Vol.2014-AL-150 No.10
2014/11/20

移動ビザンチン故障の移動性とビザンチン合意問題の困難さ

佐々木 徹[1,a)]　山内 由紀子[1,b)]　来嶋 秀治[1,c)]　山下 雅史[1,d)]

概要：複数のプロセスから構成される分散システム上の最も基本的な協調動作の一つに、プロセス間で合意を形成する合意問題がある。特に、一部のプロセスが最悪の行動をとると仮定するビザンチン故障を起こす場合をビザンチン合意問題と呼び、さらに、ビザンチン故障を起こすプロセスが移動する可能性がある場合を移動ビザンチン合意問題と呼ぶ。移動ビザンチン合意問題については、これまでに４つの異なる故障モデルの下で合意アルゴリズムが提案されており、それぞれ許容できる故障プロセス数も異なる。本研究では、移動時間、送信基本操作、故障感知性の３つの観点によって分類される１２個の異なる故障モデルを定式化し、これらのモデルのすべてに対して適応できる統一的な合意アルゴリズムを提案し、このアルゴリズムが許容する故障プロセス数の上下界について考察する。また、故障が移動しない場合の許容される故障プロセス数の上下界と比較することで、故障の移動が許容できる故障プロセス数に影響を与えないことを示す。

Toru Sasaki[1,a)]　Yukiko Yamauchi[1,b)]　Shuji Kijima[1,c)]　Masafumi Yamashita[1,d)]

1. はじめに

分散システムの協調動作の最も基本的な形として，各プロセス間で合意を形成する合意問題と呼ばれる問題がある．特に，一部のプロセスがアルゴリズムを無視して任意の行動をとるビザンチン故障を起こした場合の合意問題をビザンチン合意問題と呼ぶ [5], [6]．ビザンチン合意問題ではプロセスの故障は永久的に続くが，Garay[4] はビザンチン故障がプロセス間を移動する可能性がある移動ビザンチン合意問題を提案した．その後，移動ビザンチン合意問題について様々な研究が成され，故障に関する仮定の異なる様々なモデルが提案されており，また，それぞれ許容できる故障プロセス数も異なる [1], [2], [3], [7]．本研究では移動ビザンチン合意問題を移動時間、送信基本操作、故障感知性の３つの観点から分類し，３つのパラメータ γ，δ，ϵ によって定式化する．その後，これらのどのモデルに対しても移動ビザンチン合意問題を解く統一的なアルゴリズムを提案する．また，新たな合意問題である $(\gamma, \delta, \varepsilon)$-合意問題を提案し，この問題が解ける条件と比較することで，故障の移動が移動ビザンチン合意問題の困難さに影響を与えないことを示す．

[1] 九州大学大学院 システム情報科学府
福岡県福岡市西区元岡 744
[a)] toru.sasaki@inf.kyushu-u.ac.jp
[b)] yamauchi@inf.kyushu-u.ac.jp
[c)] kijima@inf.kyushu-u.ac.jp
[d)] mak@inf.kyushu-u.ac.jp

2. 移動ビザンチン故障モデル

2.1 システムモデル

n 個のプロセスによって構成される分散システムを考える．通信ネットワークは任意の２つのプロセス間に通信リンクが存在する完全ネットワークを仮定し，これらのプロセスは互いにメッセージの送受信が可能である．各プロセスには $1, 2, ..., n$ のいずれかの固有の ID を持っており，メッセージを受信したプロセスは，そのメッセージの送信元のプロセスの ID を識別することができる．各プロセスの基本送信操作には２種類があり，各プロセスは一度の送信で同じメッセージしか送信できない**放送型**と，各プロセスが一度の送信でそれぞれのプロセスに対して異なるメッセージを送信することができる**二地点間型**がある．また，通信リンク内でのメッセージの消失及び改ざんはないものとする．本研究では送信、受信、内部計算を１ラウンドとし，これらがすべてのプロセス間で同期的に実行される同期システムを仮定する．

2.2 故障モデル

プロセスのうちのいくつかは**ビザンチン故障**と呼ばれる故障を起こす．この故障を起こしたプロセスは，アルゴリ

移動時間	故障感知性	送信基本操作	移動ビザンチン合意問題を解くための条件（*：必要十分条件）
SR-移動	あり	放送型	
		二地点間型	$n > 3t$ [3]*
	なし	放送型	
		二地点間型	$n > 3t$ [3]*
RC-移動	あり	放送型	
		二地点間型	
	なし	放送型	
		二地点間型	
CS-移動	あり	放送型	
		二地点間型	$n > 4t$ [1]
	なし	放送型	$n > 5t$ [2]*
		二地点間型	$n > 6t$ [7]*

表 1 異なる移動ビザンチン故障モデル

ズムを無視して任意の行動をとる．また，本研究では，ビザンチン故障を起こすプロセスが時間ごとに変わる**移動ビザンチン故障**を仮定する．故障があるプロセスから他のプロセスに移動するタイミングを移動時間と呼ぶ．故障の移動は送信，受信，内部計算のいずれかの操作の間に起こるものとし，移動時間が送信と受信の間である場合を **SR-移動**，受信と内部計算の間である場合を **RC-移動**，内部計算と送信の間である場合を **CS-移動**と呼ぶ．また，故障が自分のもとから去ったプロセスは，その直前にまで滞在していた故障により内部変数やアルゴリズムのコードなどが任意に書き換えられている．これらのプロセスは次の受信時に他のプロセスから正しい情報を受け取ることで，正常な動作に復帰することができる．このように，故障が去った瞬間から次の受信フェーズが開始するまでの状態にあるプロセスを**復帰プロセス**と呼ぶ．また，故障が滞在しているプロセスを**故障プロセス**と呼び，復帰プロセスでも故障プロセスでもないプロセスを**正常プロセス**と呼ぶ．ここで，SR-移動の際には復帰プロセスは存在しない．また，前述したように，復帰プロセスの内部変数は直前まで滞在した故障に書き換えられるので，送信の際には通信基本操作に基づいた上で任意の値を送信する．ここで，自分が直前まで故障しており，現在復帰状態にあることを感知できる性質を**故障感知性**と呼ぶ．故障感知性を仮定した場合，復帰プロセスは自分の送信するメッセージが改変されることを認識できるため，送信を行わない．各ラウンドごとに故障を起こすプロセス数の上限を t とする．

2.3 移動ビザンチン合意問題

移動ビザンチン合意問題は，移動ビザンチン故障が起こる分散システム上で，各プロセスに 0 または 1 が与えられた状態から，各プロセスが同じ値を合意値として決定し，合意が取れた状態を目指す問題である．移動ビザンチン合意問題は以下の 4 つの条件を満たさなければならない．

(決定性) 永久に故障しているプロセス以外のプロセスはいずれ合意値を決定する．

(合意性) 故障しているプロセス以外のプロセスは同じ値を合意値とする．

(妥当性) 故障しているプロセス以外のすべてのプロセスの初期値が同じ値ならば，合意値はこの値である．

(合意維持性) 一度故障しているプロセス以外のプロセス間で合意が形成された場合，それ以降の各ラウンド終了時に，新たな非故障プロセスはその値で合意を維持する．

この問題は，すべてのプロセスが少なくとも一時的に故障する場合，$t = 0$ でないと解けないことが知られている [4]．そこで，以下では永続的に故障しないプロセスが少なくとも 1 つ存在することを仮定する．

また，本研究において”プロセス i が合意値を決定する”とは，プロセス i が合意値を格納するための特別な変数 w_i に値を格納することを指す．ただし，i が故障から復帰した場合，w_i の内容が書き換えられている可能性があり，合意維持性を満たすために，一度合意値を決定した後も w_i を更新し続ける必要がある．

2.4 異なる移動ビザンチン故障モデル

移動ビザンチン合意問題については様々な研究が成されており [1], [2], [3], [4], [7]，それぞれ移動時間、送信基本操作、故障感知性が異なるモデルが提案されている．移動時間、送信基本操作、故障感知性によって分類される 12 個のモデルと，明らかになっている移動ビザンチン合意問題を解くための条件を表 1 に示す．

2.5 ３つのパラメータによる移動ビザンチン故障モデルの定式化

2.4 節で示した 12 個の移動ビザンチン故障モデルを定式化するために，以下に 3 つのパラメータ γ，δ，ε を定義する．

γ: 移動時間が RC-移動または CS-移動の場合は $\gamma = 1$，

移動時間	故障感知性	送信基本操作	γ	δ	ε
SR-移動	あり	放送型	0	0	0
		二地点間型	0	0	0
	なし	放送型	0	0	0
		二地点間型	0	0	0
RC-移動	あり	放送型	1	0	0
		二地点間型	1	0	0
	なし	放送型	1	1	0
		二地点間型	1	1	1
CS-移動	あり	放送型	1	0	0
		二地点間型	1	0	0
	なし	放送型	1	1	0
		二地点間型	1	1	1

表 2 各移動ビザンチン故障モデルに対応する γ，δ，ε の値

SR-移動の場合は $\gamma = 0$ とする．すなわち，復帰プロセスが存在するモデルの場合は $\gamma = 1$，復帰プロセスが存在しないモデルでは $\gamma = 0$ となる．

δ: $\gamma = 1$ かつ故障感知性がない場合，$\delta = 1$ とする．それ以外の場合，すなわち $\gamma = 0$ または故障感知性がある場合は $\delta = 0$ とする．

ε: $\delta = 1$ かつ送信基本操作が二地点間型の場合，$\varepsilon = 1$ とする．それ以外の場合，すなわち $\delta = 0$ または通信基本操作が放送型の場合，$\varepsilon = 0$ とする．

各故障モデルに対応する γ，δ，ε の値を表 2 に示す．

3. 統一的な移動ビザンチン合意アルゴリズム

本節では，2.5 節で定義した γ, δ, ε を用いた統一的な移動ビザンチン合意アルゴリズム UmBA を提案する．UmBA は 2.4 節で挙げた 12 個の故障モデルのいずれに対しても，$n > (3 + \gamma + \delta + \varepsilon)$ のときに移動ビザンチン合意問題を解く．

UmBA の概要を示す．UmBA は連続する 3 ラウンドを 1 フェーズとし，このフェーズを連続して行うことで合意を目指す．また，各プロセスはフェーズごとに順番にコーディネータを担当する．コーディネータは現在のラウンド数で決まるため，各プロセスはどのプロセスがコーディネータであるかを知ることができる．また，各プロセス i は，自分が合意しようとしている値を v_i に格納しておく（実行開始時は v_i はプロセス i の初期値で初期化する）．以下に各フェーズの動作を示す．1 つ目のラウンドでは各プロセス i は v_i をすべてのプロセスに送信し，他のプロセスから受け取った値を配列 $PV_i[1..n]$ に格納する．さらに，$PV_i[1..n]$ に値 $v \in \{0, 1\}$ が一定数以上出現する場合，v_i に v を格納する．2 つ目のラウンドでは，各プロセス i は 1 つ目のラウンドと同様に v_i をすべてのプロセスに送信し，受け取った値を配列 $SV_i[1..n]$ に格納する．3 つ目のラウンドでは，各プロセス i は $SV_i[1..n]$ をすべてのプロセスに送信し，受け取った配列を 2 次元配列 $EV_i[1..n][1..n]$ に格納する．さらに，新たな配列 $MV_i[1..n]$ に対して，$EV_i[1..n][1..n]$ の j 列目の値のうち，一定数以上出現する値 $v \in \{0, 1\}$ を $MV_i[j]$ に格納する．$MV_i[1..n]$ に一定数以上出現する値 $v \in \{0, 1\}$ が存在するならば，v_i に v を格納する．それ以外の場合は，コーディネータから受け取った配列をもとに v_i を決定する．以上の操作を 1 フェーズとして繰り返す．コーディネータが故障している場合は，そのフェーズ内で合意を達成することができないが，正常プロセスがコーディネータである場合には合意が達成可能である．また，一度合意が達成された後は，故障プロセスがコーディネータになってもその合意は維持されたままとなる．

仮定より，絶対に故障しないプロセスが 1 つ存在しているので，n フェーズ，つまり $3n$ ラウンド後にはすべての

Algorithm UmBA

1: $v_i \leftarrow d_i$;
2: **for** $s = 0$ to $n - 1$ **do**
3: //Round $3s + 1$
4: send v_i to all processes;
5: $PV_i[1..n] \leftarrow [\perp, ..., \perp]$;
6: **for all** $i \in \Pi$ **do**
7: $PV_i[j] \leftarrow v_j$;
8: **if** $(\exists v \neq \perp : \#_v(PV_i[1..n]) \geq n - (\gamma + 1)t$ and $\#_v(PV_i[1..n]) + \#_\perp \geq n - (\delta + 1)t)$ **then**
9: $v_i \leftarrow v$;
10: **else**
11: $v_i \leftarrow \perp$;
12: $w_i \leftarrow \perp$;
13: //Round $3s + 2$
14: send v_i to all processes;
15: $SV_i[1..n] \leftarrow [\perp, ..., \perp]$;
16: **for all** $j \in \Pi$ **do**
17: $SV_i[j] \leftarrow v_j$;
18: $w_i \leftarrow \perp$;
19: //Round $3s + 3$
20: send $SV_i[1..n]$ to all processes;
21: $EV_i[1..n][1..n] \leftarrow [\perp, ..., \perp][\perp, ..., \perp]$;
22: $MV_i[1..n] \leftarrow [\perp, ..., \perp]$;
23: **for all** $j \in \Pi$ **do**
24: $EV_i[j][1..n] \leftarrow SV_j[1..n]$;
25: **for all** $j \in \Pi$ **do**
26: **if** $(\exists v \neq \perp : \#_v(EV[1..n][j]) \geq n - (\gamma + 1)t)$ **then**
27: $MV_i[j] \leftarrow v$;
28: **else**
29: $MV_i[j] \leftarrow \perp$;
30: $c_i \leftarrow (s \mod n) + 1$;
31: **if** $(\exists v \neq \perp : \#_v(EV_i[c_i][1..n]) > (\delta + 1)t)$ **then**
32: $cv_i \leftarrow v$;
33: **else**
34: $cv_i \leftarrow 0$;
35: **if** $(\exists v \neq \perp : \#_v(MV_i[1..n]) \geq n - (\gamma + 1)t)$ **then**
36: $v_i \leftarrow v$;
37: **else**
38: $v_i \leftarrow cv_i$;
39: $w_i \leftarrow \perp$;
40: $w_i \leftarrow v_i$;
41: **for** $r = 3n + 1$ to ∞ **do**
42: send w_i to all processes;
43: $w_i \leftarrow$ the value received at least $n - (\gamma + 1)t$ times;

非故障プロセス i について，$v_i = v \in \{0,1\}$ となり，各プロセス i は v_i の値を w_i に格納することで合意が達成される．また，ラウンド $3n+1$ 以降は，各プロセス i は w_i をすべてのプロセスに対して送信し，受け取った値のなかで一定数を占める値を新たに w_i とすることで，新たに故障から復帰したプロセスを含めて合意を再形成し，合意維持性を満たすようにする．

アルゴリズム中では表記の簡潔化のために，配列 V 中に値 v の出現する個数を ${}_v(V)$ と表記する．

定理 1 アルゴリズム UmBA は，$n > (3+\gamma+\delta+\varepsilon)t$ のとき，移動ビザンチン合意問題を解く．

4. $(\gamma, \delta, \varepsilon)$-合意問題

2.3 節では，3 つのパラメータ γ, δ, ε によって定式化されたモデル上での移動ビザンチン合意アルゴリズムを提案した．本節では γ, δ, ε によって定義される $(\gamma, \delta, \varepsilon)$-合意問題を提案し，移動ビザンチン合意問題との関係を示す．

$(\gamma, \delta, \varepsilon)$-合意問題では，高々 t 個のプロセスが永久的に 2.3 節で γ, δ, ε によって定義されたモデルにおける復帰プロセスと同じ振る舞いを行う．ただし，$\gamma = 0$ の場合は復帰プロセスが存在しないため，$\gamma = 1$ の場合のみを考える．

4.1 $(1,0,0)$-合意問題

$(1,0,0)$-合意問題では，高々 t 個のプロセスが 2.3 節における $(\gamma, \delta, \varepsilon) = (1,0,0)$ の場合の復帰プロセスと同様に，各ラウンドの送信時にメッセージを送信しない動作を行う．$(1,0,0)$-合意問題は単純なアルゴリズム $(1,*,0)$-Agree によって解くことができる．$(1,*,0)$-Agree は 1 ラウンドのみで構成されている．まず，各プロセスは自分の初期値をすべてのプロセスに送信する．その後，他のプロセスから受け取った値を配列 $MV_i[1..n]$ に格納し，各プロセスは $MV_i[1..n]$ に最も多く出現する値を合意値として決定する．各プロセスはすべてのプロセスに同じメッセージを送信するか，あるいはすべてのプロセスにメッセージを送信しないので，それぞれのプロセスの配列の内容は一致する．従って，すべてのプロセスが同じ合意値を決定する．

Algorithm $(1,*,0)$-Agree

```
1: send d_i to all processes;
2: MV_i[1..n] ← [⊥, ..., ⊥];
3: for all j ∈ Π do
4:     MV_i[j] ← d_j;
5: if (∃v ∈ {0,1} : #_v(MV_i[1..n]) > #_{|v−1|}(MV_i[1..n])) then
6:     w_i ← v;
7: else
8:     w_i ← 0;
```

定理 2 アルゴリズム $(1,*,0)$-Agree は $(1,0,0)$-合意問題を $n > t$ のときに解く．

また，$(1,0,0)$-合意問題の不可能性についても容易に証明することができる．すべてのプロセスが復帰状態でメッセージを送信しない場合，明らかに妥当性を満たすことはできないため，$n \leq t$ の場合には $(1,0,0)$-合意問題は解けない．

定理 3 $n \leq t$ の場合，どんなアルゴリズムも $(1,0,0)$-合意問題を解くことはできない．

4.2 $(1,1,0)$-合意問題

$(1,1,0)$-合意問題では高々 t 個の復帰プロセスが 2.3 節における $(\gamma, \delta, \varepsilon) = (1,1,0)$ の場合の復帰プロセスの振る舞いを行う．これらのプロセスは任意の値を送信することはできるが，各プロセスに別々の値を送信することはできないので，$(1,*,0)$-Agree を用いることで合意が達成される．

定理 4 アルゴリズム $(1,*,0)$-Agree は $(1,1,0)$-合意問題を $n > 2t$ のときに解く．

また，詳細は省略するが，$(1,0,0)$-合意問題と同様に不可能性についても証明することができる．

定理 5 $n \leq 2t$ の場合，どんなアルゴリズムも $(1,1,0)$-合意問題を解くことはできない．

4.3 $(1,1,1)$-合意問題

$(1,1,1)$-合意問題では高々 t 個の復帰プロセスが，各プロセスに対して任意の値を送信する．この問題は一見ビザンチン合意問題 [5] と似ているが，$(1,1,1)$-合意問題では復帰プロセスを含むすべてのプロセスが同じ合意値を決定する必要がある．そこで $(1,1,1)$-合意問題を解くアルゴリズム $(1,1,1)$-Agree では，最初にビザンチン合意アルゴリズムを実行し，正常プロセス間のみで合意を形成する．その後，それぞれのプロセスが自分の合意値を各プロセスに送信し，受け取った値のうち過半数を占める値を合意値とすることで，すべてのプロセス間で合意を形成する．

Algorithm $(1,1,1)$-Agree

```
1: run Byzantine agreement algorithm and let v_i be the output
2: send v_i to all processes;
3: MV_i[1..n] ← [⊥, ..., ⊥];
4: for all j ∈ Π do
5:     MV_i[j] ← v_j;
6: if (∃v ∈ {0,1} : #_v(MV_i[1..n]) > #_{|v−1|}(MV_i[1..n])) then
7:     w_i ← v;
8: else
9:     w_i ← 0;
```

定理 6 アルゴリズム $(1,1,1)$-Agree は $(1,1,1)$-合意問題を $n > 3t$ のときに解く．

ビザンチン合意問題の不可能性 [5] より，$(1,1,1)$-合意問題の不可能性も明らかである．

定理 7 $n \leq 3t$ の場合，どんなアルゴリズムも $(1,1,1)$-合意問題を解くことはできない．

系 1 $(\gamma, \delta, \varepsilon)$-合意問題を解く必要十分条件は，$n >$

移動時間	故障感知性	送信基本操作	移動ビザンチン合意問題を解くための条件（*：必要十分条件）
SR-移動	あり	放送型	$n > 3t$*
		二地点間型	$n > 3t$ [3]*
	なし	放送型	$n > 3t$*
		二地点間型	$n > 3t$ [3]*
RC-移動	あり	放送型	$n > 4t$
		二地点間型	$n > 4t$
	なし	放送型	$n > 5t$*
		二地点間型	$n > 6t$*
CS-移動	あり	放送型	$n > 4t$
		二地点間型	$n > 4t$ [1]
	なし	放送型	$n > 5t$ [2]*
		二地点間型	$n > 6t$ [7]*

表 3 異なる移動ビザンチン故障モデルと移動ビザンチン合意問題を解く条件

$(\gamma + \delta + \varepsilon)t$ である．

4.4 移動ビザンチン合意問題との関係

2.3 節では，γ，δ，ε によって定式化された移動ビザンチン合意問題が $n > (3 + \gamma + \delta + \varepsilon)t$ のときに解けることを示した．また，各ビザンチン故障が移動を行わずに同じプロセスに滞在しているビザンチン合意問題を解く必要十分条件は $n > 3t$ である [5]．これら 2 つの式に加えて，系 1 の必要十分条件を加えた 3 つの式を比較してみると，移動ビザンチン合意問題を解く条件は，ビザンチン合意問題を解く条件と $(\gamma, \delta, \varepsilon)$-合意問題を解く条件の単純な加算であることが分かる．このことから，移動ビザンチン故障の移動性は移動ビザンチン合意問題を解く条件に直接影響を与えていないと考えることができる．しかし，この考察では移動ビザンチン合意問題において仮定した永久的に故障しないプロセスの存在に考慮していないなど不十分な点がいくつかある．この考察を裏付ける明確な根拠を見つけることが今後の課題となる．

5. まとめ及び今後の課題

本研究では，移動ビザンチン合意問題を移動時間、送信基本操作、故障感知性の 3 つの観点から分類し，γ，δ，ϵ によって定式化することで，$n > (3 + \gamma + \delta + \varepsilon)t$ のときに移動ビザンチン合意問題を解く統一的なアルゴリズムを提案した．先行研究及び本研究によって明らかになった各モデルに対する移動ビザンチン合意問題を解く条件を表 3 に示す．その後，移動ビザンチン合意問題の復帰プロセスのみが静止した状態で存在する $(\gamma, \delta, \varepsilon)$-合意問題を提案し，この問題を解くための必要十分条件を示した．これら 2 つの式及びビザンチン合意問題を解く必要十分条件である $n > 3t$ を比較すると，単純な加算であり，故障の移動が移動ビザンチン合意問題の困難さに直接影響を与えないことが分かった．しかし，この考察は不十分な点が多く，この考察を裏付けることが今後の課題である．

参考文献

[1] N. Banu, S. Souissi, T. Izumi, and K. Wada, “An improved Byzantine agreement algorithm for synchronous systems with mobile faults,” *Int’l J. Computer Applications,* 43, 21, 2011.

[2] F. Bonnet, X.Défago, T.D.Nguyen, and M.P.Butucaru, “Tight bound on mobile Byzantine agreement,” *Proc. 28th International Symposium on Distributed Computing,* (DISC 2014) (to appear).

[3] H. Buhrman, J. A. Garay, and J. Hoepman, “Optimal resiliency against mobile faults,” *Proc. 25th Int’l Symp. Fault-Tolerant Computing,* (FTCS’95) 83–88, 1995.

[4] J. A. Garay, “Reaching (and maintaining) agreement in the presence of mobile faults,” *Proc. Workshop on Distributed Algorithms,* 253–264, 1994.

[5] L. Lamport, R. Shostak, and M. Pease, “The Byzantine generals problem,” *ACM Trans. Programming Languages and Systems,* 4, 382–401, 1982.

[6] M. Pease, R. Shostak, and L. Lamport, “Reaching agreement in the presence of fault,” *J. ACM,* 27, 2, 228–234, 1980.

[7] T. Sasaki, Y. Yamauchi, S. Kijima, M. Yamashita, “Mobile Byzantine agreement on arbitrary network,” *Proc. 17th International Conference on Principles of Distributed Systems,* (OPODIS 2013) 236–250, 2013.