アルゴリズム 47－5
（1995．9．21）

分散合意のための 1-ビットメッセージ最適早期停止アルゴリズム

依田 邦和[1]　岡部 寿男[2]　金澤 正憲[2]
[1] 京都大学大学院工学研究科応用システム科学専攻
[2] 京都大学大型計算機センター

同期式通信を行う分散システムを考える．n 個のプロセッサからなり，そのうち最大 t 個は故障しているとする．各プロセッサは初期バイナリー値を持つとする．分散合意問題は，正常プロセッサ達が，故障プロセッサ達の如何なる振舞にも関わらず，自分達のうちのどれかが持っていた初期値を共通に決定するという問題である．この問題に対する多くの分散アルゴリズムが研究されている．

本研究ではラウンド数と最大メッセージビット数が同時に最適の早期停止アルゴリズム (すなわち、アルゴリズム終了までのラウンド数が実際の故障数 f に比例する) を示す。このアルゴリズムでは全プロセッサ数 $n \geq (4t+1)(t+1)$、ラウンド数 $r = \min\{f+2, t+1\}$、最大メッセージビットサイズ $m = 1$ である．

An optimal early stopping algorithm with one-bit messages for distributed consensus

Kunikazu Yoda[1], Yasuo Okabe[2], Masanori Kanazawa[2]
[1] Division of Applied Systems Science, Faculty of Engineering, Kyoto University
[2] Data Processing Center, Kyoto University

We consider a synchronous distributed system of n processors at most t of which are faulty. Each correct processors has an initial binary value. The Distributed Consensus problem is a problem for all correct processors to agree on a common value that was held by one of them despite any behavior of faulty processors. Many distributed algorithms have been studied for this problem.

This paper presents an early stopping algorithm (i.e., the number of rounds is proportional to the number of actual faults f) which achives round and message optimality simultaneously. This algorithm requires number of prosessors $n \geq (4t+1)(t+1)$, number of rounds $r = \min\{f+2, t+1\}$, maximal message bit $m = 1$.

1 はじめに

分散システムにおける耐故障アルゴリズムの研究分野において合意問題は最も基本的な問題の1つである．分散合意問題(またはこれと同値な問題であるビザンティン合意問題 [LSP82]) とは，信頼できるネットワークで継ったn個のプロセッサの集合(そのうち最大t個故障している)において，正常プロセッサ達が，たとえ故障プロセッサが正常プロセッサに矛盾する情報を送信するといった悪意をもった振舞をしたとしても，正常プロセッサのどれか1つが持っていた初期値を共通に決定するための，通信アルゴリズムを求めるものである．

この合意問題ではアルゴリズムの性能を評価するパラメータとして，全プロセッサ数nと最大故障プロセッサ数tの比，アルゴリズム終了までのラウンド数r，1メッセージの最大ビット数mの3つが知られていて，それぞれ別々の下限も知られている．Pease, Shostak, および Lamport は最大故障数tに対するプロセッサ数nの下限が$n \geq 3t+1$であることを示した [PSL80]．また，Dolev, Strong [DS83] および DeMillo, Lynch, Merritt [DLM82] はラウンド数の下限が$r \geq t+1$であることをそれぞれ独立に示した．

Pease 等はプロセッサ数が $n \geq 3t+1$，ラウンドが $r = t+1$ の最適アルゴリズムを示した [PSL80]．ただしこのアルゴリズムでは最大メッセージビット数は$m = O(n^t)$である．Bar-Noy, Dolev [BD88] や Coan, Welch [CW93] は1ビットメッセージでラウンド数が最適のアルゴリズムを示した．これらのアルゴリズムでは，実行の際に，たとえ故障が全く無かったとしても，起こり得る最大の故障tの場合と同じ$t+1$ラウンドも必要としている．

ラウンドの下限$t+1$を突破する方法として，1つはランダムアルゴリズムを用いる方法があるが [CD89]，もう1つの方法として，実際の故障数fが最大故障数tより少なければそれだけ早いラウンドで停止する早期停止アルゴリズムが考えられた．Dolev, Reishuk, および Strong は，プロセッサが(これまでのアルゴリズムのように同じラウンドで同時に停止するのではなく)別々のラウンドで停止しても良いならば，ラウンド数がfに比例して早く停止する早期停止アルゴリズムを示し，この場合を含めたラウンド数の下限が$\min\{f+2, t+1\}$であることを示した [DRS90]．さらに Berman, Garay, および Perry は，プロセッサ数とラウンド数に関して最適な早期停止アルゴリズム，およびプロセッサ数に関して最適な1ビットメッセージ早期停止アルゴリズムを示した [BGP92]．

本論文では，これまで示されていなかった，ラウンド数に関して最適な1ビットメッセージ早期停止アルゴリズムを示す．ここで示す方法は，早期停止ではないラウンド数に関して最適な1ビットメッセージアルゴリズムである The Beep Once algorithm [BD88] の改良である．The Beep Once algorithm と同じように，全nプロセッサを$t+1$個の交わりの無い集合に分割して，この集合に1から$t+1$までの順番を付ける．そして第kラウンドでは集合kのプロセッサのみが送信というように，各プロセッサは自分の属する集合の順番に相当するラウンドでのみ値を送信して，それ以外のラウンドでは受信のみをする．The Beep Once algorithm と異なる点は，集合kのプロセッサは集合$k+1$のプロセッサへだけに送信するのではなく，全てのプロセッサへ送信すること，そして各プロセッサは0,1のうち一方の値をある一定個以上受信したらその値を決定して停止することの2点である．このアルゴリズムではプロセッサ数は$n \geq (4t+1)(t+1)$, ラウンド数は$r = \min\{f+2, t+1\}$，最大メッセージビット数は$m = 1$である．

この論文の残りの構成は次のようになっている．第2章で問題の定義を示す．第3章でこれまでの合意アルゴリズムの研究をまとめる．第4章で本論文のアルゴリズムを説明する．第5章で今後の展望に付いて述べる．

2 問題の定義

2.1 システムのモデル

本論文では, 複数のプロセッサがネットワークでつながっていてお互いに同期式に通信する分散システムを考える. ただし, プロセッサの中には故障しているものも存在するとする. まず始めに記号の説明をする.

n: プロセッサの数 (既知)

t: 故障プロセッサ数の上限 (既知)
(最大 t 個の故障にまで耐えられるようにアルゴリズムを設計している.)

f: 実際の故障プロセッサ数 (未知)

次にシステムのモデルの説明をする.

- n 個のプロセッサはプロセッサ番号で一意に識別される.
- n 個のプロセッサは完全連結の信頼できるネットワークで結ばれている. すなわち通信は $n(n-1)/2$ 本の双方向の通信リンクを通して誤り無く行われる.
- プロセッサ達には共有メモリは無く, 自分以外のプロセッサの情報は通信リンクを通して直接 (または他を通して間接的に) 知ることができるだけである.
- プロセッサはラウンドという単位で同期を取りながら通信を行なう. 1ラウンドの間に1つのプロセッサは次の3つの動作を行なう.
  1. 前ラウンドまでに記憶している情報を基に, このラウンドでメッセージを送るべきプロセッサ達ごとにおくるべきメッセージをそれぞれのリンクを通して送信する. (メッセージは同じラウンド内に確実に届く.)
  2. このラウンド内に自分に送られて来た全てのメッセージを受信する. (各プロセッサは受信したメッセージそれぞれの送信者を特定できる.)
  3. 受信メッセージ (必要ならばこれらを記憶する) を使ってこのラウンドですべき計算を行ない, 結果を記憶する.

  これを, 第1ラウンド, 第2ラウンド,... というように繰り返して行なっていく.
- 正常プロセッサはあらかじめ決められた共通のアルゴリズムに従い, 最初はどれが故障プロセッサか知らない.
- 故障プロセッサもラウンド単位で通信を行うが, 正常なプロセッサの従うアルゴリズムには従わず, 各ラウンドで誰にどんなメッセージを送っても (または送らなくても) よい. そして自分宛のメッセージは受け取る.

故障プロセッサ達は, 正常プロセッサ間の通信の妨害や, 正常プロセッサになり済ましてメッセージを送る事などはできない. 故障プロセッサの振舞で正常プロセッサ達の目的達成を妨げる原因となる行為は, 複数のプロセッサに対し, 正常プロセッサならば同一のメッセージを送るようアルゴリズムで決められているときに, 別々のメッセージを送るということにある.

2.2 分散合意問題

前節で説明した分散システムを考える. 各プロセッサは初期値 (0 or 1) をそれぞれ独立に持つとする. プロセッサ間で通信を行なうことで, 全ての正常なプロセッサの持つ値を以下の2つの条件の下に, 共通の値 (0 or 1) にセットしたい.

(合意条件) 全ての正常なプロセッサは同じ値を決定する.

(正当条件) もし全ての正常なプロセッサの初期値が同じだったならば, その値を最終的に決定する.

分散システム全体が上の条件を満たす状態になったことを各正常プロセッサが知っている状態にすることが分散合意問題であり，これを実現するための各正常プロセッサで共通に動かすアルゴリズムが合意アルゴリズムである．

2.3 合意アルゴリズムの性能評価パラメータ

合意アルゴリズムは次の3つのパラメータによって評価される．

n: 全プロセッサ数（最大故障数tの関数で表されることで，どれだけの故障に耐えられるかを表す．）

r: 全ての正常なプロセッサが値を決定するまでに要するラウンド数

m: 1メッセージの最大ビット数

3 合意アルゴリズムのこれまでの研究

3.1 合意アルゴリズムの評価パラメータ

ここでは各プロセッサにランダムな動作の無い決定的アルゴリズムのみを考える．合意アルゴリズムを評価するパラメータ各々の下限（理論上の最適値）は次であることが知られている．

- $n = 3t+1$ [PSL80]
  （故障は全体の1/3未満まで）
- $r = \min\{f+2, t+1\}$ [DRS90]
- $m = 1$ (明らか[1])

これら3つを同時に最適にするアルゴリズムはまだ見つかっておらず，また同時に最適にするアルゴリズムは存在しないという結果も出されていない．

[1] 1より大きいビット数のメッセージは複数のラウンドに分けて送れば良い．

3.2 主な合意アルゴリズム

分散合意問題は評価パラメータが3つあるがこれらを同時に最適にすることは困難であり，そのためこれらを同時に小さくする，特にパラメータの2つが最適で，残り一つができるだけ小さいアルゴリズムが発表されている（表1参照）．

表 1: 分散合意問題の様々なアルゴリズム

文献	n	r	m
[BGP92]	$3t+1$	$\min\{f+2, t+1\}$	$O(t!)$
[BGP92]	$3t+1$	$4\min\{f+2, t+1\}$	1
[ZIP92]	$8t+1$	$\min\{f+2, t+1\}$	$O(n)$
[BD88]	$(2t+1)(t+1)$	$t+1$	1
[CW93]	$O(t \cdot \log t)$	$t+1$	1

Berman, Garay, および Perry は [BGP92] で

- プロセッサ数とラウンド数が最適な早期停止アルゴリズム
  $(n \geq 3t+1, r = \min\{f+2, t+1\}, m = O(t!))$
- プロセッサ数が最適な1ビットメッセージ早期停止アルゴリズム
  $(n \geq 3t+1, r = 4\min\{f+2, t+1\}, m = 1)$

をそれぞれ示した．

早期停止でラウンドが最適，nは$O(t)$で，mはなるべく小さいというアルゴリズムでは

- Zamski, Israeli, および Pinter [ZIP92] のアルゴリズム
  $(n \geq 8t+1, r = \min\{f+2, t+1\}, m = O(n))$

が挙げられる．

早期停止ではないアルゴリズムでラウンドが$r = t+1$をとり，1ビットメッセージのものは

- Bar-Noy, Dolev [BD88] の The Beep Once algorithm
  $(n \geq (2t+1)(t+1), r = t+1, m = 1)$

- Coan, Welch [CW93] のアルゴリズム
  $(n = O(t \cdot \log t), r = t + 1, m = 1)$

がある.

3.3 The Beep Once algorithm

次の基本的なアルゴリズムは [BD88] の中で示されたもので，本論文で示すアルゴリズムのもとになったものである．このアルゴリズムではプロセッサの数は $n \geq (2t + 1)(t + 1)$ であり，$r = t + 1$ ラウンド必要とし，1 ビットのメッセージを使う．以下にこのアルゴリズムを記す.

まず，n 個のプロセッサを $t + 1$ 個の共通部分のない集合に分割する．各集合は $2t + 1$ 個のプロセッサからなる．これら集合を $S_1, S_2, \ldots, S_{t+1}$ と示す．このアルゴリズムには $t + 1$ のラウンドがあり，ラウンド k では集合 S_k に属するプロセッサのみがメッセージを送信する．送られてくるメッセージが届かなかった場合は，受信プロセッサはデフォルト値 0 を受け取ったとする.

- $k = 1$: S_1 に属する全てのプロセッサは初期値を S_2 に属する全てのプロセッサへ送る.
- $1 < k < t+1$: 各プロセッサ $p \in S_k$ はラウンド $k - 1$ で $2t + 1$ ビットを受けとる．p はこの $2t + 1$ 個のバイナリー値のうち過半数を占める値を S_{k+1} に属する全てのプロセッサへ送る.
- $k = t + 1$: S_{t+1} に属するプロセッサは S_t から受けとった $2t + 1$ 個の値のうち過半数の値を n 個の全プロセッサへ送る.
- **決定:** 各プロセッサの決定値は最終ラウンドで S_{t+1} から送られてきた $2t+1$ ビットの値の過半数の値とする.

証明： 各集合 S_k の $2t+1$ 個のプロセッサのうち少なくとも $t + 1$ 個は正常である．もしこれら $t + 1$ 個のプロセッサが S_{k+1} へ（$k = t + 1$ のときは全プロセッサへ）共通の値を送ったならばその値が S_{k+1} の正常プロセッサが受け取る値の過半数となる.

この場合，その値が最終ラウンドまで引き継がれ，決定値となる．よって正当条件が成立することがわかる.

交わりの無い $t + 1$ 個の集合があり，故障は最大 t 個なので，故障プロセッサを含まない集合 S_k が少なくとも 1 つ存在する．したがって S_{k+1} の正常なプロセッサ（または $k = t + 1$ のときは全プロセッサ）は同じ $2t + 1$ ビットを受け取り，同じ過半数値を計算する．前の議論よりこの値が決定値となり，合意条件が成立する.

4 ラウンド数が最適の 1 ビットメッセージ早期停止アルゴリズム

この節ではラウンド数が最適の 1 ビットメッセージ早期停止アルゴリズムの説明と合意できることの証明を与える．このアルゴリズムでは効率を表す 3 つのパラメータは以下のとおりである.

- プロセッサ数は $n \geq (4t + 1)(t + 1)$
- ラウンド数は $r = \min\{f + 2, t + 1\}$
- 最大メッセージビット数は $m = 1$

ここでは簡単のため $n = (4t + 1)(t + 1)$ の場合を説明する.

n 個のプロセッサを $t + 1$ 個の互いに重なりのない集合に分割する．各集合はそれぞれ $4t + 1$ 個のプロセッサを要素として持つ．この集合を $S_1, S_2, \ldots, S_{t+1}$ と表示する.

4.1 第 k ラウンドでの各プロセッサの動作

- 集合 S_k に属するプロセッサのみが自分の記憶している値 ($k = 1$ の場合は初期値) を自分自身を含む他の全てのプロセッサへ送信する.
- 集合 S_k から送信された $4t + 1$ 個の値を受け取る．（値が送られてこなかったプロセッサ

についてはこの第 k ラウンドの始めに自分が記憶していた値が送られてきたとみなす.) 送られてきた値のうち, 過半数を占める方の値を自分の値として記憶する.

- もしその記憶した方の値が $3t$ より多く送られてきていたのなら, その値を決定値として停止する. (以後, 送受信は行わない.) $3t$ 以下であれば次の第 $k+1$ ラウンドに進む.
- もし途中で終了しないで $t+1$ ラウンドが終ったら, そのとき記憶していた値を決定値とする.

各プロセッサ p に対するコード

```
for k := 1 to t + 1 do
    if p ∈ S_k then send(V);
    for i := 0 to 1
        C[i] := number of recieved i's
    end;
    V := C[1] > C[0];
    if C[V] > 3t then
        halt
    endif;
end;
```

コードの中の変数や命令の説明

k: ラウンド数

V: プロセッサ p が記憶している値. (第 1 ラウンドの始めでは初期値)

send(V): 自分自身を含む全てのプロセッサへ値 V を送信する.

$C[i]$ $i=0,1$: 受け取った値 i の個数. ($C[0]+C[1]=4t+1$ である.)

halt: 現在記憶している値 V を決定値として停止する.

4.2 合意できることの証明

正当条件: 各集合 S_j の中にはそれぞれ $4t+1$ 個のプロセッサがあり, そのうち $3t+1$ 個以上は正常である. もし第 k ラウンドの始めに, 全正常プロセッサが, 停止しているものもしていないものも同じ値 V を記憶している状態になったならば, 各々が S_k から受信する値の $3t+1$ 個以上は V であるので, このラウンドの終りに, 停止していなかったものも V を決定して停止する. したがって初期値が同じ値 V のとき, 全正常プロセッサは第 1 ラウンドの終りに V を決定して停止する. よって正当条件は成り立つ.

合意条件: $t+1$ 個の互いに重なりのない集合 S_j があり, 故障プロセッサは最大 t 個なので, 故障プロセッサを 1 つも含まない最初の集合 S_h が存在する $(h \leq t+1)$.

もし全正常プロセッサが停止せずに第 h ラウンドまで来た場合, このラウンドでは受け取る 0, 1 の個数 $C[0], C[1]$ は同じであるので, 同じ値 V を記憶する. すると $h < t+1$ ならば次の第 $h+1$ ラウンドの始めには, 全正常プロセッサは同じ値 V を記憶している状態となる. 正当条件のところで示したことにより, V を決定値として停止する. $h = t+1$ ならばこの第 $h(=t+1)$ ラウンドの終りに全正常プロセッサは同じ値 V を記憶している状態で $t+1$ 回目 (最終) のラウンドが終了して停止する.

また, 第 $l(< h)$ ラウンドで初めて値 V を決定して停止した正常なプロセッサがあったとする. このときその停止したプロセッサでは $C[V] > 3t$ であったのだから, S_l の中の正常な $2t+1$ 個以上のプロセッサ (これは S_l の中のプロセッサの総数 $4t+1$ の過半数を占める) は V を送信していたことになる. よって次の第 $l+1$ ラウンドの始めには全正常プロセッサは停止しているものもしていないものも同じ値 V を記憶している状態となる. 正当条件のところで示したことにより, V を決定して残りの正常プロセッサも停止する. 以上より合意条件が成立する.

4.3 合意までのラウンド数

故障プロセッサ数が f のとき，もし，S_h が故障プロセッサを含まない最初の集合だとすると，$h \leq f+1$ である．上の議論から，遅くとも第 $h+1$ ($\leq f+2$) ラウンドには全ての正常なプロセッサは終了する．すなわちアルゴリズム終了までのラウンド数は $\min\{f+2, t+1\}$ である．

4.4 例

ここでは $n = 52$, $t = 3$ の場合の例を示す．全52個のプロセッサは4つの集合 S_1, S_2, S_3, S_4 に分割される．

ラウンド1で，メッセージを送信する集合 S_1 が故障プロセッサを含んでいて，正常プロセッサ達の持つ0,1の個数がほぼ同じで極端な差が付いていないとき，故障プロセッサ達が自分の値として，ある正常プロセッサには0を，別の正常プロセッサには1を送信することで，正常プロセッサが記憶する値を操作できる (図の (A) の場合)．すなわち，次にメッセージを送信する集合 S_2 の正常プロセッサ達の持つ値の0,1の数を半々にする事ができる．同様に S_2 の故障プロセッサの仕業で S_3 の正常プロセッサ達の持つ値の0,1の数を半々にすることができる．このように送信集合に故障が存在する限り，正常プロセッサの値は一致しない．

しかし故障プロセッサを1つも含まない集合 S_3 が存在するので，ラウンド3では全プロセッサには0,1が同じ数だけ送信され，全正常プロセッサは多数決によって皆同じ値を記憶する．すると，ラウンド4の始めには図の (B) のように S_4 の正常プロセッサは全て共通の値を記憶している．よって，ラウンド4の終りには全正常プロセッサは共通の値を決定して停止する．

もし故障プロセッサを1つも含まない集合 S_3 が送信するラウンドより前 (第1，2ラウンド) で一部の正常プロセッサが値を決定して停止してしまった場合 (図の (C) に相当)，このとき停止したものもそうでないものもすべて同じ値を記憶している．よって，次のラウンドの始めには図 (B) のように送信をする集合の正常プロセッサは同じ値を持っている状態になり，このラウンドの終りに全正常プロセッサは停止する．

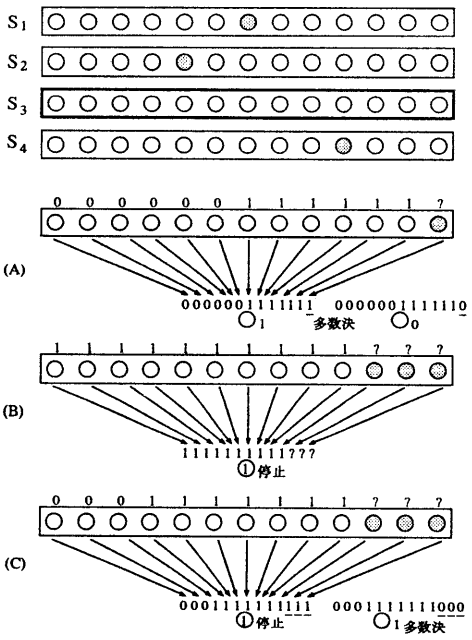

図 1: $n = 52, t = 3$ の例

5 おわりに

本論文では1ビットメッセージの早期停止のラウンド数最適なアルゴリズムを示したが，プロセッサ数に付いては $n = O(t^2)$ であり，最適値 $n = 3t+1$ よりもかなり大きくなっている．これをラウンド数とメッセージはそのままでプロセッサ数を $n = O(t)$ になるべく近づけるようなアルゴリズムを考えるのが今後の目標である．今回は [BD88] を参考にして早期停止アルゴリズムを考えたが，[CW93] をもとにすれば n が $O(t \cdot \log t)$ の

早期停止アルゴリズムを構築できる可能性があると考えている.

参考文献

[BD88] A. Bar-Noy and D. Dolev. "Families of Concensus Algorithms". In *Proceedings 3rd Aegean Workshop on Computing*, pages 380–390, 1988.

[BGP92] P. Berman, J. Garay, and K. J. Perry. "Optimal Early Stopping in Distributed Consensus". In *Proceedings 6th International Workshop on Distributed Algorithms*, pages 221–237, 1992.

[CD89] B. Chor and C. Dwork. "Randomization in Byzantine Agreement". In S. Micali, editor, **Randomness and Computation* Advances in Computer Reaserch*, volume 5. JAI Press, 1989.

[CW93] B. A. Coan and J. L. Welch. "Modular Construction of an Efficient 1-Bit Byzantine Agreement Protocol". *Mathematical System Theory*, pages 131–155, 1993.

[DLM82] R. DeMillo, N. A. Lynch, and M. Merritt. "Cryptographic Protocols". In *Proceedings 14th Annual ACM Symposium on Theory of Computing*, pages 383–400, 1982.

[DRS90] D. Dolev, R. Reischuk, and H. R. Strong. "Early Stopping in Byzantine Agreement". *Journal of the ACM*, pages 720–741, 1990.

[DS83] D. Dolev and H. R. Strong. "Authenticated Algorithms for Byzantine Agreement". *SIAM Journal of Computing*, pages 656–666, 1983.

[LSP82] L. Lamport, R. Shostak, and M. Pease. "The Byzantine Generals Problem". *ACM Trans. Program. Lang. Syst.*, pages 382–401, 1982.

[PSL80] M. Pease, R. Shostak, and L. Lamport. "Reaching Agreement in the Presence of Faults". *Journal of the ACM*, pages 228–234, 1980.

[ZIP92] A. Zamsky, A. Israeli, and S. S. Pinter. "Optimal Time Byzantine Agreement for $t < n/8$ with linear messages". In *Proceedings 6th International Workshop on Distributed Algorithms*, pages 136–152, 1992.