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

2003－ＭＢＬ－27　(19)
2003－ＩＴＳ－15　(19)
2003／11／14

モバイルアドホックネットワークシミュレーションの規模適応性を向上させる技法の検討

廣森聡仁[†]　山口弘純[†]　安本慶一[††]
東野輝夫[†]　谷口健一[†]

ネットワーク規模の増大やネットワーク構造の複雑化に伴い，大規模ネットワークにおけるシミュレーションが必要とされている．しかし，計算機で使用できる資源は限られているため，シミュレートできるネットワークの大きさ，シミュレーションシナリオの規模は制限される．シミュレーションにおいて，計算機資源 (特にメモリ容量) を消費する要因の一つは，各ノードにおいて，到着したパケットを宛先ノード毎にどの隣接ノードに転送すべきかを示す，ルーティングテーブルであることが知られている．我々の研究グループでは，被覆木のデータ構造をもったルーティングテーブルと，従来のルーティングテーブルを組み合わせた被覆木併用ルーティングテーブルを構築することで，任意のルーティングを表現し，かつそのルーティングテーブルの容量を削減する方法を提案している．しかしながら，ルーティングテーブル容量削減に効果的な被覆木の構築には時間がかかるため，ネットワークトポロジが動的に変化し経路が変更する度に，最適な被覆木を求め，ルーティングテーブルの容量を小さく維持することは現実的ではない．本稿では，被覆木併用ルーティングテーブルを用いた方式を，モバイルアドホックネットワークのように，ネットワークトポロジが動的に変わるような環境においても適応できるよう，被覆木を部分的に再構築するアルゴリズムを提案する．提案手法の評価を行った結果，被覆木を再構築することで，単純なルーティングテーブルと比較し，ルーティングテーブルの容量を 10% 弱と小さく維持できることがわかった．

An Adaptive Routing Table Reconstruction Method for Mobile Ad-Hoc Network Simulation

Akihito Hiromori,[†] Hirozumi Yamaguchi,[†] Keiichi Yasumoto,[††]
Teruo Higashino[†] and Kenichi Taniguchi[†]

In simulating large-scale networks, due to the limitation of available resources on computers, the size of the networks and the scale of simulation scenarios are often restricted. Especially, routing tables, which indicate the directions to forward packets, are considered to consume memory space. We have proposed a new method to reduce the capacity of routing tables which is applicable to any routing strategy. In our method, an *algorithmic routing-based table* is used to represent a part of the given routing table to reduce the size of the routing table as well as to allow any routing strategy. Our method takes a lot of time to find a near-optimal algorithmic routing-based table. In this paper, we propose a new method to reconstruct an algorithmic routing-based table for route changes. Our experimental results have shown that our method could reduce about 90 % of the routing table size compared with a general routing table.

1. はじめに

近年，端末同士で通信することができるモバイルアドホックネットワーク上でのプロトコルやアプリケーションの研究が盛んに行われている．モバイルアドホックネットワークでは，モバイル端末間の通信は基地局を介さずに，端末同士で直接通信するか、もしくは他の端末を介して通信を行う．そのため，端末間の経路が端末の移動により切断されたり，新しい経路が生成される状況が頻繁に生じる．モバイルアドホックネットワークに限らず，通常のネットワーク上でのプロトコルやアプリケーションの性能評価の方法として，ネットワークシミュレーションが多く用いられている．ネットワークシミュレーションでは，計算機上でネッ

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†† 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

トワークを仮想的に構築し，実ネットワーク上におけるパケットの挙動をシミュレートすることで，プロトコルの評価を様々な面から行うことができる．

しかし，多くのネットワークシミュレータはネットワーク上の全てのノードの振舞いを計算機上でシミュレートするため，計算機能力によってシミュレートできるネットワーク規模や想定するシミュレーションシナリオが制限される場合が多い．既存のシミュレータでは，このような規模適応性の問題を改善する方法として，シミュレーションの並列化とネットワークの抽象化が主に用いられている．例えば，GlomoSim[1] では，使用する計算機数を増やすことでシミュレーションの並列化を行い，実行速度向上を図っている．また，ネットワークの振舞いを事細かにシミュレートするのではなく，一部の振舞いをシミュレートしないことで，ネットワークの振舞いを抽象化し，シミュレーションの効率を上げる方法もある．例えば，ns[2] では，各ノード間の複数ホップ通信を，伝送遅延が等しい単一ホップ通信に置き換え，ルータでのパケット転送処理を省略し，シミュレータの負荷軽減を図る機能を提供している．しかし，この方法では，各リンクにおけるパケットの流れを忠実にシミュレートしていることにはならず，評価目的によっては適さない場合もある．

シミュレータが計算機資源 (特にメモリ容量) を占有する主な要因の一つに，各ノードが各宛先ノードごとどの隣接ノードにパケットを転送するかを示す，ルーティングテーブルの容量が挙げられる．一般のネットワークシミュレーションでは，パケットの転送先を高速に求めるために，ネットワークトポロジとルーティングプロトコルから，各ノードにおいてどのようにパケットを転送するかを，あらかじめ求めておき，ルーティングテーブルとして保持しおく．一般的なルーティングテーブル (以下，汎用ルーティングテーブル) のデータ構造を図 1 に示す．このテーブルは，到着ノード X，宛先ノード Y，転送先ノード Z の三項を一つのエントリとしたエントリ集合であり，各エントリは，ノード X に到着した宛先ノード Y のパケットは，転送先ノード Z に転送すべきであることを示している．したがって，各エントリの参照は $O(1)$ で行える一方，全エントリ数は，ノード数を N とした場合，$N \times (N-1)$ となっており，ルーティングテーブルの容量は $O(N^2)$ となる．メモリ容量は限られているため，$O(N^2)$ で増加するルーティングテーブルの容量は，大規模ネットワークにおけるネットワークシミュレーションの妨げとなる場合も多い．

近年，ルーティングテーブルのデータ構造を工夫して，その容量を小さくし，シミュレーションの効率を向上させる方法がいくつか提案されている[3)~5)]．そのような方法の一つである被覆木ルーティング法[5] は，2 ノード間の経路が全て，全ノードを被覆するある一つの木 (被覆木) に含まれる場合のみ，ルーティングテーブルのデータ構造を被覆木そのものとする方法であり，転送先ノードを求める処理に $O(logN)$ の計算量がかかる一方，ルーティングテーブルの容量を $O(N)$ とすることができる．文献 5) では，全ての経路を被覆木に含まれるような経路に制限した場合も，多くの経路は最小ホップ数の経路となることを示しており，最小ホップに基づくルーティングプロトコル上でのシミュレーションに対しては，被覆木ルーティング法は有効であるとしている．しかし，一般のルーティングに適用することができず汎用性に欠ける．

到着ノード X	宛先ノード Y	転送先ノード Z
1	2	2
⋮	⋮	⋮
1	n	2
2	1	1
⋮	⋮	⋮
n	$n-1$	$n-1$

図 1 汎用ルーティングテーブルのデータ構造

我々が提案している被覆木併用ルーティング法[6] では，与えられた汎用ルーティングテーブルを，被覆木によるルーティングテーブルと，汎用ルーティングテーブルを組み合わせた被覆木併用ルーティングテーブルに変換することで，任意のルーティングにおけるルーティングテーブルの容量を削減する (図 2)．被覆木併用ルーティングテーブル全体の容量は，与えられた汎用ルーティングテーブルのエントリを，どの程度被覆木によるルーティングテーブルで表現できるかに依存する．文献 6) では，ルーティングテーブル容量の削減に有効な被覆木を求めるヒューリスティックな解法を提案し，その評価を行った結果，インターネットにおいて一般的なトポロジに対し，ルーティングテーブルの容量を 10 % 程度にできることがわかった．しかしながら，被覆木を求める処理に時間がかかるため，ネットワークトポロジが動的に変化し経路が変更する度に，最適な被覆木を求めることは困難であり，モバイルアドホックネットワークなどに適用することは困難であった．

本稿では，被覆木併用ルーティング法をもとに，ネッ

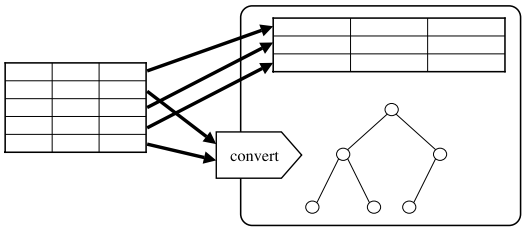

図 2 被覆木併用ルーティングテーブルのデータ構造

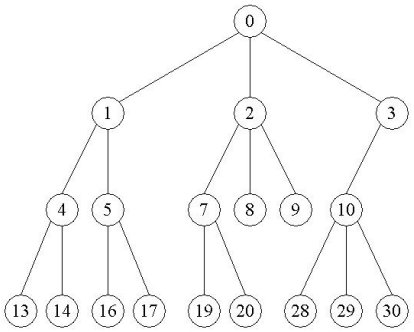

図 3 被覆木の例

トワークトポロジが動的に変わるような環境においても適応できるよう，被覆木を部分的に再構築する方法を提案する．その評価実験を行った結果，提案手法により被覆木を再構築することで，ルーティングテーブルの容量を 10% 弱と小さく維持できることがわかった．

以下，2 章では関連研究について述べる．3 章及び 4 章では提案方式の概要とその詳細についてそれぞれ述べ，5 章では評価実験の結果と考察を述べる．6 章では本稿のまとめと今後の課題について述べる．

2. 関 連 研 究

2.1 被覆木ルーティング法

被覆木ルーティング法[5] は，2 ノード間の全経路が，ある一つの被覆木に含まれる場合，各ノードに一定の規則にしたがって専用のノード番号を付与することで，ルーティングテーブル容量を削減する方法である．被覆木ルーティング法では，独自のノード番号を用いることで，各ノードに到着したパケットの宛先ノードのノード番号に対し，そのパケットの転送先ノードのノード番号を，ノード番号の付与規則に基づくアルゴリズムを用いて計算できる．ノード番号が付与された被覆木の例を図 3 に挙げる． 付与規則では，ノード番号 0 を付与したノードを被覆木の根としたとき，ノード番号 n の子ノードのノード番号は $n \times k + 1$ から $(n+1) \times k$ (ただし，k は被覆木の最大次数 - 1) となるように付与する．このノード番号により，与えられた 2 ノード間の親子関係があるかどうかを判別することができる．その結果，各宛先ノードごとに転送先ノードを汎用ルーティングテーブルのように直接保持しておく必要がなくなる（すなわち，アルゴリズムを用いて on-the-fly で計算できる）ため，転送先ノードを求める手間はかかる一方，ルーティングテーブルとしては，被覆木内のノード間の接続関係のみを保持すればよく，ルーティングテーブルの容量を $O(N)$ とすることができる．以下，被覆木ルーティング法で用いる木構造によるルーティングテーブルを，被覆木ルーティングテーブルと呼ぶ．

```
next_hop( A, B )
begin
  while ( B > 0 )
   B_parent = (B-1)/k
   if ( B_parent == A)
    return B
   end
   B = B_parent
  end
  return (A-1)/k
end
```

図 4 被覆木ルーティングにおける転送先ノード計算アルゴリズム

ノード番号 A に到着した，ノード番号 B 宛のパケットを転送すべき転送先ノードを計算するアルゴリズム next_hop(A, B) は図 4 のようになり，計算量は $O(logN)$ となっている． このアルゴリズムでは，宛先ノード B のノード番号が，到着ノード A の子孫か否かを，ノード B の先祖をたどる (ノード B からノード 0 のノードに向けて木をたどる) ことで判定し，そうである場合には，パケットを転送すべきノードとして，ノード A の子ノードのうち，ノード B の先祖にあたるノードを返す．一方，ノード 0 に到着するまでノード A を 発見できなかった場合，パケットを転送すべきノードとしてノード A の親ノードを返す．例えば，ノード 2 において，ノード 19 はノード 2 の子ノード 7 の子ノード (すなわち，ノード 2 の子孫ノード) として存在していると判断できるため，ノード 19 宛へのパケットはノード 7 へ転送する．一方，ノード 5 は，ノード 2 でないことが判定できるため，ノード 5 宛てのパケットはノード 2 の親ノードであるノード 0 に転送する．

2.2 被覆木併用ルーティング法

被覆木ルーティング法では，2 ノード間の経路は全

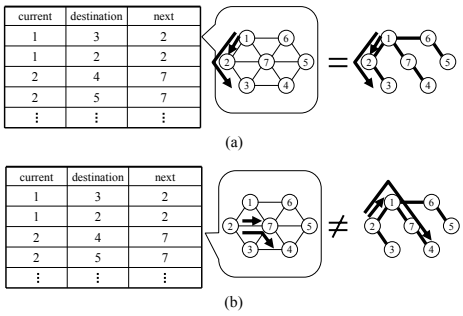

図 5 実現エントリ，非実現エントリの例

て被覆木上を通らなければならないため，任意のルーティングを表現できず汎用性に欠ける．我々が提案している被覆木併用ルーティング法[6] では，汎用ルーティングテーブルと被覆木ルーティングテーブルを併用することにより，与えられた汎用ルーティングテーブルで表現される任意のルーティングを表現できる．

入力として与えられた汎用ルーティングテーブルにおいて，到着ノード n_i，宛先ノード n_j であるようなエントリ ($< n_i, n_j >$ の二次組で表す) に示された転送先ノードを $next(n_i, n_j)$ とし，$< n_i, n_j >$ において，被覆木 st における被覆木ルーティングで求められる転送先ノードを $next_{sp}(st, n_i, n_j)$ とする．$next(n_i, n_j)$ と $next_{sp}(st, n_i, n_j)$ が同じであれば，エントリ $< n_i, n_j >$ は被覆木 st における被覆木ルーティングで実現できる．以降，$next(n_i, n_j)$ と $next_{sp}(st, n_i, n_j)$ が等しいエントリ $< n_i, n_j >$ を実現エントリ，そうでないエントリを非実現エントリと呼ぶ． 被覆木併用ルーティング法を図 5 を用いて説明する．被覆木 st は太線から構成されるとする．入力として与えられたルーティングテーブルでは，エントリ $< 1, 3 >$ の転送先ノードとしてノード 2 を示している．図 5 (a) では，被覆木 st をたどることにより求められた転送先ノード $next_{sp}(st, 1, 3)$ も 2 を示しており，$< 1, 3 >$ は被覆木 st を用いた被覆木ルーティングにより表現できる．よって，$< 1, 3 >$ は実現エントリであることがわかる．一方，図 5 (b) では，エントリ $< 2, 4 >$ について，st を用いた被覆木ルーティングの転送先ノード $next(st, 2, 4)$ としてノード 1 を示しており，$next(2, 4) = 7$ と異なっている．従って，$< 2, 4 >$ は非実現エントリであることがわかる．被覆木併用ルーティング法では，ある一つの被覆木を設定し，与えられた汎用ルーティングテーブルのエントリのうち，その被覆木を用いた被覆木ルーティングにより実現できるエントリは被覆木ルーティングテーブルで表現し，そうでないエントリはもとの汎用ルーティングテーブルで残してそのまま用いる．この際，被覆木ルーティングテーブルは，サイズが $O(N)$ の木構造により実現され，被覆木ルーティングで実現できるエントリ数に関わらず，その容量は固定である．そこで，被覆木ルーティングにより実現されるエントリ数をできるだけ大きくするような被覆木を発見することで，ルーティングテーブルの総容量削減を図っている．具体的には，ある辺を初期辺として含むような被覆木をグリーディに構築する手続きを，ネットワークの各辺を初期辺として適用する．得られた被覆木群のうち最も多くのエントリを実現する被覆木を，被覆木ルーティングで用いる被覆木とする．我々が提案したこのアルゴリズムは，テーブル容量削減にきわめて高い効果を発揮する (階層型ネットワークにおいて削減率 90 %)．一方で，この方式における被覆木構築の計算量は $O(k^2N^4)$ (k はネットワークの最大次数) であり，被覆木構築には時間がかかるため，経路やネットワークトポロジが頻繁に変化するようなネットワークシミュレーションには不向きであった．

3. 被覆木併用ルーティングテーブルの動的更新

本稿では，モバイルアドホックネットワーク等，ネットワークトポロジが動的に変化するネットワークシミュレーションを想定し，経路の変化に応じて，被覆木併用ルーティングテーブルを再構築することにより，ルーティングテーブル容量を少なく維持する方法を提案する．

ネットワークシミュレーション開始時には，被覆木併用ルーティング法により被覆木を構築する．以下，この被覆木を用いた被覆木併用ルーティングテーブルを rt (被覆木ルーティングテーブル部を t，汎用ルーティングテーブル部を gt) とする．提案方式では，経路が動的に変化するネットワークシミュレーションを想定し，経路の変化に従って 常に変更後の経路を正しく表現できるように，rt 内の t，gt を更新する．経路の変化は，エントリの追加，エントリの変更，エントリに示された転送先ノードの変更により表すことができる．それぞれについて，rt をどのように更新するか説明する．

新たに追加されるエントリ $< n_i, n_j >$ の転送先ノードを $next_{new}(n_i, n_j)$ とする．$< n_i, n_j >$ に対する rt の更新は以下のようになる．

- $next_{new}(n_i, n_j) = next_{sp}(t, n_i, n_j)$ である場合には，$< n_i, n_j >$ を被覆木ルーティングにより

表現する．

- $next_{new}(n_i, n_j) \neq next_{sp}(t, n_i, n_j)$ である場合には，$< n_i, n_j >$ を汎用ルーティングテーブルにより表現し (gt にいれる)，gt 内の $< n_i, n_j >$ の転送先ノードを $next_{new}(n_i, n_j)$ とする．

転送先ノードが変更されるエントリ $< n_v, n_w >$ について，経路変更前の転送先ノードを $next_{before}(n_v, n_w)$ とし，経路変更後の転送先ノードを $next_{after}(n_v, n_w)$ とする ($next_{before}(n_v, n_w) \neq next_{after}(n_v, n_w)$)．$< n_v, n_w >$ に対する rt の更新は以下のようになる．

- gt に $< n_v, n_w >$ が含まれ，$next_{after}(n_v, n_w) = next_{sp}(t, n_v, n_w)$ である場合には，そのエントリを被覆木ルーティングにより表現する (gt から削除する)．
- gt に $< n_v, n_w >$ が含まれ，$next_{after}(n_v, n_w) \neq next_{sp}(t, n_v, n_w)$ である場合には，gt 内の $< n_v, n_w >$ の転送先ノードを $next_{after}(n_v, n_w)$ におきかえる．
- t に $< n_v, n_w >$ が含まれる場合には，汎用ルーティングテーブルにより表現し (gt にいれる) し，gt 内の $< n_v, n_w >$ の転送先ノードを $next_{after}(n_v, n_w)$ とする．

削除されるエントリを $< n_x, n_y >$ とする．$< n_x, n_y >$ に対する rt の更新は以下のようになる．

- gt に $< n_x, n_y >$ が含まれる場合には，そのエントリを gt から削除する．
- t に $< n_x, n_y >$ が含まれる場合には，rt の更新はなにも行わない．

上記のように rt を更新することで，経路の変更を正しく表現できるものの，t は始めに与えられた汎用ルーティングテーブルに対し構築されているため，経路の変更後においても t で表現できるエントリの数が多いとは限らない．従って，経路の変更の度に，gt に含まれるエントリ数が多くなると考えられる．さらに，トポロジ変更に伴い経路が変化する場合には，t はその時点におけるネットワークトポロジ上での被覆木ではなくなるため，t で表現できるエントリの数はより少なくなる．

提案手法では，gt に含まれるエントリの数が増えてきた場合に，t の再構築を行う．被覆木併用ルーティングテーブル再構築アルゴリズムでは，被覆木ルーティングで用いる被覆木から，ある一つの辺を削除し，別の辺を挿入する処理を繰り返すことで，被覆木を再構築し，被覆木ルーティングにより実現されるエントリ数を増加させ，被覆木併用ルーティングテーブルの総容量を少なくする．

4. 被覆木併用ルーティングテーブル再構築アルゴリズム

被覆木併用ルーティングテーブル再構築アルゴリズムにおいては，以下が入力として与えられる．

- ネットワークトポロジを表したグラフ g
  グラフ g 上における辺の存在は，辺の両端のノードが互いの通信範囲内に存在し，それらノード間で通信可能であることを示す．
- 被覆木併用ルーティングテーブル $rt = (t, gt)$

被覆木併用ルーティングテーブル再構築アルゴリズムでは，入力として与えられた t を部分的に再構築し，被覆木ルーティングにより実現されるエントリ数を増やすことで，ルーティングテーブルの総容量を小さくする．t の再構築は，入力として被覆木 t_i と t_i 上のある一つの辺 e_d を与えたとき，t_i から e_d を削除し，別の一辺を加えることにより得られる被覆木のうち，最も実現エントリ数が大きな被覆木 t_o を求める手続きを複数回適用することにより行われる．この手続きの説明を 4.1 節で行う．

一般に，g は動的に変化し，t が g 上の被覆木となっていない場合には，t における被覆木ルーティングでは，g に存在しないような辺を通る経路を表現していることが考えられる．このような経路は rt において指定されないため，t には無駄な部分が存在する．そこで，提案アルゴリズムでは，まず，t に含まれ，かつ g に含まれない全ての辺について，それぞれを e_d として上記の手続きを適用し，t を g 上の被覆木として再構築する．その後，t に全ての辺について，上記の手続きを適用し，得られた被覆木群のうち，最も実現エントリ数がよくなるものを被覆木とする．

4.1 削除辺に対し最適な被覆木を求める手続き

手続きの入力として与えられる被覆木を t_i，t_i から取り除く辺 (削除辺) を e_d で表す．t_i から e_d を削除すると，t_i は二つの木 t_1, t_2 に分割される．本手続きは，t_1, t_2 を部分木として含むような被覆木のうち，最も実現エントリ数が大きい被覆木 t_o を構築する．但し，t_o は t_i と異なるものとする．t_o の構築は次のようにして行う．t_1, t_2 を部分木として含むような被覆木群を，t_1, t_2 に e_d とは異なる辺を挿入することにより構築し，その被覆木群の中で最も実現エントリ数が最も大きな被覆木を t_o とする．

以下で t_o の構築について詳しく説明する．t_1, t_2 を部分木として含む被覆木の集合を T とする．T から t_o を求める際には，t_i を除く T に属する全ての被覆木に対し，その被覆木を用いた被覆木ルーティングで

実現されるエントリ数を調べる必要がある．T に属する各被覆木 t' に対して，rt の各エントリ $< n_i, n_j >$ が t' を用いた被覆木ルーティングで実現できるか否か (t' における $< n_i, n_j >$ の実現状態と呼ぶ) を調べる処理には $O(logN)$ の計算量を必要とする．rt の全エントリ数は $N \times (N-1)$ となっており，各被覆木において実現されるエントリ数を調べる処理には，$N \times (N-1) \times O(logN)$ の計算量を要する．被覆木ルーティングにおいては，用いる被覆木により，実現されるエントリ群が異なるが，T に属する被覆木は，t_i から一つの辺を削除し，別の辺を加えた被覆木であるため，互いの構造は似通っている．そのため，T に属する全ての被覆木に対し，実現状態が同じとなっているエントリは少なくない．このようなエントリを確定エントリと呼び，そうでないようなエントリを未確定エントリと呼ぶ．本手続きでは，rt の全エントリについて未確定エントリであるか否かの判定を行い，T に属する各被覆木に対し，未確定エントリについてのみ実現状態を調べることで，被覆木ルーティングにおける実現エントリ数を計算する処理を軽減している．

未確定エントリのみを抽出するために，与えられたエントリが確定エントリであると判定できる条件について説明する．被覆木ルーティングにおいては，被覆木をたどることにより転送先ノードを求めるため，あるエントリ $< n_i, n_j >$ を実現されるための必要条件として，転送先ノード $next_h(rt, n_i, n_j)$ が，被覆木上において，到着ノード n_i の隣接ノードとなるように，n_i と $next_h(rt, n_i, n_j)$ を両端とする辺 (e' とする) が被覆木に含まれることが挙げられる．よって，t_1, t_2 に到着ノードと転送先ノードが含まれているエントリのうち，e' を t_1, t_2 に加えることにより t_1, t_2 が閉路を構成するエントリは，t_1, t_2 を基にして構築されるいかなる被覆木においても実現されることはない．すなわち，$< n_i, n_j >$ は T に属する全ての被覆木において非実現エントリであるため，これは確定エントリである．例を図 6 (a) に示す． 今，エントリ $< 2, 5 >$ の転送先ノードを 7 とすると，e' は (2, 7) となっており，この辺を太線で示されている木 t_1 に加えると閉路が生じる．そのため，ノード 7 はノード 2 の隣接ノードとなることはなく，転送先ノードが 7 であるエントリ $< 2, 5 >$ は，t_1, t_2 を部分木として含むいかなる被覆木においても実現されないエントリと決定される．

t_1, t_2 は最終的に得られる被覆木の部分木であるため，t_1, t_2 による被覆木ルーティングの経路は，最終的に得られる被覆木においても含まれている．t_1, t_2 のいずれか同一の木に含まれる任意の 2 ノード間の，被覆木ルーティングによる経路は，t_1, t_2 を部分木として含むいかなる被覆木においても変わらない．従って，t_1, t_2 のいずれか一つの木に到着ノードと宛先ノードが共に含まれるエントリ $< n_x, n_y >$ の，被覆木ルーティングによる転送先ノードは既に定まっており，$next_h(rt, n_x, n_y)$ と比較することにより，$< n_x, n_y >$ が実現エントリか非実現エントリであるか否かが決定される．よって，$< n_x, n_y >$ は確定エントリと決定される．

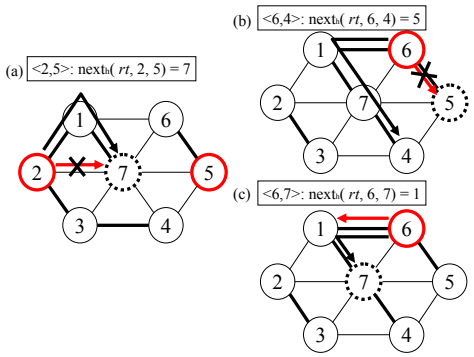

図 6 確定エントリの例

例を図 6 (b), (c) に示す．図 6 (b) において，エントリ $< 6, 4 >$ の転送先ノード $next_h(rt, 6, 3)$ は 5 となっているが，部分木によれば，到着ノード 6 から宛先ノード 4 への経路は，ノード 1, 7 を経由することになる．よって，転送先ノードが 5 であるエントリ $< 6, 4 >$ は非実現エントリと決定され，確定エントリとなる．また，図 6 (b) において，エントリ $< 6, 7 >$ の転送先ノード $next_h(rt, 6, 7)$ は 1 となっており，転送先ノードが 1 であるエントリ $< 6, 7 >$ は実現エントリと決定され，確定エントリとなる．

以上より，未確定エントリとなるエントリ $< n_i, n_j >$ の条件は次のようになる．ノード n_i, n_j が t_1, t_2 の別々の木に属する場合には，未確定エントリと判断できる．但し，辺 $e' = (n_i, next_h(rt, n_i, n_j))$ を t_1, t_2 に追加すると閉路を構成する場合は除く．この未確定エントリの数はおおよそ $2 \times (n-1)$ から $n^2/2$ となっており，削除辺 e_d により異なる．

rt の全エントリから未確定エントリを抽出し，T に属する被覆木について，未確定エントリと決定された各エントリが実現されるか否かを調べる．辺の端点は t_1 に，別の端点は t_2 に属する辺を，t_1, t_2 に加えることで，T に属する被覆木群を構築し，各被覆木について，未確定エントリのうちその被覆木により実現さ

表 1 被覆木構築アルゴリズムと被覆木再構築アルゴリズムのおける測定結果

ノード数	実現エントリ数の割合		被覆木構築時間	
	再構築	全構築	再構築	全構築
25	80.1%	99.3%	0.003s	0.05s
50	89.3%	97.8%	0.36s	2.27s
75	85.5%	97.9%	2.74s	12.64s
100	91.0%	95.5%	8.33s	132s
125	88.3%	95.5%	21s	240s
150	88.2%	93.5%	37s	564s
175	90.0%	92.3%	60s	1234s
200	85.0%	89.3%	466s	3610s
225	86.5%	87.1%	631s	4290s
250	84.8%	88.9%	1273s	8229s

れるエントリ数を計算する．その結果，最も実現エントリ数が大きい被覆木を t_o とする．

5. 評 価 実 験

本章では，提案する被覆木再構築アルゴリズムにより，再構築された被覆木併用ルーティングテーブルにおいて，ルーティングテーブルの全エントリ数のうち，どの程度のエントリを実現できているか確かめる評価実験を行った．

評価実験は，被覆木併用ルーティングテーブルを用いた，モバイルアドホックネットワークのネットワークシミュレーション上で行う．1000×1000 の平面上にノードをランダムに配置し，各ノードの通信範囲は，ノードを中心とした半径 100 の円内とする．シミュレーションが開始すると，各ノードは，それぞれに定められた目的地に向かって，ランダムウォークに従い，タイムスロット毎平均 50 ほど進んでいくものとする．シミュレーション開始前には，被覆木併用ルーティング法により被覆木を構築し，被覆木再構築アルゴリズムへの最初の入力被覆木とする．シミュレーション開始後には，タイムスロット毎に，提案アルゴリズムを適用し，被覆木併用ルーティングテーブルを再構築する．

評価実験では，ノード数 25, 50, 75, 100, 125, 150, 175, 200, 225, 250 のネットワーク環境でそれぞれシミュレーションを行った．シミュレーションの総タイムスロット数は 20 とし，各タイムスロットにおいて，被覆木により実現されるエントリ数の割合と，被覆木の再構築に要した時間を計測した．それらの平均値を表 1 に示す．また，比較のために，シミュレーション開始前に被覆木併用ルーティング法で構築した被覆木により実現されるエントリ数の割合と，構築にかかった時間もあわせて示す．実験結果より，提案アルゴリズムで再構築した被覆木により実現できたエントリの割合は 9 割近くとなっており，提案する被覆木再構築アルゴリズムを用いることで，ルーティングテーブルの容量を 1/10 に維持できることがわかった．また，再構築に要する時間は，被覆木併用ルーティング法の被覆木構築アルゴリズムに要する時間の 1/20 〜 1/7 となっており，頻繁な経路の変更に対して，ある程度対応できると考えられる．

6. ま と め

本稿では，文献 6) で提案された被覆木併用ルーティング法を，ネットワークトポロジが動的に変わるような環境においても適応できるよう，被覆木を再構築するアルゴリズムを提案した．また，提案方式について評価実験を行い，提案手法を用いることにより，被覆木併用ルーティングテーブルの容量を平均して 90 % 近く削減することができることがわかった．また，被覆木の再構築は少ない時間で行えることがわかった．

今後の課題としては，より少ない時間で被覆木を再構築するアルゴリズムの考案，単一の被覆木ではなく複数の被覆木を用いて，ルーティングテーブルの容量をさらに削減する方法を考案する予定である．

参 考 文 献

1) Zeng, X., Bagrodia, R. and Gerla, M.: GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks, *Proceedings of the 12th Workshop on Parallel and Distributed Simulations(PADS'98)* (1998).
2) MASH Research Group, University of California, Berkeley: The Network Simulator ns-2 (2000). http://www-mash.cs.berkeley.edu/ns/.
3) Riley, G. F., Fujimoto, R. and Ammar, M. H.: Stateless Routing in Network Simulations, *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (2000).
4) Riley, G.F., Ammar, M.H. and Zegura, E.W.: Efficient Routing With Nix-Vectors, *Proceedings of IEEE Workshop on High Performance Switching and RoutingHPSR 2001* (2001).
5) Huang, P. and Heidemann, J.: Minimizing Routing State for Light-Weight Network Simulation, *Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (2001).
6) Hiromori, A., Yamaguchi, H., Yasumoto, K., Higashino, T. and Taniguchi, K.: Reducing the Size of Routing Tables for Large-scale Network Simulation, *Proc 17th Workshop on Parallel and Distributed Simuation(PADS'03)*, pp. 115–122 (2003).