A-013

局所名前付け自己安定アルゴリズム
A Self-Stabilizing Algorithm for Local Naming Problem

川井 裕之 † 角川 裕次 †
Hiroyuki Kawai Hirotsugu Kakugawa

1. はじめに

分散システムとは、地理的に離れた地点にある複数のプロセスがネットワークで接続され、それらが強調してある共通の目的を達成するものである. 分散システムを構築することにより、一部のプロセスや通信路に故障が発生しても残りの部分でシステムが動作を行うことができるようになる. 分散システムを実現するためのアルゴリズムを分散アルゴリズムという.

通常の分散アルゴリズムでは各プロセスの状態やネットワーク中に存在するメッセージの存在をあらかじめ決めておくが、分散アルゴリズムの一種である自己安定アルゴリズムは各プロセスの状態やメッセージの初期状態を仮定しない. このことによりプロセスや通信路に故障が起こったり、ネットワークの形状が変化する動的ネットワークにおいても問題を解くことができる.

1.1 局所名前付け問題

分散アルゴリズムの議論では同一のプログラムを実行するとしながらも実際には各プロセスが相異なる識別子を持ち、その識別子をパラメータとするプログラムを実行することを許すことが多い. これに対し、匿名ネットワークと呼ばれるすべてのプロセスが同一の状態機械であるとするネットワークでは識別子が与えられていない. 名前付け問題とはこうしたネットワークにおいて各プロセスに固有の識別子を与える問題である.

しかし大規模分散システムでは、大域的に一意的な識別子の割り当ては困難である. そこで通常は局所的な一意性を保証するだけにし、大域的な一意性が必要となれば、その時点で名前付けアルゴリズムを用いて、大域的に固有の識別子を構成することが考えられる.

名前付け問題と関連する研究としてカラーリング問題があげられる. カラーリング問題とはネットワークに存在する全ての隣接する 2 つのプロセスをみても、同じ色であるものが存在しないと言うものである. 局所名前付け問題との違いは、カラーリング問題はあるプロセスを中心としてみた時の隣接プロセス間に同じ色 (識別子) を持つプロセスの存在を許すのに対し、局所名前付け問題は隣接プロセス間の識別子の重複を許さないところである.

本稿ではあるプロセスを中心として、半径距離 2 以内に存在するプロセスに一意な識別子を与える自己安定アルゴリズムを提案し、その正当性の証明および、アルゴリズムを実行するにあたり必要となる識別子の範囲を求めた.

2. 定義

定義 **1.** (自己安定) Γ をすべての状況の集合とし、Λ を解状況の集合とする. アルゴリズムが次の *2* つの条件を満たす場合に限り、アルゴリズムは $\Lambda \subseteq \Gamma$ に対し自己安定である.

- 収束性
  - 任意の状況からはじめていつかは正当な状況 Λ のうち *1* つとなる.
- 閉包性
  - 状況 $\lambda \in \Lambda$ について、λ に続くあらゆる状況 Γ もまた Λ の状況である.

定義 **2.** (名前付け問題) 次の条件を満たす限り、計算状況 λ は名前付け問題の解状況である.

- ネットワークに存在するどのプロセスの識別子をとってみても同じ識別子が現れない.

定義 **3.** (局所名前付け問題) 次の条件を満たす限り、計算状況 λ は半径距離 k 以下での局所名前付け問題の解状況である.

- ネットワークに存在するどのプロセスをとってもみても半径距離 k 以内に同じ識別子を持つプロセスが存在せず、かつ自分と同じ識別子を持つプロセスが存在しない.

定義 **4.** (システムモデル) ネットワークに存在する各プロセスは状態機械である. 各プロセスは自分の局所状態を示す局所変数を持っているものとする. そして各プロセスは自分の局所状態と隣接プロセスの状態から次の状態を決定する. また全てのプロセスの状態集合が分散システムの状況であるとする. ネットワークの状態は *[1]* による半径距離 *1* 以下に存在するプロセスに一意な識別子を与える自己安定アルゴリズムを適用した状態であるとする.

通信モデルとしては、状態通信モデルを採用する. この通信モデルでは、各プロセスは遅延なく隣接プロセスのもつ局所状態を読んでくることが出来る. 各プロセスは隣接プロセスの状態を読むことは出来るが、書き換えることはできない. 各プロセスが書き換えることが出来るのは、自身の持つ局所状態のみである.

また本アルゴリズムが対象とするネットワークの形状はツリーであるとする.

3. アルゴリズム

アルゴリズムの概要は以下の通りである.

まず各プロセスは自分の隣接プロセスの識別子を保持する. その保持した隣接プロセスの識別子を参照することにより、距離 2 離れたプロセスの状態を知ることができ、もし距離 2 以内に同じ識別子を持つプロセスを検出

†広島大学大学院 工学研究科 情報工学専攻

した場合は、$flag_2$ という変数にその重複している識別子を代入する．$flag_2$ に代入された重複している識別子は隣接プロセスの $flag$ にコピーされ重複している識別子を持つプロセスに伝えられることになる．そして自分の識別子が半径距離 2 以下で重複していることがわかったプロセスは自分の識別子を安全に変更できる範囲からランダムに変更する．

提案する自己安定アルゴリズムは以下の通りである．各プロセスはアルゴリズム中の各ステップを同期して実行するものとする．

Algorithm on the processor i

Constant

LID :半径距離 1 以下で一意な識別子

Local Variable

id :半径距離 2 以下で一意な識別子となる変数
$flag$:重複を伝える変数.$flag_2$ の複製
$flag_2$:重複を伝える変数
$N_{id}[i]$:プロセス i の隣接プロセスの LID と id の組 $[LID, id]$ の集合
$N_{id}2[i]$:プロセス i の隣接プロセスの N_{id} の集合に含まれる id の集合
r :変更する id の候補

Macro

$changeID$:自分の id を $flag[id, r]$ の r と $N_{id}2$ を参考に変更する

```
LOOP {
    /* Step 1 */
    if ( ( id.i = id.j ∨ id.i = id.l | ∃j,k ∈ N[i],
    ∃[LID.l ≠ LID.i, id.l] ∈ N_id[k] )
    ∧ ( id.i ≠ LID.i ) ∧ (id.i ≠ LID.i) ) then
            id.i := LID.i ;

    /* Step 2 */
    if ( id.j = id ∈ flag_2k ∧ id.l ≠ id ∈ flag.i |
    ∃j,k,l ∈ N[i] ) then
            flag.i := flag_2.k ;

    /* Step 3 */
    if ( id.j = p ∨ p = q | ∃j,k,l ∈ N[i], ∃p ∈ N_id[k],
    ∃q ∈ N_id[l] ∧ ( id ∈ flag_2.i ≠ r ∧ id ∈ flag.i
    ≠ id.n | ∃m,n ∈ N[i], ∃r ∈ N_id[m] ) ) then
            flag_2.i := [p, r] ;

    /* Step 4 */
    if ( id.i = flag.j | ∃j ∈ N[i] ) then
            id.i := changeID ;
}
```

以下のことを証明した．証明の詳細は紙面の都合上省略する．

定理 1. 各プロセスが取る id の範囲の最大値は $\frac{1}{4}(n+7)^2$ 以上である．

定理 2. アルゴリズムは、半径距離 2 以下での局所名前付け問題を解く自己安定アルゴリズムである．

証明の概要は以下の通りである．

ネットワークに存在するプロセスの数を n とする．まず重複している識別子を持つプロセスが自身の識別子を変更する際、変更後の識別子が再び他の識別子と重複する可能性はあるが、ネットワーク全体で重複している識別子の総数は増えることはない．また変更後に他の識別子と重複することがない場合、ネットワーク全体で重複している識別子の総数は減少する．これらのことより重複している識別子の総数は単調減少する．

また局所名前付け問題の解状態において、どのプロセスも実行できない．

以上のことより、定義 1. の自己安定の条件である収束性と閉包性を満たし、提案アルゴリズムが問題を解く自己安定アルゴリズムであることを証明した．

4. おわりに

本稿ではツリー状のネットワークにおいて半径距離 2 以下に存在するプロセスに一意な識別子を与える局所名前付け自己安定アルゴリズムを考案し、その正当性を証明した．

今後の課題としては局所名前付けを行う範囲の拡大、任意のグラフにおいての局所名前付けを行えるための改良などが挙げられる．

参考文献

[1] Maria Gradinariu, Colette Johnen. Self-stabilizing Neighborhood Unique Naming Under Unfair Scheduler. in *Proc. of the 7th International Euro-Par Conference on Parallel processing*, pp. 458-465. 2001.

[2] Maria Gradinariu and Sebastien Tixeuil. Self-stabilizing Vertex Coloring of Arbitrary Graphs. OPODIS, pp. 55-70. 2000.

[3] Edsger W.Dijkstra. Self-stabilizing Systems in Spite of Distributed Control, in *Communications of the ACM*, Vol. 17, No.11, pp. 643-644. 1974.

[4] Sukumar Ghosh, Mehmet Hakan Karaata. A Self-Stabilizing Algorithm for Coloring Planar Graphs. Distributed Computing, pages 7:55-59. 1993.