぶ．

各計算機 u が保持している計算機網情報として次のような情報を考える．

- u の識別子 $id(u)$
- u の次数 $deg(u)$
- u の隣接計算機の識別子 $adid(p)$ $(1 \leq p \leq deg(u))$
  但し，$\rho(u,p) = (u,v)$ のとき，$adid(p) = id(v)$
- 計算機網 N のサイズ $size(N)$

このうち，$id(u)$ と $deg(u)$ を基本情報という．

各計算機 u の計算機網情報によって解ける問題のクラスやアルゴリズムの効率が異なることがある．

計算機網 N の各計算機 u の計算機網情報によって，計算機網を次のように分類する．

- 基本情報モデル　　：基本情報のみ．
- 隣接識別子既知モデル：基本情報と $adid$ のみ．
- サイズ既知モデル　　：基本情報と $size(N)$ のみ．

計算機網上の計算機は，指定された開始方法によって，指定された時間に，指定された初期設定を行った後，プログラムの実行を開始する．プログラムの実行を開始していない計算機は，隣接計算機からメッセージを受信することによってプログラムの実行を開始する．隣接計算機からメッセージを受信することなしに，プログラムの開始条件を満たした計算機は，自主的にプログラムの実行を開始する．このような計算機を始動計算機と呼ぶ．

ある1つの始動計算機がプログラムの実行を開始したとき，アルゴリズムが開始されたという．また，すべての計算機がプログラムを停止したとき，アルゴリズムが終了したという．同様に，連結成分 C のある1つの始動計算機がプログラムの実行を開始したとき，C においてアルゴリズムが開始されたといい，C のすべての計算機がプログラムを停止したとき，C においてアルゴリズムが終了したという．

アルゴリズム $\mathcal{A}$ に対して，アルゴリズムが開始されてから終了するまでに通信されるメッセージ数の最大値を $\mathcal{A}$ の通信複雑度といい，$M(\mathcal{A})$ で表す．メッセージに関して次の仮定をおく．

［仮定6］1つのメッセージによって送信できる情報のデータ量は $O(\log n)$ ビットとする．但し，n は計算機網のサイズとする．

各リンク (u,v) におけるメッセージの伝達遅延を1単位時間と仮定した時間を理想時間という．アルゴリズム $\mathcal{A}$ に対して，各計算機の処理時間を無視した場合の，アルゴリズムの開始から終了までの理想時間計算量の最大値を $\mathcal{A}$ の理想時間複雑度といい，$T(\mathcal{A})$ で表す．

故障に関して次の仮定をおく．

［仮定7］計算機の故障が発生すると，その計算機に隣接している計算機はその計算機の故障を自動的に検知することができる．同様に，リンクの故障が発生すると，そのリンクに接続している計算機はそのリンクの故障を自動的に検知することができる．

［仮定8］故障した計算機及びリンクはまったく使用できない．

［仮定9］アルゴリズム実行中に計算機やリンクが新たに故障したり，故障した計算機やリンクが復旧することはない．

故障していない計算機，リンクをそれぞれ正常計算機，正常リンクと呼び，故障している計算機，リンクをそれぞれ故障計算機，故障リンクと呼ぶ．計算機網 $N = (P, L)$ に対して，F_P $(\subseteq P)$ を故障計算機集合と呼び，$F_L (\subseteq L)$ を故障リンク集合と呼ぶ．$F = F_P \cup F_L$ とするとき，次のような計算機網 $N - F = (P', L')$ を正常計算機網と呼ぶ．

$$P' = P - F_P$$

$$L' = \{(x,y) \mid (x,y) \in L - F_L, \{x,y\} \cap F_P = \phi\}$$

計算機網 N において，計算機やリンクの故障が発生したとき，N の正常計算機網が連結であるかどうかを判定する問題を故障発生時の連結性判定問題 (以下，PC と呼ぶ) という．

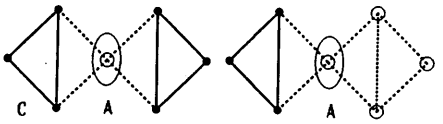
3 生成木構成分散アルゴリズム

本節では，計算機網のサイズを n，リンク数を e，アルゴリズムの始動計算機数を k とする．

ここでは，PC を解くのに必要になる計算機網の生成木を構成する新たな分散アルゴリズム ST を示し，$M(ST) = O(e + n \log k)$，$T(ST) = O(n \log k)$ となることを示す．従来にも，生成木を構成するアルゴリズムはいろいろ示されているが [1, 3, 4]，$M = O(e + n \log k)$ である従来のアルゴリズムは $T = O(e)$ であり [4]，本稿の結果は理想時間複雑度を改善したものになっている．なお，$T = O(n)$ となるアルゴリズムも知られているが [1]，このアルゴリズムは $M = O(e + n \log n)$ であり，k をパラメータにした場合もこれ以上良くならない．

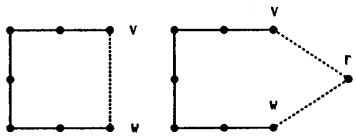
3.1 アルゴリズム ST の概略

（第1段階）　各始動計算機が隣接計算機にメッセージを送ることにより，すべての計算機を始動状態にするとともに，自分が最初に受信したメッセージの送信者を親とすることにより，各始動計算機を根とする部分木を構成する．（計算機網全体とし



(a) N-A　　(b) N'

図 1: 定理 2 の証明



(a) N=(P,L)　　(b) N'=(P',L')

図 2: 定理 3 の証明

ては始動計算機が根である木を連結成分とする生成林が構成される.)

(第 2 段階)　第 1 段階の結果により得られた各生成部分木をレベル 0 のフラグメント [3] とし, 文献 [3] の手法を用いて計算機網全体の生成木を構成する. このようにすると, フラグメントの最大レベルは高々 $\log k$ になる.

3.2 アルゴリズム ST の正当性及び効率

定理 1 *ST* は, $M(ST) = O(e + n \log k)$ かつ $T(ST) = O(n \log k)$ で有限時間内に正しく計算機網の生成木を構成する.

略証 ST の正当性は文献 [3] と同様にして示される. 各複雑度については ST の第 1 段階における通信量及び理想実行時間がそれぞれ高々 $2e + n$ 及び $3n$ であり, 第 2 段階は最終フラグメントのレベルが高々 $\log k$ となるので, 文献 [3] と同様にそれぞれ $O(e + n \log k)$ 及び $O(n \log k)$ を示すことができる. □

4 故障発生時の連結性判定問題

次の定理より計算機の故障を許した場合には PC を解く分散アルゴリズムは存在しない.

定理 2 任意のモデルの計算機網に対して, 計算機の故障を許した場合には *PC* を解く分散アルゴリズムは存在しない.

証明 $N = (P, L)$ を分離集合 A をもつ計算機網とする.
次のような 2 つの状況を考える (図 1).

(1) A 内の計算機がすべて故障した場合.
正常計算機網 $N - A$ は非連結となる. このとき, 連結成分の 1 つを $C = (P', L')$ とする.

(2) A 及び $P - P'$ 内のすべての計算機が故障した場合.
正常計算機網 $N'(= N - (A \cup (P - P')))$ は連結である.

(1)(2) の状況では, 明らかに $N' = C$ である. また, 故障の検知は, それに隣接する計算機のみが可能なので, N', C では A に含まれる計算機に隣接する計算機がそれらの故障を検知するのみである. 即ち, (1) の状況における C 内の計算機が持つ情報と (2) の状況における N' 内の計算機が持つ情報はそれぞれまったく同じである. 従って, 任意のモデルにおける計算機網において連結性判定問題を解く任意の分散アルゴリズムを $\mathcal{A}$ とすると, それを (1) の状況で稼働させた場合と (2) の状況で稼働させた場合ではまったく同じ動きをする場合がある. その場合, アルゴリズム終了時に各計算機が持つ結果が同じとなるので, (1)(2) のどちらか一方の状況に対しては間違った判定する. 故に, 任意のモデルの計算機網に対して, 計算機の故障を許した場合には PC を解く分散アルゴリズムは存在しない. □

以下では, 故障をリンクのみに限定し基本情報モデル, 隣接計算機既知モデル, サイズ既知モデルのそれぞれに対する PC を解く分散アルゴリズムを考察する.

4.1 基本情報モデル

定理 3 基本情報モデルの計算機網において, リンク故障時の *PC* を解く分散アルゴリズムは存在しない.

証明 ある 2 つの計算機 v, w $(\in P)$ 間のリンク (v, w) $(\in L)$ を除去しても連結であるような任意の計算機網を $N = (P, L)$ とする. 但し, $\rho_N(v, b) = \rho_N(w, c) = (v, w)$ とする (b,c:$1 \leq b \leq deg(v)$,$1 \leq c \leq deg(w)$). ここで, $N' = (P', L')$ を次のように定義する.

$$P' = P \cup \{r\} \ (r \notin P)$$
$$L' = (L - \{(v, w)\}) + (\{(v, r), (w, r)\})$$

また, 各計算機 $u(u \in P')$, 各ポート $a(1 \leq a \leq deg(u))$ に対し, N' のポートとリンクの対応を次のように定める.

$$\rho_{N'}(u, a) = \begin{cases} (v, r) & (u = v, \ a = b \text{ のとき}) \\ (v, r) & (u = r, \ a = 1 \text{ のとき}) \\ (w, r) & (u = w, \ a = c \text{ のとき}) \\ (w, r) & (u = r, \ a = 2 \text{ のとき}) \\ \rho_N(u, a) & (\text{その他の場合}) \end{cases}$$

基本情報モデルの場合, 各計算機の計算機網情報は識別子と次数だけなので, r 以外の各計算機の N' での計算機網情報は, N の場合と同じである. ここで, N においてリンク (v, w) だけが故障する場合と N' においてリンク (v, r), (w, r) だけが故障する 2 つの状況を考える (図 2) と, 定理 2 と同様にして基本情報モデルの計算機網においてリンク故障時の PC を解く分散アルゴリズムは存在しないことが示される. □

4.2 隣接計算機既知モデル

ここでは, 隣接計算機既知モデルの計算機網上でリンク故障時の PC を解くアルゴリズム CN(the algorithm for Connectivity of networks in Neighbors-known model) を示す. 次の補題で, 連結性を判定するのに利用する性質を示す.

補題 1 計算機網におけるすべての故障リンクを $(u_1, v_1), \cdots, (u_f, v_f)$ とすると, 正常計算機網が連結である必要十分条件は, $u_1, \cdots, u_f, v_1, \cdots, v_f$ が同じ連結成分に存在することである. □

4.2.1 アルゴリズム CN の概略

(第 1 段階)　故障を検知した計算機が始動計算機として 3 節のアルゴリズム ST で, 正常計算機網 N-F の各連結成分 C ごとに, 生成木 T_C を構成する. (但し, 自分のすべてのポートで故障を検知した計算機は ST を実行せずに結果を "非連結" として終了する.)

(第 2 段階)　C 内で故障を検知した各計算機の識別子の集合を D_C とする. 故障を検知した各計算機は, T_C を介して自分の識別子を親に送り, T_C の根である計算機に D_C の各要素

を集める．根である計算機は D_C の各要素をすべての計算機に放送する．D_C の要素を受け取った各計算機は，次のような条件で連結であるか（"連結"）または非連結であるか（"非連結"）を T_C における親に送る．

(1) 自分に接続する故障リンクの反対側の計算機の識別子がすべて存在し，自分のすべての子から"連結"を受け取るならば"連結"を送る．

(2) その他の場合は"非連結"を送る．

最終的に，根である計算機が"連結"を得たならば"連結"をすべての計算機に放送し，"非連結"を得たならば"非連結"を放送する．表 1に CN の第 2 段階の詳細を示す．

4.2.2 CN で利用するメッセージ

(1) ≪COL(nid)≫: 計算機網内で，故障を検知した計算機を子から親へ伝えるメッセージである．nid は故障を検知した計算機の識別子あるいは $null$(≪COL≫ による識別子の送信終了を示す) を表す．

(2) ≪REP(nid)≫: ≪COL≫ と同様だが，親から子へのメッセージである．

(3) ≪LO(con)≫: 各計算機が判定した局所的な連結性を子から親に伝えるメッセージである．con はこのメッセージを送信する計算機が判定した (局所的な) 連結性を表す．

(4) ≪GL(con)≫: 根が判定した大域的な連結性を親から子へ伝えるメッセージである．con は根が判定した (大域的な) 連結性を表す．

4.2.3 CN で利用する変数

(1) fat: 生成木における親につながるポート番号が格納される．

(2) Son: 生成木における子につながるポート番号の集合が格納される．

(3) N_Son: 生成木における子の数が格納される．

(4) 連結性: 計算機網の連結性を表す変数であり，アルゴリズム終了時には結果が格納される (連結，非連結)．初期値は連結とする．

(5) F_Set: 故障リンクによって隣接していた計算機のうち，同じ連結成分に属していることがまだ確認されていない計算機の識別子の集合を表す．初期値は故障を検知したポートの反対側の計算機の識別子の集合とする．

(6) D_Set: 同一連結成分内の計算機のうち故障を検知したと確認された計算機の識別子の集合を表す．初期値は空集合とする．

(7) N_Mes: ≪COL($null$)≫ および ≪LO≫ を受信した回数を評価するための変数．初期値は 0 とする．

4.2.4 CN の正当性

任意の計算機網 N においてリンクに故障が発生したとき，N の正常計算機網 N' の任意の連結成分を C とする．C 内の計算機数が 1 台のときは明らかに CN はその計算機においてリンク故障時の PC を正しく解き，有限時間で停止するので，以下では C の計算機数が 2 台以上のときについて考える．

補題 2 C 内の計算機数が 2 台以上のとき，C 内のすべての計算機は CN の実行を開始し，C における生成木が構成される．□

以下で記述される親，子，先祖，子孫，根，葉という表現は連結成分 C の生成木における親，子，先祖，子孫，根，葉を表すとする．

表 1: CN の第 2 段階

初期化
$F\_Set \leftarrow \{x | x$ は故障を検知したポートの反対側の計算機の識別子 $\}$
連結性 $\leftarrow$ 連結, $D\_Set \leftarrow \phi$, $N\_Mes \leftarrow 0$

根に対するアルゴリズム

(1) ST 終了時:
$F\_Set \neq \phi \Rightarrow D\_Set \leftarrow \{id\}$

(2) ≪COL(nid)≫ 受信:
$nid = null \Rightarrow$
$N\_Mes \leftarrow N\_Mes + 1$,
$N\_Mes = N\_Son \Rightarrow$
$F\_Set \neq \phi \Rightarrow$ 連結性 $\leftarrow$ 非連結,
$F\_Set = \phi \Rightarrow$ 連結性 $\leftarrow$ 連結,
各 $nid \in D\_Set$ に対して, send(≪REP(nid)≫,Son),
send(≪REP($null$)≫,Son)
$nid \neq null \Rightarrow (F\_Set \leftarrow F\_Set - \{nid\}, D\_Set \leftarrow D\_Set + \{nid\})$

(3) ≪LO(con)≫ 受信:
$N\_Mes \leftarrow N\_Mes - 1$, $con =$ 非連結 $\Rightarrow$ 連結性 $\leftarrow$ 非連結,
$N\_Mes = 0 \Rightarrow$(send(≪GL(連結性)≫,Son), 終了)

葉に対するアルゴリズム

(1) ST 終了時:
$F_Set \neq \phi \Rightarrow$send(≪COL($id$)≫,$fat$), send(≪COL($null$)≫,$fat$)

(2) ≪REP(nid)≫ 受信:
$nid = null \Rightarrow$
$F\_Set \neq \phi \Rightarrow$ 連結性 $\leftarrow$ 非連結,
$F\_Set = \phi \Rightarrow$ 連結性 $\leftarrow$ 連結,
send(≪LO(連結性)≫,fat)
$nid \neq null \Rightarrow F\_Set \leftarrow F\_Set - \{nid\}$

(3) ≪GL(con)≫ 受信:
連結性 $\leftarrow$ con, 終了

その他に対するアルゴリズム

(1) ST 終了時:
$F_Set \neq \phi \Rightarrow$send(≪COL($id$)≫,$fat$)

(2) ≪COL(nid)≫ 受信:
$nid = null \Rightarrow$
$N\_Mes \leftarrow N\_Mes + 1$,
$N_Mes = N_Son \Rightarrow$send(≪COL(null)≫,$fat$))
$nid \neq null \Rightarrow (F\_Set \leftarrow F\_Set - \{nid\}$, send(≪COL($nid$)≫,$fat$))

(3) ≪REP(nid)≫ 受信:
$nid = null \Rightarrow$
$F\_Set \neq \phi \Rightarrow$ 連結性 $\leftarrow$ 非連結,
$F\_Set = \phi \Rightarrow$ 連結性 $\leftarrow$ 連結,
send(≪REP($null$)≫,Son)
$nid \neq null \Rightarrow (F\_Set \leftarrow F\_Set - \{nid\}$, send(≪REP($nid$)≫,$Son$))

(4) ≪LO(con)≫ 受信:
$N\_Mes \leftarrow N\_Mes - 1$, $con =$ 非連結 $\Rightarrow$ 連結性 $\leftarrow$ 非連結,
$N_Mes = 0 \Rightarrow$send(≪LO(連結性)≫,fat)

(5) ≪GL(con)≫ 受信:
連結性 $\leftarrow$ con, send(≪GL(con)≫,Son), 終了

(注 1)send(≪M≫,X) の X が集合の場合は，X の各要素 x に対して send(≪M≫,x) を実行することを表す．
(注 2) 初期化以外のアルゴリズムでは，各動作の後に自分が終了していなければ，"receive" を実行する．

補題 3 故障を検知した計算機の識別子は，C 内のすべての計算機によって受信される．

証明 故障を検知した根以外の計算機は，ST の実行後に自分の識別子 id を $\ll$COL(id)$\gg$ により親に送信する．根でも葉でもない計算機が $\ll$COL(nid)$\gg$ を受信したとき，そのメッセージは親に送信される．また，葉は ST の実行後に (故障を検知した場合は $\ll$COL$\gg$ で自分の識別子を送信した後に)$\ll$COL($null$)$\gg$ を親に送信する．根でも葉でもない計算機が $\ll$COL($null$)$\gg$ をすべての子から受信したとき，$\ll$COL($null$)$\gg$ を親に送信する．従って，仮定 4 を考慮すれば必ず根である計算機に $\ll$COL(id)$\gg$ が受信される．同様に $\ll$COL($null$)$\gg$ も必ず受信される．CN より，根は受信したこれらの識別子 (自分が故障を検知した計算機であるならば，自分の識別子も含む) を D_Set に記憶して，すべての子から $\ll$COL($null$)$\gg$ を受信したとき，D_Set 内の各要素をすべての子に $\ll$REP$\gg$ で送信する．また，根でも葉でもない計算機は $\ll$REP(nid)$\gg$ を受信したとき，そのメッセージをすべての子に送信する．従って，仮定 4 より補題が示される．□

補題 4 根以外の計算機が $\ll REP(null)\gg$ を受信したとき，あるいは根がすべての子から $\ll COL(null)\gg$ を受信したとき，自分に接続する故障リンクの反対側の計算機がすべて C 内にあるかどうかを判定し，すべてが C 内にあれば連結性 = 連結，そうでなければ連結性 = 非連結となる．

証明 補題の状況を満たすとき，CN より各計算機は C 内の故障を検知したすべての計算機の識別子をすでに受信している (補題 3)．各計算機は故障を検知した計算機の識別子を受信したとき，その識別子が F_Set 内にあればそれを F_Set から削除する．よって，補題の状況のとき各計算機においては自分に接続する故障リンクの反対側の計算機がすべて C 内にあれば F_Set は空となり，そうでなければ空ではない．CN では F_Set が空でないならば，連結性を非連結にする．従って，補題が成り立つ．□

補題 5 C の計算機数が 2 台以上のとき，C のすべての計算機は N' の連結性について正しい判定結果を持ってアルゴリズムを停止する．

証明 補題 4より各計算機は自分の局所的な連結性を正しく判定する．CN によれば，各計算機は子から $\ll$LO(con)$\gg$ を受信したとき，con = 非連結ならば連結性を非連結にし，すべての子から $\ll$LO$\gg$ を受信したとき親に $\ll$LO(連結性)$\gg$ を送信する．よって，根がすべての子から $\ll$LO$\gg$ を受信したときの連結性は補題 1から正常計算機網 N' の連結性に等しいことが導かれる．根は連結性を $\ll$GL$\gg$ ですべての子に送信し終了する．根以外の計算機は $\ll$GL(con)$\gg$ 受信すると，連結性を con にしてすべての子に $\ll$GL(con)$\gg$ を送信し終了する．従って，補題が成り立つ．□

以上から，次の定理が成り立つ．

定理 4 任意の隣接計算機既知モデルの計算機網において，アルゴリズム CN は，リンク故障時の PC を正しく解き，有限時間で停止する．□

4.2.5 CN の効率

効率の評価を行うためのパラメータとして，計算機網のサイズ，正常リンク数，故障リンク数をそれぞれ n，e_f，f で表す．また，以下では任意の連結成分を C と表し，C 内の計算機数を p，故障を検知した計算機数を s とする．

定理 5 $M(CN) = O(e_f + \min(fn, n^2))$

証明 CN における始動計算機は故障を検知した計算機であり，高々 $\min(2f, n)$ 台である．CN では，まず必要な設定した後，前節の生成木構成アルゴリズム ST をサブルーチンとして実行する．ST によって必要とされるメッセージ数は $O(e_f + n\log(\min(2f, n)))$ である．以下，それ以外のメッセージ数を評価する．

N' の任意の連結成分 C について考える．C_k 内の各計算機が送信するメッセージは $\ll$COL$\gg$, $\ll$REP$\gg$, $\ll$LO$\gg$, $\ll$GL$\gg$ の 4 種類である．CN より $\ll$COL($null$)$\gg$ 及び $\ll$LO$\gg$ は，根以外の計算機によってちょうど 1 回ずつ送信される．また，$\ll$REP($null$)$\gg$ 及び $\ll$GL$\gg$ は，葉以外の計算機によってすべての子にちょうど 1 つずつ送信される．更に，故障を検知した各計算機の識別子が $\ll$COL(nid)$\gg$ により根に送られ，そこからすべての計算機に $\ll$REP(nid)$\gg$ によって送信される．従って，高々 $4(p-1) + s(p-1)$ となる．これをすべての連結成分に対して和をとると，

$$\begin{aligned} & \sum(4(p-1) + s(p-1)) \\ \leq\ & \sum(4(p-1) + s(n-1)) \qquad (\because p \leq n) \\ \leq\ & \sum(4p + s(n-1)) \\ \leq\ & 4n + \min(2f, n)(n-1) \\ & (\because \textstyle\sum p = n, \sum s \leq \min(2f, n)) \end{aligned}$$

となる．従って，メッセージ数は $O(\min(fn, n^2))$ である．以上より，定理が成り立つ．□

計算機網 N の正常計算機網 N' の任意の連結成分 C について，アルゴリズム CN の C での理想実行時間 IT_C を考える．

$p = 1$ の場合，$IT_C = 0$ であるので，以下では $p \geq 2$ の場合を考える．C の生成木の深さを d とする．また，C における ST の終了時刻を H^C_{ST}，CN の終了時刻を H^C_{CN} とする．このとき，次の補題が成り立つ．

補題 6 $H^C_{ST} = O(p\log s)$ である．□

補題 7 $H^C_{CN} \leq H^C_{ST} + 4d + 2s$

証明 付録参照．□

補題 6,7より次の補題が示される．

補題 8 $IT_C = O(p\log s)$ である．

証明 補題 6,7と $d \leq p$，$s \leq p$ を考慮すれば，$H^C_{CN} = O(p\log s)$ となり，補題が示される．□

定理 6 $T(CN) = O(n\log f)$

証明 補題 8により，$IT_C = O(p\log s)$ である．N' 全体の理想実行時間はこれらの最大のものであるので，$p \leq n$，$s \leq \min(2f, n) \leq 2f$ を考慮すれば，$T(CN) = O(n\log f)$ である．□

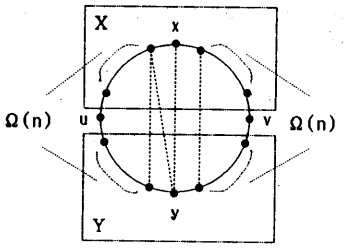
4.2.6 通信複雑度および理想時間複雑度の下界

この問題に対する通信複雑度の下界を示すために，図 3に対して次の補題が成立することを示す．

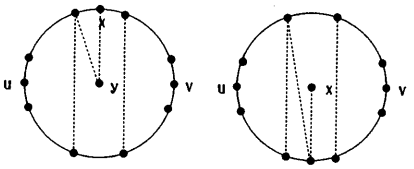
補題 9 図 3(a) において，ある計算機 x，$y((x,y) \in F_L)$ が存在して，それらの計算機を識別するのに必要な情報 (必ずしも識別子であるとは限らない) がいずれも，u と v のどちらの計算機にも受信されないならば，どの計算機も連結性を判定できない．

証明 ある計算機 x，$y((x,y) \in F_L)$ が存在して，それらの計算機を識別するのに必要な情報 (必ずしも識別子であるとは限らない) がいずれも，u と v のどちらの計算機にも受信されないとする．このとき，図 3 (a) における Y に属する計算機は，計算機 x の接続関係やそれに対する故障の状況がわからない．同様に，X に属する計算機は，計算機 y の接続関係やそれに対する故障の状況がわからない．よって，x 及び y がそれぞれ図 3(b) (c) の場合と区別できない．従って，補題が成り立つ．□

定理 7 隣接計算機既知モデルの計算機網におけるリンク故障時の PC を解く任意のアルゴリズム $\mathcal{A}$ に対して，$M(\mathcal{A}) = \Omega(\max(n, \min(fn\frac{\log f}{\log n}, n^2)))$，$T(\mathcal{A}) = \Omega(n)$ となる計算機網が存在する．



(a) N=(P,L)



(b) X に対するN　　(c) Y に対するN

図 3: 補題 9 と定理 7 の証明

証明 $T(\mathcal{A}) = \Omega(n)$ となる計算機網が存在することは明らかである.

図 3(a) の計算機網においては, 補題 9より故障を検知した任意の計算機 x, $y((x,y) \in F_L)$ に対して, それらの計算機を他と区別するのに必要な情報のどちらかが, u または v のどちらかに受信されなければならない. このとき, 故障を検知する計算機数が, ある正数 c に対して $f \leq cn$ が成り立つならば X 内と Y 内にそれぞれ f 個ずつ, $f \geq cn$ が成り立つならば X 内と Y 内にそれぞれ cn 個ずつとなるような計算機網が構成できる. この場合, これらの情報を区別するには $\min(\log 2f, \log 2cn)$ ビットが必要となり, このような情報が $\min(f, cn)$ 個必要なので, 全体で $\Omega(\min(f \log f, n \log n))$ ビットを $\Omega(n)$ 送る必要がある. ここで, メッセージ長は $O(\log n)$ ビットと仮定しているので, メッセージ数は $\Omega(\min(fn\frac{\log f}{\log n}, n^2))$ となる.

また, 明らかに $\Omega(n)$ メッセージを必要とする計算機網及び故障が存在する. 従って, $M(\mathcal{A}) = \Omega(\max(n, \min(fn\frac{\log f}{\log n}, n^2)))$ となる. □

4.3 サイズ既知モデル

ここでは, サイズ既知モデルの計算機網上でのリンク故障時の PC を解く分散アルゴリズム CS(the algorithm for Connectivity of networks in Size-known model) の概略のみを示す. 前節と同様に連結性を判定するのに利用する性質を示す.

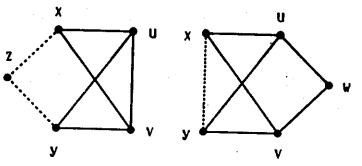
補題 10 サイズが n の計算機網において, リンクに故障が起きた場合, 正常計算機網が連結である必要十分条件は, 正常計算機網が n 個の計算機からなる連結成分を持つことである. □

4.3.1 アルゴリズム CS の概略

(第 1 段階)　CN の第 1 段階と同様.

(第 2 段階)　T_C の葉である計算機から根である計算機へと順に親の計算機に自分の子孫の数を送ることによって, C 内の計算機数 n_C を根に持たせて, 根がそれと計算機総数 n を比較することにより連結性を判定し, 結果を T_C を用いて放送する.

(1) $n_C < n \Rightarrow$ 連結



(a) N=(P,L)　　(b) N'=(P',L')

図 4: 定理 1 0 の証明

(2) $n_C = n \Rightarrow$ 非連結

4.3.2 CS の正当性および効率

定理 8 任意のサイズ既知モデルの計算機網において, CS はリンク故障時の PC を正しく解き, 有限時間で停止する.

略証 第 1 段階において, 各連結成分における生成木が有限時間内に構成されることは定理 1から明らかである. 第 2 段階では, この生成木を利用して, 連結成分内に属する計算機数を根に集め, 根がそれと計算機網のサイズを比較する. 根の判定の仕方と補題 10より, 根は正常計算機網の連結性を正しく判定する. 根はその結果を生成木をを利用して, 同一連結成分内のすべての計算機に伝える. よって, 定理が示される. □

定理 9 $M(CS) = O(e_f + n \log f)$, $T(CS) = O(n \log f)$

略証 N' の任意の連結成分 C_i に対して, 通信量および理想実行時間を評価する. C_i における計算機数, 正常リンク数, 故障リンク数をそれぞれ n_i,e_{f_i},f_i とする. このとき, C_i 内における始動計算機数は高々 $2f_i$ である. よって, 第 1 段階における通信量および理想時間量は, 定理 1よりそれぞれ $O(e_{f_i} + n_i \log f_i)$, $O(n_i \log f_i)$ である. 第 2 段階においては, 第 1 段階で構成した生成木を介した通信で, C_i 内の計算機数を根に集め, 根が連結性を判定して結果を C_i 内のすべての計算機に放送するので, 通信量および理想実行時間はそれぞれ $O(n_i)$, $O(n_i)$ である. 従って, CS 全体では $O(e_{f_i}+n_i \log f_i)$, $O(n_i \log f_i)$ となる.

これらを正常計算機網全体で考えると, 通信複雑度は $O(e_f + n \log f)$, 理想時間複雑度は $O(n \log f)$ となる. □

4.3.3 通信複雑度および理想時間複雑度の下界

定理 10 サイズ既知モデルの計算機網におけるリンク故障時の PC を解く任意のアルゴリズム $\mathcal{A}$ に対して, $M(\mathcal{A}) = \Omega(e_f)$, $T(\mathcal{A}) = \Omega(n)$ となる計算機網が存在する.

証明 $T(\mathcal{A}) = \Omega(n)$ となる計算機網が存在することは明らかである.

通信複雑度は, 以下のように考える. $G = (V, E)$ を互いに隣接していない計算機組を持つ連結な計算機網とし, それらを x, y とする. $N = (P, L)$ を次のように定義する.

$$P = V + \{z\}$$
$$L = E + \{(x,z),(y,z)\}$$

また, 故障リンクの集合を $F_L = \{(x,z),(y,z)\}$ とする.

ここで, PC に対する任意のアルゴリズム $\mathcal{A}$ が必要とするメッセージ数が $e_f - 1$ 以下であるとすると, メッセージが送られない正常リンクが存在する. 計算機網 N において, そのようなリンクを (u,v) とする (図 4 (a)). このとき, 次のような計算機網 N' 及び故障集合を考える (図 4 (b)).

$$P' = P - \{z\} + \{w\}$$

$$L' = L - \{(x,z),(y,z)\} + \{(x,y)\} - \{(u,v)\} + \{(u,w),(v,w)\}$$
$$F' = \{(x,y)\}$$

また，各計算機 $s(\in P')$ のポート対応を次のように定義する．各 a $(1 \le a \le deg(s))$ に対して，

$$\rho_{N'}(s,a) = \begin{cases} (u,w) & (s = u \text{ かつ } \rho_N(u,a) = (u,v)) \\ (v,w) & (s = v \text{ かつ } \rho_N(v.a) = (u,v)) \\ (u,w) & (s = w \text{ かつ } a = 1) \\ (v,w) & (s = w \text{ かつ } a = 2) \\ (x,y) & (s = x \text{ かつ } \rho_N(x,a) = (x,z)) \\ (x,y) & (s = y \text{ かつ } \rho_N(y,a) = (y,z)) \\ \rho_N(s,a) & (\text{その他}) \end{cases}$$

w が始動計算機でないならば，u または v からメッセージを受信しない限り u，v のどちらにもメッセージを送信しない．また，各計算機がアルゴリズム開始時に保持している計算機網情報は，自分の識別子と接続リンクに対応するポート番号と計算機網のサイズのみであるので，N の連結成分 C と N' で問題を解く場合，各対応する計算機が同じ動作をする場合がある．従って，C と N' における各計算機は同じ結果を出力する．しかし，N の正常計算機網は非連結であるが，N' の方は連結であるので，どちらかに対して，間違った結果を出力する．従って，メッセージ数は e_f 以上となり，$M(A) = \Omega(e_f)$ である．□

5 むすび

本稿では，故障が発生した計算機網における連結性を判定する分散アルゴリズムについて考察した．特に，故障をリンクのみに限定した場合には，隣接計算機既知モデル及びサイズ既知モデルに対して，PC を解く分散アルゴリズムを示し，それぞれの効率については理想時間複雑度はオーダー的に変わらないが，通信複雑度は各計算機がより大域的な情報をもつと考えられるサイズ既知モデルの方が小さくなるという結果を得た．残された問題としては，上界および下界の間のすき間を埋めることが考えられる．

参考文献

[1] B.Awerbuch, "Optimal distributed algorithms for minimun weight spanning tree, counting, leader election and related problems", *Proc.19th Symp.Th.Comp* 230–240, (1987).

[2] 朴, 増澤, 萩原, 都倉, "ネットワークの連結性関連問題を解く効率のよい分散アルゴリズム", 信学論 J72-D-I,5,343–356, (1989).

[3] R.G.Gallager, P.A.Humblet and P.M.Spira, "A distributed algorithm for minimum-weight spanninig trees", *ACM Trans.on Prog.Lang. and Syst.* 5,1,66–77,(1983).

[4] R.G.Gallager, "Finding a leader in a network with $O(E)+O(N \log N)$ messages", *Internal Memorandum* (1983).

[5] 萩原, 都倉, 増澤, "分散アルゴリズムの複雑度について", セル構造に基づく高度並列情報処理システムに関する総合的研究 104–113,(1988).

付録

補題 7の証明

補題 7を証明するために，以下に 3 つの補題を示す．

各計算機 u に対して，次のような時刻に関する記号を定義する．

(1) $t_{COL}(u)$: u がすべての子から ≪COL($null$)≫ を受信する時刻を表す．(u が葉である場合は u が ST を終了する時刻を表す．)

(2) $t_{REP}(u)$: u が親から ≪REP($null$)≫ を受信する時刻を表す．

(3) $t_{LO}(u)$: u がすべての子から ≪LO≫ を受信する時刻を表す．(u が葉である場合は $t_{LO}(u) = t_{REP}(u)$ とする．)

同様に，生成木に関する記号を定義する．

(1) $L(u)$: u の生成木における深さを表す．

(2) $S(u)$: u の子孫のうちで故障を検知した計算機の数を表す．

以下では r を根とし，深さが d である計算機の 1 つを v とする．

補題 11 任意の計算機 u について，

$$t_{COL}(u) \le H^C_{ST} + d - L(u) + S(u) \qquad (1)$$

が成り立つ．

証明 (a)u が葉の場合は明らか．
(b)u のすべての子で式 (1) が成立すると仮定し，t_{COL} の値を最大とする計算機を w とする．CN より，w は時刻 $t_{COL}(w) - S(w)$ までには u へ少なくとも 1 つの ≪COL(nid)≫ を送信する．また，時刻 $t_{COL}(w)$ までには $S(w)$ 個目の ≪COL(nid)≫ を送信する．よって，u は $t_{COL}(w)+1$ までに少なくとも $S(w)$ 個の ≪COL(nid)≫ を送信し，w の選び方よりこれ以降 u が ≪COL(nid)≫ を受信することはない．従って，u は $t_{COL}(w)+1+S(u)-S(w)$ までに ≪COL(nid)≫ 送信し終え，u は $H^C_{ST}+d-L(w)+1+S(u)$ までにすべての子から ≪COL($null$)≫ を受信する．故に，$L(u) = L(w) - 1$ を考慮すれば u についても式 (1) が成立する．

(a)(b) よりすべての計算機に対して，式 (1) が成立する．□

補題 12 $t_{REP}(v) \le t_{COL}(r) + d + s$

証明 時刻 $t_{COL}(r)$ においては，各計算機の受信待行列は空である．また，r はこの時刻に D_Set 内の識別子をすべての子に送信しはじめ，時刻 $t_{COL}(r)+s$ において，≪REP($null$)≫ を送信する．v の深さは d であるので，v は時刻 $t_{COL}(r)+d$ 以降 ≪REP≫ を受信し続け，時刻 $t_{COL}(r)+d+s$ に ≪REP($null$)≫ を受信する．□

補題 13 任意の計算機 u に対して，

$$t_{LO}(u) = t_{REP}(v) + d - L(u) \qquad (2)$$

が成り立つ．

証明 時刻 $t_{REP}(v)$ までには明らかにすべての計算機が ≪REP($null$)≫ を受信する．従って，すべての計算機 u に対して次の式が成り立つことを式 (1) と同様に示すことができる．□

補題 11,12,13より，次の 3 つの式が成り立つ．

$$t_{COL}(r) \le H^C_{ST} + d + s$$
$$t_{REP}(v) \le t_{COL}(r) + d + s$$
$$t_{LO}(r) = t_{REP}(v) + d$$

また，明らかに $H^C_{CN} = t_{LO}(r) + d$ が成り立つ．従って，補題 7が示される．□