一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IA2018-27 (2018-09)

Skip Graphにおける平均経路長の短い範囲検索クエリルーティング手法

坂野　遼平†　　首藤　一幸†

† 東京工業大学　〒152–8550 東京都目黒区大岡山 2–12–1
E-mail: † banno@computer.org, shudo@is.titech.ac.jp

あらまし　多数のノード群が自律分散的に相互発見を行うための技術として，構造化オーバレイが知られている．Skip Graph は，構造化オーバレイのアルゴリズムの一種であり，範囲検索が可能であるという特長を有する．これまで，範囲検索クエリのルーティングについてはいくつかの手法が提案されてきたが，経路長の長大化や，メッセージ数の肥大化といった問題があった．本稿では，新たな範囲検索クエリのルーティング手法を提案する．提案手法では，クエリを受信した各ノードが，隣接ノードのキーに基づいて当該クエリの対象範囲を分割し，部分領域を隣接ノードに委譲していく．これにより，範囲内の全ノードに対し，最小のメッセージ数かつ短い平均経路長でクエリを配送することが可能となる．シミュレーションによる評価を行い，既存手法に対して平均経路長を概ね 30% 以上削減可能であることを確認した．

キーワード　Skip Graph, オーバレイネットワーク, P2P ネットワーク，ルーティングアルゴリズム，範囲検索クエリ

Reducing Average Path Lengths of Range Queries on Skip Graphs

Ryohei BANNO† and Kazuyuki SHUDO†

† Tokyo Institute of Technology　2–12–1 Ookayama, Meguro-ku, Tokyo, 152–8550 Japan
E-mail: † banno@computer.org, shudo@is.titech.ac.jp

Abstract　Structured overlay networks are useful techniques to enable a large number of nodes to discover each other in an autonomous manner. Skip Graph is one of the algorithms of structured overlay networks, which has the capability of range queries. Although there are some existing methods for routing range queries, they have problems of long path length or a large number of messages. In this paper, we propose a novel routing method for range queries in Skip Graph. In the proposed method, each node which receives a range query divides its target range into subranges by the keys of its neighbor nodes, and delegates the subranges to the nodes. The proposed method enables to deliver a range query to all nodes within its target range with a shorter average path length and the minimum number of messages. By simulation experiments, we confirmed that our method can reduce the average path length roughly 30% or more compared to an existing method.

Key words　Skip Graph, Overlay networks, Peer-to-peer networks, Routing algorithms, Range queries

1. はじめに

構造化オーバレイは，多数のノード群が自律分散的に相互発見を行うことを可能とする技術である．高いスケーラビリティや耐障害性を備えることから，データストレージ [1] やビデオストリーミング [2]，pub/sub メッセージング [3] 等の大規模システムへの応用で注目を集めており，近年ではブロックチェーン [4] や IoT [5] といった分野においても適用が検討されている．

Skip Graph [6] は構造化オーバレイのアルゴリズムの一種であり，範囲検索が可能であるという特長を有する．構造化オーバレイの代表的な仕組みである分散ハッシュテーブル [7]〜[9] とは異なり，Skip Graph ではキーをハッシュしないため，キーの順序関係を保ったトポロジを構築することができる．これにより，Skip Graph のノードは，範囲を指定したクエリを発行し，当該範囲内に含まれるキーを持つノードを発見することが可能となっている．

Skip Graph における範囲検索クエリのルーティングについては，いくつかの手法が提案されている [10], [11] が，それら既存手法には経路長の長大化やメッセージ数の肥大化といった問題が存在する．例えば範囲内においてクエリをブロードキャストする方式を用いた場合，各ノードが同じクエリを 2 回以上受信する可能性があり，メッセージ数が増大してしまう．一方，

This is an unrefereed paper.

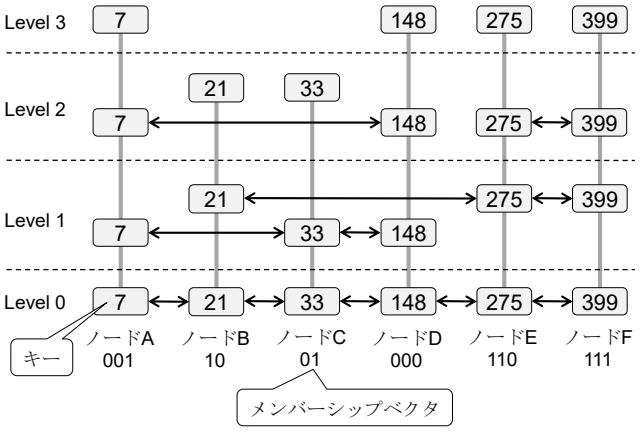

図 1　Skip Graph の例
Fig. 1　Example of Skip Graph.

範囲内各ノードがバケツリレー式にクエリを転送した場合には，範囲内ノード数が多い場合に経路長が非常に長くなる問題がある．これらの問題点を改善する手法も提案されているが，クエリの配送木が二分木を構成するという性質上，半数近くのノードが配送木の末端にて最長経路長でクエリを受信するために，依然として平均経路長が長大化する傾向を有している．

本稿では，Skip Graph における範囲検索クエリのルーティング手法として，SFB（Split-Forward Broadcasting）を新たに提案する．SFB では，クエリを受信した各ノードが，隣接ノードのキーに基づいて当該クエリの対象範囲を分割し，部分領域を隣接ノードに委譲していく．これにより，範囲内の全ノードに対し，最小のメッセージ数かつ短い平均経路長でクエリを配送することが可能となる．なお，本稿は，我々のこれまでの取り組み [12] を発展させたものであり，擬似コードを用いた提案手法の詳細な説明や，シミュレーション実験による評価を追加している．

以降では，まず 2. 章にて Skip Graph における範囲検索クエリのルーティングに関する既存手法を概説する．続いて 3. 章にて提案手法を述べ，4. 章ではその特性について解析的に考察を行う．5. 章でシミュレーション実験による評価について述べた後，6. 章でまとめと今後の課題について述べる．

2. 関 連 研 究

Skip Graph [6] は，範囲検索が可能な構造化オーバレイである．各ノードはキーとメンバーシップベクタと呼ばれるランダムな文字列を持つ．本稿では，メンバーシップベクタを構成する文字種の集合を有限アルファベット Σ として表す．$\Sigma = \{0, 1\}$ である場合の，Skip Graph が構成するトポロジの例を図 1 に示す．

Skip Graph のトポロジは，Skip List [13] を多重化した構造となっている．最下位層はキー順にソートされたノードの双方向リストであり，レベル i ではメンバーシップベクタの接頭 i 桁が一致するノード同士で双方向リストを形成する．

キーの完全一致検索を行う場合，検索開始ノードの最上位レベルからスタートし，Skip List と同様のルーティングを行う．即ち，検索対象のキーを通り過ぎない範囲でホップし，レベルをひとつ下げる，という動作を繰り返す．これにより，上位レベルが長距離をショートカットするリンクとして働き，検索対象のキーを持つノードまで短い経路長で到達することが可能となっている．例えば図 1 において，ノード A がノード E のキー 275 を検索する場合，ノード A からノード D（レベル 2），ノード D からノード E（レベル 0）という経路にて，経路長 2 でたどり着くことができる．

N ノードの Skip Graph において，完全一致検索クエリの経路長は $\mathcal{O}(\log N)$ となる．また，各ノードが保持する経路表のサイズも $\mathcal{O}(\log N)$ である．

2.1 Skip Graph における範囲検索クエリのルーティング

Skip Graph において範囲検索クエリのルーティングを行う場合，ターゲット範囲内の任意の 1 ノードに至るまでは前述の完全一致検索クエリと同様のルーティングを行い，その後，ターゲット範囲内においてマルチキャストを行うという手順を踏む．この，ターゲット範囲内におけるマルチキャストの方式については，複数の手法が提案されている．Beltran ら [10] は，以下 4 つの方式を示している．

シーケンシャル転送
各ノードは，レベル 0 の隣接ノードがターゲット範囲内である場合，クエリを転送する．

ブロードキャスト転送
各ノードは，ターゲット範囲内の全ての隣接ノードにクエリを転送する．同時に，受信済みクエリを記録し，同一クエリが届いた場合には破棄する．

ノード記録型ブロードキャスト転送
本方式は，ブロードキャスト転送を拡張したものである．各ノードは，クエリ転送時，当該クエリを受信済みのノードについて知り得る範囲でリスト化し，クエリに付加する．これにより，重複転送を部分的に避けることが可能となる．

ツリーベース転送
各ノードは，クエリを Skip Tree Graph [14] 固有のリンクを用いて転送する．

本稿では，汎用性の観点から，追加リンク等を持たない通常の Skip Graph を議論の対象としている．このため，独自の追加リンクを必要とするツリーベース転送については，本稿のスコープ外とする．

シーケンシャル転送では，ターゲット範囲内各ノードが同一クエリを 1 回しか受信しない動作となるため，メッセージ数は最小限に抑えられる．しかしながら，特にターゲット範囲内のノード数が多い場合には，経路長が非常に長くなるという問題がある．一方，ブロードキャスト転送（ノード記録型を含む）の場合，経路長の観点ではシーケンシャル転送よりも優位であるものの，重複転送によりメッセージ数が肥大化する点が問題となる．Beltran ら [10] による検証では，ノード記録型ブロードキャスト転送を用いた場合であっても，シーケンシャル転送と比べメッセージ数が増大する結果となっており，重複転送を完全には排除できないことが示されている．

上記 3 方式では経路長とメッセージ数の両方を抑えること

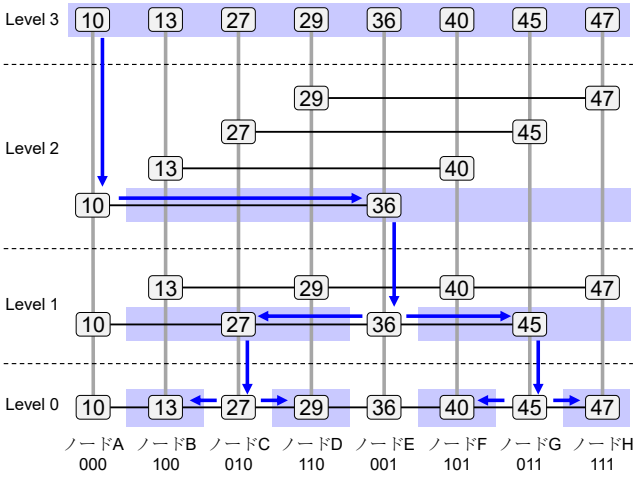

図 2　MRF によるルーティングの例
Fig. 2　Example of routing with MRF.

が困難であるのに対し，短い経路長と少ないメッセージ数を両立可能な手法として，MRF（Multi-Range Forwarding）が存在する [11]．MRF は，Skip Graph において同一ノードに複数キーを保持可能とする手法 Multi-key Skip Graph の要素技術であるが，通常の Skip Graph においても範囲検索クエリのルーティング手法として有用である．

ここでは，範囲検索クエリのターゲット範囲を R とする．MRF では，R 内のノードがクエリを受け取った時，R を当該ノードのキー位置にて 2 つの部分領域に分割する．その後，当該部分領域に含まれる隣接ノードの中で，最も高いレベルで隣接しているノードに対し，当該部分領域を委譲する．即ち，R の代わりに当該部分領域を付与して，クエリを転送する．クエリを受け取った各ノードが同様の動作を繰り返すことで，最終的にターゲット範囲内の全ノードが当該クエリを受信することができる．

図 2 に，MRF を用いたルーティングの例を示す．青い矩形は，ターゲット範囲もしくはその部分領域を表している．例として，ノード A が $R = [8, 50]$ なる範囲検索クエリを受け取った場合について考える．ノード A は，自身のキー 10 を用いて，ターゲット範囲 R を，$[8, 10)$ と $(10, 50]$ という 2 つの部分領域に分割する．$(10, 50]$ の中で，ノード A と最も高いレベルで隣接するノードは E であるから，ノード A はクエリに部分領域 $(10, 50]$ を付加してノード E へ転送する．同様にして，ノード E はノード C とノード G へ，部分領域 $(10, 36)$ と $(36, 50]$ をそれぞれ委譲する．さらに，ノード C はノード B とノード D へ，ノード G はノード F とノード H へとクエリを転送する．これにより，R 内の全ノードがクエリを受信することとなる．

MRF では重複転送が生じないため，最小限のメッセージ数にて範囲検索クエリを処理可能である．加えて，シーケンシャル転送と比べて短い経路長でクエリを配送することができる．これは，シーケンシャル転送では範囲内の各ノード（但し，最初のクエリ受信ノードを除く）がひとつの隣接ノードにのみクエリを転送するのに対し，MRF では最大 2 つの隣接ノードにクエリを転送するためである．

しかしながら，MRF を用いた場合，経路長が不必要に長くなってしまうケースが存在する．例えば図 2 において，ノード B はノード A とレベル 0 で隣接しているにも関わらず，ノード E とノード C を経由してクエリを受信するため，経路長は 3 となる．このような非効率性を解消するため，本稿では，新たな範囲検索クエリのルーティング手法を提案する．なお，本章で述べた既存手法については，4. 章にて提案手法を交えたより詳細な比較を行う．

3. 提 案 手 法

2. 章で述べた既存手法 MRF に対し，その平均経路長を削減可能な手法として，SFB（Split-Forward Broadcasting）を新たに提案する．

SFB では，MRF と同様，ターゲット範囲を部分領域に分割して隣接ノードに委譲していく．但し，MRF とは異なる方針で分割を行う．具体的には，範囲検索クエリを受け取った各ノードは，自身のキーではなく，隣接ノードのキーにて部分領域への分割を行う．

ターゲット範囲が R である範囲検索クエリについて，R 内で最初にそのクエリを受信したノードは，まず，R を当該ノードのキー位置にて 2 つの部分領域に分割する．その後，左右それぞれの部分領域について，隣接ノードのキーを用いた部分領域の委譲を行っていく．即ち，各部分領域について，その領域内に含まれる全ての隣接ノードのキーを用いて，より細かい部分領域へと分割し，それら隣接ノードへ委譲する．クエリを受け取った各ノードが同様の動作を繰り返すことで，最終的にターゲット範囲内の全ノードが当該クエリを受信することができる．

アルゴリズム 1 は，SFB の擬似コードである．**uponReceivingAtStart** は，ターゲット範囲内で最初にそのクエリを受信したノードにおいて一度のみ呼び出される．それ以降にクエリを受信するノードにおいては，**uponReceiving** が呼び出される．

2 行目から 5 行目にかけて，まず，クエリを受信したノードが自身のキーによってターゲット範囲を 2 つの部分領域に分割する．続いて，当該ノードは，部分領域それぞれについて，**uponReceiving** を呼び出す．(6 行目から 7 行目). **uponReceiving** では，9 行目から 22 行目において，引数で渡された部分領域に含まれる隣接ノードの中から，最も高いレベルで隣接しているノードを抽出する．該当する隣接ノードが存在した場合，部分領域はその隣接ノードのキーによってさらに 2 つの部分領域に分割され，当該隣接ノードに近い側の部分領域が委譲される．その後，残った部分領域について，再帰的に **uponReceiving** が呼び出され (33 行目)，徐々にレベルを下げながら同様の処理が繰り返されることとなる．なお，擬似コードにおいて，**setRightClosedBound** と **setLeftClosedBound** はそれぞれ，範囲の端点を引数で指定された値に変更して右閉もしくは左閉とする関数であり，変更後の範囲はその指定値を含むものとなる．同様に，**setRightOpenBound** と **setLeftOpenBound** は，範囲の端点を引数で指定された値に変更して右開もしくは左開とする関数であり，変更後の範囲はその指

Algorithm 1: SFB のルーティング処理

```
1  uponReceivingAtStart(range, query)
2      leftSubRange ← range.clone();
3      leftSubRange.setRightClosedBound( localNode.key );
4      rightSubRange ← range.clone();
5      rightSubRange.setLeftClosedBound( localNode.key );
6      uponReceiving( leftSubRange, query, FALSE);
7      uponReceiving( rightSubRange, query, TRUE);
8  uponReceiving(range, query, isRightSide)
9      delegationNode ← NULL;
10     neighbors ← NULL;
11     if isRightSide = TRUE then
12         neighbors ← routingTable.getRightNeighbors();
13     else
14         neighbors ← routingTable.getLeftNeighbors();
15     end
16     maxLevel ← −1;
17     foreach neighbor ∈ neighbors do
18         if neighbor.key is within range and
             neighbor.level > maxLevel then
19             maxLevel ← neighbor.level;
20             delegationNode ← neighbor;
21         end
22     end
23     if delegationNode ≠ NULL then
24         subRange ← range.clone();
25         if isRightSide = TRUE then
26             subRange.setLeftClosedBound(
                 delegationNode.key );
27             range.setRightOpenBound(
                 delegationNode.key );
28         else
29             subRange.setRightClosedBound(
                 delegationNode.key );
30             range.setLeftOpenBound( delegationNode.key
                 );
31         end
32         Send subRange, query and isRightSide to
             delegationNode;
33         uponReceiving( range, query, isRightSide);
34     end
```

定値を含まないものとなる．

図 3 は SFB を用いたルーティングの例を示している．図 2 と同様，青い矩形はターゲット範囲またはその部分領域を表す．例として，ノード A が $R = [8, 50]$ なる範囲検索クエリを受け取った場合について考える．ノード A は，自身のキー 10 を用いて，ターゲット範囲 R を $[8, 10]$ と $[10, 50]$ という 2 つの部分領域に分割する．$[10, 50]$ の中で，ノード A と最も高いレベルで隣接するノードは E であるから，ノード A は $[10, 50]$ をノード E のキー 36 で分割し，$[10, 36)$ と $[36, 50]$ を得る．このうち

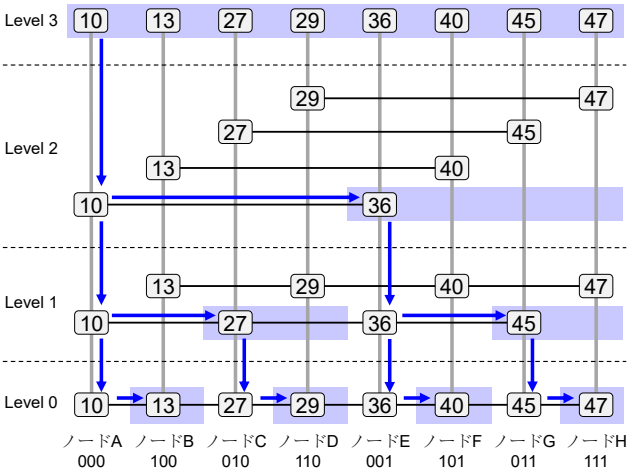

図 3　SFB によるルーティングの例
Fig. 3　Example of routing with SFB.

後者についてはクエリに付加してノード E へと転送し，前者についてはレベルを下げて同様の処理を繰り返す．即ち，レベル 1 ではノード C に部分領域 $[27, 36)$ を委譲し，レベル 0 ではノード B に部分領域 $[13, 27)$ を委譲する．ノード A からクエリを受け取った各ノードが同様の処理を行うことによって，最終的に，R 内の全ノードがクエリを受信することとなる．

4. 特性分析

本章では，提案手法 SFB について，2. 章で述べた既存手法との解析的な比較を行う．

表 1 は，ターゲット範囲内のノード数を N_R とした場合の各手法の比較であり，ターゲット範囲内における経路長とメッセージ数を表している．シーケンシャル転送では，範囲内ノードが一列に並んだリストの上をクエリが転送されていくため，経路長とメッセージ数のいずれも N_R に対する線形オーダーとなる．ブロードキャスト転送（ノード記録型を含む）では，Skip Graph のショートカットリンクを用いてクエリが転送されるため，経路長は対数オーダとなるが，メッセージ数については $\mathcal{O}(N_R \log N_R)$ となる．これは，範囲内にある N_R 個の各ノードが，当該範囲内に $\mathcal{O}(\log N_R)$ の隣接ノードを持つためである．MRF と SFB は，いずれも，範囲内の最上位レベルから徐々にレベルを下げつつクエリを転送していく．各レベルにおける転送の回数は $|\Sigma|$ の線形オーダーとなり，Σ はオーバレイ構築前に定められる定数パラメータであるから，$\mathcal{O}(1)$ とみなせる．クエリ転送中に移動するレベル数は $\mathcal{O}(\log N_R)$ であるため，経路長は $\mathcal{O}(\log N_R)$ となる．また，MRF 及び SFB においては，範囲内各ノードは同一クエリを一度のみ受信するため，メッセージ数については $\mathcal{O}(N_R)$ となる．以上より，MRF 及び SFB は，シーケンシャル転送やブロードキャスト転送と比べ，経路長とメッセージ数の双方について優位性を持っていると言える．

このように，MRF と SFB は近い性質を持っているが，2. 章で述べたとおり，MRF は実際の経路長が長大化するという傾向を有する．この点について，以降では，配送木の構造の観点

表 1　範囲検索クエリのルーティング手法の比較
Table 1　Comparison of routing methods for range queries.

	経路長	メッセージ数
シーケンシャル転送	$\mathcal{O}(N_R)$	$\mathcal{O}(N_R)$
ブロードキャスト転送	$\mathcal{O}(\log N_R)$	$\mathcal{O}(N_R \log N_R)$
ノード記録型ブロードキャスト転送	$\mathcal{O}(\log N_R)$	$\mathcal{O}(N_R \log N_R)$
MRF	$\mathcal{O}(\log N_R)$	$\mathcal{O}(N_R)$
SFB	$\mathcal{O}(\log N_R)$	$\mathcal{O}(N_R)$

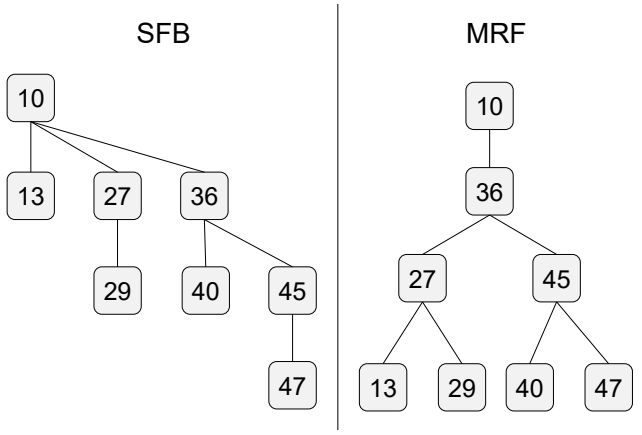

図 4　配送木の構造の違い
Fig. 4　Difference of delivery tree structures.

から考察する．

図 4 は，図 2 と図 3 の例における配送木の構造の違いを表したものである．SFB が偏った木を形成するのに対し，MRF では二分木に近い構造となっている．なお，これらの例では $|\Sigma| = 2$ となっており，簡単のため，理想的にバランスしたトポロジを用いている点に留意されたい．即ち，これら例のトポロジにおいては，レベル i のショートカットリンクは必ず $2^i - 1$ 個のノードをスキップするものとなっている．

この SFB の木構造は，二項木として知られている [15]．二項木は，図 5 に示すような再帰的な構造で定義される．根ノードが p 個の子ノードを持つような二項木を B_p と表すこととする．このとき，B_0 は根ノードのみから成る木である．$p > 0$ について，B_p の根ノードは，$B_{p-1}, B_{p-2}, \cdots, B_0$ の根ノードを子ノードとして持つ．図 4 に示した SFB の木は，B_3 と同一の構造となっている．

二項木は，各深さにおけるノード数が二項係数と一致する性質を持つことが知られている．即ち，二項木 B_p において，深さ d には $\binom{p}{d}$ 個のノードが存在する [15]．図 4 の例においても，SFB が形成する木の各深さにおけるノード数は 1, 3, 3, 1 となっており，二項係数と一致していることがわかる．一方，MRF の場合，二分木となっていることから，各深さにおけるノード数は 2 の累乗となる．

これらの性質を用いて，ターゲット範囲内のノード数が $2^{13} = 8,192$ である場合の，配送木の深さ毎のノード数を計算すると，図 6 のようになる．なお，ここでは検索開始ノード（根ノード）は範囲内の左端ノードであることを想定している．

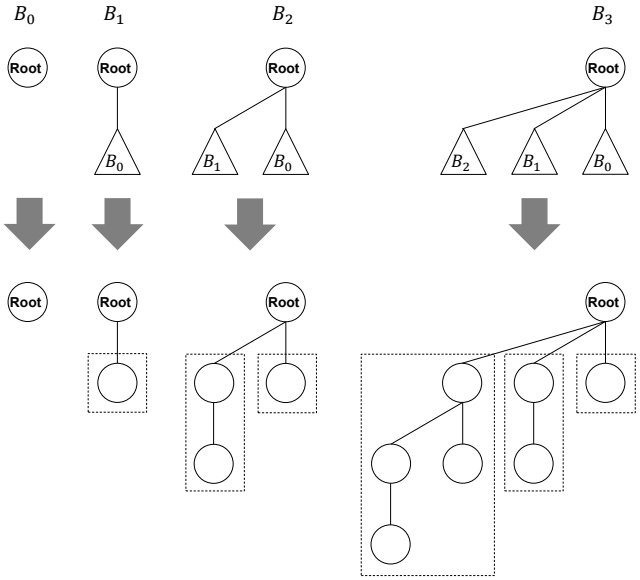

図 5　二項木の再帰的構造
Fig. 5　Recursive structure of binomial trees.

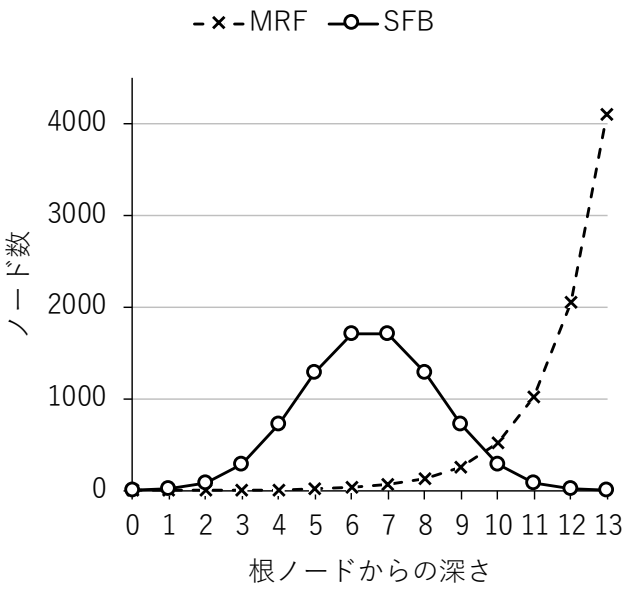

図 6　配送木の深さ毎のノード数の違い
Fig. 6　Difference of number of nodes at each depth.

MRF では深さ 13 のノード数が最大となっているのに対し，SFB の場合，深さ 6 及び 7 のノード数が最大となっていることがわかる．これは，SFB ではより多くのノードがより短い経路長でクエリを受信可能であることを意味している．

上記二項木の性質より，SFB と MRF におけるターゲット範囲内の平均経路長を導出することが可能である．以降の計算では，以下の前提条件を置く．

- $|\Sigma| = 2$
- 理想的にバランスした Skip Graph トポロジ
- 範囲内のノード数は 2 の累乗
- 検索開始ノードは範囲内の左端ノード

まず，SFB の平均経路長 L_{SFB} については，以下のように計

算することができる．

$$L_{SFB} = \frac{1}{N_R} \sum_{d=0}^{\log N_R} \left\{ \binom{\log N_R}{d} \cdot d \right\} = \frac{\log N_R}{2}$$

また，MRF の平均経路長 L_{MRF} は，以下の式で表すことができる．

$$L_{MRF} = \frac{1}{N_R} \sum_{d=0}^{\log N_R - 1} \left\{ (d+1) \cdot 2^d \right\} = \log N_R - 1 + \frac{1}{N_R}$$

L_{SFB} と L_{MRF} を比較すると，SFB は MRF の平均経路長を概ね半減できることがわかる．平均経路長の削減は，レイテンシ性能の向上に有効である．SFB では MRF と比べ，各ノードがクエリを転送するノード数（即ち，子ノード数）が多くなり得るが，一般に処理遅延と比べて通信遅延の方が大きいことに加え，SFB の子ノード数は最大でも $\mathcal{O}(\log N_R)$ であることから，子ノード数増加によるレイテンシ性能への影響は限定的と考えられる．また，平均経路長が短くなることによって，耐障害性が向上する効果も得られる．これは，クエリの転送がノード障害に遭遇する確率が減少することが期待できるためである．

5. 評　　価

シミュレーション実験により，提案手法 SFB の評価を行った．なお，シミュレーションプログラムは Java 言語にて作成した．

まず，ターゲット範囲内における平均経路長に関する実験を行った．10,000 ノードにより Skip Graph を構成し，SFB と MRF のそれぞれについて，ターゲット範囲内のノード数（N_R）を変えながら，平均経路長を測定した．測定の流れは以下のとおりである．

（1） ターゲット範囲内の左端ノードが，範囲検索クエリを発行．

（2） 各ノードは，SFB または MRF によってクエリを転送．同時に，検索開始ノードからターゲット範囲内各ノードまでの経路長を記録．

（3） クエリの配送完了後，経路長情報を集計し，平均値を算出．

（4） 以上の手順を 5 回繰り返し，5 回の試行の平均値を算出．

図 7 に結果を示す．なお，エラーバーは標準偏差を表している．SFB と MRF は N_R の変化に対し同様の傾向を示しているものの，SFB の方がより短い平均経路長となっていることがわかる．図 7 のグラフにおいて，各 N_R における平均経路長の削減率は 26.0%（$N_R = 10$），35.23%（$N_R = 100$），33.19%（$N_R = 1,000$），35.96%（$N_R = 10,000$）となっている．

SFB は概ね 30% 以上の平均経路長削減を達成しているものの，4. 章で述べた，MRF の平均経路長を概ね半減するという考察結果と比べると，少ない削減率となっている．これは，4. 章の議論では Skip Graph のトポロジが理想的にバランスしている想定を置いていたためである．実際にはメンバーシップベクタはランダムに生成されるため，完全にバランスしたトポロジにはならず，理想的トポロジにおける理論値を下回る削減率になっているものと思われる．Skip Graph の拡張として，ランダムなメンバーシップベクタに基づいて構築されたトポロジを理想的なトポロジに近づける手法が提案されており [16]，このような手法を用いた場合には，より 4. 章の考察結果に近い削減率になることが予想される．

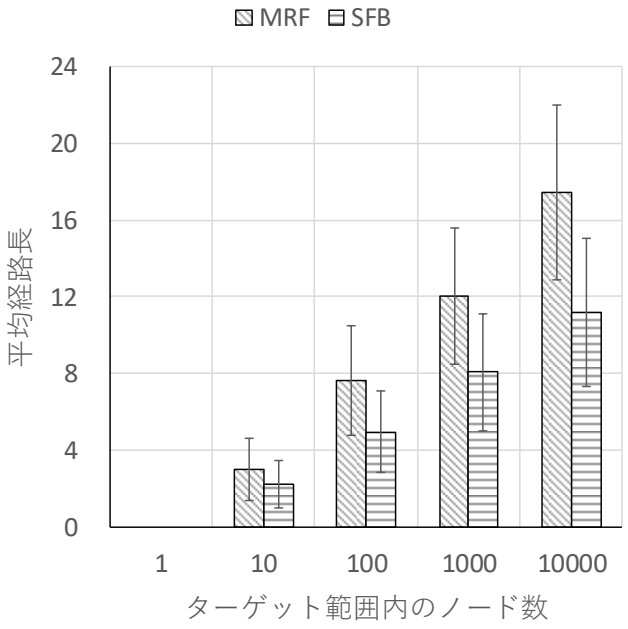

図 7　平均経路長
Fig. 7　Difference of average path length.

5.1 配送遅延の比較

4. 章で述べたように，平均経路長の削減はレイテンシ性能の向上に有効と考えられる．このことを確認するため，簡易的なモデル化に基づく平均配送遅延の比較を行った．各種遅延に関する想定は以下の通りである．

- 各ノードにおいて，範囲検索クエリの処理に要する固定的な処理遅延を 1 ミリ秒とする．
- ノード間の通信遅延を 10 ミリ秒とする．
- 各ノードにおいて，子ノード毎に要する処理遅延を t ミリ秒とする．

ここで，配送遅延とは範囲検索クエリが発行されてからあるノードに当該クエリが到達するまでの所要時間を指し，平均配送遅延はその平均値を指すものとする．あるノードに関する配送遅延は，経路長に応じた固定処理遅延及び通信遅延の累積と，経路中の各ノードにおいて子ノード毎に要する処理遅延の累積によって算出される．例えば図 3 の例において，ノード B は経路長 1 でクエリを受け取るため，固定処理遅延と通信遅延の累積は 11 ミリ秒となる．また，ノード A の子ノードの中では 3 番目に処理されることから，子ノード毎の処理遅延の累積は $3t$ ミリ秒である．従って，ノード B に関する配送遅延は $11 + 3t$

表 2　配送遅延の比較
Table 2　Comparison of latency.

子ノード毎に要する処理遅延（ミリ秒）	0.01	0.1	1	10	100
MRF の平均遅延（ミリ秒）	184.27	186.41	213.90	427.04	2626.25
SFB の平均遅延（ミリ秒）	113.98	125.13	134.94	295.67	1849.79

ミリ秒となる．

$N_R = 10,000$ において，t を 0.01 ミリ秒から 100 ミリ秒まで変化させた際の，SFB と MRF における平均配送遅延を表 2 に示す．t が増加するにしたがって，MRF の平均配送遅延に対する SFB の平均配送遅延の比は緩やかに増加する傾向が見られ，例えば $t = 0.01$ では約 0.62 倍となっているのに対し，$t = 100$ では約 0.70 倍である．しかしながら，いずれの t においても SFB の方が平均配送遅延が小さい結果となっている．実際的な t の値の範囲を想定すると，子ノードとの通信を非同期/同期のいずれの方式で処理した場合にも，表 2 の範囲を大きくは超えないケースが多いと思われ，SFB がレイテンシ性能の観点で優位性を持っていることがわかる．

5.2　アルファベットサイズの影響

Skip Graph におけるパラメータのひとつとして，メンバーシップベクタのアルファベットサイズ $|\Sigma|$ がある．Skip Graph では，レベル i においてメンバーシップベクタの接頭 i 桁が一致するノード同士で双方向リストを形成するため，$|\Sigma|$ を大きくすると，ノード同士の接頭 i 桁が一致する確率が下がることになる．即ち，各レベルのショートカットリンクがスキップするノード数は増大し，各ノードの最上位レベルはより低くなる．このようなトポロジの変化は，範囲検索クエリのルーティング効率に影響を与えるものと思われる．

そこで，アルファベットサイズの変化がもたらす影響を確認するための実験を行った．本実験では，10,000 ノードにより Skip Graph を構成し，$|\Sigma|$ の値を変えながら，ターゲット範囲内における平均経路長と，平均経路表サイズとを測定した．各ノードの経路表には，各レベルにおける左右の隣接ノードがエントリとして格納されているが，これらエントリは同一ノードである場合がある．具体的には，レベル i においてメンバーシップベクタの接頭 i 桁が一致する隣接ノードがあった時，レベル $i+1$ においても接頭 $i+1$ 桁が一致すると，当該隣接ノードとはレベル i とレベル $i+1$ の 2 つのレベルにおいて隣接することになる．そこで本実験では，経路表エントリの重複を含む値と，重複を含まない値の，2 種類について平均経路表サイズを測定した．N_R は 10,000 で固定とし，各測定を 5 回実施して平均値を算出した．

図 8 に結果を示す．平均経路表サイズは，$|\Sigma|$ の増加に伴い，指数的に減少している．また，$|\Sigma| = 2$ においては経路表エントリの半数以上は重複エントリであり，この重複率は $|\Sigma|$ が大きくなるにつれて減少していることが見て取れる．これは，前述のとおり，アルファベットサイズが大きくなるほど，あるレベルで隣接している 2 ノードがひとつ上のレベルでも隣接する確率が低くなるためである．

一方，平均経路長については，$|\Sigma|$ の増加に伴い概ね線形に増加しており，増加の傾向については SFB と MRF の間に大きな違いは見られない結果となっている．平均経路表サイズと比べると，平均経路長の増加速度は限定的である．例えば，$|\Sigma|$ を 2 から 10 へ変化させた場合，重複を含む平均経路表サイズが半分以下となるのに対し，SFB における平均経路長は依然として $|\Sigma| = 2$ における MRF の平均経路長よりも短いものとなっている．

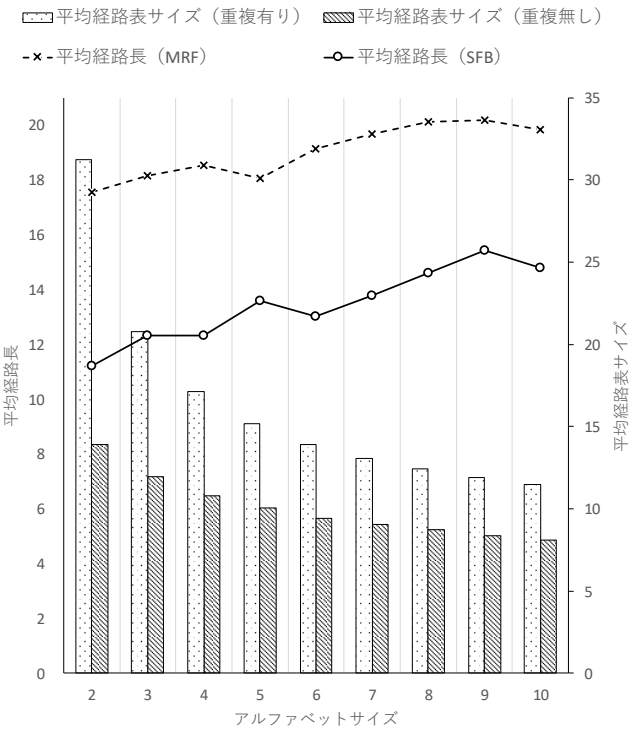

図 8　アルファベットサイズの影響
Fig. 8　Influence of alphabet size.

6. おわりに

本稿では，Skip Graph における新たな範囲検索クエリのルーティング手法として SFB を提案した．SFB は，最小限のメッセージ数，かつ，既存手法よりも短い経路長で，ターゲット範囲内の全ノードにクエリを配送することが可能である．シミュレーション実験により，SFB が既存手法 MRF と比べ，平均経路長を概ね 30% 以上削減できることを示した．また，アルファベットサイズが平均経路長及び平均経路表サイズにもたらす影響を明らかにした．

今後の課題として，Skip Graph 以外の範囲検索可能な構造化オーバレイアルゴリズムへの適用可能性の検討や，SFB と他のルーティング手法の適応的組み合わせの検討等を行う予定である．

謝辞 本研究は（公財）セコム科学技術振興財団 一般研究助成の支援を受けたものです．本研究は JSPS 科研費 16K12406 の助成を受けたものです．本研究の一部は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務として行われました．

文　献

[1] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: Amazon’s Highly Available Key-value Store,” Proceedings of the ACM SIGOPS Symposium on Operating Systems Principles, pp.205–220, 2007.

[2] N. Ramzan, H. Park, and E. Izquierdo, “Video streaming over P2P networks: Challenges and opportunities,” Signal Processing: Image Communication, vol.27, no.5, pp.401–411, 2012.

[3] Y. Teranishi, R. Banno, and T. Akiyama, “Scalable and Locality-Aware Distributed Topic-based Pub/Sub Messaging for IoT,” Proceedings of the IEEE Global Communications Conference, pp.1–7, 2015.

[4] J. Chen, “Flowchain: A Distributed Ledger Designed for Peer-to-Peer IoT Networks and Real-time Data Transactions,” Proceedings of the International Workshop on Linked Data and Distributed Ledgers, 2017.

[5] D. Wu, D.I. Arkhipov, E. Asmare, Z. Qin, and J.A. McCann, “UbiFlow: Mobility management in urban-scale software defined IoT,” Proceedings of the IEEE Conference on Computer Communications, pp.208–216, April 2015.

[6] J. Aspnes and G. Shah, “Skip Graphs,” ACM Transactions on Algorithms, vol.3, no.4, pp.37:1–37:25, 2007.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A Scalable Content-addressable Network,” Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications, pp.161–172, 2001.

[8] A. Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems,” Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, pp.329–350, 2001.

[9] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, “Chord : A Scalable Peer-to-peer Lookup Service for Internet Applications,” ACM SIGCOMM Computer Communication Review, vol.31, no.4, pp.149–160, 2001.

[10] A.G. Beltran, P. Milligan, and P. Sage, “Range Queries Over Skip Tree Graphs,” Computer Communications, vol.31, no.2, pp.358–374, 2008.

[11] 小西佑治，吉田 幹，竹内 亨，寺西裕一，春本 要，下條真司，“単一ノードに複数キーを保持可能とする Skip Graph 拡張，” 情報処理学会論文誌，vol.49，no.9，pp.3223–3233，2008.

[12] R. Banno, T. Fujino, S. Takeuchi, and M. Takemoto, “SFB: A Scalable Method for Handling Range Queries on Skip Graphs,” IEICE Communications Express, vol.4, no.1, pp.14–19, 2015.

[13] W. Pugh, “Skip Lists : A Probabilistic Alternative to Balanced Trees,” Communications of the ACM, vol.33, no.6, pp.668–676, 1990.

[14] A.G. Beltran, P. Sage, and P. Milligan, “Skip Tree Graph: a Distributed and Balanced Search Tree for Peer-to-Peer Networks,” Proceedings of the International Conference on Communications, pp.1881–1886, 2007.

[15] J. Vuillemin, “A Data Structure for Manipulating Priority Queues,” Communications of the ACM, vol.21, pp.309–315, April 1978.

[16] T. Kawaguchi, R. Banno, M. Hojo, M. Ohnishi, and K. Shudo, “Self-refining skip graph: Skip graph approaching to an ideal topology,” IEEE Annual Consumer Communications Networking Conference, pp.441–448, Jan. 2017.