社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－DPS－130 (30)
2007－CSEC－ 36 (30)
2007／3／1

トポロジ変化の影響を抑えたモバイルアドホックネットワーク向け自己安定相互排除プロトコル

西川元， 山内由紀子，大下福仁， 角川裕次， 増澤利光
大阪大学大学院情報科学研究科

自己安定プロトコルは，任意の初期状況から実行を開始しても，いずれ正しい状況に収束するという性質を持ち，一時故障に対する耐性とネットワークトポロジ変化に対する適応性に優れている．しかし，自己安定プロトコルをモバイルアドホックネットワークのような環境に適用すると，正しい状況に収束する前にトポロジ変化が発生し，正しい状況に収束できない可能性がある．そのため，トポロジ変化へのより高度な対応が必要となっている．Chen ら [1] はモバイルアドホックネットワーク上での自己安定相互排除プロトコルを提案している．Chen らの手法はノードの移動によるトポロジ変化に対して優れた適応性を持つが，ノードの離脱によるトポロジ変化に対する適応性は十分ではない．本稿では Chen らの手法を改善し，ノードの移動，離脱の両方に対して優れた適応性を持つ自己安定相互排除プロトコルを提案する．

A self-stabilizing mutual exclusion protocol minimizing effect of topology changes for mobile ad hoc networks

Gen Nishikawa， Yukiko Yamauchi , Fukuhito Ooshita， Hirotsugu Kakugawa , and Toshimitsu Masuzawa
Graduate School of Information Science and Technology, Osaka University

A Self-stabilizing protocol converges to the desired behavior regardless of an arbitary initial configuration. Thus, a self-stabilizing protocol has strong fault tolerance. However, to apply self-stabilizing protocols in dynamic networks such as mobile ad hoc networks, it is necessary to highly adapt topology changes. Chen et. al. introduced a self-stabilizing mutual exclusion protocol for mobile ad hoc networks that is adaptive for node mobility. However, it has less adaptability for node departure[1]. In this paper we propose a self-stabilizing mutual exclusion protocol that improves adaptability for node departure of [1], and adaptability for node mobility and departure is evaluated by simulation.

1 はじめに

自己安定プロトコルは，任意の初期状況から開始しても，有限時間で正しい状況に収束する性質を持つ，一時故障に対する耐性とトポロジ変化に対する適応性に優れたプロトコルであり，Dijkstra[2] により提案された．

近年，**モバイルアドホックネットワーク**が発達している．モバイルアドホックネットワークは移動ノードのみで構成されたネットワークで，各ノードは無線通信範囲内にあるノードとのみ通信できる．各ノードはネットワーク中を自由に移動・参加・離脱を行ない，トポロジ変化が頻発するという特徴を持つ．

既存の自己安定プロトコルの多くはモバイルアドホックネットワークに適用しても，トポロジ変化の頻発によりシステムが永久に収束しない可能性がある．

増澤ら [3] は自己安定プロトコルについて，トポロジ変化に応じた正しい状況への収束に必要な実行ステップ数に対する定量的な評価法を提案し，リング上での自己安定相互排除プロトコルに対して適用した．

Vasudevan ら [4] はモバイルアドホックネットワークにおける，弱自己安定リーダ選択アルゴリズムを提案した．Tholeyre ら [5] はモバイルアドホックネットワークにおける，連結 k 支配集合を構成する自己安定プロトコルを提案した．[4, 5] のプロトコルはノード間の情報交換を頻繁に行なうことにより，正しい状況に高速に収束することを可能としている．しかし，トポロジ変化が発生し続けることにより，正しい状況へ収束することが保証できないという問題はまだ残されている．

角川ら [6] はトポロジ変化に対して，素早く安全性を回復し，そこから最適な状況へ収束するという，安定収束を定式化し，この性質を満たす自己安定極小支配集合プロトコルを提案した．

Chen ら [1] はモバイルアドホックネットワークにおける相互排除問題を解く自己安定プロトコルを提案した．相互排除はファイルなどの共有リソースに対して排他的なアクセスを保証し，グループコミュニケーションの実装 [7] などに用いられる．Chen ら

の手法は状況が正しい状況に収束しているときに，任意のトポロジ変化が発生しても，正しい状況に収束する．しかし，Chen らの手法はノードの移動・参加に対する適応性は高いが，ノードの離脱がシステムに与える影響が大きく，ノードの離脱に対する適応性は十分とはいえない．

本稿では Chen らの手法を改善し，Chen らの手法のノードの移動・参加に対する適応性を維持したまま，ノードの離脱に対する適応性を向上した自己安定相互排除プロトコルを提案する．

まず，2 節でシステムモデルを定義し，3 節で Chen らの手法 [1] の概要を説明する．4 節では Chen らの手法の問題点を改善する提案手法を示す．5 節でシミュレーション実験により Chen らの手法と提案手法の評価を行ない，提案手法の有効性を示す．

2 システムモデル

モバイルアドホックネットワーク $G = (V, E)$ はノード集合 V とリンク集合 E で定義される．p_i,p_j 間にリンク (p_i, p_j) が存在するとき，ノード p_i, p_j は隣接すると呼ぶ．リンク集合 E，ノード集合 V は動的に変化する．V と E の変化を以下ではトポロジ変化と呼ぶ．ノードを頂点，リンクを辺とすると，モバイルアドホックネットワーク G はグラフとみなせるため，グラフに対する用語をモバイルアドホックネットワークについても用いる．各ノードには固有の識別子が存在し，システム中のノード数を n と表記する．V は動的に変化するため，n の値も動的に変化する．また，通信モデルにはメッセージパッシングモデルを用い，メッセージが送信されてから受信されるまでの通信遅延の上限を d とする．通信中にメッセージが消失することはあるが，改変されることはないとする．ノード p_i はローカル変数を持ち，p_i の**状態**は全ローカル変数の値により表される．各ノード p_i はノード数の上限 N を知っているものとする．システムの**状況**は全ノードの状態の集合 S，伝送中のメッセージの集合 M，モバイルアドホックネットワーク G の 3 組 (S, M, G) で定義する．

システムの**実行**はイベントとシステムの状況の交替列により表す．**イベント**とは，あるノードのプロトコルの計算，もしくはトポロジ変化から成る．ノード p_i におけるプロトコルの計算とは p_i の現在の状態と受信したメッセージから p_i の次の状態を計算し，メッセージを送信するものである．プロトコルの計算はメッセージの受信もしくはタイマーの値が 0 になることにより発生する．各ノードは進行速度が互いに同じである時計とタイマーを持っており，タイムアウトにより動作ができると仮定する．トポロジ変化は連続する 2 つのプロトコルの計算間に発生する．状況 c_{k-1}，c_k を相異なる状況とし，イベント e_k により c_{k-1} から c_k に遷移するとする．この時，$c_{k-1} \xrightarrow{e_k} c_k$ と表記する．プロトコルの計算 e_{l-1}，e_{l+1} の間に起こるトポロジ変化 e_l は，システムのトポロジを，c_{l-1} のトポロジから c_l のトポロジに変化させる．e_l はトポロジを変化させないこともある．$E = c_0 \xrightarrow{e_1} c_1 \xrightarrow{e_2} \ldots$ と表記し，E をシステムの実行という．実行 E が無限長の系列である時，E は無限の実行であるという．

相互排除問題とは，あるリソースに対して，2 つ以上のノードが同時にリソースにアクセスすることがないように制御する問題である．以下では，ノード p_i がリソースへアクセスする状態を p_i が**クリティカルセクションに入る**と呼ぶ．

相互排除問題を解くプロトコルに要求される性質として，排他的制御，デッドロックなし，ロックアウトなしがある．**排他的制御**とは任意の状況においてクリティカルセクションに入っているノードはたかだか 1 つであるという性質である．**デッドロックなし**とは，無限の実行においてはいずれかのノードが無限にしばしばクリティカルセクションに入ることができるという性質である．**ロックアウトなし**とは，無限の実行においてはどのノードも無限にしばしばクリティカルセクションに入ることができるという性質である．

自己安定プロトコルとは任意の状況から実行を開始しても，システムはいずれ正しい状況に収束するプロトコルである．

ある状況 c においてクリティカルセクションに入っているノードの集合を $in\_CS(c)$ と表す．ある状況 c_k にシステムに存在するノード集合を V_k と表し，$\forall k : V_k$ の和集合を V_{all} と表す．

定義 1 システムが取りうる全ての状況の集合を Γ とする．プロトコル P が自己安定相互排除プロトコルであるとは，以下の状況を満たす C が存在する．C の状況を正しい状況と呼ぶ．

収束性：任意の $c_0 \in \Gamma$ より実行を開始しても，有限時間で C に含まれる状況に到達する．

閉包性：任意の正しい状況 $c \in C$，任意の状況 $c' \in \Gamma$ について，$c \xrightarrow{e} c'$ であれば，$c' \in C$．

排他的制御：すべての正しい状況 $c \in C$ について，$|in\_CS(c)| \leq 1$．

本稿では自己安定相互排除プロトコルのクラスとしてデッドロックなし自己安定相互排除プロトコル，ロックアウトなし自己安定相互排除プロトコルを定義する．

定義 2 自己安定相互排除プロトコル P が正しい状況 C についてデッドロックなし自己安定相互排除プロトコルであるとは，ある状況 $c \in C$ からはじまる

P の実行 $E = c_0 \stackrel{e_1}{\rightarrow} c_1 \stackrel{e_2}{\rightarrow} \ldots$ について，以下の条件を満たすことである：

$$\forall k \geq 0, \exists l \geq k : |in\_CS(c_l)| = 1$$

定義 3 自己安定相互排除プロトコル P が正しい状況 C についてロックアウトなし自己安定相互排除プロトコルであるとは，ある状況 $c \in C$ からはじまる P の実行 $E = c_0 \stackrel{e_1}{\rightarrow} c_1 \stackrel{e_2}{\rightarrow} \ldots$ について，以下の条件を満たすことである：

$$\forall p_i \in V_{all}, \forall k \geq 0, \forall l \geq k : p_i \in V_l \Rightarrow \exists m \geq k : in\_CS(c_m) = \{p_i\}$$

定義 3 の条件は，無限に V に含まれる任意のノード p_i について，p_i だけがクリティカルセクションに入っているという状況が無限に存在することを表す．

$\forall k \geq 0 : p_i \in V_k$ となる p_i が少なくとも 1 つは存在するとき，定義 2，3 より，ロックアウトなし自己安定相互排除プロトコルはデッドロックなし自己安定相互排除プロトコルであることは自明である．

本稿で提案する自己安定相互排除プロトコルは，トークンの巡回性能により，デッドロックなし，もしくはロックアウトなしのいずれかの性質を保証する．

3 Chen らによるプロトコル

本節では提案手法で改良の対象にしている Chen らの手法 [1] の概要と問題点について述べる．

Chen らの手法は**トークン**と呼ばれるクリティカルセクションに入る特権をネットワーク中に巡回させる．トークンはただ 1 つだけ存在する特別なノード p_0 が定期的に生成する (以下 p_0 以外のノードを一般ノードと呼ぶ)．p_0 はシステムに常に存在すると仮定する．トークンは複数存在することもあるが，以下で述べる**仮想リング**と**フェイズ**を用いて有効なトークンを 1 つに限定し，有効なトークンを持つノードのみがクリティカルセクションに入ることができる．仮想リングはクリティカルセクションに入るノードの順番を制御し，フェイズはノードがクリティカルセクションに入る回数を制御する．これにより，各ノードは仮想リングに定められた順に 1 フェイズで 1 度ずつクリティカルセクションに入る．

3.1 トークン巡回プロトコル:LRV_L

Chen らの手法ではトークン巡回プロトコルに LRV_L を用いる [1]．トークンが，あるノードから隣接する別のノードに移動することを，1 ホップ移動すると呼ぶ．LRV_L は全てのノードを巡回するために同一のノードに複数回移動することもありうる．文献 [8, 9] で LRV_L は動的なネットワークでは，$O(n)$ ホップで全てのノードを 1 回以上訪問する巡回を行なうことがシミュレーションにより示されている．各トークンは移動できるホップ数の上限 L を持ち，トークンは生成されてから L ホップ後に消滅させる．

3.2 排他的制御

本節では排他的制御の方法について述べる．

3.2.1 フェイズ・仮想リング・トークン

1 フェイズはフェイズ開始から仮想リングに含まれるノード全てがちょうど 1 回ずつクリティカルセクションに入るまで，もしくは後述するタイムアウトが発生するまでの実行の部分系列を指す．各ノードは局所変数でフェイズ番号を持ち，これは $[0..M_{tid}]$(M_{tid} はフェイズ番号の上限) の範囲の整数値をとる．フェイズ番号に対する演算は全て $(M_{tid}+1)$ を法とする演算である．p_0 の持つフェイズ番号をシステムの現在のフェイズとする．

仮想リングは p_0 が保持するノードの識別子の順列である．仮想リングに含まれるノードが現在のフェイズの相互排除参加ノードであり，各ノードは仮想リングに出現する順にクリティカルセクションに入る．

トークン t はフェイズ番号 $t.phase$，仮想リング $t.ring$，訪問リスト $t.visit$，経路リスト $t.route$ の情報を持つ．訪問リストはノードの識別子の集合であり，そのフェイズでクリティカルセクションに入ったノードの集合を表す．経路リストはノードの識別子の順列であり，トークンの巡回経路を表す．フェイズ番号はトークンが生成されたフェイズを表し，仮想リングはクリティカルセクションに入るノードの順を表す．トークンから情報を得るために p_0 も訪問リスト $visit_0$ と経路リスト $route_0$ を持つ．

3.2.2 一般ノードの動作

一般ノード p_i が持つフェイズ番号 $phase_i$ は p_i が最後にクリティカルセクションに入ったフェイズを表し，$phase_i + 1$ が p_i が次にクリティカルセクションに入るフェイズである．ノード p_i がトークン t を受信した時，以下の処理を行なう．

1: $t.ring$ から $t.visit$ 中のノードを除いた順列の先頭のノードが p_i ならば以下を実行．
  1-1: $t.visit$ に p_i を追加
  1-2: $t.phase = phase_i + 1$ ならば以下を実行．
    1-2-1: クリティカルセクションの実行．
    1-2-2: $phase_i$ を $t.phase$ に更新．
2: $t.route$ に p_i が含まれなければ，
  $t.route$ の末尾に p_i を追加．
3: LRV_L に従って t を隣接ノードに送信．

3.2.3 p_0 の動作

p_0 は自分が持つ仮想リング $ring_0$ とフェイズ番号 $phase_0$ を持つトークンを定期的に生成する．p_0 と同じフェイズ番号 $phase_0$ を持つトークン t を受信すると以下の処理を行なう．

1: $visit_0$ に $t.visit$ の要素を追加．

2: $route_0$ に $t.route$ を代入.
3: $visit_0$ と $ring_0$ 中のノードが一致すれば,
フェイズ終了. そうでなければ,
LRV_L に従って隣接ノードに t を送信.

$ring_0$ 中の全てのノードがクリティカルセクションに入るか, フェイズ開始から一定時間経過すると次のフェイズを開始する. 一定時間の経過によるフェイズの終了をタイムアウト発生と呼ぶ. タイムアウト発生時は以下の処理を行なう.

1:$ring_0$ から $visit_0$ に含まれないノードを削除.
2:トークンが全て消滅するまで (Ld 間) 待機.

$visit_0$ に含まれるノードはクリティカルセクションに入ったノードであるため, タイムアウト発生時はクリティカルセクションに入らなかったノードが $ring_0$ から除かれる.

新しいフェイズの開始は以下のように行なう.

1: フェイズ番号 $phase_0$ を $phase_0 + 1$ に更新.
2: p_0 がクリティカルセクションへ入る.
3: $route_0$ から新しい $ring_0$ の順を決定.

ノードは, 一定時間クリティカルセクションに入らなければ, 仮想リングに含まれないと判断する. その後, p_0 へ要求を送信し, その要求が p_0 に処理されれば, p_0 はそのノードを仮想リングに含める (詳細は割愛).

3.2.2 節の 1~1-1 の処理により, 各ノードは仮想リングの順でのみクリティカルセクションに入り, 1-2~2 の処理により, 同一フェイズでクリティカルセクションに入るのはたかだか 1 回である. これにより, 排他的制御が保証される.

3.3 問題点

仮想リング中のノード p_i がクリティカルセクションに入らずに離脱した場合を考える. p_i はトークンの訪問リストに入らないため, 仮想リング中で p_i より後方のノードは, クリティカルセクションに入ることができない. このため, これらのノードは p_0 でタイムアウト発生時に, 仮想リングから削除される. これらのノードは仮想リング参加要求を p_0 に送信し, 要求が p_0 に処理されるまで相互排除に参加できない.

図 1 でフェイズ k での p_0 の仮想リングが $ring_0 = \langle p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7 \rangle$ であり, p_2 がクリティカルセクションに入る前に離脱したとする. このとき, $ring_0$ で p_2 より後方に存在するノード $p_3, \ldots, p_7$ はフェイズ k ではクリティカルセクションに入ることができず, タイムアウト発生時に $ring_0$ から削除される. よって, フェイズ $k+1$ での p_0 の仮想リングは $ring_0 = \langle p_0, p_1 \rangle$ となる.

仮想リングに含まれるノード数を n, 離脱が起こる確率が各ノードで同一とし, ある 1 つのノードが

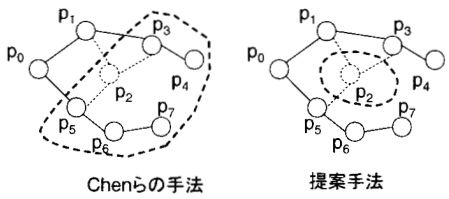

図 1: 離脱により削除されるノードの例

離脱することによりタイムアウトが発生したとする. このとき, Chen らの手法では期待値的に $n/2$ 個のノードが削除される. このように, Chen らの手法では 1 つのノードの離脱により, 多くのノードが仮想リングから削除され, しばらくの間クリティカルセクションに入ることができなくなる. これにより, Chen らの手法では仮想リングから削除されるノードとされないノードの間でクリティカルセクションに入る待ち時間の公平性に問題を生じさせる.

4 提案手法

Chen らの手法の問題点は, タイムアウト発生時に, そのフェイズでクリティカルセクションに入らなかったノード全てを仮想リングから削除することが原因である. 提案手法はタイムアウト発生時に, システムから離脱した可能性が高いノードのみを仮想リングから削除することによりこの問題を解決する. つまり, タイムアウト発生時に仮想リングの先頭にあるノード (以下このようなノードを離脱候補ノードと呼ぶ) とそのフェイズ中で 1 回もトークンを受信していないノードのみ削除する. 離脱をしていないノードが 1 フェイズでそのフェイズのトークンを必ず受信するとすると, 提案手法では, ある 1 つのノードの離脱によりタイムアウトが発生した時に仮想リングから削除されるノード数は必ず 1 つとなる. 各フェイズでトークンを受信しているノードの情報は, トークンの経路リストから集められる. p_0 が持つ情報にトークン受信ノードリスト rec_list を追加する.

また, Chen らの手法では, タイムアウト発生時にクリティカルセクションに入っていないノードはフェイズ番号が更新されず, 仮想リングから削除されなくても, 次のフェイズではクリティカルセクションに入ることができない. このため, 提案手法は仮想リング中のノードがトークンを受信した時に, 現在のフェイズのフェイズ番号から一定範囲内に収まるようにフェイズ番号を更新する.

4.1 一般ノードの動作

提案手法は, 一般ノード p_i がトークンを受信したときに, p_i のフェイズ番号を一定範囲内に収める処

```
1:  if (p_i ∈ t.ring)
2:    if(t.phase = phase_i + 2)
3:      phase_i := phase_i + 1
4:    if (p_i が t.ring から
5:      t.visit のノードを除いた順列で先頭)
6:      t.visit := t.visit ∪ {p_i}
7:      if(phase_t = phase_i + 1)
8:        クリティカルセクション実行
9:        phase_i := t.phase
10:   if((p_i ∉ t.route)
11:     ∧((t.phase = phase_i + 1)
12:       ∨(t.phase = phase_i)))
13:     t.route の末尾に p_i を追加
14: LRV_L でトークンを送信
```

図 2: 提案手法：一般ノードの動作

理を追加する．ノード p_i がトークン t を受信した時の動作を図 2 に示す．

まず，仮想リングに含まれるノードのフェイズ番号 $phase_i$ が更新されていない場合は，トークン t が持つフェイズ番号 $t.phase$ から一定の範囲内に収まるように処理する (1 ～ 3 行目)．

次に，Chen らの手法と同様の処理により，p_i がクリティカルセクションに入ることができるかを決定し，t を受信したノードの順を記録するために t の経路リスト $t.route$ に p_i を加える．ただし，仮想リングに含まれないノードや不正なフェイズ番号を持つノードは加えられない (4 ～ 13 行目)．

t による処理が終了すれば，LRV_L プロトコルに従い隣接ノードに t を送信する (14 行目)．

4.2 p_0 の動作

提案手法で p_0 が自分と同じフェイズ番号 $phase_0$ を持つトークン t を受信すると以下の処理を行なう．

1: $visit_0$ に $t.visit$ の要素を追加．
2: $route_0$ に $t.route$ を代入．
3: $receive\_list$ に $t.route$ の要素を追加．
4: $visit_0$ と $ring_0$ 中のノードが一致すれば，
フェイズ終了．そうでなければ，
LRV_L に従って隣接ノードに t を送信．

タイムアウト発時生は以下のように処理を行なう．

1: $ring_0$ から離脱候補ノードと
$receive\_list$ に含まれないノードを削除．
2: Ld 間待機．

その後，Chen らの手法と同様の新規フェイズ開始処理を行なう．

図 1 において，3.3 節と同様に，フェイズ k での p_0 の仮想リングが $ring = \langle p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7 \rangle$ であり，p_2 がクリティカルセクションに入る前に離脱したとする．また，p_2 以外の全てのノードはタイムアウト発生前にフェイズ k で生成されたトークンを受信しているとする．このとき，タイムアウト発生により $ring$ から削除されるのは p_2 のみである．よって，フェイズ $k+1$ での p_0 の仮想リングは $ring = \langle p_0, p_1, p_3, p_4, p_5, p_6, p_7 \rangle$ となり，提案手法は Chen らの手法よりも仮想リングから削除されるノード数が大きく減少していることがわかる．

4.3 提案手法の正当性

提案手法について以下の定理 1，定理 2 が成立する．

定理 1 提案手法はデッドロックなし自己安定相互排除プロトコルである．

文献 [8] では，LRV_L は，静的なネットワークにおいて，有限ホップ数 L_{max} で全てのノードを巡回することが示されている．

定理 2 以下の仮定が全て成立すれば，提案手法はロックアウトなし自己安定相互排除プロトコルである．

- トポロジ変化が発生しない．
- トークンの移動ホップ数の上限が $L \geq NL_{max}$

動的なネットワークでの提案手法の性能は 5 節のシミュレーション実験により示す．

5 シミュレーション実験

この節ではクリティカルセクションに入るまでの平均待ち時間についてのシミュレーションによる評価結果を示す．

5.1 評価指標

シミュレーション実験では各ノードのクリティカルセクションに入るまでの平均待ち時間について Chen らの手法 [1] と提案手法を比較する．**平均待ち時間**を (ノードが動作したラウンド数)/(クリティカルセクションに入った回数) と定義する．

シミュレーション実験で各ノードはラウンドにより動作する．全てのノードは 1 ラウンドに丁度 1 回ずつプロトコルの計算と移動を行ない，ラウンドを 1 単位時間とする．

全ノードの平均待ち時間が長い 5% にあるノードを下位ノードと呼ぶ．シミュレーション実験では，全ノードの平均待ち時間と下位ノードの平均待ち時間を比較することにより，各ノードの待ち時間の公平性を確認する．

5.2 評価環境

シミュレーション実験は，ノード数 $n = 100, 120, 140, 160, 180, 200$，フィールドサイズ $10n \times 10n$ のランダムにノードが配置されたユニットディスクグラフ

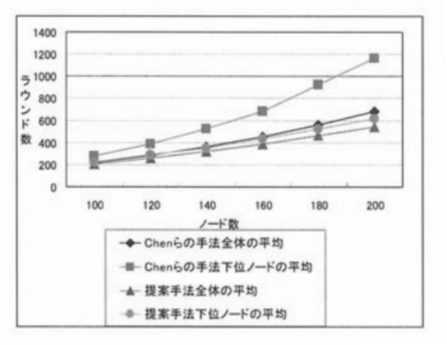

図 3: 平均待ち時間の差

に対して100回ずつ行なった．1回のシミュレーション時間は1620000ラウンドである．

各ノードは一定の確率でシステムから離脱し，ノードが送信するメッセージは1ラウンドで到着する．ノードの移動モデルは各ノードがランダムに目的地と速度を決定，目的地まで一定速度で移動，目的地に到着後一定時間停止という動作を繰り返すランダムウェイポイントモデルを用いる．各ノードは確率的に移動開始・停止を行なうとした．

また，シミュレーション開始からクリティカルセクションに入る前に離脱したノードやp_0と連結でないためシミュレーション中トークンを受信できないノードは，プロトコルの性能によらずクリティカルセクションに入らないため，評価の対象外とする．

その他のパラメータは以下の通りである．

- 通信半径：200
- ノード移動速度の上限：0.01/ラウンド
- 停止時間の上限：1200 ラウンド
- 停止後に移動する確率：1/100
- 移動中に停止する確率：1/600(/ラウンド)
- トークンのライフタイム L：$3n$ ラウンド
- 1フェイズの上限：$3n$ ラウンド
- 1ラウンドのノード離脱確率：1/1620000

5.3 評価結果

図3にシミュレーション実験の結果を示す．

まず，全体の平均待ち時間に着目すると，提案手法がChenらの手法よりも，平均待ち時間が短く，提案手法はChenらの手法よりも各ノードがクリティカルセクションに入る回数が多いことがわかる．

下位ノードの平均値と全体の平均値の差に着目すると，Chenらの手法ではノード数が増えるに従って差が大きくなることがわかる．これは，ノード数が増えれば離脱ノードが発生する確率が増え，仮想リングから削除されるノードの平均待ち時間が長くなるためであると考えられる．それに対し，提案手法では，離脱ノードが発生しても仮想リングから削除されるノード数が少ないため，ノード数が増加しても下位ノードの平均値と全体の平均値との差は大きく広がらないことがわかる．

以上から，Chenらの手法の問題点であった，各ノードの待ち時間の公平性の問題を提案手法により改善していることがわかる．

6 おわりに

本稿では既存のChenらの手法をもとに，ノードの離脱に対する適応性を向上した自己安定相互排除プロトコルを提案した．シミュレーションにより，提案手法がChenらの手法の問題点を改善していることを確認した．

今後の課題は，解析により各ノードの待ち時間の公平性がどの程度向上したのかを理論的に評価することである．

謝辞 本研究の一部は，文部科学省21世紀COEプログラム（研究拠点形成費補助金）の研究助成，日本学術振興会科学研究費補助金（基盤研究(B)15300017，基盤研究(B)17300020），大川情報通信基金，文部科学省科学研究費補助金（特定領域研究16092215，若手研究(B)18700059），および，総務省戦略的情報通信研究開発推進制度（SCOPE）によるものである．

参考文献

[1] Y. Chen and J. Welch. Self-stabilizing dynamic mutual exclusion for mobile ad hoc networks. JPDC ,vol 65, pp. 1072-1089, 2005.

[2] E. Dijkstra. Self-stabilizing systems in spite of distributed control. Communications of the ACM , vol. 17, no. 11, pp. 643-644, 1974.

[3] T. Masuzawa and H. Kakugawa. Self-stabilization in spite of frequent change of networks:Case study of mutual exclusion on dynamic rings. Proc. of SSS, LNCS 3764, pp. 183-197, 2005.

[4] S. Vasudevan, J. Kurose and D. Towsley. Design and analysis of leader election algorithm for mobile ad hoc networks. Proc. of the 12th ICNP, pp. 350-360, 2004.

[5] F. Theoleyre and F. Valois. About the self-stabilization of a virtual topology for self-organization in ad hoc networks. Proc. of the 7th SSS, pp. 214-228, 2005.

[6] H. Kakugawa and T. Masuzawa. A self-stabilizing minimal dominating set algorithm with safe convergence. Proc. of the 8th APDCM , pp 263-271 , 2006.

[7] M. Kwon and S. Fahmy. Topology-aware overlay networks for group communication. Proc. of the 12th NOSSDAV, pp. 127-136, 2002.

[8] N. Malpani, Y. Chen, N. Vaidya and J. Welch. Distributed token circulation in mobile ad hoc networks. IEEE Transactions on Mobile Computing, vol. 04, no. 2, pp. 154-165, 2005.

[9] N. Malpani, N. Vaidya and J. Welch. Distributed token circulation in mobile ad hoc networks. Proc. of the 9th ICNP, 2001.