P2P ネットワークにおける経路長あるいは経路表サイズの最大値を柔軟に設定可能な経路表構築方式の提案

A Method for Constructing Routing Tables That Allows Configurable Max Path Length or Routing Table Size

呉 承彦† 安倍 広多† 石橋 勇人† 松浦 敏雄†
Seung-eon Oh Kota Abe Hayato Ishibashi Toshio Matsuura

1 はじめに

構造化 P2P ネットワークは，各ノードが保持する経路表によってルーティングを行い，宛先ノードに到達する．Chord [1]，Pastry [2]，Kademlia [3] など，既存の構造化 P2P ネットワークの多くはノード数 n に対して，経路長が $O(\log n)$ である（対数オーダルーティング）．対数の底は固定されているもの（例えば Chord では 2）と，選択できるもの（例えば Pastry や Kademlia では 2^b，ただし b は定数）がある．対数の底の値を大きくすれば経路長は短く，経路表サイズは大きくなり，逆に小さくすれば経路長は長く，経路表サイズは小さくなる．ただし，既存の方式では，底の値を P2P ネットワークの運用中に動的に変更することはできなかった．

本稿では，対数の底の値を動的に変更可能な経路表構築手法を提案する．また，これを用いることで，以下のような構造化 P2P ネットワークを実現する．

- 最大経路長が，指定した任意の値を上回らないようにする方式（**経路長優先方式**と呼ぶ）．このとき，経路表サイズはノード数に応じて増減する．
- 経路表サイズを，指定した任意の値とする方式（**経路表サイズ優先方式**と呼ぶ）．このとき，経路長はノード数に応じて増減する．

本手法を用いることによって，1 ホップルーティングから対数オーダルーティングまでをカバーすることができる．ここで，1 ホップルーティングとなるのは，経路長優先方式において最大経路長を 1 と指定するか，経路表サイズ優先方式において経路表サイズよりもノード数が少ない場合である．

提案手法は，範囲検索が可能な構造化 P2P ネットワークの 1 つである Chord$^\#$ をベースとしている．オリジナルの Chord$^\#$ ではキー空間を 2 分探索する 1 次元の経路表（finger table）を構築するが，提案手法では finger table を 2 次元に拡張し，これによって k 分探索を可能とする．その上で，設定した最大経路長あるいは最大経路表サイズと推定ノード数に基づいて k を動的に変更する．

本稿の構成は次のとおりである．2 章では Chord$^\#$ について述べる．3 章では提案手法について述べ，4 章で関連研究について触れる．5 章で提案手法の評価と考察を行い，最後に 6 章でまとめと今後の課題について述べる．

† 大阪市立大学，Osaka City University

2 Chord$^\#$

本章では，提案手法の基礎となる Chord$^\#$ について述べる．

2.1 概要

Chord$^\#$ はリングベースの構造化 P2P ネットワークの 1 つである．各ノードは *key* を保持しており，リング上の key 空間にマップされる．リングは，時計回りに key の値が大きくなるものとする．Chord とは異なり，リングの key 空間の大きさをあらかじめ決めておく必要はない．

Chord$^\#$ の各ノードはいくつかの変数を保持している（本節では初出の変数名をイタリックで表記する．上述の *key* はその 1 つである）．以後，ノード u が保持する変数 x を $u.x$ のように表記する．また，Chord$^\#$ の（あるいは後で述べる提案手法の）全ノードの集合を $V = \{N_0, N_1, \cdots, N_{n-1}\}$ とし（ただし n はノード数），各ノードは時計回りに $N_0, N_1, \cdots, N_{n-1}$ の順に並べられるものとする ($N_i.key < N_{i+1}.key$)．さらに，$a \oplus b \equiv (a + b) \bmod n$ とする．

Chord$^\#$ の各ノードは，変数として，key 以外に *successor, predecessor, finger table* および長さ r の *successor list* を持つ．successor は key からリング上で時計回りに最も近いノードへのポインタ，predecessor は反時計回りに最も近いノードへのポインタである．すなわち，ノード N_i の successor は $N_{i\oplus 1}$，predecessor は $N_{i\oplus(n-1)}$ と表すことができる．なお，ここでのポインタはノードのロケータ（IP アドレスなど）とノードの key の両方を含むものとする．finger table に関しては次の節で述べる．successor list は，時計回りに最も近い r 個のノードへのポインタであり，successor ノードが離脱または故障した場合に，successor を付け替えてリングを修復するために用いる．

2.2 finger table

Chord$^\#$ では，検索効率を向上させるために finger table を持つ．finger table はノード数 n に対して，$\lceil \log_2 n \rceil$ 個のエントリを持つ 1 次元配列である．finger table の各エントリは式 (1) で決定される．

$$finger[i] = \begin{cases} successor & (i = 0) \\ finger[i-1].finger[i-1] & (i > 0) \end{cases} \quad (1)$$

例として，ノード N_0 の finger table を考える．N_0.finger[0] には，N_0 の successor すなわち N_1 へのポインタが代入され

i \ j	0	1	2
0	N_1	N_2	N_3
1	N_4	N_8	N_{12}
2	N_{16}	N_{32}	N_{48}
⋮	⋮	⋮	⋮

図 1　ノード N_0 における finger table($k = 4$)

る．N_0.finger[1] には，N_0.finger[0] が指すノードである N_1 が保持する finger[0]，すなわち，N_2 へのポインタが代入される．同様に，N_0.finger[2] には，N_0.finger[1] が指すノードである N_2 が保持する finger[1]，すなわち，N_4 へのポインタが代入される．

このように，ノード N_k の finger[i] には，$N_{k\oplus 2^i}$ へのポインタが代入されるため，finger table の高さは $\lceil \log_2 n \rceil$ となる．この finger table を用いると，検索クエリを他のノードへ転送するごとに検索範囲を 1/2 ずつ絞り込めるため，検索に要するホップ数は $O(\log_2 n)$ となる．

各ノードの finger table はノードの挿入や削除，障害などによって理想的な状態（式 (1) が満たされた状態）ではなくなるため，各ノードは finger table を定期的に更新する必要がある．上で述べたように，ノード u の finger[i] ($i > 0$) を更新するためには，u は自身の finger[$i-1$] が指すノードの finger[$i-1$] を取得する必要がある．1 つのノードが持つ finger table のすべての行を更新するために必要なメッセージ数は，$2\lceil \log_2 n \rceil$ (反復ルーティングで取得した場合)，または $\lceil \log_2 n \rceil + 1$（再帰ルーティングで取得した場合）となる．

3　提案手法

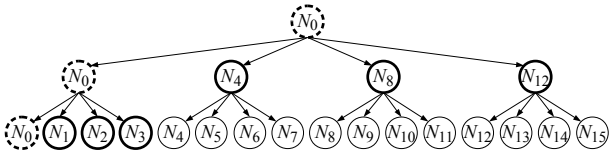
本章では，提案手法が Chord$^\#$ と異なる点に焦点を絞って，提案手法の概要および詳細を述べる．

3.1　データ構造

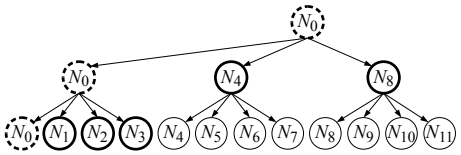
1 章で述べたように，提案手法ではキー空間を k 分探索できるように finger table を 2 次元化する[*1]．2 次元化した finger table を fingers[][] で表す．便宜上，第 1 次元を高さ方向とし，第 2 次元を幅方向とする．finger table の幅は $k-1$ であり，高さは $\lceil \log_k n \rceil$ である．なお，$k = 2^m$ とする（m は 1 以上の整数）．ノード N_u の fingers[i][j] は，定常状態（ノードの参加・離脱がない状態）でノード $N_{u\oplus(j+1)k^i}$ へのポインタを保持する（すなわち，N_u.fingers[i][j] $= N_{u\oplus(j+1)k^i}$ である）．$k = 4$ のときのノード N_0 の finger table の例を図 1 に示す．

このような finger table を用いると，図 2 のように k 分探索木を構成できる．図 2 はノード N_0 を起点（根）とした場合の 4 分探索木の例である（上段は $n = 16$，下段は $n = 12$ の場合）．ノード x の finger table がノード y を含むときに，x から y に矢印を描いている．この図より，最大経路長と経路

[*1] この構造は文献 [4] で提案されている．



$n = 16 \quad k = 4$



$n = 12 \quad k = 4$

図 2　ノード数 $n = 12, 16$ のとき，ノード N_0 における k-分探索木の例（$k = 4$）

表の高さはどちらも $\lceil \log_k n \rceil$ となることがわかる．

提案方式では，finger table のすべてのエントリが埋まっているとは限らない．本稿では，埋まっているエントリの数を経路表サイズとして扱う．経路表サイズの上限は 2 次元配列の大きさであり，$(k-1)\lceil \log_k n \rceil$ となる．

3.2　finger table の更新

N_u から時計回りに 2 の整数乗個離れたノードの集合 $S_u = \{N_{u\oplus 2^p} \mid 0 \le p < \lceil \log_k n \rceil, p \in \mathbb{Z}\}$ を考える．

N_u の finger table，N_u.fingers[i][j] において S_u の要素は必ず列 $j = 2^x - 1$ ($0 \le x < m$) に存在する．例えば，図 1 において S_0 の要素（$N_1, N_2, N_4, N_8, N_{16}, N_{32}, \ldots$）は，すべて列 0 と列 1 に存在している．一般に N_u の finger table は S_u のすべての要素を含む（定理 1）．

定理 1. 定常状態で，ノード N_u の *finger table* は，$0 \le p < \lceil \log_2 n \rceil$ を満たす任意の整数 p について $N_{u\oplus 2^p}$ へのポインタを含む．

証明. 定常状態で，$N_{u\oplus 2^p} = N_u$.fingers[x][y]（ただし $x = \lfloor p/m \rfloor$，$y = 2^{p-mx} - 1$）であることを示す．

N_u.fingers[i][j] $= N_{u\oplus(j+1)k^i}$ を用いて N_u.fingers[x][y] を変形すると，N_u.fingers[x][$2^{p-mx} - 1$] $= N_{u\oplus 2^{p-mx}k^x} = N_{u\oplus 2^{p-mx}2^{mx}} = N_{u\oplus 2^p}$ が得られる．

次に，y が finger table の幅の制約 $0 \le y < k - 1 = 2^m - 1$ を満たすことを示す．$y = 2^{p-m\lfloor p/m \rfloor} - 1$ であるから，$0 \le p - m\lfloor p/m \rfloor < m$ を示せば良いが，これは関係式 $\frac{p}{m} - 1 < \lfloor \frac{p}{m} \rfloor \le \frac{p}{m}$ から容易に導ける．□

提案手法では，ノード N_u が finger table を更新する際，S_u に含まれるエントリについては Chord$^\#$ と同様の方法で取得する．すなわち，N_u は $N_{u\oplus 2^p}$ に問い合わせることで $N_{u\oplus 2^{p+1}}$ を取得する．

S_u に含まれないエントリ N_u.fingers[i][j] は，finger table の第 i 行の中で，第 j 列より左側の最も近い S_u の要素から取得できる．例えば，図 1 において S_0 ではない要素 N_3 や N_{12} は，それぞれ N_2，N_8 から取得できる．一般に，以下の定理が

成り立つ．

定理 2. 定常状態で，ノード N_u の S_u に含まれない任意のエントリは N_u.fingers$[i][2^x-1+t]$（ただし $0 \le x < m$, $1 \le t < 2^x$）と表せる．このとき，N_u.fingers$[i][2^x-1+t]$ は，左側で最も近い S_u の要素，すなわち N_u.fingers$[i][2^x-1]$ から取得できる．

証明. ノード $N_v = N_u$.fingers$[i][2^x-1]$ とする．N_v が，N_u.fingers$[i][2^x-1+t]$ を保持していることを示す．

N_u.fingers$[i][2^x-1+t] = N_{u\oplus(2^x-1+t-1)k^i} = N_{v\oplus(t-1)k^i} = N_v$.fingers$[i][t]$ □

以上をまとめると次のようになる．

1) N_u が，S_u の要素であるエントリ N_u.fingers$[i][2^x-1]$ を更新する際，次の式 (2) を用いて問い合わせ先ノードとその finger table 上の位置を求める．

$$fingers[i][2^x-1]=\begin{cases} successor & (i=0,\ x=0) \\ fingers[i-1][k/2-1].fingers[i-1][k/2-1] & (0<i,\ x=0) \\ finger[i][2^{x-1}-1].finger[i][2^{x-1}-1] & (0\le i,\ 0<x<m) \end{cases} \quad (2)$$

2) S_u に含まれない要素は，上の処理で N_u.finger$[i][2^x-1]$ の指すノードに問い合わせるときに同時に取得する．取得するエントリは式 (3) を用いて求められる．

$$fingers[i][2^x+j] = finger[i][2^x-1].finger[i][j] \quad (3)$$
$$(0 \le i,\ 0 < x < m,\ 0 \le j < 2^{x-1})$$

ノード N_0 における finger table の更新例を図 3 に示す．ここでは $k=4$ である．N_0 はまず successor である N_1 に問い合わせて N_2 へのポインタを取得し（太枠），次に N_2 に問い合わせることで N_4 へのポインタを取得し，というようにして finger table を更新する．また，ノード N_2，N_8 に問い合わせる際には，それぞれノード N_3，N_{12} へのポインタも取得する (灰色)．

提案手法における finger table の更新は，k がどのような値の場合でも，2 の整数乗個離れたノードに問い合わせるだけで完了する．このため，finger table の更新に必要なメッセージ数は，k の値に関わらず，Chord$^\#$ と同様 $2\lceil \log_2 n \rceil$（反復ルーティングの場合）あるいは $\lceil \log_2 n \rceil + 1$（再帰ルーティングの場合）である（2.2 節参照）．ただし，k が大きくなると，式 (3) を用いて取得するエントリ数が増大するため，メッセージサイズは大きくなる．

3.3 ノード数の推定

Chord$^\#$ に基づくシステムでは，P2P ネットワークに参加しているノード数を，ローカルな情報のみを用いて容易に推定できる．経路長優先方式および経路表サイズ優先方式のいずれも推定したノード数に基づいて k を変更するため，ここでノード数の推定方法を述べる．

ノード N_u が finger table を更新する際，$N_{u\oplus 2^x}$ に問い合わせることで $N_{u\oplus 2^{x+1}}$ を得る．この処理は $N_{u\oplus 2^{x+1}}$ がリングを一周して N_u を超えるまで継続する．N_u を超えた時点における x を x' とすると，$2^{x'} \le n < 2^{x'+1}$ という関係が成り立つ．提案手法では，ノード数の推定値 n_c として $2^{x'+1}$ を用いる．

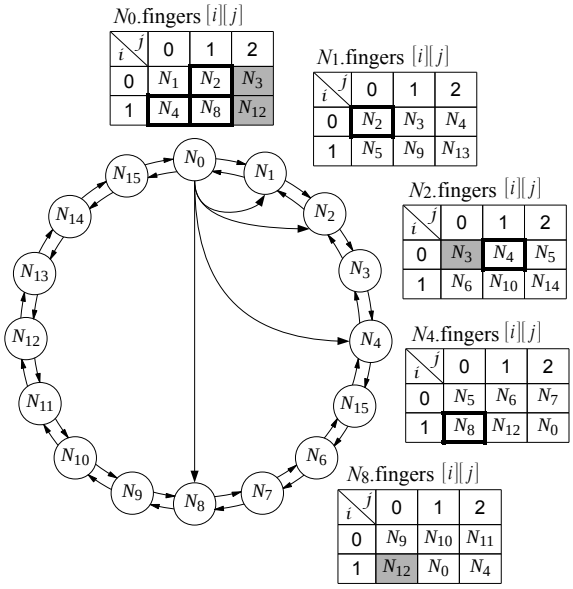


図 3 提案手法における finger table の更新例 ($k=4$)

3.4 k の変更

ここでは，経路長優先方式と経路表サイズ優先方式のそれぞれの方式の詳細を述べる．

3.4.1 経路長優先方式

経路長優先方式は，P2P ネットワーク上のノード数に関わらず，検索時の最大経路長が，設定した経路長（L_{max}）を上回らないようにする方式である．このとき，最大経路長は定数オーダとなる．

各ノードは，検索時の最大経路長が L_{max} を上回らないように，推定ノード数 n_c に基づいて自律的に k を変更する．なお，k の初期値は 4 とする．

k と推定ノード数 n_c に対し，予測される最大経路長を $L(k)$ とする．$L(k) = \lceil \log_k n_c \rceil$ である．各ノードは，finger table の更新処理を開始する前に，以下のようにして k を調整する．

- $L(k) \ge L_{max}$ の場合，推定ノード数に対して k が小さすぎる．このため，$L(k) < L_{max}$ になるまで，k の値を増加させる（$k = 2^m$ であるため，k を 2 倍する）．
- 逆に k に対して推定ノード数が少なすぎる場合，k を減少させて（k を 1/2 にする）経路表サイズを削減する．このとき，単純に $L(k) < L_{max}$ のときに k を減少させると，ノード数のわずかな変動により k が振動する可能性があるため，ここでは $L(k/2) < L_{max}$ のときに k を減少させる（1/2 にする）こととした．ただし，k は 4 を下回らないようにする．

ノード数 n が単調に増加するとき，$k = 2^{\lceil \frac{log_2 n}{L_{max}} \rceil}$ となり，ま

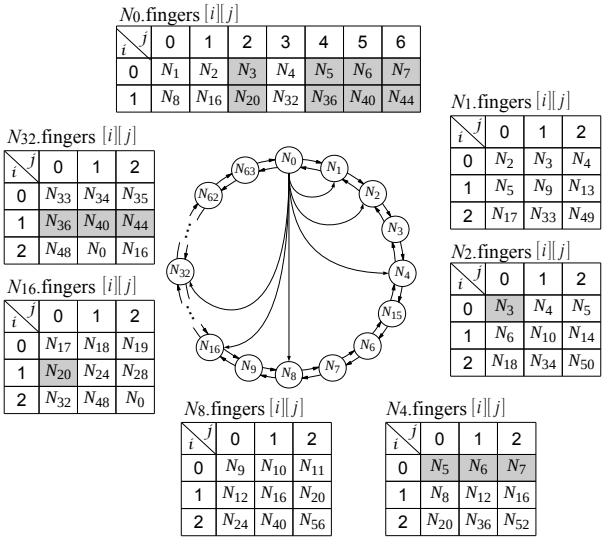


図 4　k が異なるときの finger table の更新例

i \ j	0	1	2
0	N_1	N_2	N_3
1	N_4	N_8	N_{12}
2	N_{16}	N_{32}	N_{48}
3	N_{64}	-	-

$k = 4$

i \ j	0	1	2	3	4	5	6
0	N_1	N_2	N_3	N_4	N_5	N_6	N_7
1	N_8	N_{16}	N_{24}	-	-	-	-

$k = 8$

図 5　$k = 4$, $k = 8$ の finger table の例（$S_{max} = 10$）

た経路表サイズは $O(L_{max} \times n^{\frac{1}{L_{max}}})$ となる[*2]．

k の変更は各ノードで別々の契機で行われるため，ノードごとに k は異なる可能性がある．このため，ノード N_u が finger table の更新のためにノード N_x に問い合わせた時に，N_u の k と N_x の k が等しくない場合がある．図 4 は，ノード N_0 のみ $k = 8$，それ以外のノードでは $k = 4$ の状態で，N_0 が finger table を更新した場合を示している．このとき，N_0 の finger table の一部のエントリには，正しいノードが入らない．例えば，N_0 の，fingers[1][2]，fingers[1][4]，fingers[1][5]，fingers[1][6] は，本来それぞれ N_{24}，N_{40}，N_{48}，N_{56} が入るべきであるが，そうなっていない．

この状況は過渡的なものであり，時間の経過によりすべてのノードの k が同一の値になれば解消される．

3.4.2　経路表サイズ優先方式

経路表サイズ優先方式は，経路表サイズの最大値（S_{max}）を固定する方式である．$n \leq S_{max}$ の場合は 1 ホップルーティングとなり，$S_{max} < n$ の場合は経路長が対数オーダで徐々に増加する．

この方式では，各ノードは与えられた経路表サイズの中で，可能な限り経路長を短くするように k を調整する．k の初期値は $2^{\lceil \log_2 S_{max} \rceil}$ である．

図 5 は，$S_{max} = 10$ のときの N_0 の finger table を，$k = 4$ および $k = 8$ の 2 つの場合について示したものである．$k = 4$ および $k = 8$ のそれぞれの場合で，N_0 から最も離れたノードは N_{64} および N_{24} である．一般に，k を与えた時の，finger table 中で最も離れたノードまでの距離（ノード数）$d(k)$ は以下の式で計算できる．

$$d(k) = (S_{max} \bmod (k' - 1))k^{\lfloor \frac{S_{max}}{k-1} \rfloor}. \qquad (4)$$

[*2] より正確には，経路表サイズは $(k-1)\lfloor \log_k n \rfloor + \left\lceil \frac{n}{k^{\lfloor \log_k n \rfloor}} \right\rceil - 1$ である．ただし $k = 2^{\lceil \frac{log_2 n}{L_{max}} \rceil}$．

$d(4) = 64$，$d(8) = 24$ である．

各ノードは，finger table の更新処理を開始する前に，以下のようにして k を調整する．

- $d(k) \geq n_c$ の場合，より広い範囲のノードをカバーするために k の値を 1/2 に減少させる（ただし 4 を下回らないようにする）．
- $d(2k) < n_c$ の場合，k の値を 2 倍に増加させる．ここで $d(k)$ ではなく $d(2k)$ を用いているのは，経路表サイズ優先方式と同じ理由である．

ノード N_u が finger table を更新する際，N_u から 2 の整数乗離れたノード群から取得したすべてのエントリを N_u の finger table に入れると，経路表サイズが S_{max} を超えてしまう場合がある．このような場合，finger table の中で，2 の整数乗離れたエントリ（つまり S_u に含まれるエントリ）を優先的に残し，それ以外のエントリを，最も遠いものから順次削除することで経路表サイズを S_{max} 以内に収める．なお，2 の整数乗離れたエントリを優先的に残すのは，finger table の更新の際に 2 の整数乗離れたエントリを利用するためである．

図 6 は $S_{max} = 7$ の場合の例を示している．N_0 は，N_8 から N_{12} を，N_{16} から N_{48} を取得するが，これらを finger table に入れるとエントリ数が 9 となり S_{max} を上回ってしまう．このため，S_0 の要素ではないエントリから最も遠いノード 2 つ（N_{48}，N_{12}）を削除している．

3.5　ルーティングアルゴリズム

提案手法では，経路長優先方式と経路表サイズ優先方式があるが，いずれの方式においても基本的なルーティングアルゴリズムは同じである．

以下はノード u がキー x を検索するアルゴリズムである．

1. u が x を保持していれば，それを結果として返し，終了する．
2. $x \in (u.key, u.successor.key)$ であれば，キー x を保持するノードが存在しないため，nil を返して終了する．なお，$x \in (x_1, x_2)$ はリング上の key 空間で x_1 から x_2 間での

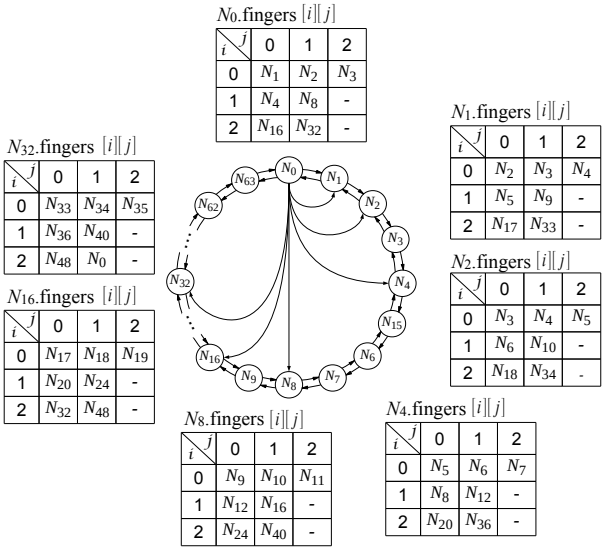


図 6　S_{max} よりエントリ数が上回ったときの finger table の更新例（$S_{max} = 7$）

範囲（ただし，両端は含まない）に x が含まれる場合に真となる．

3. u の finger table にキー x のエントリがあれば，当該エントリの指すノードに検索クエリを転送し，終了する．
4. ノード集合 $X = \{v \mid v \in u.\text{fingers}[i][j] \wedge v.key \in (u.key, x)\}$ の中からキーが x に最も近いノードに検索クエリを転送し，終了する．

4　関連研究

既存の構造化 P2P ネットワークの多くは検索時の経路長が $O(\log n)$ である．そこで，対数の底を選択することによって経路長を調整する方式が提案されている [2–6]．これらの方式は，底の値を大きくするほど経路長が短く，経路表サイズが大きくなり，逆に底の値を小さくするほど経路長が長く，経路表サイズが小さくなるという特徴を持つ．ただし，既存の方式はいずれも底の値を P2P ネットワークの運用中に動的に変更することは考慮されていない．提案手法では，底の値を動的に変更することができるため，ノード数に関わらず最大経路長を制限したり，固定された経路表サイズに応じて検索時の経路長を調節することができる．

また，提案手法の経路表サイズ優先方式と同様に，ノード数が経路表サイズより小さい場合は 1 ホップ，大きい場合は対数オーダのルーティングが可能な P2P ネットワークとして FRT-Chord [7] がある．FRT-Chord は経路表の管理を経路表エントリ分布の管理と捉え，システムで与えられた目的分布に近づくように経路表エントリを管理する．FRT-Chord は，ノード間に送受信される検索クエリを利用して経路表の更新を行うため，経路表の維持管理に追加のメッセージを必要としない．しかし，ノードの参加・離脱の頻度が高い場合や検索クエリの頻度が低い場合は経路表エントリ分布の管理が困難となる．5.2 節では，提案手法の経路表サイズ優先方式と FRT-Chord の定量的比較を行う．

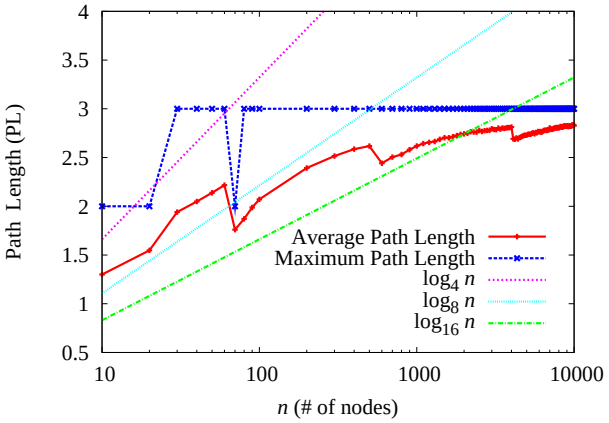


図 7　経路長優先方式における経路長の変化（$L_{max} = 3$）

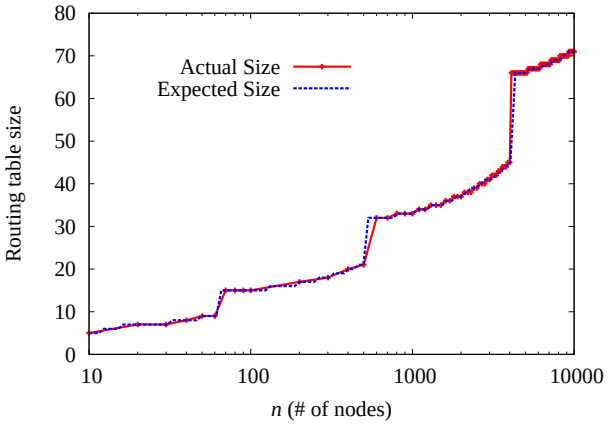


図 8　経路長優先方式における経路表サイズの変化（$L_{max} = 3$）

5　評価

経路長優先方式と経路表サイズ優先方式をシミュレーションにより評価した．

5.1　経路長優先方式

経路長優先方式において，経路長の最大値（L_{max}）を 3 と設定し，ノード数を $10, 20, 30, \cdots, 100, 200, 300, \cdots, 10000$ と変化させたときの平均経路長と最大経路長を測定した．各ノードには 0 から $2^{31} - 1$ の範囲の乱数で key を付与した．各ノード数において，ランダムに選択した 2 つのノード間の経路長の測定を 10,000 回試行した．結果を図 7 に示す．横軸はノード数，縦軸は経路長である．また，k が変化するタイミングを確認するために，$\log_4 n$，$\log_8 n$，$\log_{16} n$ もプロットした．

グラフより，ノード数が増加しても最大経路長は常に $L_{max} = 3$ 以下であり，定数オーダルーティングが実現できていることが確認できる．また，$Y = L_{max}$ と，$\log_4 n$，$\log_8 n$，$\log_{16} n$ が交差している付近（それぞれ $n = 64$，$n = 512$，$n = 4096$）で，平均経路長が一旦減少している．これは，ここで k が増加し，平均経路長が減少したためである．

上記の実験を行ったときの，各ノードの経路表サイズの平均値を図 8 に示す．横軸がノード数，縦軸は経路表サイズであ

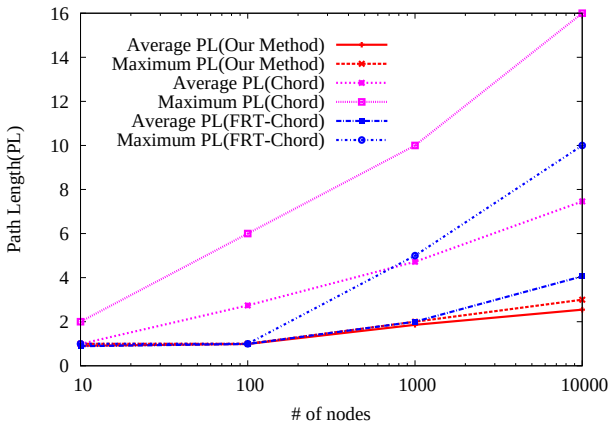


図 9　経路表サイズ優先方式における経路長の変化（$S_{max} = 160$）

る．比較のため理論値もプロットしている．

グラフより，ノード数が増加するに従い，経路表サイズは緩やかに増加している．経路表サイズが急激に増えているところは，k が増加したところである．

5.2　経路表サイズ優先方式

経路表サイズ優先方式において，経路表サイズの最大値（S_{max}）を 160 と設定し，ノード数 $n = 10, 100, 1000, 10000$ と変化させたときの平均経路長と最大経路長を測定した．実験方法は 5.1 節と同様である．

結果を図 9 に示す．横軸はノード数，縦軸は経路長である．文献 [7] では FRT-Chord と Chord に対して同様の実験を行った結果が示されているため，参考のためにその結果もプロットした．なお，FRT-Chord と Chord におけるルーティングアルゴリズムは，ノード u からノード v に到達する際に v の predecessor に到達してから，v に到達するため，提案手法のルーティングアルゴリズムより平均 1 ポップ大きい．公平性を保つために，1 ホップずつ減らした値をプロットしている．

グラフより，ノード数が S_{max} より小さい場合（$n = 10$, $n = 100$），提案手法は 1 ホップルーティングが実現できていること，また，ノード数が S_{max} より大きい場合（$n = 1{,}000$, $n = 10{,}000$）は最大経路長・平均経路長とも対数オーダで増加することがわかる．

Chord や FRT-Chord に比べて提案手法は最大経路長が短いことも確認できる．例えば，$n = 10,000$ において，提案手法の最大経路長は 3 である．これは，提案手法が（定常状態では）正確に k 分探索を行えるためである．ただし，提案手法は経路表を定期的に更新する必要があり，検索クエリによって経路表を学習する FRT-Chord よりも経路表を維持するために必要なトラフィックは大きくなる．

6　おわりに

本稿では，検索時の経路長あるいは経路表サイズを柔軟に設定可能な経路表構築手法を提案した．提案手法では，Chord$^{\#}$ の finger table を 2 次元化し，k 分探索が可能な経路表を構築している．その上で，k の値を動的に変更可能にすることによって，検索時の最大経路長を一定以下に保つことができる経路長優先方式と，ノード数に応じて 1 ホップからシームレスに対数オーダルーティングに移行可能な経路表サイズ優先方式を実現した．

以下に本研究の今後の課題をいくつか挙げる．

- 著者らは，これまでに，Chord$^{\#}$ における finger table の更新コスト削減手法（Chord$^{\#\#}$）を提案した [8]．Chord$^{\#\#}$ では，ノード離脱時の finger table 更新処理の高速化やネットワーク近接性を利用したルーティングを実現している．本稿における提案手法に関しても，Chord$^{\#\#}$ と同様な拡張の適用を検討する．
- 提案手法を実ネットワーク上で実装し，評価を行う．

参考文献

[1] Stoica, I., Morris, R., Liben-Lowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F. and Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications, *IEEE/ACM Transactions on Networking*, Vol. 11, No. 1, pp. 17–32 (2003).

[2] Rowstron, A. and Druschel, P.: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350 (2001).

[3] Maymounkov, P. and Mazières, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, *Interantional Workshop on Peer-to-Peer Systems* (2002).

[4] El-Ansary, S., Alima, L. O., Brand, P. and Haridi, S.: A Framework for Peer-to-Peer Lookup Services based on k-ary search, Technical Report TR-2002-06, SISC (2002).

[5] Zhao, B. Y., Kubiatowicz, J. D. and Joseph, A. D.: Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing, Technical Report UCB/CSD-01-1141, UC Berkeley (2001).

[6] Alima, L., El-Ansary, S., Brand, P. and Haridi, S.: DKS(N, k, f): A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications, *the 3rd International workshop on Global and P2P Computing on Large Scale Distributed Systems, CCGRID 2003* (2003).

[7] Nagao, H. and Shudo, K.: Flexible Routing Tables: Designing Routing Algorithms for Overlays Based on a Tatal Order on a Routing Table Set, *Proc. IEEE P2P'11*, pp. 72–81 (2011).

[8] 呉　承彦，安倍広多，石橋勇人，松浦敏雄：Chord$^{\#}$ における経路表の維持管理コスト削減手法の提案とその評価，情報処理学会論文誌，Vol. 53, No. 12, pp. 2752–2761 (2012).