ビザンチン将軍問題におけるメッセージ内容に基づく故障の特定

藤代 康誠　長谷部 浩二

ビザンチン将軍問題は，故障プロセスに影響されることなく，すべての正常プロセスが妥当かつ一致した判断をする問題である．通常，ビザンチン将軍問題では故障プロセスの特定までは要求されないが，実システムへの応用を考えると故障プロセスを可能な限り特定できることが望ましい．そこで本研究では，Lamport の提案したビザンチン将軍問題に対するアルゴリズム OM において，可能な範囲で故障を特定する方法を示す．具体的には，故障が特定された状態を，すべての正常プロセスが故障プロセスの識別子を知っていることとして定義し，故障プロセスが送信する誤ったメッセージをもとに，グラフ理論的な考察によって故障を特定するアルゴリズムを示す．さらに，故障を特定するためには，メッセージのやり取りを通して形成される正常プロセスの間の信頼関係が連結であれば十分であることを示す．

1　研究の背景と目的

分散システムは，複数の計算機上で実行される独立したプログラムが協調することで処理を実現するシステムである．独立したプログラムの実行単位をプロセスという．プロセスを協調させる上で重要な課題の一つは，故障したプロセスに影響されることなく，すべての正常なプロセスが一致した正しい判断に至ることである．この問題はビザンチン将軍問題と呼ばれ，この問題を解くさまざまなアルゴリズムが提案されている．

通常，ビザンチン将軍問題では故障プロセスの特定までは要求されないが，実システムへの応用を考えると，故障プロセスを可能な限り特定できることが望ましい．その理由の一つとして，ビザンチン将軍問題を解くアルゴリズムには，署名を用いない場合，耐えることのできる故障プロセスの数に限界があることが挙げられる．処理の過程で故障プロセスを特定し，取り除くことができれば，より確実に合意を達成することができる．さらに，故障プロセスは，システムの協調を妨げるために悪意を持って実行されるプログラムであると考えることもできる．したがって，故障プロセスの特定はセキュリティの観点からも重要な課題といえる．

そこで本研究では，Lamport が提案したビザンチン将軍問題に対するアルゴリズム OM において，故障プロセスが送信する誤った内容のメッセージをもとに故障を特定する方法を示す．メッセージを特定の手がかりとして用いる理由は，最も厄介な故障，すなわちシステムの協調を妨げつつ正常なプロセスに特定されないよう努める故障に対しても利用できる情報であると考えられるためである．

本研究の手法で重要な役割を果たしている概念は信頼関係である．プロセス P_i が，これまでに受信したメッセージの内容から，プロセス P_j は確実に正常であると結論できるとき，P_i は P_j を信頼しているという．プロセスは，アルゴリズム OM において想定した内容と矛盾したメッセージを得たとき，それをも

---

* Fault-Detection with Content of Messages in Byzantine Generals Problem.

Kosei Fujishiro, 筑波大学情報理工学位プログラム, Master's Program in Computer Science, University of Tsukuba.
Koji Hasebe, 筑波大学システム情報系, Faculty of Engineering, Information and Systems, University of Tsukuba.

とに故障の可能性があるプロセスを限定し，それ以外のプロセスを信頼することができる．もしこの作業によってすべての正常なプロセスを信頼することができたなら，そのプロセスは故障を特定したことになる．さらに，この作業で故障が特定できなかったとしても，信頼しているプロセスからは，誤った情報を送られる心配なく情報を聞き出すことができる．したがって，信頼関係のあるプロセスどうしで情報を共有することで故障が特定できる場合がある．そこで本研究の手法は，まず各プロセスが自分の受信したメッセージをもとに信頼関係を形成し，次にその信頼関係に基づいて情報を共有することで故障を特定する，という 2 段階のアプローチをとる．この手法を用いることで，メッセージをもとに正常プロセスの間で形成される信頼関係が連結であれば，プロセスどうしの情報共有によって故障が特定されることを証明する．

本稿の構成は次の通りである．まず，第 2 章で関連研究について述べる．第 3 章では，本研究で扱う分散システムのモデルについて述べる．第 4 章では，ビザンチン将軍問題の定義とそれに対するアルゴリズム OM について述べる．第 5 章では，アルゴリズム OM において送信されるメッセージの内容をもとに故障を特定する方法を示す．最後に，第 6 章で結論と今後の課題を述べる．

2 関連研究

ビザンチン将軍問題とそれに対するアルゴリズム OM は Lamport によって 1982 年に提案された [4]．これをきっかけにビザンチン故障に耐性のあるアルゴリズムの研究が活発に行われるようになり，実用的なアルゴリズムも提案されている [1]．このような従来のアルゴリズムは，ビザンチン故障の影響を覆い隠せるものの，故障の特定まではしない．この点に関連して Yumerefendi らの論文 [6] は，信頼性が求められる分散システムの設計目標として説明責任と呼ばれる要件を最優先事項にすべきであるとしている．説明責任とは，システムの参加者の行動が証明可能であり，その行動を第三者が検証できるような仕組みのことである．Haeberlen らの論文 [3] では，説明責任を満たすシステムと，そのシステムにおいて故障を特定する手法が提案されている．この手法では，各ノードがメッセージの送受信やアプリケーションの入出力に関する偽装できないログを記録する必要がある．これに対し，本研究ではそのようなログを必要としない．

その他に，本研究と同様にメッセージの内容に基づく手法を取る研究としては，クラッシュ故障の特定 [2] や，フェイクニュースの発信源の特定 [5] などが挙げられる．分散システムの一般的な内容，特に第 3,4 章で述べるシステムのモデルとビザンチン故障問題の定式化は米田らの文献 [7] に基づいている．

3 対象とするシステムのモデル

本研究で扱う分散システムのモデルは，ネットワークを介して互いにメッセージを送受信することで通信するプロセスの集合によって構成される．各プロセスは固有の識別子を持ち，それらは $1, \ldots, n$ で表される．ここで，n はプロセスの総数である．さらに，識別子 i を持つプロセスを P_i で表す．

分散システムは，プロセスの処理速度，メッセージ遅延，プロセスが持つ局所時計と実時刻の同期の程度によって，同期システムと非同期システムに分けられる．同期システムでは，プロセスの処理速度とメッセージ遅延に上限が存在し，プロセスの局所時計と実時刻との間に一定の関係が成り立つ．本研究では，同期システムに基づく同期ラウンドモデルを採用する．同期ラウンドモデルでは，プロセスはラウンドと呼ばれる単位で処理を行い，ラウンドは全プロセスで同期して実行される．具体的には，プロセスは各ラウンドのはじめに直前のラウンドで送信されたメッセージをすべて受信し，その内容をもとに現在のラウンドの処理とメッセージの送信を行う．

ビザンチン将軍問題をはじめとする合意問題では，プロセスの故障を想定する．そこで，プロセスの故障時の動作についても規定しておく必要がある．故障のモデルにはいくつか種類があるが，本研究ではビザンチン故障を仮定する．ビザンチン故障は，故障プロセスの動作に仮定を置かない故障のモデルである．そのため，故障プロセスは処理の途中で動作を停止したり，正しい処理を行った場合とは異なる内容のメッセージを送信するなど，あらゆる動作を起こす可能

性がある．本研究では，故障プロセスが送信する誤った内容のメッセージをもとに，正常プロセスが故障プロセスを特定する方法を示す．

その他に，メッセージに関して以下を仮定する．

- 送信されたメッセージは必ず正しく届く．
- メッセージの受信者はそのメッセージの送信者がわかる．
- どのプロセスも他のすべてのプロセスに直接メッセージを送ることができる．

1 つ目と 2 つ目の仮定は，プロセスが利用する通信ネットワークの信頼性が十分に高く，いったん送信されたメッセージがネットワークの障害によって失われたり改ざんされたりしないことを表す．3 つ目の仮定は，プロセスの間の通信ネットワークが完全グラフであることを表す．

4 ビザンチン将軍問題の概要

ビザンチン将軍問題は，故障プロセスに影響されることなく，すべての正常なプロセスが正しい判断で一致する問題である．正確な定義を以下に示す．

定義 1 (ビザンチン将軍問題)．発信者と呼ばれる特別なプロセス P_O がある値 v_0 を保持しているとする．P_O 以外のすべてのプロセス P_i は以下の条件を満たすように変数 w_i に値をセットせよ．

- P_O 以外の正常なプロセス P_i はいずれ必ず w_i の値をセットする．(停止性)
- P_O 以外の正常なプロセス P_i, P_j が w_i, w_j に値をセットした場合，セットされた値は同じである．(合意)
- P_O が正常ならば，P_O 以外の正常なプロセスが w_i にセットした値は v_0 である．(妥当性)

以降，プロセス P_O は値 v_0 を「提案する」といい，P_O 以外のプロセス P_i が w_i に値 v をセットすることを，P_i が v を「決定する」という．

故障プロセスの数を $k(< n)$ としたとき，$n \leq 3k$ のもとではビザンチン将軍問題を解くアルゴリズムは存在しない．一方，$n \geq 3k+1$ の場合には，この問題を解くさまざまなアルゴリズムが提案されている．本研究では，Lamport が提案したアルゴリズム $OM(m, \mathbf{P}, g, x)$ を考える．OM は，プロセスの集合 $\mathbf{P}$ に値を返す再帰的なアルゴリズムである．OM の実行開始時には，引数として，m に故障プロセスの数 k を，$\mathbf{P}$ に発信者以外のすべてのプロセスの集合を，g に発信者プロセスを，x に g が提案する値を指定する．OM の結果として各プロセスに返される値が，各プロセスの決定すべき値である．具体的なアルゴリズムを以下に示す．

アルゴリズム $OM(0, \mathbf{P}, g, x)$

1. プロセス g が値 x を $\mathbf{P}$ に含まれるすべてのプロセスに送信する．
2. 各プロセス $P_i \in \mathbf{P}$ に受信した値を返す．

アルゴリズム $OM(m, \mathbf{P}, g, x), m > 0$

1. プロセス g が値 x を $\mathbf{P}$ に含まれるすべてのプロセスに送信する．
2. 各プロセス $P_i \in \mathbf{P}$ が受信した値 y_i を用いて，$|\mathbf{P}|$ 個の $OM(m-1, \mathbf{P}-\{P_i\}, P_i, y_i)$ を実行する．
3. 各プロセス P_i は，ステップ 1 で 1 つの値を受信し，ステップ 2 で $|\mathbf{P}|-1$ 個の値を返されている．それら合計 $|\mathbf{P}|$ 個の値に過半数を占める同一の値があればその値を，なければあらかじめ決められたデフォルトの値を P_i に返す．

3 章で述べたように，ここでは同期ラウンドモデルを仮定しているため，プロセスはこのアルゴリズムが正しく実行された場合に各ラウンドで受信すべきメッセージを予想することができる．したがって，故障プロセスがあるラウンドで送信すべきメッセージを送信していない場合，受信側のプロセスはそれを検出することができる．その場合は，あらかじめ決められたデフォルトのメッセージを受信したと見なす．この考え方を採用することで，故障プロセス P_i の動作を，アルゴリズム $OM(m, \mathbf{P}, g, x)$ のステップ 1 において x とは異なる値 y を送信する，というものに限定しても一般性を失わない．

5 故障の特定

本章では，アルゴリズム OM において送受信されるメッセージをもとに故障を特定する具体的な方法を示す．はじめに故障プロセス数 k が 1 の場合の方法を示し，次にそれを一般化したアルゴリズムを示す．プロセスの総数 n はアルゴリズム OM によってビザ

ンチン将軍問題を解くことができる数以上，すなわち $n \geq 3k+1$ を満たすと仮定する．

故障の特定に至るまでの大まかな流れは次の通りである．各プロセス P_i がそれぞれ発信者となって $OM(k, \mathbf{P}, P_i, v_i)$ を開始し，n 個のビザンチン将軍問題を同時に解くことを考える．OM の実行過程では，プロセスの間でさまざまなメッセージが送受信される．特に故障プロセスの存在下では，OM が正常に実行された場合とは整合しないメッセージの集合を 1 つのプロセスが受信することがある．そのようなメッセージの集合を受信したプロセスは，故障が疑われるプロセスの範囲を絞り込むことができる．メッセージの送受信を通して故障が疑われるプロセスが 1 つに絞られるならば，故障は特定されたことになる．

5.1 基本的な定義と仮定

はじめに，故障が特定されている状態の定義をはっきりさせておく．

定義 2. 各プロセス P_i は，正常と判断したプロセスの集合 $\mathbf{R}_i$ を保持しているとする．集合 $\mathbf{R}_i$ がすべての正常プロセスを含み，かつ故障プロセスを含まないとき，プロセス P_i は故障を特定しているという．また，すべての正常プロセスが故障を特定しているとき，故障が特定されているという．

さらに，プロセスは以下の事実をあらかじめ知っていると仮定する．

A1. システムのモデルは 3 章で述べたものである．

A2. 実行されるアルゴリズムは OM である．

A3. 故障プロセス数は k である．

今後しばらくは議論をわかりやすくするため故障プロセス数 k が 1，プロセスの総数 n が 4 の場合について説明する．

図 1 は，OM におけるメッセージの送受信関係を，OM の再起呼び出しを最初に開始する発信者ごとに示したものである．このような発信者を「おおもとの発信者」と呼ぶ．グラフの頂点はプロセスを表し，頂点の間を結ぶ矢印は，矢印が向かう方向へメッセージが送信されることを表す．太い線で描かれている頂点は，おおもとの発信者を表す．アルゴリズム OM でメッセージが送信されるのは 4 章で示したアルゴリズムにおけるステップ 1 のときであり，メッセージの内容は提案する値 x である．

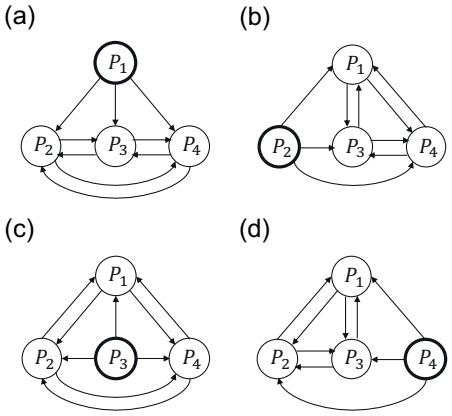

図 1 OM におけるメッセージの送受信関係

図からわかる通り，同じプロセスの間でも複数のメッセージが送信されるため，それらが図のどの矢印に相当するのかをプロセスが区別できる必要がある．そこで，OM で送信されるメッセージには，そのメッセージがこれまでにどのプロセスに受信されてきたかを表す履歴が含まれるとする．履歴はプロセスの列で表される．例えば，図 1(a) の P_2 から P_3 への矢印に相当するメッセージには，P_1 からのメッセージが P_2 を経由して送信されたことを表す列 $\langle P_1, P_2 \rangle$ が付加される．この履歴は，プロセスがメッセージを送信するタイミングで自分の識別子を履歴の末尾に追加することで成長していく．この情報を付加したとしても，故障プロセスの動作を誤った値の送信に限定してよい点は変わらない．今後，故障プロセスが誤った値を送信することを，「嘘をつく」とも表現する．

5.2 メッセージ内容に基づく故障の特定

OM の実行の中で，故障プロセスが嘘をつくタイミングは複数存在する．以下では例として，P_2 が故障プロセスである場合を考える．図 1 からわかる通り，P_2 がメッセージを送信する機会は，P_2 がおおもとの発信者である場合に 3 回，そうでない場合に 2 回ずつの計 9 回ある．そのそれぞれについて，P_2 が嘘をつくと，他のプロセスは何を知ることができるか考える．仮に，P_1 をおおもとの発信者とする OM（図

1(a)) において，P_2 が P_3 に対して嘘をついたとする．すると，P_3 は P_1 の提案値として決定すべき値の候補を 2 通り受け取ることになる．P_1 から受け取った値を v_1，P_2 から受け取った値を $v_1'(\neq v_1)$ とする．

このとき，P_3 は P_1 か P_2 のいずれかが嘘をついていることがわかる．実際，P_1 も P_2 も正常であるのに P_3 が 2 通りの値を受け取ることはない．一方で，P_1 と P_2 どちらが嘘をついたのかまではわからない．なぜなら，仮に P_2 ではなく P_1 が故障していて，P_2 には v_1' を，P_3 には v_1 を送っていたとしても，P_3 が P_1 と P_2 から受け取る値の組は P_2 が嘘をついた場合と同じであり，区別がつかないからである．

さらに故障プロセス数が 1 であるという前提知識から，P_3 は P_1 と P_2 以外のプロセス，すなわち P_4 は確実に正常であることがわかる．したがって，P_3 は正常と判断したプロセスの集合 $\mathbf{R}_3$ に P_4 を加えてよい．一般に $P_j \in \mathbf{R}_i$ のとき P_i は P_j を信頼しているといい，このことを関係 R を用いて $(P_i, P_j) \in R$ で表す．R を信頼関係と呼ぶ．

同様の議論により，P_2 がメッセージを送信する 9 回の機会すべてについて，どのタイミングで嘘をつくと，どのプロセスの間に信頼関係ができるのかを導き出すことができる．

信頼関係の言葉を用いれば，故障が特定されている状態は，すべての正常プロセス P_i, P_j の間で $(P_i, P_j) \in R$ が成り立つことといえる．これは，正常プロセスの集合を頂点とし，信頼関係 R を辺とするグラフが完全グラフであることともいえる．明らかに，P_2 が自分以外をおおもとの発信者とする OM において常に嘘をつけば，グラフは完全グラフとなり故障は特定される．

5.3 アルゴリズム TC による信頼関係の拡張

上で示した通り，故障プロセスが常に嘘をつけば故障は特定されるが，実用的な観点からは，故障プロセスが嘘をつく回数が少なくても故障が特定できることが望ましい．少ない嘘から故障を特定するためのアプローチは少なくとも 2 つある．1 つは，各プロセスが持つメッセージの集合からさらに得られる情報を考えるアプローチである．上で示した方法はプロセスが持つ情報をすべて使っているとは限らないため，より詳細な議論によって，少ない嘘からでも故障が特定できる可能性がある．もう 1 つは，プロセスどうしで何らかの情報交換をすることによって信頼関係を拡張するアプローチである．ここでは後者のアプローチをとる．

次のアルゴリズム TC は，信頼関係を安全に推移閉包に拡張する．より具体的には，どの正常プロセスも正常プロセスのみを信頼しているならば，このアルゴリズムによって推移閉包に拡張された信頼関係もその性質を保持する．そのうえ，このアルゴリズムは故障に影響されない．

アルゴリズム TC

1. 各プロセス P_i は，自分が信頼している各プロセス $P_j \in \mathbf{R}_i$ に，P_j が信頼しているプロセスの集合 $\mathbf{R}_j$ を答えるよう要求する．要求を受けた P_j は，$\mathbf{R}_j$ を P_i に送信する．
2. 直前のステップにおける要求の結果として P_i が受信したすべての $\mathbf{R}_j$ の和集合を $\mathbf{R}_i'$ とする．各プロセス P_i は，$\mathbf{R}_i' - \mathbf{R}_i = \emptyset$ ならば，このアルゴリズムを終了する．そうでなければ，各プロセス $P_j \in \mathbf{R}_i' - \mathbf{R}_i$ に $\mathbf{R}_j$ を答えるよう要求した後，$\mathbf{R}_i$ に $\mathbf{R}_i'$ の要素をすべて加える．要求を受けた P_j は，$\mathbf{R}_j$ を P_i に送信する．その後，ステップ 2 をもう一度行う．

正常プロセスが他のプロセスに情報を要求する回数は高々n 回であり，同じプロセスに 2 回以上情報を要求することはないため，正常プロセスはいつか必ずこのアルゴリズムを終了できる．アルゴリズム TC に関して次の定理が成り立つ．

定理 1. 正常なプロセスの間で形成されている信頼関係のグラフが連結であり，かつ任意の正常プロセス P_i について，$(P_i, P_j) \in R$ ならば P_j は正常であるとする．このとき，アルゴリズム TC を実行することで故障が特定される．

この定理は，故障を特定するためには正常プロセスの間で連結な信頼関係を形成できれば十分であることを示している．例えば，故障プロセスを P_2 とし，アルゴリズム OM の実行後に形成されている信頼関係を $R = \{(P_1, P_3), (P_3, P_4), (P_4, P_1)\}$ とする．この時

点では故障は特定されていない．それどころか，どのプロセスも故障を特定できていない．しかし，この信頼関係は定理 1 の仮定を満たしているため，この状態でアルゴリズム TC を実行すれば故障が特定できる．

5.4 故障を特定する一般的なアルゴリズム

この節では，前節で示した故障の特定方法を一般化し，アルゴリズムとして示す．故障を特定するアルゴリズムを DetectFault と呼ぶことにする．故障プロセス数を k，プロセスの総数を $n(\geq 3k+1)$ とし，その他の仮定は前節までと同様とする．DetectFault を実行するタイミングは，各プロセスをおおもとの発信者とする n 個の OM がすべて終了した直後である．各プロセスは OM において受信したメッセージをすべて保持しているとする．メッセージは値 v とプロセスの有限列 h の組 (v,h) で表される．v は OM における提案値であり，h はこれまでにそのメッセージを受信したプロセスの履歴である．提案値が取りうる値の集合を V とし，プロセスの有限列の集合を H とする．プロセス P_i が OM において受信したメッセージの集合を $M_i \subseteq V \times H$ で表す．DetectFault は次の 2 段階からなるアルゴリズムである．

1. 各プロセス P_i はメッセージの集合 M_i をもとにいくつかのプロセスを信頼する．
2. アルゴリズム TC によって信頼関係を拡張する．

このうち，第 2 段階のアルゴリズム TC については前節で説明した通りである．なお，定理 1 の結果は故障プロセスの数に依存しないため，k が 2 以上であっても成り立つ．以下では第 1 段階で各 P_i が実行するアルゴリズムに焦点を当てる．このアルゴリズムを FormReliance と呼ぶ．

FormReliance の処理内容を図 2 に示す．このアルゴリズムではプロセスどうしの通信は行われず，プロセスの内部で逐次的に処理が実行される．2 行目から 9 行目までの処理では，メッセージの集合 M_i を用いて故障プロセスを少なくとも 1 つ含むプロセスの集合をいくつか求め，それらを集合 $\mathcal{S} \subseteq 2^{\mathbf{P}}$ の要素として追加している．ここで，$\mathbf{P} = \{P_1, \ldots, P_n\}$ である．4 行目の集合 M_i^j は，M_i に含まれるメッセージのうち，履歴が P_j から始まるもの，すなわち P_j をおおもとの発信者とするメッセージの集合である．

```
1: procedure FormReliance
2:     S ← ∅
3:     for all j ∈ {1, ..., n} do
4:         for all (v, h), (v', h') ∈ M_i^j do
5:             if v ≠ v' then
6:                 S ← S ∪ {branch(h, h')}
7:             end if
8:         end for
9:     end for
10:    R_i ← {P_i}
11:    for all S_1, ..., S_k ∈ S do
12:        if ∀l, m ∈ {1, ..., k}, S_l ∩ S_m = ∅ then
13:            R_i ← R_i ∪ (P − ⋃_{l=1}^{k} S_l))
14:        end if
15:    end for
16: end procedure
```

図 2 アルゴリズム FormReliance

ここでの基本的な考え方は，おおもとの発信者が等しく，値が異なる 2 つのメッセージからは，故障プロセスを少なくとも 1 つ含むプロセスの集合が導ける，というものである．例えば，$(0, \langle P_1, P_2 \rangle)$ と $(1, \langle P_1 \rangle)$ という 2 つのメッセージからは，前述の議論により P_1, P_2 の少なくとも一方が故障していると結論づけられる．また，$(0, \langle P_1, P_2 \rangle)$ と $(1, \langle P_1, P_2, P_3 \rangle)$ からは，P_2, P_3 の少なくとも一方が故障していると結論づけられる．ここで P_1 が故障の候補とならない理由は，OM における 2 つのメッセージの成り立ちによる．具体的には，2 つのメッセージは，P_1 から P_2 に送信されたメッセージを P_2 が複製し，別々のプロセスに送信することで初めて履歴の異なる 2 つのメッセージとなる．したがって，2 つのメッセージの値が異なっているならば，それは履歴における P_2 以降のプロセスが嘘をついたためである．一般に OM において，おおもとの発信者が等しく，値が異なる 2 つのメッセージが得られているとき，2 つの履歴の分岐点となるプロセス以降のプロセスの集合は故障プロセスを少なくとも 1 つ含む．6 行目の関数 $\mathrm{branch} : H \times H \to 2^{\mathbf{P}}$

は 2 つのメッセージの履歴を入力として，履歴の分岐点となるプロセス以降のプロセスの集合を返す．例えば，$h = \langle P_1, P_2, P_4 \rangle, h' = \langle P_1, P_2, P_3 \rangle$ のとき $\mathrm{branch}(h, h') = \{P_2, P_3, P_4\}$ であり，$h = \langle P_1, P_2, \rangle, h' = \langle P_1, P_3 \rangle$ のとき $\mathrm{branch}(h, h') = \{P_1, P_2, P_3\}$ である．

10 行目以降の処理では，集合 $\mathcal{S}$ を用いて信頼できるプロセスを求める．はじめに 10 行目で自分自身を信頼する．11 行目以降では，$\mathcal{S}$ の要素，すなわち故障プロセスを少なくとも 1 つ含むことがわかっているプロセスの集合を k 個選ぶ．選ばれた集合を $S_1, \ldots, S_k$ とする．それらがどの 2 つも共通部分を持たないならば，前提知識 A3 により $S_1, \ldots, S_k$ はちょうど 1 つずつ故障プロセスを含む．さらに，$S_1, \ldots, S_k$ のどの 1 つにも属さないプロセスは確実に正常である．したがって，そのようなプロセスを $\mathbf{R}_i$ に追加する．これにより，すべての正常プロセスは正常プロセスだけを信頼する．

6 結論と今後の課題

本研究では，アルゴリズム OM において送受信されるメッセージをもとに故障を特定する方法を示し，それを具体的なアルゴリズムで記述した．さらに，本研究の方法で故障を特定するためには，メッセージをもとに形成される正常プロセス間の信頼関係が連結であれば十分であることを示した．本研究の方法を応用することで，一般的な分散システムのアルゴリズムにおいて故障が特定できるかどうかを検証する仕組みや，故障を可能な限り漏れなく特定するアルゴリズムの設計などが可能になると考えられる．

今後の課題としては，提案手法をより広い範囲で適用できるように改良することが挙げられる．現状では，システムが同期システムであることや，プロセスが故障プロセスの数を知っていることなどを仮定しているが，実際のシステムではこれらの仮定が成り立たない場面が多いと考えられる．したがって，より弱い仮定のもとでも手法を適用できるようにする必要がある．また，現状ではプロセスが自分の持っている情報から信頼関係を形成する際の推論については十分検討されていない．プロセスの推論方法を改良することで，同じ情報からより強い結論を導ける可能性がある．この課題に対しては，充足可能性問題による定式化や認識論理を用いた分析など数理論理学を用いた手法が有効であると考えられる．

参 考 文 献

[1] Castro, M. and Liskov, B.: Practical Byzantine Fault Tolerance, *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, USA, USENIX Association, 1999, pp. 173–186.

[2] Fujishiro, K. and Hasebe, K.: Robustness and Failure Detection in Epistemic Gossip Protocols, *ICFEM 2020*, (to appear).

[3] Haeberlen, A., Kouznetsov, P., and Druschel, P.: PeerReview: Practical accountability for distributed systems, *ACM SIGOPS operating systems review*, Vol. 41, No. 6(2007), pp. 175–188.

[4] Lamport, L., Shostak, R., and Pease, M.: The Byzantine Generals Problem, *ACM Transactions on Programming Languages and Systems*, Vol. 4, No. 3(1982), pp. 382–401.

[5] van den Berg, L.: Unreliable Gossip, Master's thesis, Universiteit van Amsterdam, 2018.

[6] Yumerefendi, A. R. and Chase, J. S.: The Role of Accountability in Dependable Distributed Systems, *Proceedings of the First Conference on Hot Topics in System Dependability*, HotDep '05, USA, USENIX Association, 2005, pp. 3.

[7] 米田友洋, 梶原誠司, 土屋達弘: ディペンダブルシステム, 共立出版, 2005.