解　説

ビザンティン合意問題

——信頼性の低い分散ネットワーク上での合意問題——†

山　下　雅　史††

1. はじめに

故障を含む分散ネットワーク上で正常なプロセッサが合意を取る問題，すなわちビザンティン合意問題を解決する分散アルゴリズム（プロトコル）について解説する．分散ネットワークは通信リンクで結合された共有メモリをもたない自律的に動作するプロセッサの集合であり，分散ネットワーク全体を統括するような特別のプロセッサは存在しない．そこで，分散ネットワーク全体に関する問題を解決するためには，（共有メモリが存在しないので）各プロセッサはある逐次アルゴリズムにしたがってそれぞれの管理する情報を通信リンクを介して手際良く交換する必要があるが，この個々のプロセッサが実行する情報の抽出，交換の逐次アルゴリズムの総体（集合）が分散アルゴリズムである．（断わらないかぎり，以下ではアルゴリズムと言えば分散アルゴリズムのことを指すものとする．）

分散ネットワークや分散アルゴリズムのような概念を含む分散計算モデルは，通常のアルゴリズム論で用いられる（RAM や RASP のような）逐次計算モデルや（PRAM のような共有メモリを仮定する）並列計算モデルとは異なるモデルであり，本来ならばその解説から始めるべきであるが，その一般的な解説は萩原，増澤[31](1990)に譲り，ここでは分散計算モデルに関する知識を仮定する．（必要最小限の用語は 2. で説明する．）

ビザンティン合意問題には相互無矛盾問題，将軍問題，そして合意問題，という三つの定式化がある．そこで，ビザンティン合意問題がそう呼ばれる理由から始め，これら三つの問題を説明する．（本解説では，合意問題による定式化を採用するので，厳密な定義は合意問題に対してだけ与える．）

> それぞれ将軍に率いられたいくつかのビザンティン部隊が敵地を包囲している．将軍たちは通信兵を使って互いに情報を交換する*．敵情を視察した結果，将軍たちは共同作戦を行うことにした．しかし，将軍の一部は裏切り者であって，将軍たちが作戦計画を合意することができないように振る舞うかも知れない．そこで，少数の裏切り者に惑わされることなく，すべての忠誠な将軍が共通の作戦計画を決定する方法を示せ[35]．

ビザンティンが情報工学・情報科学と関係をもった最初の場面である．将軍が n 人いるとし，$q(i)$ $(1\leq i\leq n)$ で将軍 i の作戦に対する意見を表す．作戦が将軍たちの意見だけから決定され（たとえば，攻撃か撤退かを多数決で決める），その決定手続きを各将軍がすでに知っているとするならば，この問題は（将軍たちの意見を表す）つぎのようなあるベクトル $\mathbf{r}=(r(1), r(2), \cdots, r(n))$ をすべての忠誠な将軍が知ることができれば解決できる．

ここに忠誠な将軍 i に対しては $r(i)=q(i)$ でなければいけないが，裏切り者である将軍は事態を混乱させるために出鱈目な意見を述べると考えられるので，裏切り者である将軍 j に対しては $r(j)=q(j)$ である必要はない**．

この問題を将軍をプロセッサに通信兵の往復す

† Byzantine Consensus Problem ——Consensus Problem in the Presence of Faults (A Survey)—— by Masafumi YAMASHITA (Department of Electrical Engineering, Faculty of Engineering, Hiroshima University).

†† 広島大学工学部第二類

* 将軍たちが一堂に会して会議をするようなことはない．

** 将軍 j が裏切り者である場合には $r(j)$ の値は重要ではない．すべての忠誠な将軍が同じ作戦を選択するために同じベクトル $\mathbf{r}$ を知ることが重要である．将軍 j が裏切り者である場合には，$r(j)$ に特別な値 *Traitor* を代入するように条件を強める必要がある，と考えられる方も多いであろう．このように条件を変更すると明らかに問題は解決できない．なぜなら，ある裏切り将軍 j は他の将軍に対して $q'(j)$ $(\neq q(j))$ を伝えることを除いてまったく忠誠な将軍と同じように振る舞うことができ，この場合将軍 j を裏切り者か否か判定する術がない．

る道を通信リンクに対応させ，裏切り者である将軍を故障プロセッサに対応させることで，**点対点通信** (*point-to-point communication*) で結合された（故障プロセッサを含む可能性がある）分散ネットワーク上の問題とみなし，**相互無矛盾問題** (*interactive consistency problem*) と呼ぶ．

点対点通信で結合された分散ネットワークにおいて，プロセッサ P_T が保持する情報 I を他のすべてのプロセッサに伝える問題を**放送問題** (*broadcast problem*) という*．故障が起こらない分散ネットワークでは放送は簡単に実現できる．だが残念なことに，分散ネットワークはさまざまな理由からプロセッサや通信リンクに故障を起こす．このような信頼性の低い分散ネットワークにおける“放送”においては，ネットワークに属する P_T 以外のすべての正常プロセッサが，P_T が正常ならば情報 I を，P_T が故障していて各プロセッサごとに異なる情報を送信しているような場合でも，ある共通の値 J を獲得できることが望まれる．この拡張された放送問題を**将軍問題** (*generals problem*) と呼ぶ**．ここにあげた二つの問題は一方が解決できるならば他方も解決できるという意味で互いに帰着可能な問題である．このことは，つぎのように示せる．将軍問題が解けるならば，相互無矛盾問題を解くためには各プロセッサ P_i が $q(i)$ を“放送”すればよい．一方相互無矛盾問題が解けるならば，$q(i)$ を P_i が放送したい情報 I として相互無矛盾問題を解くことによって将軍問題を解決できる．

さらに**合意問題** (*consensus problem* または *agreement problem*) と呼ばれる重要な分散問題がある．たとえば，分散データベースシステムにおいて，システムに属するあるサイトがデータを更新する場合には，そのデータに関係するすべてのサイトがデータの更新に合意する必要がある．合意問題はつぎのように定義される．

[合意問題] 各プロセッサ P_i が初期データ $v(i)$ と局所変数 $w(i)$ をもつとする．すべての正常プロセッサ P_i の変数 $w(i)$ を以下の条件を満たす値 $w(\in W)$ にセットせよ．

1. $|W|\geq 2$，かつ，任意の $w\in W$ に対して $w(i)=w$ で停止する初期データの組 $v=(v(1),\cdots,v(n))$ が存在する*．
2. すべての正常プロセッサ P_i が同じ初期データ $v(i)=v$ をもつならば $w=v$**．

合意問題が相互無矛盾問題と互いに帰着可能であることは，つぎのように示すことができる．相互無矛盾問題が解ければ合意問題が解けることは明らかである．一方，合意問題が解けるならば，“放送”を試みるプロセッサ P_T は放送したい情報 I をすべてのプロセッサに送信し，続いて各プロセッサ P_i が受信した値 $v(i)$ を初期データとして合意問題を解くことで将軍問題を解決できる．文献では，これら三つの互いに帰着可能な問題は区別されず合意問題と呼ばれることも多い．本稿では，合意問題（最後に示した問題）を解説するが，すべての結果は他の二つの問題に対しても（わずかな修正の後）成立する．

これらの問題が見かけほど簡単でないことはつぎの例から容易に推測できる．

例1 互いに通信可能な3台のプロセッサ P_1, P_2, P_3 からなる分散ネットワークを考えよう．最大1台のプロセッサが故障する可能性があり，故障プロセッサはまったく出鱈目に振る舞う可能性もあると仮定する．この分散ネットワーク上で合意問題を解くアルゴリズム A が存在すると仮定して，以下の三つのシナリオに対して A がどのように振る舞うか観察する．

1. P_1 と P_2 は正常で $v(1)=v(2)=0$ であるが，P_3 は故障している．
2. P_2 と P_3 は正常で $v(2)=v(3)=1$ であるが，P_1 は故障している．
3. P_1 と P_3 は正常で $v(1)=0$ かつ $v(3)=1$ であるが，P_2 は故障している．

シナリオ1を考える．P_1 と P_2 は合意問題の条件2を満足するために $w(1)=w(2)=0$ を出力す

---

* プロセッサの基本通信命令は点対点通信の基本通信命令である Send と Receive であって，放送を行う基本通信命令はサポートされていない．2. を参照せよ．

** 将軍（司令官）が部下の部隊長たちに正しく命令を伝達する問題として提起された．部隊長たちは少なくとも同じ行動を取る必要がある．しかも，将軍が裏切り者でなければ部隊長たちは将軍の命令に従う必要がある．

* この仮定がなければ，すべてのプロセッサ P_i が前もって定められたある値 w_0 を常に $w(i)$ の値とする，という自明な解が存在する．それゆえこの条件を**非自明条件** (*non-triviality condition*) と呼ぶ．

** この条件は**強合意条件** (*strong unanimity condition*) と呼ばれている．この条件を**弱合意条件** (*weak unanimity condition*) と呼ばれる条件“故障が実際に発生しなければ，すべての正常プロセッサ P_i が同じ初期データ $v(i)=v$ をもつならば $w=v$”に置き換えた弱ビザンティン合意問題（とそれに対応する相互無矛盾問題あるいは将軍問題）もしばしば取りあげられてきた[10]．

る．シナリオ2の場合には同じ理由から $w(2)=w(3)=1$ を出力する．最後にシナリオ3を考える．仮定から，P_2 はどのようにでも振る舞うことができるので，P_2 は P_1 に対しては $v(2)=0$ である正常プロセッサのように，P_3 に対しては $v(2)=1$ である正常プロセッサのように振る舞うことができる．このとき，P_1 はシナリオ1にしたがって振る舞い*，一方，P_3 はシナリオ2にしたがって振る舞う．すなわち，$w(1)=0$ かつ $w(3)=1$ となり矛盾が導かれた．したがって，この分散ネットワーク上で合意問題を解くアルゴリズムは存在しない．□

このように，単純な分散ネットワーク上でも故障の可能性を考慮したときには合意問題は解決困難となり，しばしば解決不可能となる．しかし，分散ネットワークの非同期さかげんや故障の程度を限定することにより故障の可能性のある分散ネットワーク上でも合意問題を解決できる場合もある．このように，少々の故障に影響されることなく正しく問題を解くアルゴリズムを**フォールトトレラントアルゴリズム**（*fault tolerant algorithm*）と呼ぶ．本稿の目的は，合意問題に対するフォールトトレラントアルゴリズムを解説することである．

2. モ デ ル

分散ネットワークは双方向通信リンクを介して結合されているプロセッサの集合 $\{P_1, \cdots, P_n\}$（n はプロセッサ数）である．ここに，プロセッサは通信命令（Send と Receive）を実行でき，局所メモリをもつ（実または仮想）計算機である**．分散ネットワークはプロセッサの集合を頂点集合に，通信リンクの集合を枝集合に取ることによって無向グラフとみなすことができる．このグラフを分散ネットワークのトポロジと呼ぶ．トポロジと合意アルゴリズムの存在性の間には重要な関係が存在するが***，以下では，分散ネットワークのトポロジを完全グラフと仮定する．すなわち，任意の二つのプロセッサ間に通信リンクが存在する．

分散ネットワークには共有メモリは存在しない．通信は点対点通信だけが許されているので，直接通信リンクによって結合されていないプロセッサどうしが情報を交換するためには，他のプロセッサを経由してメッセージを伝送する必要がある．

分散ネットワークは大きく**非同期ネットワーク**（*asynchronous network*）と**同期ネットワーク**（*synchronous network*）に分類できる*．非同期ネットワークは**大域時計**（*global clock*）をプロセッサが参照できず，しかもプロセッサの相対的な速度やメッセージ遅延に上限値を仮定しないネットワークである．したがって，プロセッサや通信リンクに故障がないかぎり送信されたメッセージは必ず有限時間内に配送されるが，配送に要する時間に上限がないので，たとえばタイムアウトのような手法を用いてメッセージが確かに届けられたかどうか（送信先のプロセッサや使用した通信リンクが故障しているかどうか）を確かめることはできない．一方，プロセッサが大域時計を参照でき，しかもプロセッサの速度や通信遅延などがそれぞれ定数であると仮定されるのが同期ネットワークである．同期ネットワークでは，大域時計の数クロックを抽象化した**ラウンド**（*round*）を考え，各プロセッサは同期して各ラウンドを実行するものと考える．ある一つのラウンドではつぎの二つのことが実行できる．

1. 直前のラウンドに自分に対して送信されたメッセージをすべて受信する．
2. そのときの内部状態と受信したメッセージから，つぎの内部状態と送信すべきメッセージとその送信先を決定し，状態遷移とメッセージの送信を行う．

本解説で対象とする故障はプロセッサ故障であり，故障の種類としてつぎの二つを想定する**．

---

* P_1 は P_2 と P_3 の間の通信内容を知ることができないので，（シナリオ1では P_2 は故障プロセッサだからどのような振る舞いもするから）P_3 の反応に係わらず分散ネットワークの状況があたかもそれがシナリオ1にしたがって動いているようにみえる．

** 紙面の制約から分散ネットワークの定義の詳細には立ち入ることはできない．分散ネットワークの基本的なモデルの詳細については文献 31) を参照されたい．

*** 合意アルゴリズムが存在するための必要条件の一つはプロセッサ数 n とトポロジの点連結度 k の間に不等式 $n>2k$ が成立することである[10),30)]．

* ここで定義される非同期ネットワークと同期ネットワークの中間にさまざまな非同期さレベルをもったネットワークモデルが考えられる．たとえば，通信遅延の上限を既知と仮定するようなモデルである．このようなモデルに対するビザンティン合意問題は文献 1) などで検討されている．

** 多くの文献では，プロセッサ故障だけが考慮されている．しかし通信リンク故障やプロセッサと通信リンクの複合故障に対するビザンティン合意アルゴリズムの研究もある[43)]．また，故障の種類として，**送信脱落故障**（*send-omission failure*—送信しなくてはならないメッセージが送信されないことがある）や**一般脱落故障**（*general-omission failure*—送信時だけでなく受信時にもメッセージが脱落することがある）なども考察されている[11),30)]．

1. **停止故障** (*crash* または *fail-stop*)：故障したプロセッサは永遠に停止する．プロセッサは故障するまで正常に動作する．

2. **ビザンティン故障** (*Byzantine failure*)：故障したプロセッサの動作に一切仮定をおかない．すなわち，故障プロセッサがアルゴリズムを正しく実行しない任意の動作をする．

ビザンティン故障を想定する合意問題をビザンティン合意問題という．

合意問題のように，分散ネットワークに属するプロセッサに分散，蓄積されている情報全体にかかわる問題を**分散問題** (*distributed problem*) といい，ある分散問題を解決するために設計されたアルゴリズムを**分散アルゴリズム** (*distributed algorithm*) という．(以下では，単にアルゴリズムと呼ぶ．) 分散ネットワークには共有メモリが存在しないから，分散問題を解決するためには，分散ネットワークに属する各プロセッサがある手順にしたがって手際よく必要な情報を交換する必要がある．個々のプロセッサが実行する情報の抽出および交換の手続きの総体が (分散) アルゴリズムである．分散ネットワーク上におけるアルゴリズムの実行は，一つあるいは複数のプロセッサが自主的にアルゴリズムの実行を開始することで開始される．このように，自主的に実行を開始するプロセッサを**始動プロセッサ** (*initiator*) といい，それらはなんらかの理由から，この分散アルゴリズムによって分散問題を解決する必要に迫られているプロセッサである．始動プロセッサ以外のプロセッサは，やがて自らも始動プロセッサとなるか，あるいは他のプロセッサからのメッセージを受信することによってアルゴリズムの実行を開始する．

すでに述べたように，故障に耐え正しく問題を解決できるアルゴリズムをフォールトトレラントアルゴリズムと呼ぶが，どのようなフォールトトレラントアルゴリズムも無制限な故障に耐えられるわけではない．あるアルゴリズムが，たとえば，最大 t 個のプロセッサ故障に耐えるときにはこのアルゴリズムを**t-耐プロセッサ故障** (t-*resilient*) である，というような言い方をする．

最後に，アルゴリズムの良さの尺度について触れておく．通常の逐次計算モデルではアルゴリズムの良さは時間と空間という二つの尺度で評価される．それに対して分散ネットワークモデルではアルゴリズムの実行が終了するまでにプロセッサ間で交換されるメッセージ（ビット）の総数を**メッセージ（ビット）複雑度** (*message* (*bit*) *complexity*) と呼び，これを用いてアルゴリズムの良さを評価する*．また，同期ネットワークの場合にはアルゴリズムの実行が終了するまでにプロセッサが実行するラウンド数を評価尺度とすることも多い．

3. ビザンティン合意問題の非可解性

非同期ネットワークにおいては，(1)初期データ $v(i)\in\{0,1\}$，(2)出力値 $w(i)\in\{0,1\}$，(3)故障は停止故障に限る，そして，(4)故障プロセッサ数の上限は1，という最も単純な場合でもビザンティン合意アルゴリズムが存在しないことを説明する．すなわち，任意にある分散ネットワーク Σ を固定し**，(一般の分散ネットワークに対するアルゴリズムはもちろん) 固定された Σ に対する 1-耐プロセッサ停止故障合意アルゴリズムでさえ存在しないことを示す．したがって，十分多いプロセッサが参加する分散ネットワークにおいても，たった1台プロセッサが故障する可能性があるだけで，たとえば多数決のような方法で正常プロセッサ間の合意を取ることはできない．非同期ネットワークのこのように困難な状況を緩和するために**ランダム化アルゴリズム** (*randomized algorithm*) を用いたビザンティン合意問題の解法が提案されているが，これについては **5.** で述べる．

A を上に述べた条件を満たすある固定された非同期ネットワーク Σ に対する (1-耐プロセッサ停止故障) 合意アルゴリズムであると仮定し，矛盾を導く．Σ の計算状況を C で表す．すなわち，C はある時点における，プロセッサの状態，通信リンクの状況など，Σ の状態を規定するすべての情報を含んでいる．A の可能な計算過程の全体は，計算状況の集合を頂点集合とする (無限かもしれない) 有向グラフ Γ として表現される．ここに，計算状況 C_1 から C_2 に向かう枝は，C_1 であるプロセッサ P である事象 e が生起し，そ

* メッセージ長は $O(\log n)$，すなわち，$O(\log n)$ ビット程度の情報が一つのメッセージで送信できると仮定するのが一般的である．
** プロセッサ数 $n\geq 2$ とする．

の結果計算状況が C_2に変化したと解釈し*，この枝を P でラベル付ける．(このとき，C_1 に e が適用され C_2 が生じた，という．) ここに，事象とは，あるプロセッサからのメッセージを受信し，対応する内部状態の推移とメッセージの送信をすることとする．アルゴリズム A は合意値（出力値）$w(i)=w$ をただ1度だけセットし，1度セットされた合意値は変更されないと仮定しても一般性を失わない．合意値 $w=i$ $(i\in\{0,1\})$ をもつプロセッサが存在するような計算状況だけに到達可能な計算状況を i-計算状況と呼び，0-計算状況と1-計算状況の両方に到達可能なまだ合意値を決めかねている計算状況を未決定計算状況と呼ぶことにする．当然ながら，未決定計算状況は i-計算状況ではない．このとき，つぎの二つの事実が証明できる．

1. 未決定初期計算状況 C_0 が存在する．

2. 未決定初期計算状況 C_0 から到達可能なある未決定計算状況 C とあるプロセッサ P が存在して以下の三つの条件を満たす．

A. P のラベルをもつ枝をちょうど一つ含む C から 0-計算状況 D_0 に至る有向道が存在する．

B. P のラベルをもつ枝をちょうど一つ含む C から 1-計算状況 D_1 に至る有向道が存在する．

C. C_0 から D_0 および D_1 に至る有向道は故障プロセッサを含まない計算に対応する．

このようなプロセッサ P を決定者と呼ぶことにする．

事実1は以下のように示すことができる．あるプロセッサを除いてすべてのプロセッサの入力データ $v(i)$ が等しいような二つの計算状況は隣接しているという．すべての初期計算状況が未決定計算状況でないと仮定する．合意問題の定義から，隣接する初期計算状況 C_0 と C_1 で，C_0 から到達可能な計算状況の中には 1-計算状況は存在せず，一方，C_1 から到達可能な計算状況の中には 0-計算状況は存在しないようなものが存在する．C_0 と C_1 ではプロセッサ P の入力だけが異なっているものとする．P が最初から故障しているというシナリオの下で C_0 と C_1 からの計算を追跡すると，明らかに両者は一致するはずであり，矛盾が導かれる．

つぎに事実2はおおよそ以下のように示すことができる．A はいずれ合意値を決定する必要があるから，未決定初期計算状況 C_0 から到達可能な未決定計算状況 C で，C から到達可能なすべての計算状況が未決定計算状況ではないものが存在する．(そうでなければ，i-計算状況を含まない無限有向道が存在し，矛盾.) C の子頂点の中には 0-計算状況 D_0 と 1-計算状況 D_1 の両方が含まれている．C から D_0 への推移がプロセッサ P における事象 e の適用によって生じたとする．このとき，C から D_1 への推移が P でラベル付けされていないならば D_1 に e は適用可能，すなわち，D_1 には P でラベル付けされた出次枝が存在する．

定理 1　非同期ネットワークに対する 1-耐停止故障合意アルゴリズムは存在しない[29]．

証明　そのようなアルゴリズム A が存在したとすると，事実2の条件を満たすある初期計算状況から到達可能な計算状況 C と決定者 P が存在する．D_1 において，P が故障したと仮定しても有限のステップであるプロセッサが合意値 $w=1$ を決定するような計算状況 D_2 に到達するはずである．C から D_2 に到達するために生起した事象の列を α とする．D_1 において，P が（故障したのではなく）P と他のプロセッサ間の通信遅延が十分に長かっただけだと考えても同じ状況が生ずるはずであるから，D_2 に到達し，D_2 に P の事象 e は適用可能である．D_2 に e を適用して生じる計算状況を D_3 とする．D_3 は 1-計算状況である．一方，D_0 に α が適用でき，その結果生じる 0-計算状況を D_4 とする．ところが，α には P の事象が含まれていないから $D_3=D_4$ となり，矛盾が導かれた．□

非同期ネットワーク上で合意問題を扱うかぎり，あるプロセッサ P の振る舞いに依存して合意値が決定されるという微妙な状況が起こり，そのような状況では，P が決定した合意値（あるいは P が故障しているのか否か）を正確に知る必要があるにも係わらず，それは不可能* であることから，合意問題の非可解性が導出された．すなわち，非可解性は決定者である P が計算の途中で突然停止し，そのことが外部から確認できない

★二つ以上のプロセッサで同時に事象が生起することはないと仮定する．このように限定された振る舞いからでも矛盾が生じることを示す．

★ Pが合意値を決定したならばその値を含むPのメッセージを待つ必要があるが，Pが故障していて合意値を決定していない可能性もあるので，Pのメッセージを待ち続けることはできない．

ことから導かれた.

それではすべての故障プロセッサは最初から故障していると仮定できる場合はどうであろうか. この場合にも, 正常プロセッサは故障プロセッサが故障しているのかそれとも通信遅延が大きいためにメッセージが受信できないのか決定できない状態のままで合意値を決定しなければならない. それゆえ, 故障プロセッサは最初から故障しているという仮定を追加しても状況はほとんど変化しないと思われるかもしれない. しかし, この場合には, $2t<n$ という条件の下で t-耐故障合意アルゴリズム B が存在する. B の概略を以下に示す[29].

$d=n-t$ とする. 各プロセッサ P は P をすべてのプロセッサに放送し, $d-1$ 個のメッセージを受信するまで待つ. (故障プロセッサ数は最大 t であるから, 少なくとも $d-1$ 個のメッセージを有限時間内に受信できる.) その結果, P は P を含めて少なくとも d 個の正常プロセッサを発見できる. 続いて, 各プロセッサ P は P がその時点で知っている正常プロセッサのリストをすべてのプロセッサに放送し, 少なくても正常であると知っているすべてのプロセッサからメッセージが到着するのを待つ. このステップを繰り返すことにより, P を含めて P がその時点で知っているすべての正常プロセッサ P' について, P' が知っている正常プロセッサの集合を一致させることができる. この正常プロセッサの集合を Δ で表す. プロセッサ $P_i \in \Delta$ のもつ初期データ $v(j)$ の最大値を Max とするとき, Max を合意値 $w(j)$ とする. $|\Delta| \geq d$ であるから, あるプロセッサの集合 $\Delta'(\Delta \cap \Delta' = \phi, |\Delta'| \geq d)$ が存在して, Max' $(\neq Max)$ を合意値とすることはない.

4. 同期ネットワークに対するビザンティン合意アルゴリズム

本章では, 同期ネットワークを検討する. 直観的に明らかなように, 過半数のプロセッサがビザンティン故障しているときには, 合意など取れるわけがないので, どのような故障にも耐える合意アルゴリズムがあるわけではない.

定理 2 n を分散ネットワークに属するプロセッサ数とする. t-耐ビザンティン故障合意アルゴリズムが存在するならば $3t<n$ である[17),18),35),40].

証明 例1に示したように, $n=3$, $t=1$ の場合には合意アルゴリズムは存在しない. この事実を一般の場合に拡張する. $3t \geq n$ であるようなある t に対して, t-耐ビザンティン故障アルゴリズム A が存在すると仮定する.

分散ネットワーク Σ に属する n 個のプロセッサを三つの集合 X, Y, Z に分割し各集合のサイズが最大 t であるようにする. 最大 t 個のプロセッサが故障するから, ある一つの集合, たとえば X に属するすべてのプロセッサが故障プロセッサであり得る. Σ の一つの集合に属するプロセッサを同一視して得られる分散ネットワークを Σ' とする. Σ' は三つのプロセッサから構成され, 各プロセッサは最大 t 個の Σ のプロセッサを代表している. A の Σ に対する振る舞いは Σ' 上でシミュレートできる. すなわち, A から Σ' に対する 1-耐ビザンティン故障アルゴリズム A' が構成できる. しかし, これは矛盾である. □

一方, 条件 $3t<n$ が t-耐ビザティン故障合意アルゴリズムが存在するための十分条件であることも示されているが[17),18),35),40], ここでは文献35)で提案された $t+1$ ラウンドで合意を達成するアルゴリズム LSP を 2), 6) にしたがって説明する.

同期ネットワークを対象としているから, あるプロセッサから (アルゴリズムから) 予想されるメッセージが予想されるラウンドに受信できない場合には, そのプロセッサは故障プロセッサであると判断できる. 一度故障プロセッサであると認定できると, そのプロセッサの振る舞いは無視して合意アルゴリズムを続行することによって, 故障プロセッサ数を実質的に1個減少させることができ, 問題が容易になる. したがって, 以下では, すべてのプロセッサは少なくともアルゴリズムが予想するラウンドにメッセージを送受信すると仮定する.

LSP は, 再帰的な多数決過程であり, 各プロセッサが高さ $t+1$ の根付き木 T を構成する手続きとみなすことができる. T の根は空系列 λ によって識別される. T の葉は, 同じプロセッサが2回以上出現しない長さ $t+1$ のプロセッサ列によって識別され, また, そのようなプロセッサ列のそれぞれに対して, その列で識別される葉頂点が存在する. ある頂点 σ の子頂点は σP という

形の列によって識別される．σ の子頂点の集合を $EXT(\sigma)$ で表し，$Ext(\sigma)=\{P|\sigma P\in EXT(\sigma)\}$ とする．各頂点 σ にはある値 $Val(\sigma)$ が対応付けられる*．LSP の記述の簡明のためつぎのマクロ通信命令を用いる．

• $SEND(\sigma)$：自分自身を含むすべてのプロセッサに $Val(\sigma)$ を送信する．

• $RECV(\sigma, P)$：T の頂点 σ に子頂点 σP を作りプロセッサ P から受信した値を $Val(\sigma P)$ にセットする．

[(プロセッサ P_i 上の) アルゴリズム LSP]

```
begin
  頂点 λ を作る；
  Val(λ):=v(i); Tips:={λ};
  for r:=0 to t do {
  /* ラウンド r*/
    for σ∈Tips do SEND(σ);
    for σ∈Tips do
      /* Val(σ) の初期値の設定*/
      for P∈Ext(σ) do RECV(σ,P);
      Tips:=∪σ∈Tips EXT(σ);
  }
/* 合意値の計算（ラウンド t の後半)*/
  repeat
    Tips:={σ|σ P∈Tips};
    for σ∈Tips do
      if 多重集合 {Val(σP)|P：プロセッサ}
        の過半数を占める値 v が存在する
      then Val(σ):=v
      else Val(σ):=0;
  until Tips:={λ};
  w(i):=Val(λ);
end.
```

LSP が終了したとき，すべての正常プロセッサの構成した木において頂点 σ がもつ値 $Val(\sigma)$ が v であるとき，σ を v-共有あるいは値共有であるという．

まず，P を正常とすると P の木において $Val(\sigma)$ の初期値が v であるための必要十分条件は σP が v-共有であることであることが $|\sigma|$ に関する下降型帰納法で証明できる．さらにこの事実から，P が正常ならば，すべての σP という形の頂点が値共有であることが導かれる．したがって，P_i が正常ならば，P_i の木において $Val(\lambda)=v(i)$ であるから，各正常プロセッサは P_i の初期データ $v(i)$ を正しく獲得できる．

一方，σ を同じプロセッサが2回以上出現しないような故障プロセッサの列とすると，σ は値共有である．なぜなら，値共有でないものの中で最長のものを σ とする．$|\sigma|=t$ ならば，σP であるようなすべての P は正常プロセッサであるから，σ は値共有である．また，$|\sigma|<t$ の場合には，仮定から P が故障プロセッサであっても σP は値共有だから，σ も値共有となる．したがって，P_i が故障であっても，P_i は値共有，すなわち，各正常プロセッサ P は同一の値を $v(i)$ の値として得ることになる*．以上から，つぎの定理が成立する．

定理 3 アルゴリズム LSP は $3t<n$ の仮定の下で，$t+1$ ラウンドで合意問題を解く t-耐ビザンティン故障アルゴリズムである． □

一方，条件 $t\leq n-2$ の下で t-耐ビザンティン故障アルゴリズムの必要とするラウンド数の下限が $t+1$ であることが知られている．証明のアイデアは以下のとおりである．

$t=2$ の場合を考える．2ラウンドで合意を達成するアルゴリズム A が存在すると仮定する．A によって正常プロセッサ P が獲得できる情報はマトリックス $M[i,j]$ によって表現できる．ここに，$M[i,j]=v$ は，プロセッサ P_j を経由して得られたプロセッサ P_i の初期データ $v(i)$ が v であることを示す．M の要素がすべて0ならば，P は0を合意値とし，M の要素がすべて1ならば，P は1を合意値とする．$M_1, M_2, \cdots, M_k$ を要素がすべて0であるマトリックスで始まり，要素がすべて1であるマトリックスで終わるようなマトリックスの系列で，すべての i に対して，ある2台の正常プロセッサ P と P' が存在し，2台の故障プロセッサの振る舞いによって，P は M_i を獲得し P' は M_{i+1} を獲得する（したがって，合意値は一致している）ようなものが構成できる．（直観的にも，マトリックスのある0要素を1に変えるには2台の故障プロセッサで十分．）これは矛盾である．以上の議論が一般の t にたいしても適用できる．

* $\sigma=P_{i_1}P_{i_2}\cdots P_{i_k}$ とするとき，$Val(\sigma)=v$ は，"私の初期データは v である，と P_{i_1} が言っている，と P_{i_2} が言っている，…，と P_{i_k} が言っている."，と解釈される．

* 将軍問題および相互無矛盾問題はこの地点ですでに解決されている．

定理 4 $t \leq n-2$ とする．任意の t-耐ビザンティン故障アルゴリズムの必要とするラウンド数は少なくとも $t+1$ である[27]．□

したがって，*LSP* は耐故障度およびラウンド数の2点で最適なアルゴリズムである．しかし，*LSP* は望まれる性質のすべてを満足しているわけではない．特に，以下の3点は重要である．

1. プロセッサ数 n および故障プロセッサ数 t の多項式でメッセージ（ビット）複雑度がおさえられる合意アルゴリズムは存在するか？
2. 実際に故障しているプロセッサ数 f が少ないときには早く終了するような合意アルゴリズムは存在するか？
3. 適当な機能を分散ネットワークにもたせることにより，故障プロセッサ数が多くても正しく動く合意アルゴリズムを構成できないか？

- **多項式アルゴリズム**

LSP は $O(n^t)$ 個のメッセージを用いて $t+1$ ラウンドでビザンティン合意問題を解決するので多項式アルゴリズムではない．Dolev と Strong[23] は $O(n^5)$ 個のメッセージを用いて $4t+4$ ラウンドでビザンティン合意問題を解決するアルゴリズムを始めて示した．最適なラウンド数 $t+1$ ラウンドで，しかも多項式個のメッセージ数でビザンティン合意問題を解決するアルゴリズムが存在するかどうかはビザンティン合意問題が検討され始めたころからの未解決問題であったが，Moses と Waarts[39] はこの問題を肯定的に解決した．彼らのアルゴリズムは *LSP* の改良とみなされる．メッセージ数を減少させる手法として，つぎの二つの方法が用いられている．

ある正常プロセッサ P が *LSP* で用いられる木で頂点 σQ の最終値 $Val(\sigma Q)$ を計算する場合を考えよう．もちろん P は Q が正常かどうか一般には分からない．しかし，σQ の子頂点の値が分かるときには $Val(\sigma Q)$ の最終値を計算できることがある．たとえば，（過半数$+t$）個以上の子頂点が同じ値を（*Val* の初期値として）もつときには，$Val(\sigma Q)$ も同じ値を最終値としてもつ．このようなばあいには，σQ の孫頂点以下の木の展開を省略できる．この手法を**早期停止** (*Early Stopping*) という．つぎに，仮にあるプロセッサ P が故障であることが分かっているとする．このとき他のプロセッサは P がある決められた振る舞いをするかのように考え，実際の P からのメッセージを無視することによって，メッセージ数を節約できる．この手法を**故障マスク** (*fault masking*) という．故障プロセッサはたとえばつぎのようにして発見できる．頂点 σP の子頂点のうち，$t+1$ 個が0を他の $t+1$ 個が1を *Val* の初期値とするならば P は故障である．

子頂点の初期値が決まっているにもかかわらず自分自身ならびにどの祖先の最終値も決定していない頂点を**堕落** (corrupted) しているということにする．故障マスクを用いることで，堕落頂点数を木の1レベルに最大1個しか現れないようにすることができるがこれでは十分でない．Moses と Waarts は，**等位訪問** (coordinated traverse) と呼ばれる手法を開発して堕落頂点数が木全体で最大1個となるようにした．

現在までに得られている主な多項式アルゴリズムを**表-1** に示す．文献 14) を参考に作成した．ただし，$o(f(t))$ は $\lim_{t\to\infty} g(t)/f(t)=0$ となるような関数 $g(t)$ の集合を示し，$\Omega(f(t))$ はある正の定数 c と t_0 が存在し任意の $t>t_0$ に対して $cf(t) \leq g(t)$ となるような関数 $g(t)$ の集合を表す．また，$\varepsilon>0$ は任意に小さい数である．）

- **早期停止アルゴリズム**

LSP は実際の故障プロセッサ数 f に依存せず，常に $t+1$ ラウンドを必要とする．*LSP* に，Moses と Waarts[39] のように，早期停止の考えを取り入れるとアルゴリズムを $\min\{t+1, f+3\}$ ラウンドで終了させることができる．このように，f が小さい場合には早く終了できるようなアルゴリズムを早期停止アルゴリズムという．Dolev ほか[22]は早期停止の概念を導入し，$\min\{2t+3, 2f+5\}$ ラウンドで合意を達成するアルゴリズムを示した．さらに，どのようなアルゴリズムも少な

表-1 多項式メッセージ複雑度合意アルゴリズムの比較

アルゴリズム	プロセッサ数	ラウンド数	ビット数
下限値[21),27),28),40)]	$3t+1$	$t+1$	$\Omega(t^2)$
Berman 他[7]	$3t+1$	$t+o(t)$	$O(t^3)$
Corn と Welch[14]	$3t+1$	$t+o(t)$	$O(t^{t+4})$
Berman と Garay[8]	$4t+1$	$(1+\varepsilon)(t+1)$	$O(t^3)$
Moses と Waarts[39]	$8t+1$	$t+1$	$O(t^6)$
Moses と Waarts[39]	$6t+1$	$t+1$	$O(t^5)$
Srikanth と Toueg[40]	$3t+1$	$2t+1$	$O(t^3 \log t)$
Coan[13]	$3t+1$	$(1+\varepsilon)(t+1)$	$O(t^{3+1/\varepsilon})$
Dolev 他[19]	$3t+1$	$2t+3$	$O(t^3 \log t)$

くても $f+2$ ラウンド必要とすること，$n>2t^2+3t+4$ という仮定の下で $f+2$ ラウンドで終了するアルゴリズムを示している*.

故障を停止故障に限ると $f+2$ ラウンドで終了するアルゴリズムが存在する．この問題を解決する一つの方法は，**巡回調停者** (*rotating coordinator*) と呼ばれる手法を用いることである[12),43)].

議論の簡単のため，将軍問題を考える．n 台のプロセッサの中から放送プロセッサ P を含む $t+1$ 台のプロセッサを前もって選び出し，P を先頭に適当に順序づける．また，すべてのプロセッサにこの順序を知らせておく．これら選ばれたプロセッサはこの順序で調停者になる．ある時点で放送される値を決定できていないプロセッサはそのときの調停者に値（ラウンド1では P によって放送される情報，他の場合（P が故障している場合）には正常プロセッサが合意する値），の送信を要請する．一方，調停者は，値の送信要求を受けたときには，まず値を送信し，続いて送信が終了したことを示すメッセージを送信する．明らかに，正常プロセッサが調停者になればアルゴリズムは終了するので，$3f+3$ ラウンドで将軍問題を解決できる．しかし，さらに調停者との通信をパイプライン的に重ね合わせて実行する**パイプライン化** (*pipelining*) を用いることによりラウンド数を $f+2$ に減少できる[11)].

- **暗号機能付ネットワーク**

通常の分散ネットワークでは t-耐故障アルゴリズムが存在するための必要条件は $n>3t$ であった．しかし，文献 16) や 44) で検討されているような**ディジタル署名** (*digital signature*) 機能を分散ネットワークが有すると仮定すると，この下限を改良できる．このような署名機能を利用するアルゴリズムを**認証** (*authenticated*) アルゴリズムと呼ぶ．署名されたメッセージは，そのメッセージが運ばれてきた経路に係わらず信頼できる．すなわち，プロセッサ P_3 がプロセッサ P_2 を経由してプロセッサ P_1 の署名のあるメッセージ M を受信したとすると，P_2 が故障プロセッサであっても，P_1 が M を送信したことを結論できる**.

* 一般の場合に，$f+2$ ラウンドで終了するアルゴリズムは知られていない（と著者は信じている）.

** したがって，P_2 は P_1 が M' を送信したように見せかけたり，M が P_1 以外のプロセッサによって送信されたように見せかけたりすることはできない.

定理 5　任意の t に対して，t-耐ビザンティン故障認証合意アルゴリズムが存在する[35),40)]. □

t-耐ビザンティン故障認証合意アルゴリズムは LSP に以下のような変更を加えることで構成できる．メッセージ M が転送されるときにはいつでもそのメッセージを転送するプロセッサ名を署名して転送するように変更を加える．したがって，頂点 σ の値は同じ署名を2回以上含まない長さ $|\sigma|$ の署名列をもつはずである．このようなメッセージを正しいメッセージと呼ぶ．木が構成されたとき，$Val(P_i)$ の最終値はつぎのように決定する．V_i を署名列 $P_i\sigma$ をもつ正しいメッセージによって運ばれた値の集合とする．$V_i=\{v\}$ ならば $Val(P_i)=v$. $|V_i|\geq 2$ ならば，P_i が故障であることを意味するある前もって決められたある値 *Traitor*. このアルゴリズムが任意の t に対して t-耐故障であることの証明は省略する．

このアルゴリズムがビザンティン合意に関する最初の論文に含まれていたことからも分かるように，認証アルゴリズムは初期の段階から活発に議論されてきた，たとえば文献 24) には t-耐故障アルゴリズムは少なくても $t+1$ ラウンドを必要とすること，また $t+1$ ラウンドと $O(nt)$ 個のメッセージ交換を必要とする t-耐故障認証将軍アルゴリズムが存在することが示されている．

5. ランダム化アルゴリズム

3. で述べたように，非同期ネットワークに対しては 1-耐停止故障合意アルゴリズムですら存在しない．このような困難な状況を克服するためにランダム化アルゴリズムが考案されている．ランダム化アルゴリズムと通常の（決定性）アルゴリズムとの相違は，ランダム化アルゴリズムでは**コイントス** (*coin toss*) と呼ばれる0か1のどちらかをランダムに返す関数を利用できることにある．もちろん，同期ネットワークに対してもランダム化アルゴリズムを考えることはできる[9)]が，詳細は省略する．

コイントスにはつぎの二つの種類がある．まず，プロセッサ個々でコイントスが行われ，その結果としてコイントス値はプロセッサに局所的な値となるいわば**局所的コイントス** (*local coin toss*) と呼ばれるべきものがある．

第2には Shamir のアルゴリズム[45)]を用いて実

現される**大局的コイントス** (*global coin toss*) と呼ばれるもので，暗号機能付き分散ネットワークを仮定している*．この方式では，ディーラと呼ばれる特別な正常プロセスを仮定する必要がある．$k=t+1$ とする．ディーラは，コイントス値となる2値ランダム列 $s_1, s_2, \cdots$ を前もって生成し，この列を暗号化し，すべての s_j に対してこの値を復元するための情報のかけら (*piece of secret*) s_{ij} を前もってすべてのプロセス P_i に送信しておく．重要な点は，どの k 個の情報 s_{ij} の組合せからでも s_j が復元でき，$k-1$ 個以下の情報のどのような組合せからも s_j が復元できないように s_{ij} が作られていることである．すなわち，正常プロセッサが k 台以上集まると情報 s_{ij} を交換することによって共通のランダム値 s_j を手に入れることができる．また，故障プロセッサは正常プロセッサの助けなしにはランダム値 s_j を知ることができない．局所的コイントスを用いるアルゴリズムには文献 4), 8) などがあり，大局的コイントスを用いるものには，41), 42), 47) などがある．

ランダム化アルゴリズムは以下のアイデアに基づいている．説明を簡単にするために，初期値 $v(i)\in\{0,1\}$ を仮定する．最初，各プロセッサは他のすべてのプロセッサに初期値 $v(i)$ を送信する．各プロセッサは，最大でも t 台のプロセッサしか故障しないので $n-t$ 台のプロセッサからのメッセージを受信を完了するまで待機できる．この $n-t$ 個のメッセージの中に最大 t 個の故障プロセッサからのメッセージが含まれている可能性があることに注意しよう．したがって，すべての正常プロセッサが同じ初期値 v をもつならば各正常プロセッサは少なくても $n-2t$ 台のプロセッサから値 v を受信する．$(n+t)/2=(n-t)/(2+t)$ 台以上のプロセッサから値 v を受信したときには $v'(i)=v$，それ以外のときには $v'(i)$ は未定義と定めると，

1. ある正常プロセッサ P_i において $v'(i)=v$ なら正常プロセッサの過半数が初期値 $v(i)=v$ をもつことになるから任意の正常プロセッサ P_j において $v'(j)=v'$ なら $v=v'$ である．

また，

2. 故障プロセッサ数が少ない場合，たとえば，$n>5t$ である場合を考えることにすると，$(n+t)/2<n-2t$ であるから，すべての正常プロセッサが同じ初期値 v をもつならば任意の正常プロセッサ P_i において $v'(i)=v$.

ある正常プロセッサ P_i において $v'(i)=v$ とする．このとき，$v'(j)$ が未定義である正常プロセッサ P_j が値 v を $v'(j)$ の値とすることができれば v を合意値として合意が達成できる．

そこで，つぎの合意手続きを考える．$v'(j)$ が未定義ならば（局所的あるいは大局的）コイントス値を $v'(j)$ として合意が達成されたかいなかを判定する．その結果，合意が達成されていなければ各プロセッサは $v'(i)$ を新しい初期値 $v(i)$ としてこの手続きを繰り返す．

局所的コイントスが可能な場合には合意を達成できるまでに必要となる繰り返しの期待値は $\Omega(2^n)$ である．大局的コイントスが可能な場合には，すべての正常プロセッサは同一のコイントス値 c を獲得できる．各繰り返しで合意が達成できる確率は0.5であるから，合意を達成できるまでに必要となる繰り返しの期待値は n や t に依存しない定数で押さえられる．

最後に，$v'(j)$ が未定義な場合の値として（故障プロセッサが推測できない）コイントス値を使わなければならない理由について触れておく．たとえば，第 r 回目の繰り返しで $v'(j)$ が未定義な場合には z_r をその値として用いると決めたとする．このとき，第 r 回目の繰り返しにおいて，ある正常プロセッサ P_i が $v'(i)=v(\neq z_r)$ となるように故障プロセッサ群は振る舞うことができ，確率1で合意が達成できない．

定理 6 t-耐ビザンティン故障ランダム化アルゴリズムが存在するための必要十分条件は $n>3t$ である．大局的コイントスが可能ならば，ステップ数の期待値を定数に押さえることができる[4),8),10),42),47)]. □

6. 応　用

従来，さまざまな分散問題が議論されてきた．そのそれぞれに対してビザンティン故障問題が考えられ，また多くの（合意問題以外の）ビザンティン問題がすでに議論されてきた．多くの場合，ビザンティン合意問題に対する定理の数々は直接

* 暗号機能を仮定しない大局的コイントスの実現方法も検討されている[33)].

適用できないが，ビザンティン合意問題に関して考案された手法の多くは適用できる．（このために，本稿ではいくつかの基本的な手法を具体的に解説してきた．）

さまざまな環境においてすべてのプロセッサの局所時計をある誤差以内の正確さで合わせる**時計合わせ問題** (*clock synchronization problem*) は代表的な例で，文献 20), 32), 34) を初め多くの研究がされている．分散ネットワークに属するプロセッサの中からある1台のプロセッサを選び出す**リーダ選挙問題** (*leader election problem*) やリーダ選挙問題を解決するために用いられリーダ選挙問題以外にも多くの応用がある**最小生成木構成問題** (*minimum spanning construction problem*) は 3), 36), 37), 38) などで検討されている．

最後になったが，本稿について有益なご助言をいただいた学会誌編集委員会の方々に謝意を表する．なお，ビザンティン合意問題の初期の成果については 26) に解説があり，参考にした．

参 考 文 献

1) Attiya, C., Dolev, D. and Gil, J.: Asynchronous Byzantine Consensus, *Proc. 3rd ACM Symposium on Distributed Computing*, pp. 119-132 (1984).
2) Bar-Noy, A., Dolev, D., Dwork, C and Strong, H. R.: Shifting Gears: Changing Algorithms on the Fly to Expedite Byzantine Agreement, *Proc. 6th ACM Symposium on Principles of Distributed Computing*, pp. 42-51 (1987).
3) Bar-Yehuda, R., Kutten, S., Wolfstahl, Y. and Zaks, S.: Making Distributed Spanning Tree Algorithms Fault-Resilient, *Proc. 4th Symposium on Theoretical Aspects of* Computer Science, Lecture Notes in Computer Science 247, pp. 432-444 (1987).
4) Ben-Or, M.: Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols, *Proc. 2nd ACM Symposium on Principles of Distributed Computing*, pp. 27-30 (1983).
5) Berman, P. and Garay, J. A.: Asymptotically Optimal Distributed Consensus, *Proc. ICALP 89, Lecture Notes in Computer Science 372*, pp. 80-94 (1989).
6) Berman, P., Garay, J. A. and Perry, K. J.: Towards Optimal Distributed Consensus, *Pcoc. 30th IEEE Symposium on Foundations of Computer Science*, pp. 410-415 (1989).
7) Berman, P., Garay, J. A. and Perry, K. J.: Recurseive Phase King Protocols for Distributed Consensus, The Pennsylvania State University, Computer Science Dept., Tech. Report CS-89-24 (1989).
8) Bracha, G.: An Asynchronous $[(n-1)/3]$-Resilient Consensus Protocol, *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, pp. 154-162 (1984).
9) Bracha, G.: An O (log n) Expected Rounds Randomized Byzantine Generals Protocol, *Proc. 17th ACM Symposium on Theory of Computing*, pp. 316-326 (1985).
10) Bracha, G. and Toueg, S.: Resilient Consensus Protocols, *Proc. 2nd ACM Symposium on Distributed Computing*, pp. 12-26 (1983).
11) Chandra, T. and Toueg, S.: Time and Message Efficient Reliable Broadcadcasts, *4th International Workshop on Distributed Algorithms*, Buri, Italy (Sep. 1990).
12) Chang, J. and Maxemchuk, N.: Reliable Broadcast Protocols, *ACM Transactions on Computer Systems 2, 3*, pp. 251-273 (1984).
13) Coan, B. A.: A Communication-Efficient Canonical Form for Fault-Tolerant Distributed Protocols, *Proc. 5th ACM Symosium on Principles of Distributed Computing*, pp. 63-72 (1986).
14) Coan, B. A. and Welch, J. L.: Modular Construction of Nearly Optimal Byzantine Agreement Protocols, *Proc. 8th ACM Symposium on Principles of Distributed Computing*, pp. 295-305 (1989).
15) Chor, B. and Coan, B.: A Simple and Efficient Randomized Byzantine Agreement Algorithm, *IEEE Transaction on Software Engineering SE-11, 6*, pp. 531-539 (1985).
16) Diffie, W. and Hellman, M.: New Directions in Cryptography, *IEEE Transaction on Information Theory IT-22*, pp. 644-654 (1976).
17) Dolev, D.: Unanimity in an Unknown and Unreliable Environment, *Proc. 22nd IEEE Symposium on Foundations of Computer Science*, pp. 159-168 (1981).
18) Dolev, D.: The Byzantine Generals Strike Again, *J. Algorithms 3*, 1, pp. 14-30 (1982).
19) Dolev, D., Fisher, M. J., Fowler, R. J., Lynch, N. A. and Strong, H. R.: An Efficient Algorithm for Byzantine Agreement without Authentication, *Information and Control 52*, pp. 257-274 (1982).
20) Dolev, D., Halpern, J. and Strong, H. R.: On the Possibility and Impossibility of Achieving Clock Synchornization, *Proc. 16th ACM Symposium on Theory of Computing*, pp. 504-511 (1984).
21) Dolev, D. and Reischuk, R.: Bounds on Information Exchange for Byzantine Agreement, *J. ACM 32*, 1, pp. 191-204 (1985).
22) Dolev, D., Reischuk, R. and Strong, H.R.:

'Eventual' is earlier than 'Immediate', *Proc. 23rd IEEE Symposium on Foundations of Computer Science*, pp. 196-203 (1982).
23) Dolev, D. and Strong, H. R.: Polynomial Algorithms for Multiple Processor Agreement, *Proc. 14th ACM Symposium on Theory of Computing*, pp. 401-407 (1982).
24) Dolev, D. and Strong, H. R.: Authenticated Algorithms for Byzantine Agreement, *SIAM J. Comput. 12*, 4, pp. 656-666 (1983).
25) Feldman, P.: Optimal Byzantine Agreement, Ph. D. thesis, Department of Mathematics, Massachusetts Institute of Technology (1988).
26) Fisher, M. J.: The Consensus Problem in Unreliable Distributed Systems (A Brief Survey), *Proc. International Foundation of Computation Theory Conference*, Borgholm, Sweden, pp. 128-140 (Aug. 1983).
27) Fisher, M. J. and Lynch, N. A.: A Lower Bund for the Time to Assure Interactive Consistency, *Inf. Proc. Lett. 14*, 4, pp. 183-186 (1982).
28) Fisher, M. J., Lynch, N. A. and Merritt, M.: Easy Impossibility Proofs for Distributed Consensus Problems, *Distributed Computing 1*, 1 pp. 26-39 (1986).
29) Fisher, M. J., Lynch, N. A. and Paterson, M. S.: Impossibility of Distributed Consensus with One Faulty Prosess, *J. ACM 32*, 2, pp. 374-382 (1985).
30) Hadzilacos, V.: Connectivity Requirements for Byzantine Agreement under Restricted Types of Failures, *Distributed Computing 2,2*, pp. 95-103 (1987).
31) 萩原, 増澤: 分散アルゴリズム, 情報処理, Vol. 31, No. 9, pp. 1245-1256 (Sep. 1990).
32) Halpern, J., Simons, B. and Strong, H. R.: Fault-Tolerant Clock Synchronization, *Proc. 3rd ACM Symposium on Distributed Computing*, pp. 89-102 (1984).
33) Lamport, L.: The Weak Byzantine Generals Problem, *J. ACM 30*, 3, pp. 668-676 (1983).
34) Lamport, L. and Melliar-Smith, P. M.: Byzantine Clock Synchronization, *Proc. 3rd ACM Symposium on Distributed Computing*, pp. 68-74 (1984).
35) Lamport, L., Shostak, R. E. and Pease, M.: The Byzantine Generals Problem, *ACM Transactions on Programming Language and Systems 4*, 3, pp. 382-401 (1982).
36) Merritt, M.: Elections in the Presence of Fault, *Proc. 3rd ACM Symposium on Distributed Computing*, pp. 134-142 (1984).
37) 増澤, 萩原, 都倉: 辺故障を考慮したある分散アルゴリズム, 電子通信学会論文誌 '86/10, Vol. J 69-D, No. 10, pp. 1394-1405 (1986).
38) Masuzawa, T., Nishikawa, N., Hagihara, K. and Tokura, N.: Optimal Fault-Tolerant Distributed Algorithms for Election in Complete Networks with a Global Sense of Direction, *Proc. 3rd International Workshop on Distributed Algorithms, Lecture Notes in Computer Science 392*, pp. 171-182 (1988).
39) Moses, Y. and Waarts, O.: Coordinated Traversal: $(t+1)$-Round Byzantine Agreement in Polynomial Time, *Proc. 29th IEEE Symposium on Foundations of Computer Science*, pp. 246-255 (1988).
40) Pease, M., Shostak, R. M. and Lamport, L.: Reaching Agreement in the Presence of Faults, *J. ACM 27*, 2, pp. 228-234 (1980).
41) Perry, K. J.: Randomized Byzantine Agreements, *IEEE Transaction on Software Engineering SE-11*, 6, pp. 539-546 (1985).
42) Rabin, M. O.: Randomized Byzantine Generals, *Proc. 24th IEEE Symposium on Foundations of Computer Science*, pp. 403-409 (1983).
43) Reischuk, R. K.: A New Solution for the Byzantine Generals Problom, *Proc. 1983 International Foundation of Computation Theory Conference*, pp. 382-393 (1983).
44) Rivest, R., Shamir, A. and Adleman, L.: A Method for Obtaining Sigital Signatures and Public-Key Cryptosystems, *Commu. ACM 21*, 2, pp. 120-126 (1978).
45) Shamir, A.: How to Share a Secret, *Commu. ACM 22*, pp. 612-613 (1979).
46) Srikanth, T. K. and Toueg, S.: Simulating Authenticated Broadcasts to Derive Simple Fault-Tolerant Algorithms, *Distributed Computing 2*, 2, pp. 80-94 (1987).
47) Toueg, S.: Randomized Byzantine Agreements, *Pcoc. 3rd ACM Symposium on Principles of Distributed Computing*, pp. 163-178 (1984).

(平成2年12年12日受付)

山下 雅史 (正会員)

昭和 49 年京都大学工学部情報卒業. 昭和 52 年同大大学院修士課程修了. 昭和 55 年名古屋大学大学院博士課程修了. 工学博士. 豊橋技術科学大学助手を経て, 現在, 広島大学工学部第二類助教授. この間, 昭和61年より約1年間, カナダサイモンフレーザー大学客員教授. マルチプロセッサの負荷分散問題, 画像処理の基礎, 組合せ問題の研究に従事. 電子情報通信学会会員.