通信リンクのブロックと切断による モバイルビザンチン故障の封じ込めと合意形成

半澤　陽[1,a)]　山内　由紀子[2,b)]

概要：任意の振る舞いをするビザンチン故障プロセスが存在している分散システムで，正常プロセス間で合意を形成する問題をビザンチン合意問題と呼ぶ．本稿では，ビザンチン故障プロセスの集合が継続的に変化する移動ビザンチン故障に着目し，通信リンクのブロックや切断によって，ビザンチン故障を分散システムの一部に封じ込める手法と，それぞれの操作を行いながら合意問題を解くアルゴリズムを示す．

キーワード：分散システム，合意問題，モバイルビザンチン故障，故障封じ込め

Containment of mobile Byzantine faults and constructing agreement by blocking and disconnecting communication links

Hinata Hanzawa[1,a)]　Yukiko Yamauchi[2,b)]

Abstract: The Byzantine agreement problem requires non-faulty processes in a distributed system to construct an agreement in the presence of Byzantine faults that make the faulty processes behave arbitrarily. We focus on the mobile Byzantine fault model that continuously changes the set of faulty processes. We consider containment of mobile Byzantine faults to a part of the system by blocking and disconnecting some communication links and propose an agreement algorithm with the two types of operations.

Keywords: Distributed system, agreement problem, mobile Byzantine fault, and containment

1. はじめに

分散システムとは，通信リンクで相互に接続された多数の計算機 (プロセス) からなるシステムである．各プロセスが通信リンクを介して他のプロセスとメッセージを送受信し，局所的に計算を行うことで，すべてのプロセスで協調して動作する．分散システムにおける最も基本的な課題は全てのプロセスで合意を形成することである．インターネットに代表されるような大規模な分散システムでは，プロセスや通信リンクの故障は不可避であるため，様々な耐故障合意形成手法が提案されている．

故障したプロセスが任意の振る舞いを行う故障を**ビザンチン故障**と呼び，この故障を想定した合意形成問題を**ビザンチン合意問題**と呼ぶ．Lamport らは，$n > 3t$ の場合，完全ネットワーク上で合意を達成するアルゴリズムを提案し，$n \leq 3$ の場合，合意を形成するアルゴリズムが存在しないことを示した [4]．ここで，n はプロセス数，t はビザンチン故障プロセス数である．Garay はビザンチン故障プロセスの集合が継続的に変化する**移動ビザンチン故障**の下での合意問題を提案し，故障プロセスになることのないプロセスが少なくとも 1 つ存在する場合に，$n > 6t$ で**移動ビザンチン合意問題**を解くアルゴリズムを提案した [2]．このモデルでは，ビザンチン故障プロセスの集合は故障を引き起こすビザンチンエージェントの移動によって変化する．Buhrman らは，ビザンチンエージェントが送信メッセー

[1] 九州大学理学部
School of Science, Kyushu Uniersity
[2] 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu Uniersity
[a)] hanzawa@tcs.inf.kyushu-u.ac.jp
[b)] yamauchi@inf.kyushu-u.ac.jp

ジと共に移動する故障モデルを提案し，$n > 3t$ の場合に完全グラフにおいて移動ビザンチン合意問題を解くアルゴリズムを提案した [1].

自己安定プロトコルとは，任意のシステム状況から開始しても自動的に目的のシステム状況へ収束することを保証する分散アルゴリズムである．Inoue らは，ビザンチンエージェントが通過した通信リンクをブロックすることで単一のビザンチンエージェントを分散システムの一部に封じ込めながら，分散システム上に全域木を構成する自己安定プロトコルを提案した [3].

本研究では，通信リンクのブロック，切断によるビザンチンエージェントの封じ込めと，合意形成を考える．ここで，切断とは通信リンクを双方向にブロックし，以後復帰させないことを指す．本稿ではまず，完全ネットワークで通信リンクを複数本ブロックしても，ビザンチンエージェントをシステムの一部分に封じ込めができないことを示す．次に，通信リンクを切断することで封じ込めができることを示す．さらに，各操作を行いながら移動ビザンチン合意問題を解くアルゴリズムを提案する．

2. 準備

プロセス集合を Π，通信リンクの集合を E とし，分散システムを無向グラフ $G = (\Pi, E)$ と表す．全プロセス数を n ($|\Pi| = n$) とする．プロセス p_i とプロセス p_j について $\{p_i, p_j\} \in E$ である時，p_i と p_j は**隣接**していると言い，p_i の隣接プロセスの集合を $N_{p_i} = \{p_j \in \Pi : \{p_i, p_j\} \in E\} \cup \{p_i\}$ と表す．隣接プロセスは相互にメッセージの送受信による直接通信を行える．プロセスは固有の識別子を持っており，メッセージの送信時に自身の識別子をメッセージに添付することで，受信プロセスは受信したメッセージがどのプロセスから送信されたのかを判別することができる．$\Pi = \{p_1, p_2, ..., p_n\}$ とし，プロセス p_i の識別子を i とする．

全プロセスは**ラウンド**毎に動作し，各ラウンドでメッセージの**送信**，**受信**，**内部計算**を行う．このモデルを**同期システムモデル**と呼ぶ．プロセスはラウンドの最初にメッセージを送信し，他のプロセスからメッセージを受信し，受信したメッセージに基づいて共通のアルゴリズムを用いて内部計算を行い，自身の状態を更新し，次のラウンドで送信するメッセージを決定する．ラウンド内で送信されたメッセージは同じラウンド内で必ず受信される．各プロセスはいくつかの内部変数を管理している．これらの変数の値がプロセスの状態を表す．

故障したプロセスが任意の振る舞いを行う故障を**ビザンチン故障**と呼ぶ．ビザンチン故障は故障プロセスの識別子を変えることはできないものとする．ビザンチン故障をしているプロセスを**故障プロセス**と呼び，故障プロセスの集合を F と表す．各プロセスは故障プロセスを知ることはできない．$\Pi \backslash F$ に含まれるプロセスを**正常プロセス**と呼ぶ．

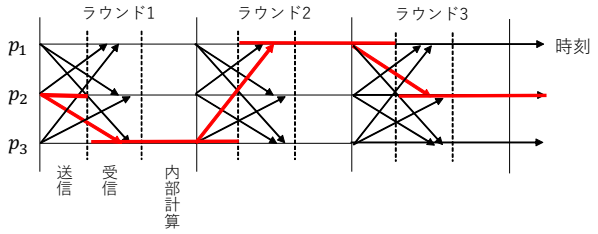

図 1 移動ビザンチン故障．各ラウンドの赤線はプロセスにビザンチンエージェントが到達し，ビザンチン故障を起こしている状態を表す．

移動ビザンチン故障とは，ビザンチン故障プロセスの集合 F が各ラウンド毎に変化する故障である (図 1)．本研究では，正常プロセスにビザンチンエージェントが訪れることで，そのプロセスがビザンチン故障となると考える．ビザンチンエージェントは送信メッセージと共に受信プロセスに移動し，高々 t 個のビザンチンエージェントが存在するとする．各プロセスは直前のラウンドでビザンチンエージェントが滞在していた場合，自身がビザンチン故障状態にあった事を認識できるとする．このような能力を**故障感知能力**と呼ぶ．直前のラウンドでビザンチンエージェントがいたプロセスを *cured* **プロセス**と呼ぶ．本研究では cured プロセスはそのラウンドでは送信を行わず，受信したメッセージをもとに正しい動作に復帰するとする．

本研究ではビザンチン故障を起こすことのない常に正常なプロセスが 1 台存在すると仮定する．

Inoue らは，ビザンチンエージェントがどのプロセスに移動したかを観測できる**探知可能モデル**と観測できない**探知不可能モデル**を提案した [3]．探知可能モデルでは，各プロセスは滞在していたビザンチンエージェントが退出する時に使用した通信リンクとその先のプロセスを，ビザンチンエージェントが退出した次のラウンド，つまり cured 状態の時にわかる能力を持っている．エージェントは変数 $dest_{p_i} \in \{p_0, p_1, ..., p_n\}$ を持ち，エージェントがプロセス p_i から隣接プロセス $p_j \in N_{p_i}$ へ移動した時，$dest_{p_i}$ が cured 状態になったラウンドの内部計算の初めに $dest_{p_i} = p_j$ となる．探知不可能モデルは，常に $dest_{p_i} = p_0$ の状態である．Inoue らは，探知可能モデルにおいて，cured 状態のプロセスがビザンチンエージェントの移動先のプロセスからメッセージを受信しない，通信リンクの**ブロック**を提案した．プロセス p_i がプロセス p_j が送信したメッセージを受信しない時，プロセスは通信リンク $\{p_i, p_j\}$ をブロックしていると言う．

本研究では，隣接プロセス同士が互いを繋ぐ通信リンクを双方向にブロックした状態である通信リンクの**切断**を導入する．切断された通信リンクはその後繋がらないものとする．

移動ビザンチン合意問題は，移動ビザンチン故障が起こ

る分散システムで，各プロセスが初期値 $d_i \in \{0,1\}$ を与えられた時に，正常プロセスが以下の 4 つの条件を満たしつつ，最終的にある値 $v_i \in \{0,1\}$ で合意した状態に到達する問題である [4].

（**決定性**）: すべての正常プロセスは合意値 v_i をいずれ決定する.

（**合意性**）: すべての正常プロセスは同じ値で合意しなければならない.

（**妥当性**）: すべての正常プロセスの初期値が同じ値であるならば，その値で合意しなければならない.

（**合意維持性**）: 一度正常プロセス間で合意に達すると，その後も正常プロセス間で合意を維持しなければならない.

3. 故障封じ込め

本節では，グラフを完全グラフに限定し，グラフ全体を移動することができるビザンチンエージェントが 1 台の場合を想定する．ビザンチンエージェントの封じ込めを以下のように定義する.

定義 3.1. ビザンチンエージェントの封じ込め
$X \subsetneq \Pi$ に含まれる全プロセスからの通信リンクを $\Pi \backslash X$ の全プロセスが全てブロックしている状態であり，かつビザンチンエージェントが X に含まれているプロセスに滞在している時，ビザンチンエージェントを X に封じ込めたと言う.

ビザンチンエージェントが 1 台の時，各プロセスがブロックする通信リンクの本数は高々一本に保つことができる．この時，ビザンチンエージェントの移動先のプロセスからの通信リンクのみブロックするならば，プロセス p_i がプロセス p_j からの通信リンクをブロックしている時，プロセス p_j はプロセス p_i からの通信リンクをブロックすることはできない．このことから，通信リンクを 1 本だけブロックしてもビザンチンエージェントがグラフ全体をハミルトンサイクルに沿って移動する場合，ビザンチンエージェントは全ての頂点を移動し続けることができ (図 2)，ビザンチンエージェントが封じ込められないことがわかる.

さらに以下の定理を示すことができる.

定理 3.1. ビザンチンエージェントが 1 台で各プロセスが通信リンクのブロックを解除せず，$(n-2)$ 本通信リンクをブロックする場合でも，完全グラフでビザンチンエージェントを封じ込めすることはできない.

一方，通信リンクを切断することで，ビザンチンエージェントを封じ込めることができる.

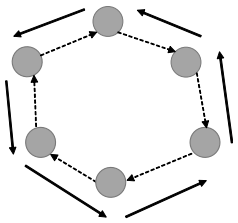

図 2 ビザンチンエージェントの封じ込めが出来ない例．実線矢印はビザンチンエージェントの移動経路，破線矢印はブロックされた通信リンクを表す.

定理 3.2. ビザンチンエージェントが 1 台で各プロセスが通信リンクを $(n-1)$ 本切断することができる場合，ビザンチンエージェントがどのように移動しても完全グラフで封じ込めすることができる.

4. ビザンチン合意アルゴリズム

本節では，グラフ G を完全グラフに限定し，通信リンクをブロックした場合と通信リンクを切断した場合の移動ビザンチン合意問題を解くアルゴリズムを与える.

各プロセスは要素がブール値を保持できる配列 block を管理し，あるプロセス p_i で，$block_{p_i}[p_j] = true$ である時，プロセス p_i はプロセス p_j からのメッセージを受信しない．各プロセスが提案する初期値を $d_i \in \{0,1\}$ とする．提案アルゴリズムでは，ラウンド 1 からラウンド 3 を 1 つのフェーズとし，各フェーズで**コーディネーター**となるプロセスを変更しながら，提案値の変換，更新を行う．$(n-4)$ 本以上の通信リンクがブロックされていない時，コーディネーターが正常プロセスであるならば，すべての正常プロセスは値 v_i で合意することができる．通信リンクをブロックもしくは切断した場合を考えており，正常プロセスがコーディネーターとなった時に，コーディネーターからの通信リンクをブロックまたは切断しているプロセスが存在する場合が考えられる．そのような場合は次のフェーズの第 1 ラウンドに正常プロセス間で合意を形成する．通信リンクをブロックした場合のプロトコルを bMopt を Algorithm1 に示す．正常プロセスは各プロセスから受信した値から自身の値 v_i を決定するが，第 3 ラウンドで cured プロセスは Algorithm2 に示した RECONSTRUCT を実行することで自身の値を決定する．通信リンクのブロックは探知可能モデルを用いていることから Algorithm3 に示した BLOCK を実行する．プロセスからメッセージを受信しなかった場合は自身の値 v_i を $\perp$ とする.

プロトコル bMopt について，以下の定理が成り立つ.

定理 4.1. $n > 3t$，$t = 1$ の時，プロトコル *bMopt* は完全グラフ上で移動ビザンチン合意問題を解く.

Algorithm 1 Protocol bMopt at process p_i

$v_i \leftarrow d_i$
for *phase* $s = 1$ *to* ∞ **do**
  Round 1
  send v_i *to all processes*
  receive(MV[·])
  if cured **then**
    $BLOCK$
  end if
  for $j = 0$ *to* 1 *do* $C[j]$ $=$ *# of* j'*s in* MV **do**
    $v_i = \begin{cases} 0 & if\ C[0] \geq n' - t \\ 1 & if\ C[1] \geq n' - t \\ \perp & otherwise \end{cases}$
  end for
  Round 2
  send v_i *to all processes*
  receive(MV[·])
  if cured **then**
    $BLOCK$
  end if
  for $j = \perp$ *to* 1 *do* $D[j]$ $=$ *# of* j'*s in* MV **do**
    $v_i = \begin{cases} 0 & if\ D[0] > t \\ 1 & if\ D[1] > t \\ \perp & otherwise \end{cases}$
  end for
  Round 3
  send MV[·] to all processes
  for *each i do* **do**
    ECHO[i, ·];=MV[·] received from i
  end for
  if cured **then**
    $BLOCK$
  end if
  if cured in Round 2 **then**
    RECONSTRUCT(EV_i)
  end if
  $c_i \leftarrow s \bmod n$
  if $v_i = \perp$ *or* $D[v_i] < n' - t$ **then**
    $v_i = max(0, v_{c_i})$
  end if
end for

Algorithm 2 RECONSTRUCT

for *all* $j \in \Pi$ *do* **do**
  if $w \in \{0, 1\}$ *occurs at least* $n' - t$ *times in column* j *of* EV_i **then**
    $SV_i[j] \leftarrow w$
  else
    $SV_i[j] \leftarrow \perp$
  end if
end for
for $j = \perp$ *to* 1 *do* $D[j]$ $=$ *# of* j'*s in* SV **do**
  $v_i = \begin{cases} 0 & (D[0] > t) \\ 1 & (D[1] > t) \\ \perp & (otherwise) \end{cases}$
end for

通信リンクを切断するアルゴリズムでは，各プロセスはそれぞれの隣接プロセスからの通信リンクをブロックした回数を記録できるモデルを考える．その記録はプロセスがビザンチン故障を起こしても書き換えられることはないとする．プロセス p_i がプロセス p_j からの通信リンクをブロックしており，プロセス p_j をブロックした回数が $(D-1)$ 回とする．D 回目のブロックの時にプロセス p_i はプロセス p_j と繋がっている通信リンクを切断する．通信リンクを 1 回で切断する場合の移動ビザンチン合意アルゴリズムで dMopt と呼ぶ．プロトコル bMopt と dMopt の相違点は以下の二点である．一点目は，プロトコル bMopt で BLOCK の操作を行う時に，プロトコル dMopt では，通信リンクを双方向ブロックする手続きを行う．二点目は，第 1 ラウンドで cured 状態のプロセスは第 2 ラウンドでは，t 個以上の値 v_i を受信した時，その値を自身の値に採用し，それ以外の時は値を $\perp$ とする．第 1 ラウンドで正常であったプロセスはプロトコル bMopt と同様の手続きを行う．

Algorithm 3 BLOCK

for $j = 1$ *to* n *do* **do**
  $block_{p_i}[p_j] \leftarrow false$
end for
if $dest_{p_j} > 0$ **then**
  $block_{p_i}[dest_{p_j}] \leftarrow true$
end if

プロトコル dMopt について，以下の定理が成り立つ．

定理 4.2. $n > 3$，$t = 1$ の時，プロトコル *dMopt* は完全グラフ上で移動ビザンチン合意問題を解く．

5. まとめ

本研究では，1 台のビザンチンエージェントを封じ込める手法と $n > 3$ で移動ビザンチン合意問題を解くアルゴリズムを提案した．今後の課題は，$t \geq 1$ の場合のビザンチンエージェントの封じ込めと移動ビザンチン合意問題を解くアルゴリズムの考案である．

参考文献

[1] H. Burhman, J. Garay, and J. Hoepman. Optimal Resiliency against Mobile Faults. Proc. of FTCS 1995, pp. 83–88, 1995.
[2] J. Garay. Reaching (and Maintaining) Agreement in the Presence of Mobile Faults. Proc. of WDAG 1994, pp. 253–264, 1994.
[3] K. Inoue, H. Kakugawa, and T. Masuzawa. Strongly stabilizing protocol for spanning tree construction with mobile Byzantine. 第 14 回情報科学ワークショップ予稿集, pp. 258–264, 2018.
[4] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4, pp. 382–402, 1982.