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[†,☆] 岡部寿男[††] 金澤正憲[††]

分散合意問題は，同期式通信を行う分散システムにおいて，0または1の初期値を持つ正常プロセッサ達が，故障プロセッサ達のいかなる振舞いにもかかわらず，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ための通信アルゴリズムを与える問題である．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を示す．早期停止アルゴリズムとは，アルゴリズム終了までのラウンド数が，最大故障数ではなく実際の故障数に比例するものである．システムが n 個のプロセッサからなり，そのうち最大 t 個が故障している可能性があり，実際の故障数は f であるとする．このアルゴリズムでは，全プロセッサ数 $n \geq (4t+1)(t+1)$ のシステムにおいて，1ビットのメッセージを用いてラウンド数 $r = \min\{f+2, t+1\}$ で合意に到達する．

An Optimal Early Stopping Algorithm with One-bit Messages for Distributed Consensus

KUNIKAZU YODA,[†,☆] YASUO OKABE[††] and MASANORI KANAZAWA[††]

The Distributed Consensus is a problem of protocols by which all correct processors agree on a common binary value that was initially held by any of them, regardless of any behavior of faulty processors, on a synchronous distributed system. We present an early stopping algorithm (i.e., the number of rounds is proportional to the number of actual faults f, not the number of possibly faulty processors t) with one-bit messages. The algorithm achieves a consensus using one-bit messages within $r = \min\{f+2, t+1\}$ rounds on a system where the number of prosessors is $n \geq (4t+1)(t+1)$.

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）とは，信頼できるネットワークでつながった n 個のプロセッサの集合において，そのうち最大 t 個故障しているとき，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持った振舞いをしたとしても，正常プロセッサ達が，正常プロセッサのどれか1台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定めるものである[8]．

分散合意問題では，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つが用いられ，それぞれ下限が知られている．Pease, Shostak, Lamport[9]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．また，Dolev, Strong[7]およびDeMillo, Lynch, Merritt[5]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らはプロセッサ数が $n \geq 3t+1$，ラウンド数が $r = t+1$ の最適アルゴリズムを示した[9]．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[1]やCoan, Welch[4]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，たとえ故障がまったくなかったとしても，起こりうる最大の故障 t の場合と同じ $t+1$ ラウンドも必要としている．

ラウンド数の下限 $t+1$ を突破する方法の1つにランダムアルゴリズムを用いる方法があるが[3]，もう1つ

† 京都大学大学院工学研究科
Faculty of Engineering, Kyoto University
☆ 現在，IBM東京基礎研究所
Presently with IBM Tokyo Research Laboratory
†† 京都大学大型計算機センター
Kyoto University Data Processing Center

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, Strong[6] 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よいならば，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．さらに Berman, Garay, Perry[2] 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．

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[1] の改良である．The Beep Once algorithm と同じように，全 n プロセッサを $t+1$ 個の交わりのない集合に分割して，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，すべ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．2章で問題の定義を示す．3章でこれまでの合意アルゴリズムの研究をまとめる．4章で本論文のアルゴリズムを説明する．5章で今後の展望について述べる．

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．以下，n, t, f の記号を次の意味で用いる．

$\boldsymbol{n}$: 全プロセッサ数（既知）

$\boldsymbol{t}$: 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する．）

$\boldsymbol{f}$: 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なく行われる．
- プロセッサ達には共有メモリはなく，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とりながら通信を行う．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送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きたすべ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) 受信したメッセージを使ってこのラウンドですべき計算を行い，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，各ラウンドでだれ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．

故障プロセッサの振舞いで正常プロセッサ達の目的達成を妨げる原因となるのは，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りうる点である．故障プロセッサ達は，正常プロセッサ間の通信の妨害や，他の正常プロセッサになりすましてメッセージを送ることはできない．

表1 分散合意問題の様々なアルゴリズム
Table 1 Known algorithms for distributed consensus.

	n	r	m
Berman, Garay, Perry [2]	$3t+1$	$\min\{f+2, t+1\}$	$O(t!)$
Berman, Garay, Perry [2]	$3t+1$	$4\min\{f+2, t+1\}$	1
Zamski, Israeli, Pinter [10]	$8t+1$	$\min\{f+2, t+1\}$	$O(n)$
Bar-Noy, Dolev [1]	$(2t+1)(t+1)$	$t+1$	1
Coan, Welch [4]	$O(t\cdot\log t)$	$t+1$	1

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うことで，すべての正常なプロセッサの持つ値を以下の2つの条件のもとに，共通の値（0 or 1）にセットしたい．

（**合意条件**）すべての正常なプロセッサは同じ値を決定する．

（**正当条件**）もしすべての正常なプロセッサの初期値が同一であったならば，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 の関数で表すことで，どれだけの故障に耐えられるか（resiliency）を示す．）

r: すべ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な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．

- $n = 3t+1$ [9]
  （故障は全体の1/3未満まで）
- $r = \min\{f+2, t+1\}$ [6]
- $m = 1$（明らか☆）

これら3つを同時に最適にするアルゴリズムが存在するかどうかについては明らかになっていない．

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ると考えられている．そのためこれらを同時に小さくし，特にパラメータの2つが最適で，残り1つができるだけ小さいアルゴリズムが研究されている（**表1** 参照）．

Berman, Garay, Perry は文献2)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 はなるべく小さいものでは

- Zamski, Israeli, Pinter のアルゴリズム[10]
  $(n \geq 8t+1, r = \min\{f+2, t+1\}, m = O(n))$

があげられる．

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 の The Beep Once algorithm [1]
  $(n \geq (2t+1)(t+1), r = t+1, m = 1)$
- Coan, Welch のアルゴリズム[4]
  $(n = O(t\cdot\log t), r = t+1, m = 1)$

がある．

3.3 The Beep Once algorithm

The Beep Once algorithm は文献1)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，1ビットのメッセージを使うという前提のもとで，プロセッサの最大故障数 t は $n \geq (2t+1)(t+1)$ であり，$r = t+1$ ラウンド必要とする．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

☆ 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よい．

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 に属するすべてのプロセッサは初期値を S_2 に属するすべ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 $2t+1$ ビットを受け取る．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} に属するすべ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 から受け取った $2t+1$ 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分かる．

交わりのない $t+1$ 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

この章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 $n=(4t+1)(t+1)$ の場合について説明する．

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すべ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見なす）．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 $t+1$ ラウンドが終わったら，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;
    if C[1] > C[0] then
        V := 1
    else
        V := 0
    endif;
    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): 自分自身を含むすべ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，このラウンドの終わりに，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ラウンドの終わりに 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 ラウンドまできた場合，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 $h(=t+1)$ ラウンドの終わ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

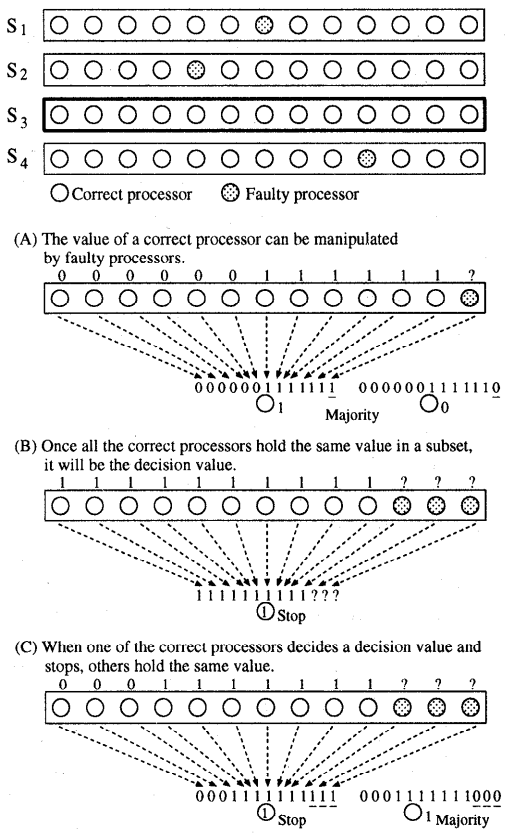
故障プロセッサ数が f のとき，もし，S_h が故障プロセッサを含まない最初の集合だとすると，$h \leq f+1$ である．上の議論から，遅くとも第 $h+1\,(\leq f+2)$ ラウンドにはすべての正常なプロセッサは終了する．すなわちアルゴリズム終了までのラウンド数は $\min\{f+2, t+1\}$ である．

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ついていないとき，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1**(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の数を半々にすることができる．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



Possible Scenarios:

1. (A)→ ····· → (A) → (B)
2. (A)→ ····· → (A) → (C) → (B)

図1 $n = 52$，$t = 3$ の例

Fig. 1 Example of the protocol with $n = 52$, $t = 3$.

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1(B)のように S_4 の正常プロセッサはすべ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1(C)に相当）は，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1(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，そのラウンドの終わりに全正常プロセッサが停止する．

5. おわりに

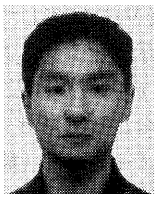
本論文では1ビットメッセージの早期停止のラウンド数最適なアルゴリズムを示した．ここで示したアルゴリズムは，プロセッサ数については $n = O(t^2)$ であり，下限 $n = 3t + 1$ よりもかなり大きくなっている．これを，ラウンド数とメッセージはそのままでプロセッサ数を $n = O(t)$ になるべく近づけるようなアルゴリズムに改良するのが今後の目標である．今回は文献1)を参考にして早期停止アルゴリズムを考えたが，文献4)をもとにすれば n が $O(t \cdot \log t)$ の早期停止アルゴリズムを構築できる可能性があると考えている．

謝辞 ご討論いただいた本学工学研究科応用システム科学教室応用情報学講座の諸氏に深謝する．

参考文献

1) Bar-Noy, A. and Dolev, D.: Families of Concensus Algorithms, *Proc. 3rd Aegean Workshop on Computing*, pp.380–390 (1988).
2) Berman, P., Garay, J. and Perry, K.J.: Optimal Early Stopping in Distributed Consensus, *Proc. 6th International Workshop on Distributed Algorithms*, pp.221–237 (1992).
3) Chor, B. and Dwork, C.: Randomization in Byzantine Agreement, **Randomness and Computation* Advances in Computer Reaserch*, Micali, S. (Ed.), Vol.5, JAI Press (1989).
4) Coan, B.A. and Welch, J.L.: Modular Construction of an Efficient 1-Bit Byzantine Agreement Protocol, *Mathematical System Theory*, Vol.26, pp.131–155 (1993).
5) DeMillo, R., Lynch, N.A. and Merritt, M.: Cryptographic Protocols, *Proc. 14th Annual ACM Symposium on Theory of Computing*, pp.383–400 (1982).
6) Dolev, D., Reischuk, R. and Strong, H.R.: Early Stopping in Byzantine Agreement, *J. ACM*, Vol.37, No.4, pp.720–741 (1990).
7) Dolev, D. and Strong, H.R.: Authenticated Algorithms for Byzantine Agreement, *SIAM Journal of Computing*, Vol.12, No.4, pp.656–666 (1983).
8) Lamport, L., Shostak, R. and Pease, M.: The Byzantine Generals Problem, *ACM Trans. Program. Lang. Syst.*, pp.382–401 (1982).
9) Pease, M., Shostak, R. and Lamport, L.: Reaching Agreement in the Presence of Faults, *J. ACM*, Vol.27, No.2, pp.228–234 (1980).
10) Zamsky, A., Israeli, A. and Pinter, S.S.: Optimal Time Byzantine Agreement for $t < n/8$ with Linear Messages, *Proc. 6th International Workshop on Distributed Algorithms*, pp.136–152 (1992).

(平成9年 8 月 7 日受付)
(平成9年 11 月 5 日採録)



依田 邦和（正会員）
1972年生．1994年京都大学工学部数理工学科卒業．1996年同大学院工学研究科応用システム科学専攻修士課程修了．同年日本アイ・ビー・エム（株）入社．現在同社東京基礎研究所においてデータベース，とくにデータマイニングの分野の研究に従事．

岡部 寿男（正会員）
1964年生．1988年京都大学大学院工学研究科情報工学専攻修士課程修了．同年同大学工学部助手．現在同大学大型計算機センター助教授．博士（工学）．並列計算の複雑さなどの研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE，ACM，EATCS各会員．

金澤 正憲（正会員）
1946年生．1971年京都大学大学院工学研究科数理工学専攻修士課程修了．1972年同大学大型計算機センター助手．同助教授をへて1995年同教授，現在に至る．工学博士．計算機システム・オペレーティングシステムの方式と性能評価，およびコンピュータネットワーク・分散システムに興味を持っている．電子情報通信学会，ACM，日本応用数理学会各会員．