Vol.2012-IOT-16 No.4
2012/3/15

距離が付加された要素集合をコンパクトに表現できるDistance Bloom Filterの提案とP2Pネットワークにおける最短経路探索への応用

西川大器[†1]　安倍広多[†1]
石橋勇人[†1]　松浦敏雄[†1]

P2P ネットワークにおいて重要な機能の 1 つは，あるデータを検索し，そのデータを持つノードを特定することである．この際，目的のノードに到達するまでの遅延時間をできるだけ小さくできることが望ましいが，従来の多くの手法では，局所的な情報のみを利用して経路を決定するため，必ずしも最短経路が選ばれるわけではない．この問題を解決するため，本稿では，経路ごとの遅延時間を空間効率良く保持できるデータ構造である Distance Bloom Filter，ならびに，これを用いて高い確率で最短経路を選択可能な手法を提案する．また，提案手法を構造化 P2P ネットワークの 1 つである Skip graph に適用したシミュレーションを行い，その有効性を確認した．

A Bloom Filter Extension for Storing Elements With Distance and Its Application for Finding Shortest Paths in P2P Networks

Taiki Nishikawa,[†1] Kota Abe,[†1] Hayato Ishibashi[†1] and Toshio Matsuura[†1]

One of the key functions of P2P networks is locating a node that stores target data. This is performed by routing a search message with the key corresponding to the data over the overlay network. Minimizing the latency of this process is not fully achieved by most P2P systems since they only use local information to determine the route. This paper proposes a novel routing method for P2P systems that finds the shortest path to the destination with high probability. *Distance Bloom Filter* as a space-efficient data structure to store distance information is introduced to support the method. Simulation results of the method applied to skip graphs, a structured P2P network, are also reported.

1. はじめに

P2P ネットワークでは，目的のデータのキーを指定した検索クエリがオーバーレイネットワーク上で転送され，最終的に目的のデータを保持するノード（目的ノード）に到達するという処理が一般に行われる．

P2P ネットワークはノードと，ノード間を接続するリンクから構成されるが，ネットワーク的に近いノード間を結ぶリンクと，遠いノード間を結ぶリンクでは通信遅延時間（ネットワーク距離）が大幅に異なる．効率が良い P2P ネットワークを実現するためには，ネットワーク距離を考慮することが必要である．

P2P アルゴリズムにおいてネットワーク距離を考慮する手法は，大きく (1) 経路表にネットワーク距離が小さいノードを格納しておく Proximity Neighbor Selection, (2) 次ホップを選択する際にネットワーク距離を考慮する Proximity Route Selection, (3) ノード ID を決定する際にネットワーク距離を考慮する Proximity Identifier Selection，に分類される[1)]．しかし，これらの手法は経路を局所的な視点で選ぶため，可能なすべての経路の中から最短な経路を選べるわけではない．

本稿では，P2P ネットワークにおいてキーの検索を行う際に，あるホップ数以内のすべての経路の中からネットワーク距離が最小となる経路を高い確率で選択する手法を提案する．また，これを実現するためのデータ構造として距離情報が付加された要素の集合をコンパクトに表現できる Distance Bloom Filter も提案する．

以下，2 章で，本研究がベースとしている Bloom Filter について概説し，3 章で提案手法である Distance Bloom Filter について述べる．4 章では Distance Bloom Filter の P2P ネットワークへの適用手法について述べ，5 章でシミュレーションによる評価を述べる．6 章で考察を加え，最後に 7 章で今後の課題とまとめを述べる．

2. Bloom Filter

本稿において提案する Distance Bloom Filter は，Bloom Filter[2)] を基本として拡張を加えたものである．そこで，本章では，Bloom Filter について簡単に説明する．

†1 大阪市立大学大学院創造都市研究科
Graduate School for Creative Cities, Osaka City University

Bloom Filter は，集合を表現するためのデータ構造である．Bloom Filter を用いると，集合の要素そのものを格納するよりもコンパクトに集合を表現できる．また，指定した要素が集合に含まれるかどうかを高速に判定できる．

Bloom Filter は，0 で初期化された長さ m のビット配列 $b[1]\dots b[m]$ と，1 から m までの値を返す独立した $k(0<k)$ 個のハッシュ関数 $(h_1, h_2, \dots, h_k)$ を用いる．

Bloom Filter B に，要素 x を追加する場合，$B.b[h_i(x)]$ を 1 にする $(1 \le i \le k)$．

Bloom Filter B に要素 x が含まれるかどうかを判定する場合，各 $B.b[h_i(x)]$ のビットを調べる $(1 \le i \le k)$．いずれかのビットが 0 ならば x は集合に含まれない．すべてのビットが 1 ならば，x は集合に含まれていると判定する．しかし，Bloom Filter では，実際には x が含まれない場合でも含むと誤って判定する場合がある（これを**偽陽性**と呼ぶ）．これは x が集合に含まれていない場合でも，別の要素の追加によって $b[h_i(x)]$ のすべてが 1 になる場合があるためである．集合に含まれる要素の数を n とするとき，偽陽性が発生する確率 f は

$$f = (1-(1-1/m)^{kn})^k \approx (1-e^{-kn/m})^k \tag{1}$$

で与えられ，また，f を最小にする k は

$$k = (m/n)\ln 2 \tag{2}$$

となる．さらに，n と f を与えたときの m は

$$m = -n(\ln f)/(\ln 2)^2 \tag{3}$$

であることが知られている．

2 つの Bloom Filter A と B があるとき，A と B のそれぞれのビット配列のビット単位 OR を計算することで A と B の和集合を表す Bloom Filter C を作ることができる（本稿ではこれを Bloom Filter の統合と呼ぶ）．統合するためには，Bloom Filter A と B のビット列の長さ m と k 個のハッシュ関数 $(h_1, h_2, \dots, h_k)$ が同一である必要がある．

なお，Bloom Filter では集合に一度追加した要素は削除できない．

3. Distance Bloom Filter

本章では，**Distance Bloom Filter**（以下 DBF）を提案する．DBF は Bloom Filter と同様，要素の集合を表現するデータ構造であるが，各要素に $p\,(>1)$ ビットの整数で表現される距離の情報を保持できることが特徴である．Bloom Filter と同様，DBF はエラーを許容することでコンパクトに集合を表現する．

3.1 データ構造

DBF では，Bloom Filter と同様，長さ m の配列 $g[1]\dots g[m]$ を用いるが，配列の各要素は p ビットの大きさを持ち，$E = 2^p - 1$ で初期化される．ハッシュ関数については Bloom Filter と同様である．

3.2 要素の追加

DBF N に要素 x を追加する場合，$N.g[h_i(x)] \leftarrow \min(N.g[h_i(x)], x.\mathrm{dist})$ とする $(1 \le i \le k)$．ただし，$x.\mathrm{dist}$ は要素 x に付随する距離で，$0 \le x.\mathrm{dist} \le M$ を満たす必要がある．ここで，M は格納できる最大の距離で，$M = 2^p - 2 = E - 1$ である．

x を格納するときに，ハッシュ関数で求めたインデックスの指す値が E ではない場合，他の要素の格納時に同一のインデックスが使用されたことを示している．これを**衝突**と呼ぶ．あるインデックスで衝突が発生した場合，当該インデックスにすでに入っていた値と $x.\mathrm{dist}$ の小さい方の値を格納する．

3.3 要素の削除

DBF では，Bloom Filter と同様に一度追加された要素は削除できない．

3.4 要素の取得

DBF では指定した要素 x が集合に含まれるかどうかを判定し，含まれる場合は x の距離を取得することができる（この操作を取得と呼ぶ）．

DBF N から，要素 x を取得する場合，各 $N.g[h_i(x)]$ の値を調べる $(1 \le i \le k)$．$N.g[h_i(x)]$ が 1 つでも E ならば x は集合に含まれず，そうでなければ含むと判定する．このとき，要素 x の距離は以下の式を用いて計算する．

$$\max_{i=1}^{k}(N.g[h_i(x)]) \tag{4}$$

Bloom Filter と同様，$N.g$ の各インデックスは複数の要素で共用しているため，正しい距離が得られない場合がある．正しい距離が得られないのは，$h_1(x)$ から $h_k(x)$ のすべてのインデックスにおいて $x.\mathrm{dist}$ より小さい値と衝突している場合に限る．このとき得られる距離は本来の距離より小さい．（別の言い方をすると，要素 x が集合に含まれる場合，x を取得して得られる距離は，本来の距離より大きくなることはない．）

3.5 Distance Bloom Filter の統合

2 つの DBF を統合し，和集合を表す DBF を作ることができる．Bloom Filter の場合と同様，配列の長さ m と k 個のハッシュ関数 $(h_1, h_2, \dots, h_k)$ は同一である必要がある．

DBF A と DBF B を統合して DBF C を得る場合，$C.g$ の各インデックスの値は以下の

式を用いて計算する．

$$C.g[i] \leftarrow \min(A.g[i], B.g[i]) \quad (5)$$

DBF A と B に共通して含まれる要素 x があるとき，A と B を統合した DBF で x を取得すると，得られる距離は A で得られる距離と B で得られる距離の小さい方になる．

3.6 Distance Bloom Filter への距離の加算

以下の操作により，DBF N が保持しているすべての要素の距離に対して任意の距離 $c(>0)$ を加算できる．距離は最大値 M を超えないようにしている．

$$N.g[i] \leftarrow \begin{cases} E & (\text{if } N.g[i] = E) \\ \min(N.g[i] + c, M) & (\text{otherwise}) \end{cases} \quad (6)$$

3.7 Distance Bloom Filter の性質

Bloom Filter と同様，DBF では要素の追加や取得は定数時間で実行できる．

DBF でも偽陽性があり，集合に含まれない要素 x が含まれると誤って判定することがある．このとき，x は式 4 で計算された（無意味な）距離を返す．DBF での偽陽性の発生確率は，Bloom Filter と同じであり，式 1 で与えられる．

3.4 節で述べたように，DBF である要素 x を取得する際，$h_1(x)$ から $h_k(x)$ のすべてのインデックスにおいて x.dist より小さい値と衝突している場合，本来の距離と異なる距離が得られる．(値の大小に関わらず）すべてのインデックスで衝突が発生する確率は偽陽性確率と等しいため，この事象が発生する確率は偽陽性確率以下である（すべてのインデックスで x.dist より小さい値と衝突する確率は x.dist の値と距離の分布に依存する）．

4. Distance Bloom Filter の P2P ネットワークへの適用

ここでは DBF を P2P ネットワークに適用することで，高い確率でネットワーク距離が最小となる経路を探索する手法について述べる．

提案手法は何らかの既存の P2P アルゴリズム（**ベース P2P アルゴリズム**と呼ぶ）と組み合わせて使用する．オーバーレイネットワークのトポロジの構築や維持はベース P2P アルゴリズムを用いる．以下詳細を述べる．

4.1 方　　針

ノード p のリンク先ノード集合を $p.N$，ノード p がノード q に接続するリンクを l_{pq} と表記する．ノード p は，各 $q\,(\in p.N)$ に対し DBF D_{pq} を割り当てる．D_{pq} には，l_{pq} を経由して到達可能な各ノード v が保持するキー k を格納する．このとき，k の距離が，p から l_{pq} を経由して v に到達するさまざまな経路の中で最小のネットワーク距離となるようにする．各ノードの DBF を構築する基本的な考え方は以下のとおりである．

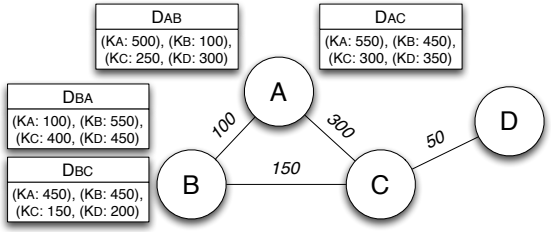

図 1　P2P ネットワークにおける Distance Bloom Filter の例

各ノード p は，隣接ノード q に対し，D_{pq} 以外のすべての DBF $\{D_{px}|x \in (p.N - q)\}$ を統合し，さらに p が保持するキー k を追加した DBF D を送信する（k.dist $= 0$ とする）．DBF D を受信したノード q では，q から p までの距離（ping などで測定しておく）を D に加算したものを D_{qp} とする．これを各ノードが繰り返すことで，各ノード v が保持するキーと，v までの最小距離が，すべてのノードの DBF に伝搬する．これは 3.5 節で述べた，2 つの DBF を統合する際に小さいほうの距離が保存される性質を利用している．

ノード p がキー k に対する検索クエリを受信した際は，p の各 DBF の中から最も k への距離が小さい DBF D_{px} を探し，ノード x に対してクエリを転送する．これを繰り返すことで，(高い確率で）k への最短経路を通るルーティングが実現できる．

図 1 を用いて説明する．図のリンク上の数字はネットワーク距離を表している．ノード A は 2 つのリンク l_{AB} および l_{AC} と，それぞれに対応した DBF D_{AB} および D_{AC} を保持している．ノード A がノード D の保持するキー k_D を検索する場合を考える．A が D_{AB} と D_{AC} から k_D を取得すると，それぞれ 300 と 350 が得られるため，小さい方のノード B に対して検索クエリを送信する．クエリを受信したノードも同様の処理を繰り返すことで最終的に k_D を保持するノード D に最短経路で到達できる．

しかし，この方法では以下の問題がある．

(1) DBF では要素の削除ができないため，ノードが離脱したり，ノード上のデータが削除されても，離脱や削除前の情報に基づいてルーティングすることになり，間違ったノードに到達する確率が上昇する．(例えば，図 1 においてノード D が離脱しても，例えば C が A に送信する DBF に含まれる D_{CB} には D のキーが格納されたままで

あるため，D のキーが消えることはない.）

(2) P2P ネットワーク中のすべてのキーが各ノードの DBF に追加されるため，偽陽性を抑えようとすると DBF のサイズ (m) をかなり大きくする必要がある．

(3) DBF では得られる距離が正しくない場合があるため，経路がループする場合がある．

以下，4.2 節で (1)，(2) の，4.3 節で (3) の問題について対処法を述べる．

4.2 Distance Bloom Filter の拡張

上記 (1)，(2) の問題を解決するため，ノード p が保持する DBF が含む要素を，p から 2^d ホップ数以内のノードが保持するキーに制限することにした（d は定数）．このために，DBF で距離情報を格納している配列 g に加え，ホップ数の情報を格納する配列 $u[1]\dots u[m]$ を追加する．配列 u の各要素は d ビットの大きさを持ち，$E' = 2^d - 1$ で初期化する．

このように拡張した DBF をノード p がノード q に送信する場合，以下のステップを実行する．

(1) 統合した DBF D を求める（4.1 節参照）．統合するとき，u に関しては式 5 と同様に計算する．

(2) $u[i] = E'$ であるエントリがあれば $D.g[i] = E$ とする．これにより q から 2^d ホップより遠いノードのみが保持する要素を削除する．

(3) $D.u$ のすべてのインデックスについて 1 を加算する．

(4) D に対して p が保持するキー k を追加する．このとき $D.u[h_i(x)] \leftarrow 0$ とする $(1 \le i \le k)$.

(5) D を q に送信する．

例えば図 1 の例で D を削除した時，D のキーに対応する u のエントリは DBF が伝搬されるごとに 1 加算されていくため，D のキーはいずれ消滅する．

4.3 ルーティング

拡張した DBF は，自ノードの周辺 2^d ホップ以内のキーしか保持しないため，それより遠いノードのみが保持するキーへの経路は見つけることができない．このため，自ノードの DBF からキーが見つからなかった場合は，ベース P2P アルゴリズムのルーティングに切り替える．ベース P2P アルゴリズムのルーティングによって目的キーを保持するノードに近づくことで，DBF にキーを発見できる場合がある．この場合は提案手法のルーティングに切り替える[⋆1]．

⋆1 この機能は 5 章で用いたたシミュレータでは未実装である．

また，DBF が返す距離は必ずしも正しくないため，経路がループする可能性がある．このため，検索クエリには通過したノードを格納しておき，各ノードが次ホップに選んだノードが通過したノードの場合はベース P2P ネットワークのルーティングに切り替える．

4.4 ネットワーク距離のマッピング

DBF では距離として 0 から M の範囲の整数を用いる．実際のネットワーク距離（例えばミリ秒単位の値）をこの範囲にマッピングするために，実際の距離を適切なパラメータ G で割った値を DBF での距離として使用する．G は，各ノードの DBF に格納されるキーの実際の距離の分布を調べ（これは d によって変化する），その大部分が 0 から M の範囲にマッピングできるように選ぶ必要がある．

4.5 Distance Bloom Filter の伝搬

ノードが DBF を他のノードに送信する契機は，(1) ノードが P2P ネットワークに参加した場合，(2) 保持するキーが変化した場合，(3) 隣接ノードとの距離が変化した場合，(4) 隣接ノードが削除された場合，(5) 他のノードから DBF を受信した場合，である．

これらの処理を効率的に行うために，各ノード p は，リンク先の各ノード q に送信した DBF を保持しておく．q に再度 DBF を送る際に，前回送信した DBF との差分を転送することでネットワーク上に流れるメッセージサイズを削減できる．また，距離（配列 g の値）が少し変化しただけという場合は，DBF を送信しないということも可能である．

4.6 Distance Bloom Filter の空間効率

拡張した DBF のサイズは $m(p+d)$ ビットである．DBF を使わずに，同じ集合を配列で表現した場合，配列のサイズは $n(s+p+d)$ ビットとなる．ここで n は要素数，s は各要素そのものの大きさである．DBF は要素自体は格納しないため，要素のサイズが大きいほど DBF の方が有利となる．

DBF と配列のサイズの比は $m(p+d)/n(s+p+d) = -\frac{(\ln f)(p+d)}{(\ln 2)^2(s+p+d)}$ である．キーのサイズ s として，分散ハッシュテーブル (DHT) の Chord などで用いられる 160 ビットを使った場合，$f = 0.1, p = 3, d = 2$ のとき，DBF は配列の約 15%，$f = 0.2$ のときは約 10%のサイズで集合を表現できる．

5. 評　　価

提案手法を評価するため，ベース P2P ネットワークとして Skip graph[3] を用いてシミュレーションを行った．

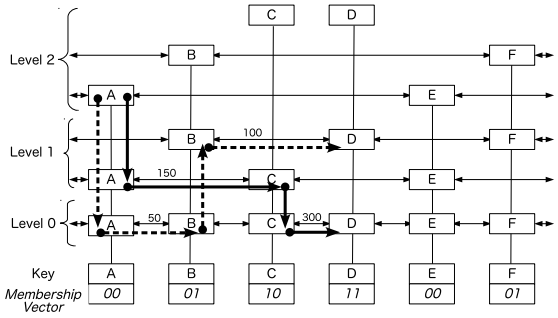

図 2 Skip graph

5.1 Skip graph

Skip graph は，構造化 P2P ネットワークの 1 つであり，キーの範囲検索が可能という特徴がある．**図 2** に Skip graph の構造を示す．縦の線 1 本が 1 つのノードを表す．各ノードはキーと，membership vector と呼ばれる w 進数の乱数を持つ（本稿では $w = 2$ とする）．横方向の細い矢印はノード間のリンクを示している．リンク上の数字はネットワーク距離である．

Skip graph には複数のレベルがあり，レベル i には高々2^i 個の双方向連結リストが存在する．レベル i では，membership vector の下位 i 桁が一致するノードが同じ連結リストに所属する（レベル 0 の連結リストはすべてのノードが所属する）．ノードは各連結リストでキーの昇順に接続される．各ノードは各連結リストの左右のノードへのリンクを保持する．ノード数を n とすると，各ノードは平均 $O(\log n)$ 個のリンクを保持する．

図 2 を用いて Skip graph の検索方式を簡単に説明する．太線はノード A がキー D を検索した場合の経路を示している．A は最上位レベルから D を超えない最大のキーを探す．レベル 1 でキー C を発見するため，ノード C に検索クエリを送信する．ノード C も同じ処理を行い，ノード D に到達できる．Skip graph では，$O(\log n)$ ホップでキーの検索が可能である．

Skip graph は特にネットワーク距離を考慮しない．上記の例では，A から D に到達するまでに 450 単位時間を要しているが，最短経路は破線で示した A → B → D という経路であり，150 単位時間で到達できる．提案手法を用いると，高い確率でこの経路を発見できる．

5.2 実験方法

Skip graph をベース P2P ネットワークとして利用し，提案手法による検索時間の削減効果を測定した．実験ではインターネットトポロジシミュレータである Inet-3.0[4] を用いて 4096 ノードの仮想ネットワークを作成し，ここからランダムにノードを選択して Skip graph を構築した．各ノードには 1 から始まるシーケンシャルな番号をキーとして付与した．

この Skip graph を用いて，通常の Skip graph と提案手法の両方で，ノード A から B を検索して B に到達するまでの時間（経由したリンクのネットワーク距離の総和）と仮想ネットワーク上でノード A と B が直接通信した場合の時間との比率（ストレッチ）の平均を求めた．なお，提案手法では，Skip graph の各レベルにある左右のリンクそれぞれに対して 1 つの DBF を付与する．参考のために，ダイクストラ法によって計算した最短経路による検索時間も求めた．ノード A，B の組み合わせは総当りで行った．

シミュレーションで用いたパラメータを**表 1** に示す．DBF は 2^d 以内のノードのキーしか格納しない．ノード数 1000 の Skip graph において各ノードがキーを 1 つ保持するとして，$d = 1, 2, 3$ のそれぞれの場合で DBF に格納されるキーの数の分布から 90 パーセンタイル値を求め，これを Bloom Filter の要素数 n とした．また，それぞれの場合について偽陽性 f が 0.1, 0.2, 0.3 の場合の m と k をそれぞれ式 3，式 2 により求めた．

G は 4.4 節で示した考え方に基づいて決定した．それぞれの d について，ノードが 2^d ホップ以内で到達可能なすべてのノードへの通信時間の分布から 90 パーセンタイル値を求めたところ，表のようになった．G は，p を与えたときに $G \times (2^p - 1) = 90$ パーセンタイル値となるように決定した．

表 1 シミュレーションで用いたパラメータ

		m			k			90 パーセンタイル
d	n	f=0.1	0.2	0.3	0.1	0.2	0.3	通信時間
1	13	76	56	37	4	3	2	44142
2	502	2897	2172	1448	4	3	2	68827
3	1000	5771	4328	2885	4	3	2	74233

5.3 実験結果

ノード数を変化させながら提案手法による平均ストレッチを求めた（**図 3**，**図 4**, **図 5**）．横軸はノード数，縦軸は平均ストレッチである．パラメータの違いによる影響を調べるため，図 3 では $p = 3, f = 0.2$ に固定し，$d = 1, 2, 3$ と変化させた．図 4 では $d = 2, f = 0.2$ に固定し，$p = 2, 3, 4$ と変化させた．図 5 では $d = 2, p = 3$ に固定し，$f = 0.1, 0.2, 0.3$ と変化させた．比較のために通常の Skip graph での値 (Normal) と，ダイクストラ法による最短経路の値 (Dijkstra) もプロットしている．

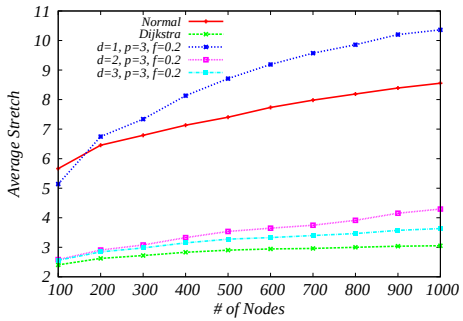

図 3 d を変化させた場合の平均ストレッチの比較

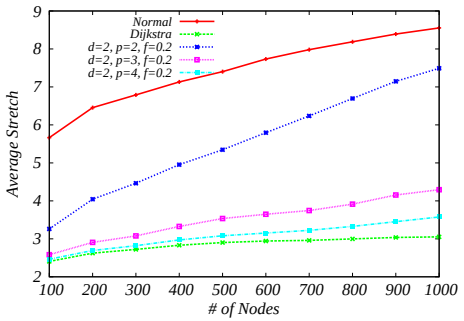

図 4 p を変化させた場合の平均ストレッチの比較

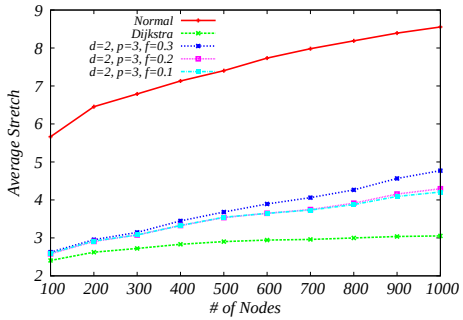

図 5 f を変化させた場合の平均ストレッチの比較

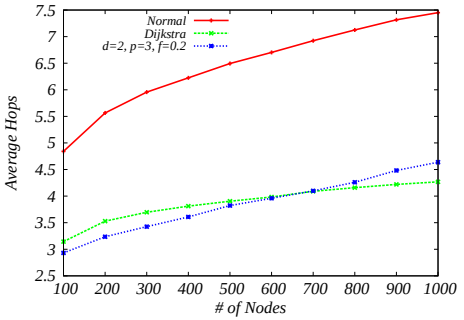

図 6 ホップ数の比較

DBF のサイズと経路の品質（最短経路との近さ）はトレードオフの関係にあるが，これらのグラフより，今回の実験環境では $d = 2, p = 3, f = 0.2$ で十分に最短経路に近い経路を求められることがわかる．このとき，1 つの DBF サイズは $2178 \times (3 + 2) = 10860$ ビット（1357.5 バイト）である．

また，このパラメータを用いてホップ数の比較も行なった（**図 6**）．グラフより，ホップ数は通常の Skip graph の約 60%に減少することがわかる．

なお，今回は最大ノード数が 1000 で，各ノードにキーが 1 つしかない環境で実験したが，適切なパラメータはノード数や各ノードが保持するキーの数によって異なる．

6. 考　　察

文献 5) には，減衰 Bloom Filter と呼ばれるデータ構造を用いて，ホップ数が最小な経路でルーティングする手法が提案されている．本稿で提案した手法は，ホップ数ではなく，通信時間を最小とする点が異なる．

提案手法では，複数のノードが同一のキーを保持している場合に，そのなかの最も近いノードにルーティングできる．これは，複製へのアクセスの効率化やオーバレイネットワークにおける Anycast などに応用できる．

7. おわりに

本稿では，距離が付与された要素の集合をコンパクトに表現できる Distance Bloom Filter を提案した．また，P2P ネットワークで DBF を用いることで，データを検索する際に高い確率でネットワーク距離が最短となる経路を選択する手法を述べた．

提案手法は，P2P ネットワーク以外でのルーティングにも応用可能と考えられる．例えば無線アドホックネットワークへの適用は興味深い．

本稿ではノードやキーが増減しない静的な環境で評価したが，より実際に近い動的な環境での評価を行なう必要がある．また，DBF のデータ転送量に関する評価や，Skip graph 以外の P2P ネットワーク（特に Proximity Neighbor Selecton や Proximity Route Selection を採用している P2P ネットワーク）における評価なども必要と考えている．

参　考　文　献

1) Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S. and Stoica, I.: The Impact of DHT Routing Geometry On Resilience and Proximity, *Proceedings of the ACM SIGCOMM '03* (2003).
2) Bloom, B.: Space/Time Trade-Offs in Hash Coding with Allowable Errors, *Communications of the ACM*, Vol.13, No.7, pp.422–426 (1970).
3) Aspnes, J. and Shah, G.: Skip Graphs, *ACM Transactions on Algorithms (TALG)*, Vol.3, No.4, pp.37:1–37:25 (2007).
4) Winick, J. and Jamin, S.: Inet-3.0: Internet Topology Generator, Technical Report CSE-TR-456-02, University of Michigan (2002).
5) Rhea, S. and Kubiatowicz, J.: Probabilistic Location and Routing, *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol.3, IEEE, pp.1248–1257 (2002).