大規模ネットワークシミュレーション向けのルーティングテーブルの容量削減

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ともない，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，シミュレータに適用可能な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 1 つは，各ノードにおいて，到着したパケットを宛先ノードごと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．N をネットワークに含まれるノードの数とすると，表形式のルーティングテーブルでは，その容量は全体として $O(N^2)$ となる．一方，任意の 2 ノード間の経路がすべて，全ノードを含むある 1 つの木（被覆木）に含まれる特殊な場合にのみ，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ことで，ルーティングテーブルの容量を $O(N)$ とする被覆木ルーティング法も提案されているが，任意のルーティングを表現できない制約がある．本論文では，被覆木ルーティング法をもとに，なるべく多くの経路を含む被覆木によるルーティングテーブルと，表形式のルーティングテーブルを組み合わせたルーティングテーブルを構築し，任意のルーティングを表現でき，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する．提案手法の評価を行った結果，階層化トポロジにおいて，単純なルーティングテーブルと比較し，容量を 90%削減できた．

Reducing the Size of Routing Tables for Large-scale 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. A general routing table requires $O(N^2)$ space where N is the number of nodes. An algorithmic routing recently proposed by Heung et al. only requires $O(N)$ space for representing routing tables, however this can be applied in the case that all the routes between two nodes are contained in a fixed spanning tree (i.e. very limited routing is allowed). In this paper, we propose a new method to reduce a capacity of routing tables which is applicable to any routing table. In our method, a (near-optimal) algorithmic routing based table is used to represent a part of the given routing table. Our experimental results have shown that our method could reduce about 90% of the routing table size compared with a general routing table in hierarchical networks.

1. はじめに

近年，プロトコルの性能評価の有用な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．ネットワークシミュレーションは，既存のネットワークプロトコルが現在のネットワークにおいてどのような挙動

† 大阪大学大学院基礎工学研究科
Graduate School of Engineering Science, Osaka University
†† 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

を示すかを測定するためだけでなく，新しいプロトコルの設計や正当性の評価等にも用いられており，ネットワークプロトコルの研究開発において，ネットワークシミュレーションの役割は重要になりつつある．

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すべてのノードの振舞いを計算機上で模倣するため，計算機能力によって模倣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．既存のネットワークシミュレータでは，この問題に対処しシミュレータの規模適応性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用いられている．たとえば，GlomoSim [1] では，ネットワークシミュレーションに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模倣するのではなく，一部の振舞いを簡略化することで，ネットワークの振舞いを抽象化し，ネットワークシミュレーションの効率を上げる方法もある．たとえば，ネットワークシミュレーター 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 1 つに，各ノードが各宛先ノードごとに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あげられる．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．一般的な表形式のルーティングテーブルのデータ構造を**表 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 3 項を 1 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．

表 1 表形式のルーティングテーブルのデータ構造
Table 1 General routing table.

到着ノード	宛先ノード	転送先ノード
1	2	2
1	3	2
1	4	5
⋮	⋮	⋮
1	n	2
2	1	1
⋮	⋮	⋮
n	$n-1$	$n-1$

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 1 つである被覆木ルーティング法 [5] は，2 ノード間の経路がすべて，全ノードを含むある固定した 1 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，すべての経路を被覆木に含まれるような経路に制限したとしても，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，経路が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．

本論文では，被覆木ルーティング法をもとに，被覆木形式のルーティングテーブルと，表形式のルーティングテーブルを組み合わせることで，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方法を提案する．提案方式では，入力として，表形式のルーティングテーブルが与えられるとし，そのルーティングテーブル内のエントリの部分集合を被覆木形式のルーティングテーブルで表現する（**図 1**）．ルーティングテーブル全体の容量は，被覆木形式のルーティングテーブルに含まれる経路（エントリ）数に依存する．含むエントリ数が最大となるような被覆木を求める方法として，グラフ上で構築可能なすべての被覆木それぞれについて，その被覆木で表現できるエントリ数を求める方法が考えられる．グラフ上で構築可能な被覆木の数は，グラフ上に存在する辺の数に対し指数的に比例するため，多項式時間でこの問題を解く

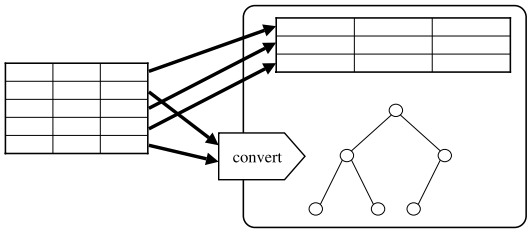


図 1 提案方式におけるルーティングテーブルのデータ構造
Fig. 1 Concept of proposed method.

ことは困難と予想される．本論文では，この問題に対し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階層型トポロジに対し，ルーティングテーブルの容量を 90%削減できることが分かった．

以下，2 章では被覆木ルーティングについて述べる．3 章では提案方式について述べ，4 章ではシミュレーション実験の結果と考察を述べる．

2. 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 1 つの被覆木に含まれる場合，各ノードに一定の規則に従って被覆木ルーティング法独自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．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は，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 2** にあげる．付加規則では，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 は被覆木中の子ノード数の最大値）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 ノード間に親子関係が存在するか否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ものの，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を保持しておけばよいこととなり，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．

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 3** のようになり，バランスのとれた木構造となるように根のノードを選ぶことによ

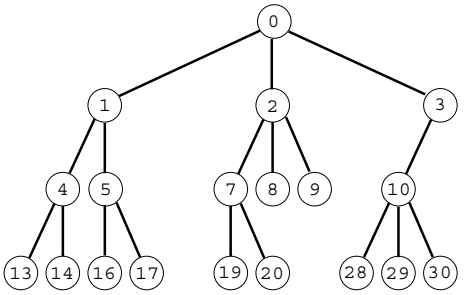


図 2 被覆木の例
Fig. 2 Example spanning tree.

```
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 3 被覆木ルーティングにおける転送先ノード探索アルゴリズム
Fig. 3 Algorithm for deciding neighbor node in algorithmic routing approach.

り，計算量は $O(logN)$ となる．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，B の先祖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，そうでない場合には，パケットを転送すべきノードとしてノード A の親ノードを返す．たとえば，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 の子孫としては存在しないことが判定できるため，ノード 5 宛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．

3. 被覆木併用ルーティング法

被覆木ルーティング法は，2 ノード間の全経路がある 1 つの被覆木に含まれる場合に限られるため，シミュレーションへの適用範囲は狭く，汎用的なシミュレーションには適さない．また，被覆木ルーティング法

を適用できるように経路を変更した場合には，シミュレーション結果の正確性が損なわれる場合が考えられる．本論文では，被覆木ルーティング法を利用し，任意のルーティングを正確に表現し，かつ容量が小さくなるようにルーティングテーブルを構築する方法を考案する．

3.1 被覆木併用ルーティングテーブル

入力として，対象とするネットワークと表 1 に示したような，シミュレーションで実現すべき経路を表現した表形式のルーティングテーブルが与えられるものとする．提案方式では，表形式のルーティングテーブルと被覆木形式のルーティングテーブルを併用するルーティングテーブル（被覆木併用ルーティングテーブル）で与えられたエントリ集合を保持する．被覆木併用ルーティングテーブルにおける被覆木形式のルーティングテーブルは，被覆木ルーティング法をもとにしており，そのデータ構造は，ネットワークシミュレーションの対象となるネットワークグラフ上の被覆木に基づく．被覆木形式のルーティングテーブルにより保持できるエントリ集合は，そのデータ構造のもととなる被覆木の形状に依存する．そのため，与えられたエントリのうち，被覆木形式のルーティングテーブルで保持できるものは被覆木形式のルーティングテーブルで保持し，残りを表形式のルーティングテーブルに残しておくことにより，全経路を正しく保持する．被覆木併用ルーティングテーブルにおいて，到着ノードと宛先ノードの組に対し，転送先ノードを参照する際には，まず，そのエントリが表形式のルーティングテーブルにおいて保持されているか否かを確かめ，表形式のルーティングテーブルで保持されている場合には，表形式のルーティングテーブルに保持された転送先ノードを返す．そうでない場合には，被覆木形式のルーティングテーブルから被覆木ルーティング法により，転送先ノードを計算し返す．

被覆木併用ルーティングテーブルの容量は，被覆木形式のルーティングテーブルと表形式のルーティングテーブルの容量の和である．被覆木形式のルーティングテーブルに要するメモリ容量は，保持するエントリ数にかかわらず一定である．そのため，なるべく多くのエントリを含むような被覆木を被覆木形式のルーティングテーブルに採用することにより，表形式のルーティングテーブルで保持すべきエントリが少なくなり，その結果被覆木併用ルーティングテーブル全体に要するメモリ容量は減少する．提案方式では，そのような被覆木を求めるヒューリスティックアルゴリズムを考案し，ルーティングテーブルの容量削減を行っている．

3.2 被覆木探索アルゴリズム

提案方式では，ネットワークのある辺を初期辺とし，グリーディに被覆木を構築する被覆木構築アルゴリズムを，ネットワーク上の各辺を初期辺として適用する．これにより得られる被覆木群のうち最も多くのエントリを保持できる被覆木を，被覆木併用ルーティングテーブルで用いる被覆木 st とする．3.3 節では提案手法で用いる（指定された 1 つの辺に対する）被覆木構築アルゴリズムを説明する．このアルゴリズムの時間計算量は $O(kN^2logN)$ となっている．よって，提案方式の計算量は，ネットワークに存在する辺の数がたかだか kN 本であることから，$O(k^2N^3logN)$ となる．

3.3 被覆木構築アルゴリズム

被覆木構築アルゴリズムでは，入力として，ネットワークの 1 つの辺 ie（初期辺と呼ぶ），ネットワークトポロジを表したグラフ g，および，表形式のルーティングテーブル rt が与えられるものとする．以下，グラフ g とルーティングテーブル rt は固定して考える．rt における各エントリは，到着ノード，宛先ノード，転送先ノードから構成され，到着ノードと宛先ノードの組により一意に定まる．表形式の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 $next(n_i, n_j)$ とする．

被覆木構築アルゴリズムでは，初期辺 ie に逐次的に辺を加えていくことで被覆木を構築する．g 上のノードすべてを含む完全グラフ上に構築され，かつ t を部分木として含む被覆木の集合を ST_t とする．構築途中の木 t に対して，各エントリの状態を「実現エントリ」,「非実現エントリ」,「未確定エントリ」の 3 種類に分類する．t における実現エントリとは，エントリに示された転送先ノードと，ST_t に属する被覆木による被覆木ルーティング法で求められた各転送先ノードが一致するエントリである．一方，t における非実現エントリとは，エントリに示された転送先ノードと，ST_t に属する被覆木による被覆木ルーティング法で求められた各転送先ノードがすべて一致しないエントリである．t における未確定エントリとは，実現エントリ，非実現エントリのいずれにも該当しないエントリである．t に辺 e を加えた際に，新たに非実現エントリと決定されるエントリ集合を $URE(t, e)$ で表し，t に隣接する辺のうち閉路を生じない辺の集合を $TE(t)$ とする．辺を加える際には，$TE(t)$ に属する辺のうち，$|URE(t, e)|$ が最も小さい辺 e を木 t に加える．この処理を繰り返し，辺を $N - 2$ 本加えたところで被覆

$t = ie$ where $ie \in E$ is a given edge
while (the number of edges in $t \neq N - 1$)
  foreach e in $TE(t)$
    compute $URE(t, e)$
  end
  find e where $|URE(t, e)|$ is minimum
  let t be a new tree t' where e is added to t
end

図 4 被覆木構築アルゴリズム
Fig. 4 Spanning-tree constraction algorithm.

木 st が得られる．被覆木構築アルゴリズムを**図 4** に示す．

以上は，各辺の付加ごと，新たに非実現エントリと決定されるエントリの数を最小とする手法であるが，一方，新たに実現エントリと決定されるエントリの数を最大とするように辺を加えていく手法も考えられる．被覆木を構築していく過程で，t における実現エントリ数は最終的に実現されるエントリ数の下限を示しているが，多くのエントリについて，実現エントリであると決定される時期が被覆木構築の終盤になるため，構築途中で決定する実現エントリ数は，最終的な実現エントリ数の目安とはならない．そのため，実現エントリ数に基づいて辺を加える場合には，実現されるエントリ数が大きい被覆木を構築することができない場合が多い．そこで，提案方式では，辺を加えていく際に，$|URE(t, e)|$ が最小，いい換えれば，実現エントリとなる可能性のあるエントリ数を最大とする方針を採用する．

被覆木構築過程において，与えられたエントリが構築途中の木 t において非実現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法においては，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 $< n_i, n_j >$ を実現する必要条件として，転送先ノード $next(n_i, n_j)$ 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 $next(n_i, n_j)$ を両端とする辺（e' とする）が被覆木に含まれることがあげられる．よって，t 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 に加えることにより t が閉路を構成するエントリは，t をもと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く，非実現エントリと決定される．例を**図 5** (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構築途中の木に加えると閉路が生じる．そのため，ノード 7 は

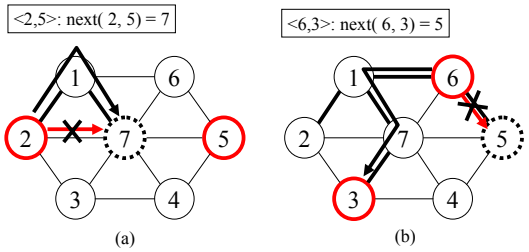


図 5 非実現エントリの例
Fig. 5 Example uncoverable entry.

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 $< 2, 5 >$ は非実現エントリと決定される．

t は最終的に得られる被覆木の部分木であるため，t 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 に含まれる任意のノード間の，被覆木ルーティングによる経路は，t を部分木として含むようないかなる被覆木においても変わらない．よって，t に到着ノードと宛先ノードが含まれるエントリ $< n_x, n_y >$ の，被覆木ルーティングによる転送先ノードはすでに定まっている．$< n_x, n_y >$ について，木 t における被覆木ルーティング法で求められる転送先ノードを $next_{sp}(t, n_x, n_y)$ で表す．到着ノードと宛先ノードが t に含まれているエントリ $< n_x, n_y >$ のうち，$next_{sp}(t, n_x, n_y)$ と $next(n_x, n_y)$ が等しくないエントリは非実現エントリと決定される．例を図 5 (b) に示す．エントリ $< 6, 3 >$ の転送先ノード $next(6, 3)$ は 5 となっているが，(b) の木においては，到着ノード 6 から宛先ノード 3 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 $< 6, 3 >$ は非実現エントリと決定される．

被覆木構築過程の各エントリ $< n_i, n_j >$ は，構築途中の木 t に (A) 到着ノードと転送先ノードが含まれている場合，(B) 到着ノードと宛先ノードが含まれている場合，それぞれ以下の条件により非実現エントリであるか判定できる．

(1) (A) のエントリ $< n_i, n_j >$ は，辺 $e' = (n_i, next(n_i, n_j))$ を木 t に追加すると t が閉路を構成する場合に，非実現エントリと判定できる．

(2) (B) のエントリ $< n_i, n_j >$ は，$next(n_i, n_j)$ と $next_{sp}(t, n_i, n_j)$ が一致しない場合には，非実現エントリと判定できる．

$URE(t, e)$ の求め方について説明する．辺 $e = (n_u, n_v)$ について，t に含まれるノードを n_u，新た

に t に加わるノードを n_v とする．また，t に含まれる各ノードを n_w とする．このとき，新たに (A) となるようなエントリは，到着ノードが n_w であるようなエントリ $< n_w, * >$ で，かつ転送先ノード $next(n_w, *)$ が n_v であるエントリ，また，到着ノードが n_v であるようなエントリ $< n_v, * >$ で，かつ転送先ノード $next(n_v, *)$ が n_w であるエントリである．これらのエントリに対して条件 (1) を判定することで，$URE(t, e)$ に属するエントリであるか否かが判定できる．また，新たに (B) となるようなエントリは，$< n_w, n_v >$，$< n_v, n_w >$ であり，これらのエントリに対して条件 (2) を判定することで，$URE(t, e)$ に属するエントリであるか否かが判定できる．ただし，すでに条件 (1) により非実現エントリと判定されたエントリは除くものとする．

被覆木構築アルゴリズムでは，木に辺を 1 つ加える際，候補となる辺の数はたかだか $k \times N$ であるため，非実現エントリ数を求める手続き試行回数もたかだか $k \times N$ 回となる．非実現エントリ数を求める計算量は $O(logN)$ であるため，辺を 1 本加えるごとに $O(kNlogN)$ の計算量が必要となる．被覆木を構築するまでに $N - 2$ 本の辺を加えるため，被覆木構築アルゴリズムの時間計算量は $O(kN^2logN)$ となる．ただし，k は被覆木の最大次数であり，一般には定数と見なせる．

4. 評価実験と考察

提案方式においては，被覆木で実現できるエントリ数が多いほどルーティングテーブルの容量は小さくなる．本章では，提案方式を用いて，与えられたルーティングテーブルのエントリ数のうち，どの程度のエントリが被覆木により実現できているか確かめるための評価実験を行った．

被覆木で実現できるエントリ数は，与えられたルーティングテーブルにより示された実現すべき経路集合の性質に大きく依存すると考えられる．評価実験では，すべての経路を最短経路とし，ネットワークトポロジを変えることで実現すべき経路集合の性質を変化させ，性質の異なる 2 種類の経路集合を対象に実験を行った．対象とするネットワークトポロジは，階層型トポロジとランダムトポロジで，階層型トポロジにおける経路集合は，特定のリンク集合に集中し比較的木の形状に近いものとなった．一方，ランダムトポロジにおける経路集合は，どのリンクにも一様に分散するものとなった．これらのトポロジに対し，提案手法において構築された被覆木により実現できたエントリ数の割合を計測した．

また，提案方式で用いた被覆木構築アルゴリズムはヒューリスティックな方法であり，実現できるエントリ数が最大となるような被覆木（最適な被覆木）を必ずしも構築することはできない．実験では，提案アルゴリズムにより構築された被覆木で実現できるエントリ数と，最適な被覆木で実現できるエントリ数の比較を行った．

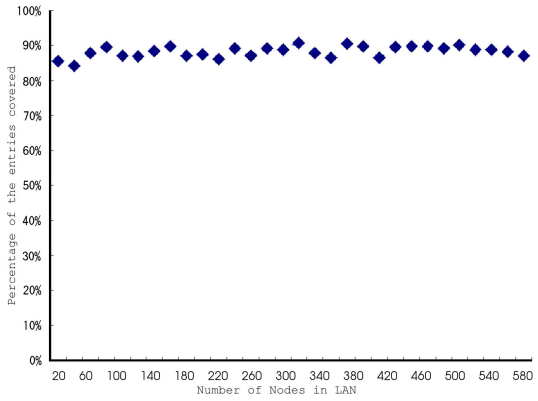
提案方式では，ルーティングテーブル容量を削減できる一方，エントリの参照時間は $O(logN)$ である．また，被覆木ルーティングで用いる被覆木の構築を行うため，表形式のルーティングテーブルのみを用いた表方式と比較し，そのための処理時間が必要となる．我々は，ネットワークシミュレータ ns[2] に対し，ns のルーティング処理部分を提案方式に従い変更することで，提案方式を実装した．この ns に対し，提案方式と表方式について，ns におけるルーティングテーブル容量，エントリ参照時間を，さらに，提案方式における被覆木構築の時間を測定した．また，メモリ容量制限下において，ルーティングテーブルのためにメモリ不足を起こすことなく，シミュレーションを実行できるか否かを調べた．

4.1 階層型トポロジに対する評価実験

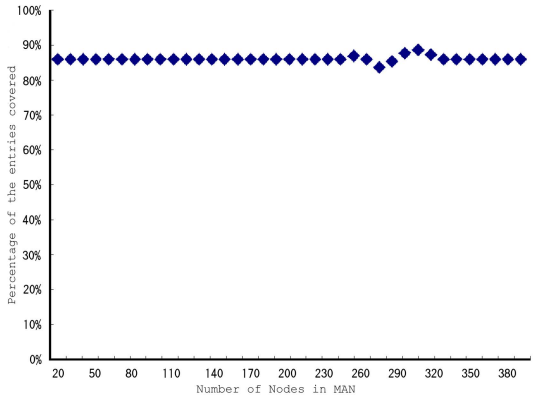
階層型トポロジに対する評価実験では，WAN，MAN，LAN からなる階層型トポロジ Tiers[6] を用いた．Tiers の形状はほぼ木構造であり，被覆木により相当量のエントリ数を削減できると期待できる．個々の LAN はスター型ネットワークとなっており，その形状は木となっている．そのため，LAN に属するノードが増えたとしても，そのノードに関わるエントリのほとんどが被覆木で実現できると考えられる．一方，中間ノードにあたる MAN に属するノードは隣接するノードが複数あると考えられ，そのノードに関わるエントリのうち被覆木で実現されるものは限られる．

評価実験では，(a) WAN，MAN に属するノード数を固定し，LAN に属するノード数を 20 から 600 まで 20 ずつ増やした場合，(b) WAN，LAN に属するノード数を固定し，MAN に属するノード数を 20 から 400 まで 10 ずつ増やした場合について，全エントリ数に対し被覆木により実現できたエントリ数の割合を調べた．

図 6 (a)，(b) にその結果を示す．縦軸は，被覆木で実現できたエントリ数の割合を示し，横軸はそれぞれ LAN，MAN に属するノード数を示している．(a)，(b) いずれの場合においても，構築された被覆木で実現できたエントリ数の割合は 90%を超える場合もあ



(a) LAN に属するノード数を増やした場合



(b) MAN に属するノード数を増やした場合

図 6 階層型トポロジにおいて被覆木により実現されたエントリ数の全エントリ数に占める割合

Fig. 6 Percentage of the entries covered by spanning-trees in hierarchical topologies.

り，提案方式は階層型トポロジに対してルーティングテーブルの削減を行う際，非常に有効であることが分かった．さらに，ノード数を増やした際に，実現されるルーティングの割合は大きな変化はなく，85%から90%に収まっていることが分かった．このことより，葉にあたるノード，中間ノードの数にかかわらず，ほぼ木構造となっているネットワークに対しては，ルーティングテーブルの容量を大きく削減できることが分かった．

被覆木ルーティングでは，与えられたルーティングテーブルに対し，被覆木で覆える部分に関するエントリのみ実現できる可能性がある．そのため，提案方式においては，階層型トポロジのように形状がほぼ木構造となっているトポロジでは，与えられたルーティングにかかわらず，被覆木ルーティングによりほとんどのエントリが実現可能で，与えられたルーティングテーブルのほとんどが削減できると考えられる．イ

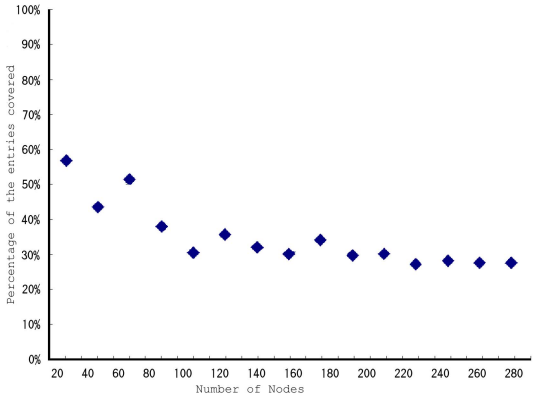


図 7 ランダムトポロジにおいて被覆木により実現されたエントリ数の割合

Fig. 7 Percentage of the entries covered by spanning-tree in random topologies.

ンターネットにおいて一般的なトポロジは階層型トポロジとなっており，インターネットを想定したシミュレーション等では，提案方式によるルーティングテーブルの容量削減は有効であると考えられる．

4.2 ランダムトポロジに対する評価実験

ネットワークトポロジの違いにより，提案方式によるルーティングテーブルの削減容量がどの程度異なっているか調べるために，ランダムトポロジに対しても，ネットワークに属するノード数を増やしたとき，被覆木によりどの程度ルーティングテーブルの容量が削減されるか評価実験を行った．評価実験では，ノード数を 20 から 240 まで 20 ずつ増やした際に，削減できるルーティングテーブルの容量を調べた．

図 7 にその結果を示す．縦軸は，被覆木で実現できたエントリ数の割合を示し，横軸はネットワークのノード数を示している．提案方式で構築された被覆木により，階層型トポロジにおいては，全エントリのうち 90%近く実現できたのに対し，ランダムトポロジにおいては，ノード数 20 の場合には 60%程度となっているが，ノード数が増えるにつれ次第にその割合が低くなり，ノード数 200 の場合には 30%程度であることが分かった．階層型トポロジと比較し，辺の数の多いランダムトポロジのようなトポロジでは，被覆木で覆える部分が限られるため，実現可能なエントリ数が少なく，ルーティングテーブル容量はそれほど削減できないと考えられる．

4.3 最適な被覆木との比較

提案アルゴリズムにより構築された被覆木により実現されるエントリ数が，最適な被覆木で実現できるエントリ数と比較し，どの程度異なっているかを確かめ

表 2 実現可能な最大エントリ数に対する，提案アルゴリズムにより構築された被覆木で実現されるエントリ数の割合の分布
Table 2 Percentage distribution of covered entries to coverable entries.

実現される	事例数	
エントリ数の割合	階層型トポロジ	ランダムトポロジ
0 〜 89%	0	0
90 〜 91%	0	0
92 〜 93%	0	0
94 〜 95%	0	0
96 〜 97%	2	10
98 〜 99%	19	10
100%	29	30

る評価実験を行った．評価実験では，ネットワーク上で構築可能なすべての被覆木に対し，被覆木ルーティングで実現可能なエントリ数を計算することで，最適な被覆木を求めた．最適な被覆木を求める処理には多くの時間を要するため，階層型トポロジではノード数 20，ランダムトポロジではノード数 15 のネットワークを対象とし，どちらのトポロジに対しても 50 回の試行を行い，最適な被覆木により実現できるエントリ数に対する，提案アルゴリズムにより構築された被覆木で実現されるエントリ数の割合を測定した．

表 2 にその結果を示す．どちらのトポロジについても，提案アルゴリズムにより実現されたエントリ数の割合が 100%，つまり最大エントリ数を実現できる被覆木が得られている回数が半分以上となった．また，すべての場合についてみても，実現されたエントリ数の割合は 95%以上となっており，構築された被覆木により実現できるエントリ数は，最大エントリ数にきわめて近いものとなっていることが分かった．

4.4 実際の環境における評価実験

階層型トポロジ Tiers を用い，ノード数 100，500，1000，2000，3000 のネットワークについて，提案方式を実装した ns を用いて実験を行った．実行環境として，CPU Pentium4 2 GHz，メモリ 512 MB の PC を用いた．実験では，提案方式と表方式について，ルーティングテーブル容量，およびエントリ参照時間の平均を，さらに，提案方式における被覆木構築の時間を測定した．また，シミュレーションで使用できるメモリ容量を 256 MB に制限し，提案方式と従来方式のそれぞれについて，シミュレーターがメモリ不足を起こすことなくシミュレーションを実行できるか否かを調べた．測定結果を**表 3** に示す．

ルーティングテーブル容量は，ネットワークに含まれるノード数にかかわらず，1 割程度となっており，4.1 節に示したように，大きく削減できることが分かった．また，提案方式のエントリ参照時間は，ノード数 3000 のネットワークにおいて，従来方式と比較し約 5 倍の 0.65 μs となっているが，総シミュレーション時間に占めるエントリ参照時間の割合は少なく，シミュレーション時間を大きく増加させることはないと考えられる．被覆木構築時間は，ノード数が増えるにつれ急激に増加しており，ノード数 3000 のネットワークにおいては，被覆木の構築に 12 時間余りを費やしている．しかし，メモリ容量を制限した環境において，表手法ではメモリ不足のため，ノード数 2000，3000 のネットワークに対するシミュレーションを実行できなかったのに対し，提案手法では，ルーティングテーブル容量を削減することで，シミュレーターの使用メモリを節約し，すべてのシミュレーションを実行することができた．このことから，提案手法を用いることで，限られたメモリ容量のもとで，より大規模なネットワークにおけるシミュレーションを実現できると考えられる．

経路が頻繁に変更されるシミュレーションにおいて，ルーティングテーブルの情報を正しく保持し，かつ容量をつねに最適に保つために，経路変更のたびに被覆木を構築し直すのは，被覆木構築に時間がかかることから効率的ではない．そのようなシミュレーションにおいては，次のようにルーティングテーブルを更新する方法が有効であると考えられる．経路変更が，追加，更新，削除の 3 種類のいずれかの情報が付加されたエントリ群により与えられるものとする．削除の情報が付加されたエントリのうち，現状の被覆木ルーティングテーブルで表現されているエントリはそのまま保持し，汎用ルーティングテーブル部で表現されているものは削除する．追加，更新の情報が付加されたエントリのうち，現状の被覆木ルーティングテーブルで表現できるエントリは被覆木ルーティングテーブル部で，そうでないエントリは汎用ルーティングテーブル部で保持する．このようにルーティングテーブルを更新することにより，経路変更を正しく表現することができ，ルーティングテーブルの容量増加は，たかだか，与えられたエントリのうち，汎用ルーティングテーブル部で保持されるエントリ分に抑えることができる．経路変更が重なった結果，汎用ルーティングテーブルに属するエントリが多くなり，ルーティングテーブルの容量が大きくなった際には，被覆木を構築し直すことにより対応できる．

5. ま と め

本論文では，被覆木を用いたルーティングテーブル容量の削減手法を応用することで，任意のルーティングを表現でき，かつネットワークシミュレータが利用

表 3 提案方式と表方式における測定結果
Table 3 Experimental results in proposed method and original method.

	ルーティングテーブル容量			エントリ参照時間			メモリ容量制限下での実行	
ノード数	提案方式	表方式	容量比	提案方式	表方式	被覆木構築時間	提案方式	表方式
100	16 KB	128 KB	12.50%	0.37 μs	0.14 μs	1 s	○	○
500	896 KB	8,000 KB	11.20%	0.42 μs	0.14 μs	2 m 09 s	○	○
1,000	1,768 KB	16,000 KB	11.05%	0.46 μs	0.14 μs	18 m 48 s	○	○
2,000	3,296 KB	32,000 KB	10.30%	0.61 μs	0.14 μs	2 h 39 m 03 s	○	×
3,000	13,768 KB	128,000 KB	10.76%	0.65 μs	0.14 μs	12 h 03 m 12 s	○	×

するルーティングテーブルの容量を削減する方法を提案した．また，階層型トポロジとランダムトポロジについて評価実験を行い，提案手法によりルーティングテーブルの容量を，インターネットに代表される階層型トポロジにおいて 90%近く削減できることを示した．

今後の課題としては，スケーラビリティの向上を目標とし，効率良く被覆木を探索する方法，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．また，ネットワークトポロジの動的な変更に対し，被覆木の辺の入替えを繰り返し，コストの少ない被覆木再構築を行い，ルーティングテーブルの容量が増加しないよう適切に被覆木を更新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, *Proc. 12th Workshop on Parallel and Distributed Simulations* (*PADS'98*) (1998).
2) MASH Research Group: The Network Simulator ns-2, University of California, Berkeley (2000). http://www-mash.cs.berkeley.edu/ns/
3) Riley, G.F., Fujimoto, R. and Ammar, M.H.: Stateless Routing in Network Simulations, *Proc. 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, *Proc. IEEE Workshop on High Performance Switching and RoutingHPSR 2001* (2001).
5) Huang, P. and Heidemann, J.: Minimizing Routing State for Light-Weight Network Simulation, *Proc. IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (2001).
6) Calvert, K., Doar, M. and Zegura, E.: Modeling Internet Topology, *IEEE Communications Magazine*, Vol.35, No.6, pp.160–163 (1997).
7) Bajaj, S., Breslau, L., Estrin, D., Fall, K., Floyd, S., Haldar, P., Handley, M., Helmy, A., Heidemann, J., Huang, P., Kumar, S., McCanne, S., Rejaie, R., Sharma, P., Varadhan, K., Xu, Y., Yu, H. and Zappala, D.: Improving Simulation for Network Research, Technical Report 99-702b, University of Southern California (1999).
8) Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y. and Yu, H.: Advances in Network Simulation, *IEEE Computer*, Vol.33, No.5, pp.59–67 (2000).
9) Floyd, S. and Paxson, V.: Difficulties in Simulating the Internet, *IEEE/ACM Trans. Networking*, Vol.9, No.7, pp.392–403 (2001).
10) Heidemann, J., Mills, K. and Kumar, S.: Expanding Confidence in Network Simulation, *IEEE Network Magazine*, Vol.15, No.5, pp.58–63 (2001).

(平成 15 年 2 月 4 日受付)
(平成 16 年 2 月 2 日採録)

廣森 聡仁（正会員）

平成 10 年大阪大学基礎工学部情報科学科中退．平成 12 年同大学大学院基礎工学研究科博士前期課程修了．同年同大学院基礎工学研究科博士後期課程入学．現在，同課程在学中．通信プロトコルの設計および性能評価の研究に従事．

山口 弘純（正会員）

平成 6 年大阪大学基礎工学部情報工学科卒業．平成 10 年同大学大学院博士後期課程修了．同年オタワ大学客員研究員．平成 11 年大阪大学大学院基礎工学研究科助手．現在，同大学院情報科学研究科助手．工学博士．分散システムや通信プロトコルの設計および実装に関する研究に従事．

安本 慶一（正会員）

平成 3 年大阪大学基礎工学部情報工学科卒業．平成 7 年同大学大学院博士後期課程退学後，滋賀大学経済学部助手．平成 9 年モントリオール大学客員研究員．現在，奈良先端科学技術大学院大学情報科学研究科助教授．工学博士．分散システム，マルチメディア通信システムに関する研究に従事．IEEE/CS 会員．

東野 輝夫（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院博士課程修了．同年同大学助手．平成 2，6 年モントリオール大学客員研究員．現在，大阪大学大学院情報科学研究科教授．工学博士．分散システム，モバイルコンピューティング，通信プロトコル等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．

谷口 健一（正会員）

昭和 40 年大阪大学工学部電子工学科卒業．昭和 45 年同大学大学院博士課程修了．同年同大学助手．現在，同大学大学院情報科学研究科教授．工学博士．この間，計算理論，ソフトウェアやハードウェアの仕様記述・実現・検証の代数的手法および支援システム，関数型言語の処理系，分散システムや通信プロトコルの設計・検証等に関する研究に従事．