Vol.2013-AL-143 No.12
2013/3/1

一般のネットワーク上の移動ビザンチン合意問題について

佐々木 徹[1,a)]　山内 由紀子[2,b)]　来嶋 秀治[2,c)]　山下 雅史[2,d)]

概要：アルゴリズムに関係なく任意の振る舞いをするようなビザンチン故障をしているプロセスが存在する分散システムにおいて，全プロセスで合意を形成する問題をビザンチン合意問題という．故障プロセス集合が時間ごとに変化する移動ビザンチン合意問題では，Garay(1994) により，$n > 6t$ の場合の完全ネットワークにおける分散アルゴリズムが提案された．ただし，n は全プロセス数，t は故障プロセス数とする．本研究では，一般のネットワークでの移動ビザンチン合意問題を解くアルゴリズムとその上下界についてネットワークの構造に着目しながら考える．まず，点連結度が d の通信ネットワーク G 上での移動ビザンチン合意問題は，$n > 6t$ かつ $d > 4t$ でないと解けないことを示す．次に，一部のプロセスと通信リンクを持たないネットワークの例として完全 k 部グラフを考え，$n - 3(\frac{n}{k} - 1) > 6t$ の場合の分散アルゴリズムを与える．また，直径の大きさが n と d によって決まるリングのべき乗 $C_n^{\frac{d}{2}}$ では $d > \min\left\{8t, \frac{n+4t-2}{2}\right\}$ の場合の分散アルゴリズムを与える．

キーワード：分散システム，合意問題，移動ビザンチン故障

Mobile Byzantine Agreement Problem in Arbitrary Networks

Abstract: The Byzantine agreement problem is to make processes in a distributed system agree on their proposed values even in the presence of Byzantine faults. Garay (1994) proposed the mobile Byzantine agreement problem where the set of Byzantine processes continuously changes, and presented a distributed algorithm in a complete network with $n > 6t$ processes where t is the number of faulty processes and n is the number of entire processes. In this paper, we consider the mobile Byzantine agreement problem in an arbitrary network. We first show that the mobile Byzantine agreement is unsolvable unless $n > 6t$ and $d > 4t$ where d is the diameter of the network. Then, we present two distributed algorithms. The first algorithm considers complete k-partite graph with $n - 3(\frac{n}{k} - 1) > 6t$ where each process does not have direct communication link to some other processes. The other algorithm considers $\frac{d}{2}$-th power of a cycle graph with $d > \min\left\{8t, \frac{n+4t-2}{2}\right\}$ where the diameter of the graph is $\lceil \frac{n-1}{d} \rceil$.

Keywords: distributed system, agreement problem, mobile Byzantine fault

1. はじめに

分散システムでは，複数のプロセスがネットワークを介して他のプロセスと情報を交換し，局所的に計算を行いながら，全体でひとつの処理を行う．分散処理を行う上でプロセス間の協調動作は必要不可欠であり，その最も単純な形として，プロセス間で合意を形成する合意問題がある．特に，プロセスの一部が，アルゴリズムに関係なく任意の行動をとるようなビザンチン故障をしている場合，合意形成は容易ではない．本研究では，プロセスのいくつかがビザンチン故障をしている場合に合意を実現するビザンチン合意問題を考える．

ビザンチン合意問題は，非同期システムの場合は 1 つでも停止故障があった場合には解けないことが知られているので，同期モデルを仮定して考えられている．n をプロセス数，t を故障プロセス数とすると，Lamport ら [3] は，$n > 3t$ の場合完全ネットワーク上でのビザンチン合意問題を解く分散アルゴリズムを提案し，$n \leq 3t$ のときは解けな

[1] 九州大学 工学部 電気情報工学科
福岡市西区元岡 7 4 4 番地
[2] 九州大学 大学院システム情報科学研究院
福岡市西区元岡 7 4 4 番地
[a)] sasaki@tcslab.csce.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

いことを示した．Dolev[1] は，一般のネットワークでは，ネットワークの点連結度を d とすると，任意のプロセス間に少なくとも d 本の疎な道があるという性質を利用することで，$n > 3t$ かつ $d > 2t$ の場合のビザンチン合意アルゴリズムを示した．また，Garay[2] は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合意問題を提案し，完全ネットワークの場合，故障しないプロセスが少なくとも 1 つ存在し，$n > 6t$ の場合に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．そこで，本研究では，一般のネットワークでの移動ビザンチン合意問題を解くアルゴリズムと故障プロセス数の上下界について考える．

本論文では，一般のネットワークで移動ビザンチン合意問題を解くための故障プロセス数の上界として，プロセス数に関しては $n > 6t$，ネットワークの点連結度に関しては $d > 4t$ であることを示す．可解性に関しては，まず一部のプロセスと通信できない場合にビザンチン故障が及ぼす影響を知るために，直径が 2 の完全 k 部グラフについて考え，各頂点分割集合の大きさを m とすると，$n - 3(m-1) > 6t$ のときに移動ビザンチン合意問題を解くアルゴリズムを与える．また，ネットワークの直径が大きくなった場合として，n と d によって定義される直径 $\lceil \frac{n-1}{d} \rceil$ のリングのべき乗 $C_n^{\frac{d}{2}}$ で，こちらも $d > \min\left\{8t, \frac{n+4t-2}{2}\right\}$ の場合のアルゴリズムを与える．これらの結果から，直接通信できるプロセスの数が少なくなり，直径が大きくなるほど故障プロセス数が少ない必要がある傾向を発見した．

2. 準備

2.1 分散システム

今回扱うモデルは，多数の計算機（プロセス）とそれらをつなぐ通信リンクから構成される**分散システム**とする．プロセス集合 Π（$|\Pi| = n$）を頂点，通信リンクの集合 E を辺とし，分散システムの通信ネットワークをグラフ $G = (\Pi, E)$ として表す．また，各プロセスは G を知っているものとする．$i, j \in \Pi$ について，$\{i, j\} \in E$ であるとき，i, j は隣接すると言い，相互にメッセージの送信，受信が可能である．また，$i \in \Pi$ に隣接する頂点の集合を N_i（$\subseteq \Pi$）と表す．$i, j \in \Pi$ に対し，G 上での i, j 間の最短路の長さを i, j の**距離**と呼ぶ．G を非連結にするために取り除く必要のある頂点の数の最小値を d とし，G の**点連結度**と呼ぶ．各プロセスは固有の識別子を持ち，$\Pi = \{1, 2, \ldots, n\}$ とする．各プロセスは状態機械であり，通信リンクにより直接接続されたプロセスとのみ通信を行い，現在の内部状態と通信内容をもとに次の内部状態を計算する．分散アルゴリズムは各プロセスの状態遷移関数に対応する．

プロセスどうしは通信リンクを介してメッセージの送信，受信により通信を行う．各通信リンクは双方向であり，メッセージの消失，重複，改変は起こらないものとする．

全プロセスはメッセージの送信，受信，内部計算をラウンドごとに行う．このモデルを**同期モデル**と呼ぶ (図 1)．各プロセスはまず最初に現在の内部状態とアルゴリズムから決まるメッセージを送信し，次に受信したメッセージをもとに内部計算を行い，状態を更新する．あるラウンドに送信されたメッセージは同一ラウンド内で必ず受信されるため，同期モデルは通信遅延を無視したモデルと解釈できる．アルゴリズムの実行開始時をラウンド 1 とし，以後，ラウンド 2，ラウンド 3, . . . とする．

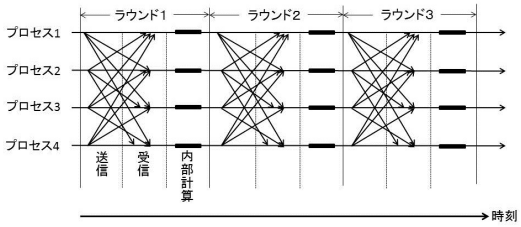

図 1 各プロセスについて横軸を時間経過とする時空間ダイアグラム 横軸を横断する矢印がメッセージの送受信を表す．同期モデルでは，各プロセスの送信，受信，内部計算は同期的に行われ，この順番が前後することはない．

2.2 ビザンチン故障と移動ビザンチン故障

各プロセスは分散アルゴリズムに従い，内部計算を行うが，本稿ではプロセスのうちのいくつかが故障する環境を考える．

故障プロセスが任意の行動をとるような故障を**ビザンチン故障**と呼ぶ．ビザンチン故障の振る舞いの例としては，送受信の際に情報を送らない，送信する値や内部変数を改変するなどが挙げられる．ビザンチン故障を起こしたプロセスは悪意のあるユーザーや，想定外のプロセスの振る舞いといった最悪の状況を反映しているため，このような振る舞いに耐性のあるアルゴリズムを設計することは重要である．

ビザンチン故障を起こしているプロセスの集合を $F \subseteq \Pi$ とし，$|F| = t$ とする．$j \in F$ を故障プロセス，$i \in \Pi \setminus F$ を正常プロセスと呼ぶ．各プロセスは自分が故障しているかどうかの識別は不可能であり，F も知らないものとする．また，ビザンチン故障プロセスは ID の詐称はできず，複数の ID を持つことはないとする．

移動ビザンチン故障とは，ラウンド毎にビザンチン故障を起こすプロセス集合が変化する故障である (図 2)．移動ビザンチン故障を仮定する場合は，故障が起こりうるプロセスの集合を $S \subseteq \Pi$，ラウンド r での故障プロセスの集合を $F_r \subseteq S$（ただし $\forall r \geq 0, |F_r| = t$）と表す．また，$\Pi \setminus F_r$ をラウンド r での正常プロセスと呼ぶ．ビザンチン故障は $i \in F_r$ の内部状態，アルゴリズムのコードを任意に書き換える．そのため，ラウンド $r+1$ で i が正常プロセスとなった場合には，送信を行わない．もしくは書き換えられた内部状態をもとに任意のメッセージを送信する可能性が

ある．ただし，各プロセスは毎ラウンドの送信の際にアルゴリズムのコードや現在のラウンド数などの情報を通信リンクをもつすべてのプロセスに送るものとする．こうすることで，あるラウンド $r > 0$ について，$i \in \Pi \setminus F_r$ であるプロセス i が，ラウンド $r+1$ で正常プロセスになった場合，受信以降は正しい動作に復帰することができる．

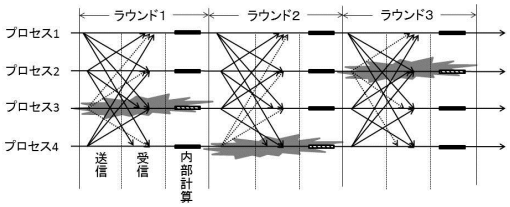

図 2 移動ビザンチン故障
移動ビザンチン故障では，ラウンド 1 ではプロセス 3，ラウンド 2 ではプロセス 4 というように時間ごとに故障するプロセスが変化する．

2.3 移動ビザンチン合意問題

まず，Lamport ら [3] が提案したビザンチン合意問題を示す．

定義 2.1 [3] 各プロセスにラウンド 0 で，それぞれ初期値 $\{0,1\}$ のいずれかが与えられたとする．また，いくつかのプロセスはビザンチン故障をしているものとする．各プロセス i は 1 回だけある値を合意値として変数 v_i に書き込むことができる．どのような初期値が与えられたとしても，各プロセスが以下を満たすような合意値を決定する問題を**ビザンチン合意問題**という．

(合意性) すべての正常プロセスは同じ値を合意値とする．

(決定性) すべての正常プロセスはいつかは合意値を決定する．

(妥当性) すべての正常プロセスの初期値が同じ値だった場合，正常プロセスはその値を合意値にしなければならない．

次に，移動ビザンチン故障を想定した合意問題である，移動ビザンチン合意問題 [2] を示す．移動ビザンチン合意問題では，ラウンド毎に故障プロセス集合が変化するので，一度合意を達成した後に故障の移動により各正常プロセスが異なる合意値を持つようなことがあってはならない．そこで，ビザンチン合意問題の制約に加え，さらに合意の維持に関する制約を要求する．

定義 2.2 [2] 各プロセスにラウンド 0 でそれぞれ初期値 $\{0,1\}$ のいずれかが与えられたとする．また，いくつかのプロセスは移動ビザンチン故障をしているものとする．各プロセス i には合意値を格納するための変数 v_i が与えられ，どのような初期値が与えられたとしても，各プロセスが以下を満たすように合意値を保持する問題を移動ビザンチン合意問題という．

(合意性) すべての正常プロセスは同じ値を合意値とする．

(決定性) すべての正常プロセスはいつかは合意値を決定する．

(妥当性) すべての正常プロセスの初期値が同じ値だった場合，プロセスはその値を合意値にしなければならない．

(合意維持性) あるラウンド r で $\forall i \in \Pi \setminus F_r, v \in \{0,1\}, v_i = v$ となった場合，ラウンド $r' \geq r$ 終了時には，$\forall r' \geq r, \forall i \in \Pi \setminus F_{r'}, v_i = v$ を満たす必要がある．

3. 一般のグラフでの移動ビザンチン合意問題の故障プロセス数の上界

まずは，故障プロセス数の上界の証明のために，移動ビザンチン故障の振る舞いに関する補題を与える．

補題 3.1 プロセス集合 Π(ただし $n > 2t$) のあるネットワーク G 上で，故障プロセス集合 F（$|F| \leq 2t$）の時にビザンチン合意問題を解くアルゴリズムを A_G とし，その任意の実行を C とする．また，A_G が停止するラウンドを r' とすると，G 上で故障プロセス数 t 以下の時に移動ビザンチン合意問題を解くアルゴリズムとして，ラウンド r' までは A_G と同じアルゴリズムであるような任意のアルゴリズムを A'_G とし，その実行を C' とする．ラウンド r にあるプロセス i からあるプロセス j に送信されるメッセージを C では $m_r(i,j)$，C' では $m'_r(i,j)$ とする．また，ラウンド r 終了時のプロセス i の内部変数を C では $S_i(r)$，C' では $S'_i(r)$ とすると，任意の C について以下の 2 つの条件を満たす C' が存在する．

$$\forall i \in \Pi, \forall j \in \Pi_i, 0 \leq r \leq r', m_r(i,j) = m'_r(i,j) \quad (1)$$

$$\forall i \in \Pi \setminus F, 0 \leq r \leq r', S_r(i) = S'_r(i) \quad (2)$$

証明. 一般性を失うことなく，ラウンド毎に各プロセスは自身の状態，ラウンド数を隣接するすべてのプロセスに送信すると仮定する．

ビザンチン合意問題に対する A_G の任意の実行 C について，F の任意の分割 F_{odd}, F_{even} を考える（ただし，$|F_{odd}| \leq t$，$|F_{even}| \leq t$，$F_{odd} \cap F_{even} = \phi$，$F_{odd} \cup F_{even} = F$）．移動ビザンチン合意問題に対する A'_G のある実行 C' について，ラウンド r での故障プロセスの集合を，偶数ラウンドでは F_{even}，奇数ラウンドでは F_{odd} とすると，補題を満たすような故障の振る舞いがあることを r に関する帰納法で示す．

$r = 0$ のとき．

ラウンド 0 では，各プロセスに初期値が与えられるだけ

なので明らかに成立.
$r \leq l$ で (1)，(2) 式が成立すると仮定する.
$l+1$ が奇数と仮定する．C' ではラウンド l で F_{even} が故障しており，これらのプロセスの内部変数を以下が成立するように変更する.

$$\forall i \in F_{even}, S'_i(l) = S_i(l) \tag{3}$$

また，ラウンド $l+1$ では，F_{odd} が故障する．この時に故障は送信直前のプロセスの状態を以下が成立するように変更する.

$$\forall i \in F_{odd}, \forall j \in \Pi_i, m_{l+1}(i,j) = m'_{l+1}(i,j) \tag{4}$$

また，F_1 のプロセスは (3) 式より，ラウンド $l+1$ 開始時の内部変数も C と C' で同じであり，仮定よりプロセスが送信する情報はそのプロセスの持っているすべての情報なので以下が成立する.

$$\forall i \in F_{even}, \forall j \in \Pi_i, m_{l+1}(i,j) = m'_{l+1}(i,j) \tag{5}$$

また F_{even}, F_{odd} 以外のプロセス i については，帰納法仮定より $S_i(l) = S'_i(l)$ であり，ラウンド $l+1$ で送信する情報も同じである．つまり，$r = l+1$ のときも (1) 式が成立し，従って F_{even}, F_{odd} 以外のプロセスは C と C' で同じ情報を受け取る．A_G と A'_G は $r \leq r'$ の範囲では同じアルゴリズムなので，これらのプロセスの内部計算の結果も等しくなる．つまり，$r = l+1$ のときも (2) 式が成立する.
$l+1$ が偶数の場合も同様である.
以上より，補題が証明された. □

補題 3.1 を使って，全プロセス数に対する故障プロセス数の上界を与える.

定理 3.2 n を全プロセス数，t を故障プロセス数とすると，任意のネットワーク G 上で，$n < 6t$ の時移動ビザンチン合意問題を解くアルゴリズムは存在しない.

証明. $n < 6t$ で移動ビザンチン合意問題を解くアルゴリズム A'_G が存在すると仮定すると，A'_G は有限ラウンド r' 終了時に合意性，妥当性，決定性を満たす．G 上で故障プロセス数 $n < 3t$ のときにビザンチン合意問題を解くアルゴリズムとして，A'_G をラウンド r' まで実行し，その後停止するアルゴリズムを A_G とする．補題 3.1 より，A_G の任意の実行に対して，ラウンド r' 終了時の正常プロセスの内部変数が等しい A'_G の実行が存在する．A'_G はラウンド r' 終了時に合意性，妥当性，決定性を満たすので A_G もこれらを満たすので，A_G は，G 上で $n < 3t$ のときにビザンチン合意問題を解く．しかし，これはビザンチン合意問題に関する故障プロセス数の上界に矛盾する．よって，$n < 6t$ で移動ビザンチン合意問題を解くアルゴリズムは存在しない. □

前節同様に補題 3.1 を使って，ネットワークの点連結度に対する故障プロセス数の上界を与える.

定理 3.3 t を故障プロセス数，d をネットワーク G の点連結度とすると，$d < 4t$ の時 G 上での移動ビザンチン合意問題を解くアルゴリズムは存在しない.

証明. G 上で $d < 4t$ で移動ビザンチン合意問題を解くアルゴリズム A'_G が存在すると仮定すると，A'_G は有限ラウンド r' 終了時に合意性，妥当性，決定性を満たす．G 上で故障プロセス数 $d < 2t$ のときにビザンチン合意問題を解くアルゴリズムとして，定理 3.2 と同様に，A'_G をラウンド r' まで実行し，その後停止するアルゴリズムを A_G とする．補題 3.1 より，A_G の任意の実行に対して，ラウンド r' 終了時の正常プロセスの内部変数が等しい A'_G の実行が存在する．A'_G はラウンド r' 終了時に合意性，妥当性，決定性を満たすので A_G もこれらを満たすので，A_G は，G 上で $d < 2t$ のときにビザンチン合意問題を解く．しかし，これはビザンチン合意問題に関する故障プロセス数の上界に矛盾する．よって，$d < 4t$ で移動ビザンチン合意問題を解くアルゴリズムは存在しない. □

4. 移動ビザンチン合意問題の可解性

一般のネットワークでビザンチン合意問題を解く際には，直接通信リンクを持たないプロセス間の情報交換に複数ラウンド費やすことで完全ネットワークと同じアルゴリズムを適用できた．しかし，移動ビザンチン合意問題の場合はラウンド毎に故障が移動してしまうため，プロセス間の距離が大きいと情報を伝達するのは困難になる．そこで，故障プロセス数に関する条件に対して，ネットワークの構造が及ぼす影響に着目しながら，２つのネットワーク上での移動ビザンチン合意問題のアルゴリズムを与える．まず，完全ネットワークから一部の通信リンクを除いた完全 k 部グラフを扱い，全プロセスと一度に情報交換ができない場合の影響を考える．その後，直径と故障プロセス数の関係性について知るため，直径が $\lceil \frac{n-1}{d} \rceil$ のリングのべき乗 $C_n^{\frac{d}{2}}$ 上でのアルゴリズムを与える.

4.1 完全 k 部グラフ

本節では，ネットワークの形を完全 k 部グラフ (図 3) に限定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．頂点集合 Π の分割を $\Pi_1, \Pi_2, ..., \Pi_k$，各分割集合 Π_i のサイズを $|\Pi_i| = m$ とすると完全 k 部グラフでは，各プロセス $i \in \Pi_j$ に対して $N_i = \Pi \setminus \Pi_j$，$|N_i| = (k-1)m$ であり，その他のプロセスとの距離は 2 である.

完全 k 部グラフで移動ビザンチン合意問題を解くアルゴ

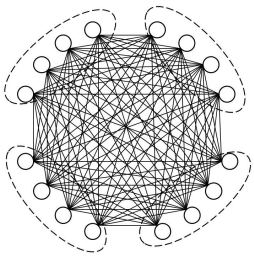

図 3 $m = 4$ の完全 4 部グラフ

リズムを以下に記述する．このアルゴリズムは Garay[2] による CompMobileByz を改良したもので，新たに 1 つのラウンドを加えた 3 つのラウンドを 1 フェーズとする．

第 1 ラウンドは CompMobileByz と同様で，各プロセス i は合意値 v_i をブロードキャストする．そして，受け取った値を $mv_i[1..n]$ に格納し，この中で多数決を取り，v_i にその値を，c_i にその個数を保存する．ただし，一部のプロセスとの通信はできないので，多数決は各プロセス $j \in N_i$ から受け取った値のみで行う．第 2 ラウンドでは，各プロセス i は $mv_i[1..n]$ を送信し，他のプロセスから受け取った配列を $table_i$ に格納する．$table_i$ は関数 RECONSTRUCT によって，第 1 ラウンドで故障していたプロセスの情報の復旧に利用される．また，フェーズキングとなったプロセスは送信の際に自身の合意値を同時に送信し，プロセス i は c_i の値によって受け取った値を合意値にするか判断する．第 1 ラウンドと同様に，フェーズキングの値も一部のプロセスは受け取ることができないため，これらのプロセスも合意ができるように，第 3 ラウンドで再び多数決を取る．

補題 4.1 k–PartByz において，$m > \frac{6t-3}{k-3}$ とする．また，任意のフェーズ l において，第 1 ラウンドでの故障プロセスの集合を $F_{l,1} \subset \Pi$，第 2 ラウンドでの故障プロセスの集合を $F_{l,2} \subset \Pi$ とする．第 1 ラウンドで，プロセス i が送信した値を v_i とすると，第 2 ラウンドの正常プロセス j は，関数 RECONSTRUCT($table_i$) の実行により，以下を満たすような mv_j を得る．

$$\forall i \in \Pi \setminus F_{l,1}, \forall j \in \Pi \setminus F_{l,2} : mv_j[i] = v_i \qquad (6)$$

定理 4.2 完全 k 部グラフ ($k \geq 4$) において，各頂点集合 Π_i の大きさを m，t を故障プロセス数，故障が移動できるプロセスの集合を S とすると，k–PartByz は $n - 3(m-1) > 6t$ かつ $|S| < n$ の時に完全ネットワーク上で移動ビザンチン合意問題を解くアルゴリズムである．

証明. このアルゴリズムが合意性，合意維持性，妥当性，停止性を満たすことを示す．

(合意性)

プロセス $j \notin S$ がフェーズキングになるフェーズについて考える．

Algorithm 1 プロセス i 上のアルゴリズム k–PartByz

```
1: v_i ← プロセス i の初期値
2: for フェーズ l=0 to ∞ do
3:     (第 1 ラウンド)
4:     v_i を各 j ∈ N_i に送信
5:     for all j ∈ N_i do
6:         mv_i[j] ←(j から受け取った値 d(ただし d ∉ {0,1} の
           場合は 0))
7:     if mv_i[1..n] の中に (k−1)m+1 / 2 個以上 1 がある then
8:         v_i ← 1
9:     else
10:        v_i ← 0
11:    c_i ← mv_i[1..n] 中の v_i の個数
12:    (第 2 ラウンド)
13:    king = (l mod n) + 1
14:    if king = i then
15:        v_i を各 j ∈ N_i に送信
16:    すべてのプロセスに mv_i[1..n] を送信
17:    for all p ∈ N_i do
18:        for all q ∈ N_p do
19:            table_i(p, q) ← mv_p[q]
20:    RECONSTRUCT(table_i)
21:    if c_i < (k − 1)m − 2t + 1 かつ v ∈ {0,1} が king から唯
       一つ到着した then
22:        v_i ← v
23:    (第 3 ラウンド)
24:    v_i を各 j ∈ N_i に送信
25:    v_i ←(受け取った (k − 1)m + 1 個の値の中で過半数を占める値)
```

Algorithm 2 プロセス i 上のアルゴリズム RECONSTRUCT($table_i$)

```
1: for all j ∈ N_i do
2:     if table_i の j 列目に v ∈ {0,1} が (k − 2)m − 2t + 2 個以
       上存在する then
3:         mv_i[j] ← v
4:     else
5:         mv_i[j] ← 0
6: if mv_i の中に (k−1)m+1 / 2 個以上 1 がある then
7:     v_i ← 1
8: else
9:     v_i ← 0
10: c_i ← mv_i[1..n] 中の v_i の個数
```

各プロセス i は第 1 ラウンドで自分自身を含む，通信リンクをもつ $(k-1)m+1$ 個のプロセスから値を受け取り，$mv_i[1..n]$，c_i，v_i を更新する．第 2 ラウンドでは，補題より，各正常プロセス i は RECONSTRUCT の実行で，第 1 ラウンドで正常プロセスによって送信された値が格納された $mv_i[1..n]$ を得る．ここで，第 2 ラウンドの内部計算終了時の各プロセスの c_i の大きさで場合分けをし，どちらの場合も第 3 ラウンド終了時にすべてのプロセスの合意値が一致することを示す．

(1) すべての正常プロセス i に関して $c_i < (k-1)m-2t+1$ である場合．

アルゴリズムより，フェーズキングから値 v を受け取ったプロセスはその値を合意値にする．フェーズキングと直接通信リンクを持つプロセスの数は $(k-1)m+1$ 個だが，そのうち高々 t 個のプロセスは故障しているので，第 2 ラウンド終了時点では $(k-1)m-t+1$ 個のプロセス i に関して $v_i = v$ である．

第 3 ラウンドでは，故障の移動によりさらに高々 t 個のプロセス j に関して $v_j \neq v$ となり，各プロセスはアルゴリズムに従い第 1 ラウンドと同様に多数決を取る．まず，プロセス i に関して $i \notin N_{king}$ の場合，$j \in N_i$ である $(k-1)m$ 個のプロセス j のうち，第 2，3 ラウンドで故障したプロセス $2t$ 個を除いた $(k-1)m-2t$ 個の v を少なくとも受け取ることができる (図 4 破線)．この数は通信できるプロセスの数の過半数，つまり $\frac{(k-1)m+1}{2}$ を超えるので，新たに v を合意値にする．一方 $i \in N_{king}$ の場合は，第 2 ラウンド終了時にすでに v を合意値にしているが，アルゴリズムに従い，同じく多数決で合意値を更新する．これらのプロセスは，$j \in N_i \cap N_{king}$ である $(k-2)m$ 個のプロセス j にフェーズキングと自分自身を足し，$2t$ 個の故障プロセス分を引いた $(k-2)m-2t+2$ 個の v を受け取る (図 4 鎖線)．この数も同様に $\frac{(k-1)m+1}{2}$ を超えるので，第 3 ラウンドですべての正常プロセスが v で合意することが証明された．

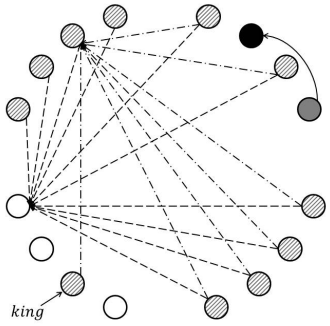

図 4 (1) の場合の第 3 ラウンド $(k=4, m=4, t=1)$
$i \notin N_{king}$ であるプロセスは 10 個のプロセスから値を受け取る (破線)．$i \in N_{king}$ のプロセスは 8 個のプロセスから値を受け取る (鎖線)．

(2) 少なくとも 1 つの正常プロセス i に関して $c_i \geq (k-1)m-2t+1$ である場合．

第 1 ラウンドで i は最低 $(k-1)m-2t+1$ 個のある値ある値 $v \in \{0,1\}$ を受信している．しかし，このうちの高々 t 個は故障プロセスが送った値で他のプロセスには違う値を送っている可能性があり，正常プロセスから送られた値は少なくとも $(k-1)m-3t+1$ 個である (図 5 破線)．この時他のプロセス j は，$p \notin N_j \cup \{j\}$ となる $m-1$ 個プロセス p を除く $(k-2)m-3t+2$ 個の v を受け取っていることが保証できる (図 5 鎖線)．この数は $\frac{(k-1)m+1}{2}$ を超えているので，第 1 ラウンドですべての正常プロセスは v を合意値にする．

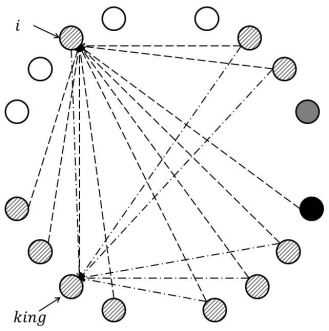

図 5 (2) の場合の第 1 ラウンド $(k=4, m=4, t=1)$
プロセス i が 11 個の v を受け取った時，フェーズキングは最低 7 個の v を受け取っている．

第 2 ラウンドでフェーズキングは v を送信するが，受け取ったプロセスはすでに合意値が v なので，フェーズキングの値を採用するか否かにかかわらず合意値は v のままである．よって，第 2 ラウンド終了時にフェーズキングと直接通信リンクを持つプロセスは合意値が v であるので，(1) の場合と同様に第 3 ラウンド終了時にすべての正常プロセスが v で合意する．1，2 より合意性が証明された．

(合意維持性)

あるフェーズ l 終了時にすべての正常プロセスの合意値がある値 $v \in \{0,1\}$ であると仮定し，それ以降のフェーズの各ラウンド終了時にもすべての正常プロセスの合意値が v であることを示す．フェーズ $l+1$ の第 1 ラウンド開始時では，t 個の故障の移動により v を合意値とするプロセスは $km-2t$ 個ある．さらに，各プロセス i はプロセス $j \notin N_i \cup \{i\}$ である $m-1$ 個のプロセス j とは通信できないので，各プロセスが受け取る v の個数は $(k-1)m-2t+1$ 個である．この数は $\frac{(k-1)m+1}{2}$ を超えているので，すべての正常プロセス i は v を合意値とし，$c_i \geq (k-1)m-2t+1$ となる．第 2 ラウンドで，各正常プロセス i は補題 4.1 より，RECONSTRUCT の実行で最低 $(k-1)m-2t+1$ 個の v が格納された $mv_i[1..n]$ を得ることができ，多数決によって v を合意値とする．また，$c_i \geq (k-1)m-2t+1$ なのですべてのプロセスはフェーズキングの値を無視する．合意性の場合と同じ議論により，第 3 ラウンド終了時にもすべての正常プロセスが v を合意値とする．よって，フェーズ $l+1$ の各ラウンド終了時にすべての正常プロセスが v を合意値としていることが示されたので，明らかにこれ以降のラウンドも合意が維持される．

(妥当性)

実行開始時にすべての正常プロセスが v で合意している

と仮定する．合意維持性の議論より，明らかにこれ以降のラウンド終了時にすべての正常プロセスは v で合意している．

(決定性)

合意性の議論より，第 2 ラウンドでフェーズキングが正常だった場合に合意が達成される．仮定より，フェーズキングは各プロセスが順番に担当するので，有限ラウンド内に $i \in \Pi \setminus S$ であるプロセス i がフェーズキングとなり，合意性の議論のように正常に動作する．よって，決定性を満たす． □

4.2 リングのべき乗 $C_n^{\frac{d}{2}}$

n 頂点のリングに対し，各頂点が距離 l 以内の頂点との間に辺をもつグラフを C_n^l と書く．今回は $C_n^{\frac{d}{2}}$ について考える．このグラフの直径は $\lceil \frac{n-1}{d} \rceil$ で表され，d に対して n が大きければ直径も大きくなり，逆に n が小さくなると完全グラフに近い形になり直径は小さくなる．

このグラフ上で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 i は CompMobileByz の各ラウンドの送信の際に，以下に示す RingBroadcast(w_i) を実行する．RingBroadcast(w_i) はプロセス i が入力として全プロセスに送りたい情報 w_i を与える．ここで，C_n^1 のみを通った時の j からの距離を j からのレベルと定義する (図 6)．ラウンド r（$2 \le r \le \lceil \frac{n-d+1}{2} \rceil$）では，$j$ から

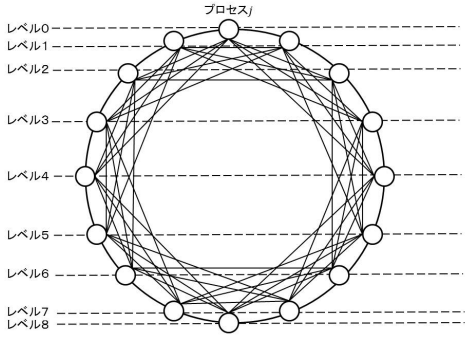

図 6 プロセス j からのレベル

のレベル $\frac{d}{2}+r-2$ 以下のプロセス i は $mw_i[j]$ を送信し，$\frac{d}{2}+r-1$ 以下のプロセスは受け取った値で多数決を取り $mw_i[j]$ を更新する．これにより，各プロセス j の入力 w_j は j からのレベルが小さいプロセスからレベルが大きいプロセスへ通信を行うたびに伝わっていく．i からのレベルが最大となるプロセスまで伝わるとアルゴリズムは終了し，RingBroadcast(w_i) 開始時の各正常プロセス j と，終了時の各正常プロセス i に対して，$mw_i[j] = w_j$ である配列が出力される．

補題 4.3 RingBroadcast(w_i) において，$d > \min\left\{8t, \frac{n+4t-2}{2}\right\}$ とする．j を開始時の正常プロセスとすると，ラウンド r（$1 \le r \le \lceil \frac{n-d+1}{2} \rceil$）終了時の j からのレベル $\frac{d}{2}+r-1$ 以下の正常プロセス i に対して，$mw_i[j] = w_j$ が成り立つ．

Algorithm 3 プロセス i 上のアルゴリズム RingBroadcast(w_i)

```
1: (ラウンド 1)
2: 各 j ∈ N_i に w_i を送信
3: for all j ∈ Π do
4:     if j からのレベルが d/2 以下 then
5:         mw_i[j] ← (j から受け取った情報 (ただし何も受け取らなかった場合は 0))
6: (ラウンド r(2 ≤ r ≤ ⌈(n−d+1)/2⌉))
7: for all j ∈ Π do
8:     if j からのレベルが d/2 + r − 2 以下 then
9:         各プロセス p ∈ N_i に mw_i[j] を送信
10: for all j ∈ Π do
11:    if j からのレベルが d/2 + r − 1 以下 then
12:        j からのレベル d/2 + r − 2 以下のプロセス p から受け取った mw_p[j] で多数決をとり mw_i[j] に格納
13: return mw_i[1..n]
```

証明. ラウンド数 r（$1 \le r \le \lceil \frac{n-d+1}{2} \rceil$）に関する帰納法で証明する．

$r = 1$ のときはアルゴリズムより，明らかに j からのレベル $\frac{d}{2}$ 以下の正常プロセス i に対して $mv_i(j) = w_j$ が成り立つ．

$r = l$（$1 \le r \le \lceil \frac{n-d+1}{2} \rceil - 1$) で補題が成り立つと仮定する．$r = l+1$ のとき，以下の 2 つの場合が考えられる．

(1) $n \ge \frac{3}{2}d + 2l$ の場合．

j からのレベル $\frac{d}{2}+l$ 以下のプロセス i はレベル $\frac{d}{2}+l-1$ のプロセスとの通信リンクを少なくとも $\frac{d}{2}$ 本もつ．この通信リンクの数を m_i とする．ラウンド l とラウンド $l+1$ の故障プロセス集合は異なる可能性があり，第 $l+1$ ラウンド開始時，i に $mw_p[j]$ を送るプロセス p のうち，高々 $2t$ 個のプロセスが w_j と異なる値を送信する．しかし，$m_i \ge \frac{d}{2}$ と $d > 8t$ より $m_i - 2t > \frac{m_i}{2}$ なので，i は受信した $mv_p[j]$ のうち過半数以上は w_j である．よって，$mv_i[j] = w_j$ となる．

(2) $n < \frac{3}{2}d + 2l$ の場合．

レベル $\frac{d}{2}+l$ 以下のプロセスが，レベル $\frac{d}{2}+l-1$ 以下のプロセスとの通信リンクをいくつ持つか考える．レベル $\frac{d}{2}+l-1$ 以下のプロセスの数は $d+2l-1$ であり，各プロセス i は自分自身を含む $d+1$ 個のプロセスと通信可能なので，鳩ノ巣原理よりレベル $\frac{d}{2}+l-1$ 以下のプロセスとの通信リンクの数を m_i とすると $m_i > 2d+2l-n$ である (図 7)．$m_i > 2d+2l-n$，$l \ge 1$，$d > \frac{n+4t-2}{2}$ より $m_i - 2t > \frac{m_i}{2}$ なので i は過半数以上の w_j を受け取り $mv_i(j) = w_j$ となる．

以上より，$r = l+1$ のときも成り立つことが証明された．

よって補題が証明された． □

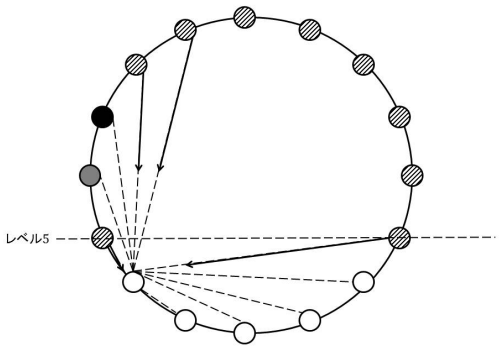

図 7 n=16，d=10 のグラフ

以上を踏まえ，$C_n^{\frac{d}{2}}$ 上で移動ビザンチン合意問題を解くアルゴリズム RingMobileByz を以下のように構成する．

- CompMobileByz の各ラウンドでプロセス i が送信する情報 w_i に対して RingBroadcast(w_i) を実行し，出力された配列を各プロセスから受け取った値として内部計算を行う．
- RingBroadcast(w_i) 実行中は，各ラウンド毎に合意値 v_i をブロードキャストし，受け取った値の中で過半数を超える値を新たに v_i とする．

系 4.4 RingMobileByz において，あるラウンド r ですべての正常プロセス j が RingBroadcast(w_j) を開始した場合，ラウンド $r+\lceil\frac{n-d+1}{2}\rceil$ で正常プロセス i が RingBroadcast(w_i) を終了し，以下が成り立つ．

$$\forall i \in \Pi \setminus F_{r+\lceil\frac{n-d+1}{2}\rceil}, \forall j \in \Pi \setminus F_r : mw_i[j] = w_j. \quad (7)$$

系 4.5 完全ネットワーク上の CompMobileByz で，あるラウンド r に各プロセス i が v_i（または $mv_i[1..n]$）を送信したとする．故障の振る舞いにかかわらず，ラウンド r の受信終了時に，任意の $i, j \in \Pi \setminus F_r$ に対して，$mv_i[p] = mv_j[p]$（または $table_i(p,\cdot) = table_j(p,\cdot)$）であるような p の個数を u_1 とする．また，$C_n^{\frac{d}{2}}$ 上でラウンド r に各正常プロセス i が RingBroadcast(w_i) を実行した場合，故障の振る舞いにかかわらず，ラウンド $r+\lceil\frac{n-d+1}{2}\rceil$ の RingBroadcast(w_i) 終了時に，任意の $i, j \in \Pi \setminus F_{r+\lceil\frac{n-d+1}{2}\rceil}$ に対して，$mw_i[p] = mw_j[p]$ であるような p の個数を u_2 とすると，$u_1 = u_2$ である．

系 4.5 より，完全ネットワークの 1 回の通信と，RingBroadcast(w_i) を 1 回実行したときの，信頼できる値の数は同じである．

定理 4.6 n を全プロセス数，t を故障プロセス数，d を点連結度，故障が移動できるプロセスの集合を S とすると，RingMobileByz は $n > 6t$ かつ $d > \min\left\{8t, \frac{n+4t-2}{2}\right\}$ かつ $|S| < n$ のとき，$C_n^{\frac{d}{2}}$ 上で移動ビザンチン合意問題を解くアルゴリズムである．

証明. CompMobileByz では，プロセス i は内部計算の際，$mv_i[1..n]$（または $table_i$）に格納されている値の個数によって変数の更新が行われる．RingBroadcast(w_i) では，開始時と終了時で故障プロセスが変わっているが，系 4.5 より，CompMobileByz を実行する上では完全ネットワークにおける 1 回の通信と同等とみなすことができ，CompMobileByz と同様に合意性，決定性，妥当性を満たす．合意維持性に関しては，RingBroadcast(w_i) 実行中も正常プロセスの合意値は一致している必要があるが，毎ラウンド毎に合意値を送信し，受け取った値で多数決を取ることで，$d > 8t$ より明らかに合意を維持できる． □

5. おわりに

本研究では，一般のネットワーク上での移動ビザンチン合意問題について考え，ネットワークの構造と故障プロセス数の上下界との関係に着目した．

まず，移動ビザンチン合意問題でのプロセス数 n に対する故障プロセス数 t の上界として $n > 6t$，ネットワークの点連結度 d に対する故障プロセス数の上界として $d > 4t$ を導出した．

可解性ついては，まず，グラフの直径が高々 2 である完全 k 部グラフについて考え，故障の起こりうるプロセスの集合を S，頂点集合の大きさを m とすると，$k \geq 4$，$|S| < n$，$n - 3(m-1) > 6t$ の場合のアルゴリズムを提案した．完全 k 部グラフのように，全プロセスに対して直接通信できるプロセスの割合が減少すれば合意できる最大の故障プロセス数も減少する．また，グラフの直径が $\lceil\frac{n-1}{d}\rceil$ であるようなリングのべき乗 $C_n^{\frac{d}{2}}$ については，$n > 6t$，$d > \min\left\{8t, \frac{n+4t-2}{2}\right\}$ の場合のアルゴリズムを提案した．ネットワークの直径が大きくなった場合にも故障プロセス数が小さい必要がある．

しかし，これらの傾向を裏付けるような，全プロセス数，点連結度，ネットワークの直径を考慮した故障プロセス数の上界の発見には至っていない．ネットワークの構造の関係性を明確にするためにより多くのネットワークを扱い，新たな上界またはアルゴリズムを発見することが今後の課題である．

参考文献

[1] D. Dolev, The Byzantine generals strike again, Jornal of Algorithms, 3, pp.14–30, 1982.
[2] J. A. Garay, Reaching (and maintaining) agreement in the presence of mobile faults, In Proceedings of Workshop on Distributed Algorithms, pp.253–264, 1994.
[3] L. Lamport, R. Shostak, and M. Pease, The Byzantine generals problem, ACM Transactions on Programming Languages and Systems, 4 , pp.382–401, 1982.