4C−3

フォールトトレラント分散システムのための自己安定アルゴリズム

江上 研介
Kensuke EGAMI
東京大学
University of Tokyo

濱田 喬
Takashi HAMADA
学術情報センター
National Center for Science Information Systems

1 はじめに

自己安定アルゴリズムとは、任意の初期状態から始めても、有限時間内に解を求める分散アルゴリズムである。これによって、フォールトトレラントな分散システムを実現することが可能となる。これまでにDデーモンの下でサイズが素数の均一なリングネットワーク上での相互排除問題が解かれている。本稿では、Dデーモンよりも制約の弱いスケジューラであるR/Wデーモンでサイズが素数の均一なリングネットワーク上での相互排除問題を考察し、Dデーモンと R/Wデーモンの間の能力差を示す。自己安定アルゴリズム (self-stabilizing algorithm) は任意のネットワーク状況から開始しても問題を解くことのできる分散アルゴリズムである。自己安定アルゴリズムには、次の二つの重要な長所がある。

- システムの初期化を行う必要がない。
- 一過性の障害から有限時間内に回復ができる。

2 計算モデル

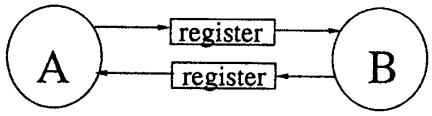
プロセス間の通信にはレジスタ通信モデルを採用する。レジスタ通信モデルとは、各リンクの各通信方向に対して、一つのレジスタが存在し、そこに隣接プロセッサへのメッセージを書込むことによって通信を行うモデルである。これによって各リンクごとに異なった情報を伝達する場合をモデル化できる。また、各プロセッサの原子動作の違いと、複数のプロセッサの同時動作を許すかどうかの違いによっていくつかのモデルがある。レジスタ通信モデルを例に説明すると、原子動作を読み書きの一連の動作として任意の並列実行を許すスケジュールが許されるモデルをDデーモンと呼ぶ。各時点で一つのプロセスしか実行されないモデルをCデーモンと呼ぶ。また、各時点で一つのプロセスしか実行されないが、原子動作が読みと書きで分かれているモデルをR/Wデーモンと呼ぶ。

3 スケジュールの包含関係

各デーモンによるスケジューリングは、直観的に次の包含関係がある。

$$S_{R/W} \supset S_D \supset S_C$$

つまり各デーモンのスケジュールは制約の弱いデーモンのスケジュールの特別な場合となっている。よって R/Wデーモンで正しく動作すれば、Dデーモンでもアルゴリズムの変更なしで動作する。



Dデーモン (A,B) ( ).....
R/Wデーモン (A-read) (B-read) (A-write) (B-write) .....

図 1:

自己安定アルゴリズムがあるデーモンのもとで正しく動作するという意味は、そのデーモンによって選択されるあらゆる公平なスケジュールに対して自己安定性を持つという意味である。よって上記の包含関係からR/Wデーモンのもとで正しく動作すれば、他のデーモンによっても正しく動作する。

4 サイズが素数の方向感覚付きリングネットワーク

このリングネットワーク上では、Burnsらの研究[1)]によってDデーモン下で相互排除問題の自己安定解がみつかっている。自己安定相互排除システムを $S_0 = (n, \delta_0, Q_0, \Delta_0)$ で表す。Q_0は状態集合、δ_0は遷移関係、Δ_0は正当状態である。各プロセス P_iの状態 $q_i \in Q_0$は $l_i.t_i$なる二つの成分よりなる。$l_i \in 0, 1, ..., n-2, t_i \in 0 \bigcup 2, 3, ...n-2$ を、それぞれラベル部、タグ部とよぶ。この問題における正当状態は次のようになる。

$$0.0\ 1.0\ ...(a-1).0\ a.0\ \underline{a.0}\ (a+1).0...(n-2).0$$

遷移関係δ_0は、以下の通りである。簡単のため$R_A(i) = (l_i \neq 0\ or\ t_{i-1} = 0\ or\ t_{i-1} \neq l_i - l_{i-1}\ or\ t_{i-1} < t_i)$とする。

$$RuleA\ If\ l_i \neq l_{i-1} + 1\ and\ R(i), then$$

$$l_{i-1}.t_{i-1}\ \underline{l_i.t_i} \longrightarrow (l_{i-1}+1).(l_i - l_{i-1})$$

$$RuleB\ If\ t_{i-1} \neq t_i\ and\ l_{i-1} + 1 \neq 0, then$$

$$l_{i-1}.t_{i-1}\ \underline{(l_{i-1}+1).t_i} \longrightarrow (l_{i-1}+1).t_{i-1}$$

この遷移関係がレジスタ通信モデルでR/Wデーモンによる選択によって行なわれる場合を考える。そのため添字$i-1$の変数の値は隣接プロセスP_{i-1}のレジスタから取得されるため必ずしもP_{i-1}の現在の値と一致していない。これがR/Wデーモンの他のデーモンとの相違である。Burnsらが行なった証明の概略を示す。状態を表すラベルの値が連続しているプロセスの列をセグメントと呼ぶ。任意の状態では高々nの複数のセグメントが存在する。正常な状態は、セグメントの数が1であり、しかもそれはwell formed(タグ部が同状態)である。規則A、規則Bのどちらを適用しても、セグメントの数は増加しない。(規則Aは、特権の移動ならびにセグメント数の減少の働きをし、規則Bはセグメントをwell formedにする働きをする)セグメントの数を一定に保とうとしても、全てのセグメントが有限時間内にwell formedになり、それに伴って必ずセグメント数が減少する。ひとたびセグメント数が1になると、有限時間内にそのセグメントはwell formedになる。このようにして、システムが安定してゆく。

5 R/Wデーモン下での証明

正当状態は先に示した通り、

$$0.0\ 1.0\ ...(a-1).0\ a.0\ \underline{a.0}\ (a+1).0...(n-2).0$$

下線部のプロセスが特権を持っている。このプロセスP_iでReadと状態遷移が実行されると特権状態にあるプロセスはなくなる。なぜなら、プロセスP_{i+1}にはP_iの状態遷移が伝わっていないからである。つまり、P_iでのWriteしか実行され得なくなる。これによって、Dデーモンにおける閉包性の証明と本質的に同じになる[1)]。収束性(ライブロックフリーともいわれる)の証明のエッセンスは次の二つのことを示すことにある。

- セグメントの非増加性
- セグメントを減少させる規則Aの無限適用

上記のことがいえれば、セグメントの数が定数に収束することがいえる。さらにその定数が1となることが示されれば自己安定性が証明される。ここではR/Wデーモン下でも上記の二つの命題が成り立つことを示す。4節で述べたようにR/Wによるスケジュールの集合は、Dデーモンによるそれよりも一般に大きい。システムにある制限があるときには二つの集合が等価となる場合があり、その場合は、Dデーモン下での正当性の証明は、そのままR/Wデーモン下でも適用される。プロセス数3のリングで下記のスケジューリングについて考える。

$$1_R\ 2_R\ 3_R\ 2_W\ 3_W\ 2_R\ 2_W\ 1_W$$

$(1_R, 3_W)(2_R, 1_W)(3_R, 1_W)$などは順序を入れ換えることはできない。一般にDデーモンによるスケジューリングをR/Wデーモンでシミュレートした場合、あるプロセスP_iのReadとWriteの間に別のP_jのRとW動作が複数回行なわれることは起こり得ない。上記のスケジューリングでは1_R1_W間に2_R2_Wが2回行われている。しかし、単方向リングの場合は、プロセスP_2の特権状態に影響を与えるのはプロセスP_1の状態だけであることを考慮すると、プロセスP_2が2_Rによって状態遷移した後は(この問題の場合は)1_Wが実行された後以外はP_2の状態遷移は起こり得ない。よって上記のようなスケジュールは起こり得ないことになる。その場合にはR/Wデーモンで実現されたスケジュールであってもすべてDデーモンのスケジュールに1対1マッピング可能であることが分かる。

双方向リングの場合、特権状態に関する入力が複数存在するためDデーモンのスケジュールに1対1マッピング可能であるとは一般に言えない。また、状態遷移によっても特権状態が持続するような遷移関係がある場合は上記のスケジューリングが起こり得るのでこの場合も1対1マッピング不可能である。

- 特権状態が持続しない。
- 特権状態に関する入力が自己の状態を除き1つである。

これら2つの条件がR/Wデーモンで実現されたスケジュールがDデーモンのスケジュールに1対1マッピング可能であることの十分条件である。

参考文献

[1] J. E. Burns, J. Pachl, "Uniform Self-Stabilizing Rings," *ACM Trans. Progrmming Lang. and Syst.*, Vol.11, No.2, PP.330-344, Apr 1989.