数理解析研究所講究録
第1894巻 2014年 67-74

2013年度冬のLAシンポジウム [S7]

移動ビザンチン合意アルゴリズムのための高信頼性伝送アルゴリズム

Reliable transmission algorithm for mobile Byzantine agreement algorithm

佐々木 徹 Toru Sasaki
山内 由紀子 Yukiko Yamauchi
来嶋 秀治 Shuji Kijima
山下 雅史 Masafumi Yamashita

九州大学
Kyushu University

1 はじめに

分散システムにおいて，プロセス間で合意を達成することは重要な問題であり，一部のプロセスがアルゴリズムを無視するビザンチン故障を起こしている場合，難しい問題となる．この問題は1980年にPease達 [7] によってビザンチン合意問題として定式化され，それ以降，多くの研究者がこの問題に取り組んできた．

ビザンチン合意問題は非同期システム上では一般に非可解であり，主に同期システム上で考えられている．n をプロセス数，t を故障プロセス数とする．Lamport達 [6] は，完全ネットワーク上で，$n > 3t$ の場合にビザンチン合意問題を解くアルゴリズムを提案し，一方，$n \leq 3t$ のときにはこの問題が解けないことを証明した．また，Dolev[4] は，点連結度が d であるネットワーク上で，$n > 3t$ かつ $d > 2t$ の移動ビザンチン合意問題を解くアルゴリズムを提案し，$d \leq 2t$ のときにはこの問題が非可解であることを示した．これらのビザンチン合意問題では，故障は永続的で，一度故障したプロセスは二度と回復しない．

Garay[5] はビザンチン故障がプロセス間を移動する場合を定式化し，移動ビザンチン合意問題を提案した．故障プロセスは任意の行動をとるが，故障が別のプロセスに移動すると正常な動作に戻る．Garayは，各ラウンド毎に故障が移動する場合には，移動ビザンチン合意問題を解くことができないことを示し，少なくとも1つの永久に故障しないプロセスが存在するという仮定の下で，$n > 6t$ 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．その後，Banu達 [1] は同じ条件の下で，$n > 4t$ で解けるアルゴリズムを示した．これらのモデルでは，故障が移動するタイミングは，各ラウンドの送信直前であった．それに対し，Buhrman達 [2] は，故障の移動を各ラウンドの送信直後とすると，$n > 3t$ で移動ビザンチン合意問題が解けることを示した．このモデルはメッセージ通信によるウイルスの伝搬等のモデルになっている．これらの先行研究では常に完全ネットワークを仮定していた。

本研究では，Buhrman達のモデルを仮定し，一般のネットワーク上で移動ビザンチン合意問題を考える．まず，任意の頂点間に長さが高々 β の点素な道が α 本存在するグラフの集合を $\mathcal{G}(\alpha, \beta)$ とし，$n > 3t$ かつ $\alpha > 2(\beta - 1)t$ 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する．次に，$n > 3t$ かつ $k > 2t$ を満たすとき，任意のグラフの k 乗グラフ

に対するアルゴリズムを示す．[1]

2 問題定義と先行研究

2.1 システムモデル

いくつかのプロセス間で双方向の通信リンクによって通信を行うことのできる分散システムを考える．Π をプロセス集合，E を通信リンクの集合とすると，ネットワークを単純無向グラフ $G = (\Pi, E)$ によって表すことができる．一般性を失うことなく，$\Pi = \{1, 2, \ldots, n\}$ ($|\Pi| = n$) を仮定し，$i \in \Pi$ をプロセス i の ID とみなす．また，$N_i = \{j \in \Pi : (i,j) \in E\} \cup \{i\}$ を，プロセス i の隣接プロセス集合とする．プロセス i は通信リンクを通して，プロセス $j \in N_i$ とメッセージの送受信を行うことができる．d を G の点連結度と呼び，グラフ 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とする．また，各プロセスは G を初期情報として与えられているものとする．

本研究では，送信，受信，内部計算を 1 ラウンドとし，各ステップが同期的に実行される同期モデルを仮定する．ラウンド $r = 1, 2, \ldots$ では，各プロセス i は各プロセス $j \in N_i$ にメッセージ m_{ij}^r を送信し，各プロセス $j \in N_i$ からメッセージ m_{ji}^r を受け取る．ただし，m_{ij}^r は前ラウンド $r-1$ の内部計算時に計算されるものとする．内部計算では，自身の新たな内部状態 s_i^r と，次のラウンド $r+1$ で隣接プロセス $j \in N_i$ へ送信するメッセージ m_{ij}^{r+1} をあるアルゴリズム A に従って計算する．

2.2 故障モデル：移動ビザンチン故障

通信リンクは信頼でき，メッセージの消失，複製，改ざんは起こらないものとする．さらに，プロセス i がプロセス $j \in N_i$ にメッセージを送信するとき，送信プロセスの ID として i が付与される．

それに対して，いくつかのプロセスは移動ビザンチン故障 [5] を起こす．ビザンチン故障が起きたプロセスはアルゴリズム A を無視して任意の行動をとる．さらに従来のビザンチン故障と異なり，移動ビザンチン故障はプロセス間を移動する．移動ビザンチン故障には，移動のタイミングや移動に必要なラウンド数などによりいくつかの種類がある．Sasaki 達 [8] は，ラウンドが変わる際に故障が移動するモデルを提案しており，故障から復帰したプロセスは，自分自身がそれまで故障していたことを知ることができない．この場合，故障しているプロセスに加えて，故障から復帰した直後のプロセスが送信する情報も信頼できない．一方，本研究では，故障の移動が各ラウンドの送信フェーズ終了後，受信フェーズの前に行われる Buhrman 達 [2] のモデルを仮定する．ラウンド $r-1$ の受信，内部計算，ラウンド r の送信時に故障しているプロセスの集合を F_r とする．ラウンド r の送信時にプロセス i が故障していたが，受信時に故障が別のプロセスに移動した場合，プロセス i の内部状態は故障により書き換えられている．しかし，プロセス i は故障から回復したとき，自分がそれまで故障していたことを認識でき，ラウンド r の内に他のプロセスから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受け取って，受信以降正常な動作に復帰できるものとする．F_r は任意に選ばれるが，ある t が与えられ，任意の r について $|F_r| \leq t$ が成立する．ラウンド r において，F_r に属するプロセスを故障プロセス，それ以外のプロセスを正常プロセスと呼ぶ．プロセス i が故障している場合，i は任意のでたらめなメッセージを送信するが，(通信リンクは信頼できるので) ID を詐称できない．

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，各プロセス i に初期値 $d_i \in \{0, 1\}$ が与えられたとき，各プロセスが同じ

[1] Sasaki 達 [8] は，異なる故障モデルの下で，本編と同様の考察を行なった．[8] では，故障は各ラウンドが終了したあとに移動でき，故障から復帰したプロセスはその事実を直接に認識できない．

合意値を決定するアルゴリズムを設計する問題である．しかし，アルゴリズム実行中には常に高々t個の故障プロセスが存在するので，これらの故障プロセスが任意の行動をとった場合にも，合意を達成するアルゴリズムを設計する必要がある．厳密に言うと，移動ビザンチン合意問題は故障プロセスの行動に関わらず，各プロセスiが以下の4つの条件を満たすように合意値を決定するためのアルゴリズムを設計する問題である．

(決定性) すべての正常プロセスはいずれ合意値を決定する．

(合意性) すべての正常プロセスは同じ値を合意値とする．

(妥当性) すべての正常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) 一度正常プロセス間で合意が形成されると，それ以降の各ラウンドでは，正常プロセスは同じ値で合意をし続ける．

故障が移動しない通常のビザンチン合意問題 [7] では，アルゴリズムが満たすべき制約は決定性，合意性，妥当性であった．しかし，移動ビザンチン合意問題では，一度合意を達成した後でも，故障が移動しても，故障から復帰したプロセスを含めた正常プロセス間で合意を維持するための制約として，合意維持性を要請する．従って，移動ビザンチン合意問題アルゴリズムは，合意達成後も停止することなく動作を続ける．

定理 1 [5] 故障が毎ラウンド移動する場合，移動ビザンチン合意問題は$t = 0$でなければ解けない．

定理1より，故障が毎ラウンド移動するならば，故障が発生する場合には移動ビザンチン合意問題を解くことは出来ない．そこで移動ビザンチン合意問題を考える際は，常に故障しないプロセスi_0が存在すると仮定する (つまり，$i_0 \notin \cup_r F_r$)．

2.4 MOPT: Buhrman達のアルゴリズム

Buhrman達 [2] は，完全ネットワーク上で$n > 3t$のときに移動ビザンチン合意問題を解くアルゴリズムMOPTを提案した．通常のビザンチン合意問題を$n \leq 3t$のときに解くアルゴリズムは存在しないから [6]，MOPTは故障プロセス数tに関して最適なアルゴリズムである．

プロセスi上のアルゴリズムMOPTの概要を説明する．MOPTは連続する3つのラウンドから構成されるフェーズを繰り返すことによって合意を目指す．各フェーズ$s = 1, 2, \ldots$では，プロセス$s \mod n$がコーディネータになる (ただし，$s \mod n = 0$のときはプロセスnがコーディネータとなる)．各フェーズの動作を以下に示す．最初のラウンドでは，他のプロセスの合意値 (最初のフェーズのときは初期値) を受け取り，各正常プロセスの合意値をある値$v \in \{0, 1\}$または$\perp$のどちらかのみとする．次のラウンドでは，再び各プロセスの合意値を受け取り，合意値及び変数$accept$の更新を行う．$accept$は，他のプロセスから受け取った合意値のほとんどが同じ値$v \in \{0, 1\}$であった場合は$False$を，それ以外の場合は$True$をとる．そして，最後のラウンドでは$accept = True$であるプロセスは，コーディネータの値を新たに合意値とする．このとき，このラウンドで故障から復帰したプロセスは他のプロセスから情報を受け取ることでデータの復旧を行う．コーディネータが常に故障していると合意を達成できないが，故障をしないプロセスi_0がコーディネータとなるフェーズで合意が達成できる．

定理 2 アルゴリズムMOPTは完全ネットワーク上で$n > 3t$のときに移動ビザンチン合意問題を解くアルゴリズムである．

通常のビザンチン合意問題では，点連結度をdとするとき，$n \leq 3t$または$d \leq 2t$であるとき，アルゴリズムは存在しない [4]．明らかに，この条件を満たすとき，移動ビザンチン合意問題も非可解である．

系 1 $n \leq 3t$または$d \leq 2t$の場合，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．

Algorithm 1 MOPT on process i

1: $v_i \leftarrow d_i$
2: **for** Phase $s = 1$ **to** ∞ **do**
3: $PV_i[1..n] \leftarrow [\perp, \ldots, \perp]$;
4: (Round 1)
5: send v_i to all processes;
6: **for all** $j \in \Pi$ **do**
7: $PV_i[j] \leftarrow v_j$ (if $v_j \notin \{0,1\}$, store $\perp$ instead of v_j);
8: **if** $w \in \{0,1\}$ occurs at least $n-t$ times in $PV_i[1..n]$ **then**
9: $v_i \leftarrow w$;
10: **else**
11: $v_i \leftarrow \perp$;
12: (Round 2)
13: $SV_i[1..n] \leftarrow [\perp, \ldots, \perp]$;
14: send v_i to all processes;
15: **for all** $j \in \Pi$ **do**
16: $SV_i[j] \leftarrow v_j$ (if $v_j \notin \{0,1\}$, store $\perp$ instead of v_j);
17: **if** $w \in \{0,1\}$ occurs more than $2t$ times in $SV_i[1..n]$ **then**
18: $v_i \leftarrow w$;
19: $accept_i = False$
20: **else if** $w \in \{0,1\}$ occurs more than t times in $SV_i[1..n]$ **then**
21: $v_i \leftarrow w$;
22: $accept_i = True$
23: **else**
24: $v_i \leftarrow \perp$;
25: $accept_i = True$
26: (Round 3)
27: $EV_i \leftarrow [\perp, \ldots, \perp][\perp, \ldots, \perp]$;
28: send $SV_i[1..n]$ to all processes;
29: **for all** $j \in \Pi$ **do**
30: $EV_i[j][1..n] \leftarrow SV_j[1..n]$;
31: **if** i recovers from fault **then**
32: RECONSTRUCT(EV_i);
33: $c_i \leftarrow s \bmod n$;
34: **if** $w \in \{0,1\}$ occurs more than t times in $EV_i[c_i][1..n]$ **then**
35: $cv_i \leftarrow w$;
36: **else**
37: $cv_i \leftarrow 0$;
38: **if** $accept_i = True$ **then**
39: $v_i \leftarrow cv_i$;

系 1 より，完全グラフを対象とするとき，MOPT は最適なアルゴリズムである．

Algorithm 2 RECONSTRUCT(EV_i) on process i

1: **for all** $j \in \Pi$ **do**
2: **if** $w \in \{0,1\}$ occurs at least $n-t$ times in column j of EV_i **then**
3: $SV_i[j] \leftarrow w$;
4: **else**
5: $SV_i[j] \leftarrow \perp$;
6: **if** $w \in \{0,1\}$ occurs more than $2t$ times in $SV_i[1..n]$ **then**
7: $v_i \leftarrow w$;
8: $accept_i \leftarrow False$;
9: **else**
10: $accept_i \leftarrow True$;

3 高信頼性伝送を用いた移動ビザンチン合意アルゴリズム

本節では，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る．完全ネットワークの場合と違い，一般のネットワークには直接通信できないプロセスの組が存在する．このようなプロセス間で情報伝達を行う場合，複数ラウンドをかけて他のプロセスを経由することで情報を伝達しなければならないが，伝達にラウンド数を費やすほど，故障の移動により多くのプロセスが影響を受ける．

本節では，直接に通信できないプロセス間でも信頼できる情報伝達を実現する高信頼性伝送アルゴリズムを 2 つ提案する．これらのアルゴリズムでは，各正常プロセス i が，放送したい情報 m_i を入力としてある時刻に実行を開始すると，アルゴリズムが終了した時刻に正常である各プロセス j は $X_j[i] = m_i$ である配列を保持する．つまり，高信頼性伝送アルゴリズムは完全ネットワーク上での 1 ラウンドの通信を模倣できる．これらを MOPT と組み合わせると，一般の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を構成できる．

3.1 アルゴリズム DP-Byz

グラフの族 $\mathcal{G}(\alpha,\beta)$ を任意のノード間に長さが高々 β の素な道が少なくとも α 本存在するグラフの集合とする．Menger の定理 [3] によると，点連結度が d のグラフの任意のノード間には，少なくとも d 本の互いに点素な道が存在するため，$d \geq \alpha$ が成り立つ．本節で提案する DP-Byz (Disjoint Path Byzantine) は，$n > 3t$ かつ $\alpha > 2(\beta-1)t$ であるときに任意の $G \in \mathcal{G}(\alpha,\beta)$ 上で移動ビザンチン合意問題を解くアルゴリズムであり，$G \in \mathcal{G}(\alpha,\beta)$ 上の高信頼性伝達アルゴリズム DPT (Disjoint Path Transmission) と MOPT から構成される．

Dolev[4] は故障が移動しない場合について，点連結度が d のグラフ上で $d > 2t$ のときに，任意の 2 つのプロセス間で信頼できる通信を実現するアルゴリズムを提案し，これを完全ネットワーク上のビザンチン合意アルゴリズムと組み合わせることで，一般のネットワーク上でビザンチン合意問題を解いた．この通信アルゴリズムでは，プロセス i から j へ情報を送信するとき，プロセス i は，Menger の定理により保証されている d 本の点素な i–j 道に沿って，j へ送りたい情報を伝達する．パスの途中に故障プロセスが存在すると，故障プロセスは誤った情報を伝達するので，d 本の道にそって伝達された情報がプロセス j まで到達したとき，高々 t 本の道から来た情報だけが信頼できない．そこで，$d > 2t$ を仮定すると，プロセス j は多数決を取ることで，プロセス i が送信した情報を正しく受信できる．

DPT はこのアイデアを利用する．プロセス i 上の DPT の，引数は w_i と配列 X_i である．w_i は放送するメッセージであり，X_i は他のプロセス $j \in \Pi$ からのメッセージ w_j を格納するための配列である．あるラウンド r で各正常プロセス i が $\mathrm{DPT}(w_i, X_i)$ を実行した場合，DPT が終了するラウンド $r+\beta$ において，各正常プロセス j では $X_j[i] = w_i$ が成立する．

存在すると仮定されている，プロセス i から j への α 本の点素な道を π_{ij}^k $(1 \leq k \leq \alpha)$ と表し，各 $1 \leq l \leq |\pi_{ij}^k|$ について，$\pi_{ij}^k[l]$ を π_{ij}^k の l 番目の頂点とする．道の長さ $(|\pi_{ij}^k|-1)$ は高々 β であり，$\pi_{ij}^k[1] = i$, $\pi_{ij}^k[|\pi_{ij}^k|] = j$ である．配列 W_i は作業用の領域である．

故障から復帰したプロセスは，合意を達成後も合意維持性を満たすために，他のプロセスから受信した合意値を元に自身の合意値を訂正する．

ここで，プロセス i がプロセス $j \in N_i$ に 2 つ以上のメッセージを送る場合，これらのメッセージは送信前に 1 つのメッセージとして集約されるものと約束する．

補題 1 $G \in \mathcal{G}(\alpha,\beta)$ とする．G 上で各プロセス i がラウンド r_0 で w_i を合意値として保持し，$\mathrm{DPT}(w_i, X_i)$ を実行した場合，$\alpha > 2(\beta-1)t$ ならば任意の $i \notin F_{r_0+\beta}$ と $j \notin F_{r_0}$ に対して，ラウンド r_f 終了時に $X_i[j] = w_j$ が成り立つ．ただし，ラウンド $r_f = r_0+\beta$ は DPT が終了するラウンドである．さらに，ラウンド r_0 から r_f の間は，常に合意維持性が満たされる．

Algorithm DP-Byz: DP–Byz は MOPT の以下の 3 つの命令列を DPT の呼び出しに置き換えたものである．

- 5–7 行目 : $\mathrm{DPT}(v_i, PV_i)$
- 14–16 行目 : $\mathrm{DPT}(v_i, SV_i)$
- 28–30 行目 : $\mathrm{DPT}(SV_i, EV_i)$

定理 3 アルゴリズム DP-Byz は $G \in \mathcal{G}(\alpha,\beta)$ 上で $n > 3t$ かつ $\alpha > 2(\beta-1)t$ のときに移動ビザンチン合意問題を解く[2]．

DP-Byz が最適となる例を紹介する．

互いに素な頂点集合の族 $V_1, V_2, \ldots, V_k$ から成る完全 k 部グラフは，$G = (V_1 \cup V_2 \cup \ldots \cup V_k, E)$ である．ただし，$E = \cup_{i \neq j, i > j} V_i \times V_j$ である．一般性を失うことなく，$|V_1| \leq |V_2| \leq \ldots \leq |V_k|$ を仮定する．

系 2 $k \geq 3$ かつ $|V_1| = |V_2| = \ldots = |V_{k-1}| = 1$ とする．$n > 3t$ かつ $d > 2t$ のとき，完全 k 部グラフ $G = (V_1 \cup V_2 \cup \cdots \cup V_k, E)$ 上で，アルゴリズム DP-Byz は移動ビザンチン合意問題を解く．

[2] MOPT と同様に，実行中に絶対に故障しないプロセスが 1 つ存在することを仮定する．

Algorithm 3 DPT(w_i, X_i) on process i

1: $W_i[1..n][1..\alpha] \leftarrow [\perp \cdots \perp][\perp \cdots \perp]$;
2: $LAST_i \leftarrow \emptyset$
3: (Round r $(1 \leq r \leq \beta - 1)$)
4: **for** $r = 1$ to $\beta - 1$ **do**
5: **if** $r = 1$ **then**
6: **for all** $j \in \Pi$ **do**
7: **for all** $k = 1$ to α **do**
8: send (w_i, k, i, j) to $\pi^k_{ij}[2]$;
9: **for all** $j \in N_i$ **do**
10: send v_i to j;
11: **else**
12: **for all** $(w, k, h, j) \in NEXT_i$ **do**
13: send (w, k, h, j) to $\pi^k_{hj}[r + 1]$;
14: **for all** $j \in N_i$ **do**
15: send v_i to j;
16: $NEXT_i \leftarrow \emptyset$
17: **for all** messages (w, k, h, j) received **do**
18: **if** (w, k, h, j) arrived from $\pi^k_{hj}[r - 1]$ **then**
19: **if** $|\pi^k_{hj}| = r + 1$ **then**
20: $LAST_i \leftarrow LAST_i \cup \{(w, k, h, j)\}$;
21: **else**
22: $NEXT_i \leftarrow NEXT_i \cup \{(w, k, h, j)\}$;
23: **if** i recovers from fault and v arrived from more than t processes **then**
24: $v_i \leftarrow v$;
25: (Round β)
26: **for all** $(w, k, h, j) \in LAST_i$ **do**
27: send (w, k, h, j) to $\pi^k_{hj}[|\pi^k_{hj}|]$;
28: **for all** messages (w, k, h, i) received **do**
29: **if** (w, k, h, i) arrived from $\pi^k_{hi}[|\pi^k_{hi}| - 1]$ **then**
30: $W_i[h][k] \leftarrow w$;
31: **for all** $j \in \Pi$ **do**
32: **if** w occurs in $W_i[j][1..\alpha]$ more than $(\beta - 1)t$ times **then**
33: $X_i[j] \leftarrow w$;

証明 G の点連結度は $d = n - |V_k|$ である．一方，任意の2つの頂点について，長さが高々2の点素な道が少なくとも $n - |V_k| - |V_{k-1}| + 1 = n - |V_k| = d$ 本存在するため，このグラフは $\mathcal{G}(d, 2)$ に属する．定理3より，$n > 3t$ かつ $d > 2t$ のとき，DP-Byz は G 上で移動ビザンチン合意問題を解く． □

3.2 Algorithm KP-Byz

グラフ G の k 乗グラフ G^k は，G と同じ頂点集合を持ち，G 上で距離が高々 k であるような任意の2つの頂点間に辺を持つグラフである．本節では G^k 上でのアルゴリズム KP-Byz を提案する．DP-Byz と同様に，KP-Byz も G^k 上で高信頼性伝送を実現するアルゴリズム PerT(Permeate Transmission) と MOPT から構成される．

PerT のアイデアを説明する．G 上でプロセス i からの距離が高々 ℓ であるプロセスの集合を $L^G_j(\ell)$ とする．プロセス i が全プロセスにある情報 m_i を放送する方法を考える．まず，最初のラウンドで i は G^k 上の隣接プロセス，つまりプロセス $j \in L^G_i(k)$ に m_i を送信する．次のラウンドでは，プロセス $j \in L^G_i(k)$ は隣接プロセスに前のラウンドで受け取った m_i を送信する．このとき高々 t 個のプロセスは故障していて，隣接プロセスに m_i と異なるメッセージを送信する可能性がある．ここで，プロセス $j \in L^G_i(k+1)$ に着目すると，G^k の性質より，j は $L^G_i(k)$ に属するプロセスのうち，少なくとも k 個のプロセスと隣接しているから，j は少なくとも k 個のメッセージを受け取る．従って，$k > 2t$ を仮定すると，j が過半数以上受け取ったメッセージが m_i である．これ以降のラウンドについても同様の動作を行うことで，ラウンド r 終了時には，プロセス $j \in L^G_i(r + k - 1)$ は m_i を保持しており，G の直径を D_G とすると，ラウンド $D_G - k + 1$ 終了時にはすべての正常プロセスが m_i を保持していることが保証される．以上の動作を，各プロセス i について同時に行うことで，G^k 上で高信頼性伝送を実現する．

また，PerT も DPT 同様に，合意維持性を満たすための処理を各ラウンドで行っている．

各プロセス i は値（あるいはベクトル）w_i と配列 X_i を引数とする．w_i はプロセス i がすべてのプロセス $j \in \Pi$ に放送したいメッセージであり，X_i は他のプロセス $j \in \Pi$ から受け取った w_j を格納するための配列である．高々 t 個の移動ビザンチン故障が存在するとき，正常プロセス i が DPT(w_i, X_i) を実行すると，PerT 終了時に各正常プロセス $j \notin F_r$ では $X_i[j] = w_j$ が成り立つ．各プロセスは G^k, G, $L_j^G(\ell)$, D_G を知っているもの仮定する．

補題 2 $k > 2t$ を仮定する．ラウンド r_0 において，各プロセス i が値 w_i を保持し，G^k 上で PerT(w_i, X_i) を実行した場合，任意のプロセス $i \notin F_{r_f}$ と任意のプロセス $j \notin F_{r_0}$ について，ラウンド r_f では $X_i[j] = w_j$ が成り立つ．ただし，$r_f = r_0 + D_G - k$ は PerT が終了するラウンドである，また，合意達成後は，ラウンド r_0 から r_f の間では合意維持性は満たされる．

Algorithm KP-Byz: KP-Byz は MOPT の以下の 3 つの命令列を DPT の呼び出しに置き換えたものである．

- 5–7 行目 : PerT(v_i, PV_i)
- 14–16 行目 : PerT(v_i, SV_i)
- 28–30 行目 : PerT(SV_i, EV_i)

定理 4 アルゴリズム KP-Byz は任意のグラフ G の k 乗 G^k 上で $n > 3t$ かつ $k > 2t$ のときに移動ビザンチン合意問題を解く．

DP-Byz と同様，KP-Byz も，あるグラフ上では最適なアルゴリズムとなる．

系 3 G が長さ k 以上の "しっぽ" を持つグラフ *(lollipop* グラフやパスグラフ*)* とする．$n > 3t$ かつ $d > 2t$ のとき，KP-Byz は G^k 上で移動ビザンチン合意問題を解く．ただし，d は G^k の点連結度である．

3.3 モデルによる結果の違い

本節では，Buhrman 達 [2] のモデル上で，移動ビザンチン合意問題を考えた．このモデル上で，完全

Algorithm 4 PerT(w_i, X_i) on process i

```
1: (Round1)
2: for all j ∈ N_i^{G^k} do
3:     send (w_i, i) and v_i to j;
4: for all messages (w, j) received do
5:     if (w, j) arrived from j then
6:         X_i[j] ← w;
7: if i recovers from fault then
8:     if v arrived from more than t processes
       then
9:         v_i ← v;
10: (Round r (2 ≤ r ≤ D_G − k + 1))
11: for r = 2 to D_G − k + 1 do
12:     W_i[1..n][1..n] ← [⊥, ··· , ⊥][⊥, ··· , ⊥];
13:     for all j ∈ Π do
14:         if i ∈ L_j^G(k + r − 2) then
15:             send (X_i[j],j) to p ∈ N_i^{G^k};
16:     for all j ∈ N_i^{G^k} do
17:         send v_i to j;
18:     for all messages (w, j) received do
19:         if (w, j) arrived from p ∈ L_j^G(k+r−2)
            then
20:             W_i[j][p] ← w;
21:     for all j ∈ {p|i ∈ L_p^G(k + r − 1)} do
22:         if w occurs in W_i[j][1..n] more than t
            times then
23:             X_i[j] ← w
24:     if i recovers from fault then
25:         if receive v more than t times then
26:             v_i ← v;
```

ネットワークでは $n > 3t$，$\mathcal{G}(\alpha, \beta)$ では $n > 3t$ かつ $\alpha > 2(\beta - 1)t$，任意のグラフの k 乗では $n > 3t$ かつ $k > 2t$ のときに移動ビザンチン合意問題を解くことができる（表 1）．一方，故障がラウンドが変わる際に移動し，さらにプロセス自身が前のラウンドで故障していたか分からないモデル [8] では，故障の影響が大きくなるため，故障プロセス数に対して必要となるプロセス数，点連結度が大きくなる（表 2）．

表 1: [2] のモデルでの結果

	アルゴリズム	不可能性
完全ネットワーク	$n > 3t$	$n \leq 3t$
$\mathcal{G}(\alpha, \beta)$	$\alpha > 2(\beta - 1)t$	or
G^k	$k > 2t$	$d \leq 2t$

表 2: [8] のモデルでの結果

	アルゴリズム	不可能性
完全ネットワーク	$n > 6t$	$n \leq 6t$
$\mathcal{G}(\alpha, \beta)$	$\alpha > 2\beta t$	or
G^k	$k > 4t$	$d \leq 4t$

4 まとめ

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について考え，ふたつの移動ビザンチン合意アルゴリズム DP-Byz と KP-Byz を提案した．DP-Byz は，任意の頂点間に長さが高々 β の点素な道が少なくとも α 本存在するようなグラフにおいて $n > 3t$ かつ $\alpha > 2(\beta - 1)t$ のときに移動ビザンチン合意問題を解く．一方，KP-Byz は，$n > 3t$ かつ $k > 2t$ のときに移動ビザンチン合意問題を解くことができる．

どちらのアルゴリズムもあるグラフ上では故障数に対して最適であるが，$n > 3t$ かつ $d > 2t$ である任意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はまだ発見されていない．故障プロセス数のよりタイトな上界についてもわかっておらず，これらは未解決で残されている．

また，故障が移動するために必要なラウンド数が 1 より大きい場合の一般のネットワークでのアルゴリズムについても，今後の課題となっている．

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] 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.

[3] R. Diestel, "Graph Theory," Springer-Verlag, 1997.

[4] D. Dolev, "The Byzantine generals strike again," *J. Algorithms,* 3, 14–30, 1982.

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

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

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

[8] 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.