A-015

アドホックネットワーク向け 自己安定トークン巡回アルゴリズムの実験的評価

Evaluation of Self-Stabilizing Token Circulation Algorithms for Mobile Ad-hoc Networks

梶原 大輔 † 角川 裕次 †
Daisuke Kajiwara Hirotsugu Kakugawa

1. はじめに

近年，携帯端末の普及，無線通信技術の発達に伴い無線端末のみで既存のインフラを必要としない一時的なネットワークを構築するアドホックネットワークに関する研究が盛んに行われている．アドホックネットワークではノードの移動によるトポロジーの変化が頻繁に起こることや，集中管理を行うサーバが存在しないという問題点から，アドホックネットワークを構築する上で，ネットワークの情報を保持したままプロセス間を移動するモバイルエージェントの利用が有効とされており，このようなアルゴリズムとしてトークンを用いる自己安定トークン巡回アルゴリズムが提案されている．自己安定アルゴリズム [1] はネットワークの初期状態に何も仮定せずに有限時間内に問題を解くことができる分散アルゴリズムである．自己安定アルゴリズムの動作は，安定するまでと安定後の2つに分けて考えることができる．安定後の動作については解析が比較的容易だが，安定するまでの動作は解析が難しい．そこで，本稿では，自己安定トークン巡回アルゴリズムを計算機上でシミュレーションを行い評価を行う．

2. Dynamic Reconfigulation Tolerant Self-Stabilization

アドホックネットワークではノードの移動によるネットワークトポロジの頻繁な変化が考えられるため，アドホックネットワーク向けの分散アルゴリズムを考える上で，耐故障性は重要な要素となる．しかし，従来の self-stabilization では安定状態までには十分長い時間が必要であり，アドホックネットワークの様に頻繁にネットワークトポロジに変化が起きる場合での使用は困難である．そこでこの問題に対して Dynamic Reconfigulation Tolerant self-stabilization が提案されている [3]．これはネットワークの変化の範囲に制約を与えることにより，アルゴリズムの動作保証を行うものである．

3. 評価アルゴリズム

本研究で評価の対象とするアルゴリズムは，文献 [2](以下 RWssGC) と文献 [3] (以下 DRTssTC) の2つである．両アルゴリズムともトークンを用いる自己安定アルゴリズムであり，以下に両アルゴリズムの概要を示す．

DRTssTC アルゴリズムは Initiator が実行する *Initiator thread* とトークンを受け取ったプロセスが実行する *Token thread* からなり，Initiator 以外は局所状態の保持が不要なステートレスなアルゴリズムである．以下にその概要を示す．まず Initiator がトークンを生成し，トークンがグラフ上を深さ優先探索を行うことによりネットワーク中のプロセスを把握し，spanning tree を形成する．その後トークンが Initiator に戻って来るたびに辺の更新を行い．ネットワークに変化が生じないネットワークにおいては最終的に minimum spaning tree に収束する．またリンクに変化が生じた場合は，通信できなくなったプロセスを根とする部分木をトークンの保持する木情報から削除し，再びトークンの巡回を続ける．一方，*RWssGC* アルゴリズムはトークンを保持しているプロセスは近隣のプロセスの中からプロセスをひとつランダムに選びこのプロセスにトークンを送る。これを繰り返すことによってネットワーク中のプロセスを巡回するアルゴリズムである．

4. 実験

評価アルゴリズムを Java 言語 (Java 2 SDK SE v.1.4.2) によって実装し，Linux kernel 2.4.22，Pentium4 3.00GHz 上でシミュレーションを行った．

4.1 概要

シミュレーションで用いるグラフは，Waxman 法 [4] により生成した，Waxman 法はランダムグラフ生成法のひとつとして，ネットワークアルゴリズムの評価に広く使用されている．本シミュレーションではまず，n 個のノードを $n \times n$ の座標平面の格子点上にランダムに配置する．そして，その中の任意の2ノード u, v 間に式 (1) で示される確率で辺を張る．

$$P(u,v) = \beta \exp \frac{-d(u,v)}{\alpha L} \quad (1)$$

ここで，$d(u,v)$ は2ノード間のマンハッタン距離，L は2ノード間の最大距離，α,β は $(0,1]$ の範囲で与えられる実数値で，これらの値によってグラフの辺の数が調整される．またリンクコストは2ノード間のマンハッタン距離とし，各ノードを少なくとも一度巡回するまでを1ラウンドとする．

評価内容　アルゴリズム *DRTssTC* と *RWssGC* において以下の内容をシミュレーションにより評価する．

1. ノードがネットワーク中を移動する速度を 0, 3, 10 とした場合のノード数別，平均 degree 別でのラウンドの長さ
2. 参加・離脱するプロセスの数を 1,10,20 とした場合の両アルゴリズムでのイベント後再び全てのプロセスを巡回するために必要なステップ数

について評価を行う．

†広島大学大学院工学研究科情報工学専攻

4.2 結果と考察

表 1: 全プロセス巡回までのステップ数：プロセスの参加・離脱

		DRTssTC			*RWssGC*		
		number of nodes$N$ ($deg = 5$)					
		50	70	100	50	70	100
join	1	94	148	220	582	918	1393
	10	125	163	229	662	1045	1498
	20	148	185	249	738	1114	1540
leave	1	101	146	232	562	881	1350
	10	83	143	191	472	817	1297
	20	67	114	192	380	672	1217

表 2: 全プロセス巡回までのステップ数：ノード数別

		DRTssTC			*RWssGC*		
		number of nodes$N$ ($deg = 5$)					
		30	50	100	30	50	100
speed v	0	54	97	193	269	648	1307
	3	89	161	431	265	588	1317
	10	101	236	609	229	612	1373

表 3: 全プロセス巡回までのステップ数：平均 degree 別

		DRTssTC			*RWssGC*		
		Avg.degree ($N = 30$)					
		5	10	20	5	10	20
speed v	0	54	52	49	269	149	127
	3	89	66	53	265	150	125
	10	101	95	70	229	149	126

4.2.1 参加・離脱に関して

表 1 より，まず *RWssGC* アルゴリズムでは，プロセスの参加・離脱発生後のステップ数については，参加・離脱するプロセス数に関係なく，そのイベント後のノード数が同じ場合は同程度のステップ数が必要となる結果となった．これは *RWssGC* においてはトークンを持つプロセスは近隣の通信できるプロセスへランダムに送るために，ネットワークの形状の変化に関係なく，1 ラウンドにかかるステップ数はノード数に依存すると言える．また *DRTssTC* については，参加の場合，イベント後もトークン中の木は辺を加えるのみで，辺の削除は行わないので，static topology での同じノード数の振る舞いとほぼ同じであると考えられ，平均ステップ数は $2n$ 程度となると予想される．また離脱する場合については，通信できなくなったプロセスが存在するために，通信できなくなったプロセスを発見する度にトークン中の木の再構成を行うので，static topology における離脱後の同じノード数での 1 ラウンドに要するステップ数よりも増加する．

4.2.2 移動速度に関して

Static topologies($v = 0$)

表 2 より，*DRTssTC* では近隣の通信できるプロセスの中からリンクコストによって送信プロセスを選び，トークンを送るため，多くても $2n$ ステップで各ノードを一巡できると示されており．また *RWssGC* ではランダムウォークに基づきトークンがネットワーク中を巡回するため，各ノードを一巡するステップ数は n^3 であることが知られている．実験結果から移動速度が 0 の場合，高々 *DRTssTC* では示されている計算時間 $2n$ 程度で各ノードを一巡できていることが確認できる，また *RWssGC* では計算時間 n^3 に対して非常に短いステップ数で各ノードを一巡することができた．ノード数が最大でも 100 と少ないために，このような結果になったと考えられる．

Dynamic topologies($v = 3, 10$)

表 2 より，*RWssGC* について，ネットワークに変化が起きる場合でも同じノード数では同程度のステップ数が必要となる，これは前述の通りである．また *DRTssTC* においては移動速度の増加によって，平均ステップ数の増加が見られる．これはネットワークの変化により通信できなくなったプロセスを根とするトークン中の部分木を削除し，spanning tree の再構成という動作を多く行ったためである．

Avg. degree

表 3 より，*RWssGC* において，平均 degree が増えるとラウンドに要するステップ数は減少している．これは近隣のプロセスの増加によって，効率良く各プロセスを巡回した為である．*DRTssTC* においては，$v = 0$ の場合ステップ数はほぼ同じであるが，$v = 3, 10$ の場合平均 degree の増加に伴ってステップ数は減少する結果となった，このようになる理由はリンクの切断の発生が減少したためと言える．

5. おわりに

本稿では二つのアドホックネットワーク向けの自己安定トークン巡回アルゴリズムを計算機上でシミュレーションを行い様々なパラメータのもとでアルゴリズムの振舞いを確認した．その結果，*DRTssTC* アルゴリズムはトポロジの変化が頻繁に起こる場合であっても，*RWssGC* アルゴリズムと比べると全ノードを約 1/2 程度のステップ数で一巡でき，動的なネットワークの変化に耐え得るアルゴリズムであると言える．今後の課題としてネットワークトポロジの違い，また *DRTssTC* においてはトークンが巡回する経路をより短い距離で形成することから，送信電力を制御することによる省電力化といった特徴が考えられ，電力消費を考慮したアルゴリズムの評価が挙げられる．

参考文献

[1] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, Vol. 17, No. 11, pp. 643–644, 1974.

[2] S. Dolev, E. Schiller, and J. L. Welch. Random walk for self-stabilizing group communication in ad hoc networks. In *Proceedings of SRDS*, pp. 70–79, 2002.

[3] H. Kakugawa and M. Yamashita. Dynamic reconfiguration tolerant self-stabilizing token circulation in ad-hoc networks. LA Symposium, 2004.

[4] B. Waxman. Routing of multipoint connections. *IEEE J. on Selected Areas in Communications*, Vol. 6, No. 9, pp. 1617–1622, 1988.