Proof-Labeling スキームに基づく故障封じ込め自己安定アルゴリズムの提案

中川遼† 大下福仁†† 角川裕次† 増澤利光†

† 大阪大学 大学院情報科学研究科

†† 奈良先端科学技術大学院大学 情報科学研究科

概要　自己安定アルゴリズムとは，任意のネットワーク状況から実行を開始しても，やがて解を求めて安定するアルゴリズムである．自己安定アルゴリズムでは，安定した状況において突発的な変化が起こっても，それを検知し自律的に安定した状態に収束することが可能である．また，故障封じ込めとは，安定した状況から少数のプロセスが故障した場合に，再び安定するまでに状態遷移するプロセスや必要とする時間を限定したものである．故障封じ込めは，自己安定アルゴリズムにおいて一般に発生しやすいと考えられている小規模な故障による故障状態からの実行において，故障の及ぶ範囲を制限しかつ素早く再安定することを目的としている．本稿では，自己安定アルゴリズムの設計において，故障の検知と安定化を分けて記述することのできる Proof-Labeling スキームを用いて，故障封じ込めの性質を付与した自己安定アルゴリズムの設計方法を提案する．

1　はじめに

分散システムは，複数のノードが相互に通信リンクで接続されたネットワークである．自己安定アルゴリズムとは，任意の初期状況から実行を始めても問題を解く（問題の要求を満たす状態に到達する）ことができる分散アルゴリズムである [1]．自己安定アルゴリズムはその性質から，ノードにおける変数の破壊などの一時的な故障（一時故障）によって問題の要求を満たす状況から任意の状況に陥っても，再び問題の要求を満たす状況に復帰して安定する．つまり任意の一時故障に対して耐故障性を持つ分散アルゴリズムであり，長期間分散システムを安定に保ち，一時故障に柔軟に対応する必要がある分散システムを動作させるのに適しており，現在盛んに研究されている分野の 1 つである．自己安定アルゴリズムは一般に，問題の要求を満たしていないことを検出する検知フェーズと，問題の要求を満たす状態に遷移する収束フェーズに分けることができる．Proof-Labeling スキームはこの考えを定式化したものであり [2]，自己安定アルゴリズムの設計を検知と収束の 2 つに分けて設計することが可能になり，自己安定アルゴリズムの設計を容易化することができる．

分散システムにおいて同時に多数のノードが故障することはまれであり，一般には少数のノードが一時故障を起こし，要求を満たす状況からわずかに変化した状況になることがほとんどである．従来の自己安定アルゴリズムではそのようなわずかな変化を考慮していないため，少数ノードの故障の影響がネットワーク全体に及ぶことがある．そこで，故障封じ込めと呼ばれる単一ノードの故障から定数時間で再安定することのできる性質を付与することは実際のシステムにおいて非常に重要である [3]．

様々な問題に対する自己安定アルゴリズムが設計されてきた．また，その自己安定アルゴリズムに故障封じ込めの性質を付与する変換手法も提案されている．それに加えて近年，自己安定アルゴリズムの設計方法として，一般に従来の設計よりも簡単と言われる Proof-Labeling スキームを用いた設計が注目を浴びている．本稿では Proof-Labeling スキームを用いて設計された自己安定アルゴリズムに，故障封じ込めの性質を付与したアルゴリズムを提案する．

2　諸定義

2.1　ネットワークモデル

本稿では，n 個のノードが通信リンクで接続されたネットワーク N を扱う．各ノードは共有レジスタを持ち，自身の共有レジスタに Read/Write を実行可能であり，隣接ノードの共有レジスタに対して Read が可能である．1 つのアトミックな動作において，各ノードは Read;Compute;Write を行うことができる．また，本稿では各ノードが少なくとも 2 つの隣接ノードを持つネットワークのみを対象とする．

2.2　分散システムの状況

故障封じ込め自己安定アルゴリズムにおいて，ネットワークは 3 つの状況に大別することが可能である．

1. Consistent
   全てのノードが目的の述語を満たしている状況．
2. 1-Faulty
   Consistent な状況から，ただ 1 つのノードの状態だけを変化させて得られる状況．状態が変化したノードを故障ノードと呼ぶ．アルゴリズムによっては，1-故障状況において，故障ノードを特定できない場合もあるが，本稿で構成する故障封じ込めアルゴリズムでは，1-故障状況で故障ノードを特定可能である．
3. Inconsistent
   Consistent でも 1-Faulty でもない状況．

2.3 自己安定性

任意の状況から実行を開始しても，いずれは正当な状況（Consistent 状況）に到達して安定する性質を自己安定性と呼び，自己安定性を持つアルゴリズムのことを自己安定アルゴリズムと呼ぶ．自己安定アルゴリズムは突発的な変化により Consistent 状況でなくなるとそれを検知し，自律的に収束する．

2.4 故障封じ込め

自己安定アルゴリズムにおいて，任意の 1-Faulty 状態から O(1) 時間で Consistent 状態に到達するならば，その自己安定アルゴリズムは故障封じ込めであると呼ぶ．

2.5 Proof-Labeling スキーム

Proof-Labeling スキーム（以下，P-L）は，自己安定アルゴリズムの設計手法の 1 つであり，ラベリングアルゴリズムとデコーダーアルゴリズムのペアに分けて自己安定アルゴリズムを構成する．以下ではオリジナルのものを紹介する．以下の P-L には故障封じ込めの性質は備わっていない．

2.5.1 ラベリングアルゴリズム

各ノードに次の条件を満たすラベルを割り当てる．

1. ネットワーク状況が与えられた述語を満たすならば，各ノードで実行するデコーダーアルゴリズムが Consistent 状況を出力するようなラベルを出力する．
2. そうでなければ，少なくとも 1 つのノードでのデコーダーアルゴリズムが Inconsistent を出力するようなラベルを出力する

2.5.2 デコーダーアルゴリズム

各ノードは自身に割り当てられたラベルと隣接ノードのラベルからネットワークの状況が Consistent か Inconsistent かを出力する．

3 提案アルゴリズムの概要

本章では P-L を拡張して，故障封じ込め自己安定アルゴリズムを構成するための枠組みを新たに提案する．提案アルゴリズムでは，従来の P-L と同様にラベリングアルゴリズムとデコーダーアルゴリズムのペアを使用して分散システムの状況が Consistent でないことを検知するが，Consistent でないときは 1-Faulty か Inconsistent かも検知可能とすることで，1 つのノードの故障からの修復を可能とする．これを実現するために，本手法ではラベリングアルゴリズムにおいて，故障封じ込めでない自己安定アルゴリズムで自身に割り当てられるラベルのコピーを隣接するノードの内の 2 つのノードにもとして割り当てたものを，新たなラベルとする．

3.1 ラベリングアルゴリズム

従来の P-L で用いられていたラベル割り当ての条件 1,2 に加えて以下の条件を追加する．

- ネットワークが 1-Faulty 状況ならば，故障しているノードで実行されるデコーダーアルゴリズムは 1-Faulty を出力

ネットワークが 1-Faulty 状況であることを判断する 1 つの方法として，本手法では隣接ノードの持つ Consistent 状況を出力するラベルのコピーと自身の現在の状態を比較して故障を検知し，かつ隣接ノードが全て Consistent 状況であるとき 1-Faulty と認識する．

3.2 デコーダーアルゴリズム

各ノードは割り当てられたラベルと隣接ノードのラベルからネットワークの状況が Consistent なのか Inconsistent なのか 1-Faulty なのかを出力する．

3.3 修復・収束アルゴリズム

各ノードはデコーダーアルゴリズムでの自身の出力により，以下の動作を行う．

1. 出力が Consistent のとき
   自身は故障していないので何もしない．
2. 出力が 1-Faulty のとき
   隣接ノードが持つ自身のラベルのコピーを元に，自身の状態を修復する．
3. 出力が Inconsistent かつ隣接ノードの出力に 1-Faulty がないとき
   元の自己安定アルゴリズムに従った動作を行う．

4 おわりに

本稿では，P-L に基づいて設計された自己安定アルゴリズムに故障封じ込めの性質を付与した自己安定アルゴリズムの P-L に基づく設計方法を提案した．提案手法により，様々な問題に対する故障封じ込め自己安定アルゴリズムを系統的に実現することが可能とする．

本提案アルゴリズムでは，各ノードが隣接 2 ノードにラベルのコピーを置く性質上，最悪時には 1 つのノードで最大次数+1 のラベルを保持しなければならない．したがって最悪時のレジスタ空間複雑度が大きくなる．

この問題を解決するために，ネットワークトポロジに何らかの制限をかけることにより最悪時のレジスタ空間複雑度を小さくすることが今後の課題である．

参考文献

[1] Dolev, S.: Self-Stabilization. MIT Press (2000)

[2] Korman, A., Kutten, S., Peleg, D.: Proof labeling schemes. Distributed Computing 22(4), 215-233 (2010)

[3] Ghosh, S., Gupta, A., Pemmaraju, S.V.: Fault-containing network protocols. In: Proceedings of the 1997 ACM Symposium on Applied Computing, pp. 431-437. ACM (1997).