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
IA2020-33(2021-01)

Skip Graphにおける 迂回経路を活用した範囲検索クエリルーティング手法

三木　友弥† 　金子　孟司† 　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
†† 工学院大学　〒192–0015 東京都八王子市中野町 2665–1
E-mail: {miki.y.ad,kaneko.t.ay}@m.titech.ac.jp, banno@computer.org, shudo@is.titech.ac.jp

あらまし　ノード群による自律分散的なネットワークを構築し，効率的なルーティングを実現するための技術として構造化オーバーレイが知られている．Skip Graph は，構造化オーバーレイのデータ構造の 1 つであり，範囲検索が可能であるという特長を有する．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を提案する．提案手法では，クエリを受信したノードが，下位ノード列の中央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，ノード数が十分に多い場合，平均経路長が既存手法よりも 20% 程度短縮できることを確認した．

キーワード　Skip Graph, 構造化オーバレイ, 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

A Routing Method for Range Queries Utilizing Detour Routes on Skip Graph

Yuya MIKI†, Takeshi KANEKO†, Ryohei BANNO†,††, and Kazuyuki SHUDO†

† Tokyo Institute of Technology　Ookayama 2–12–1, Meguro–ku, Tokyo, 152–8552 Japan
†† Kogakuin University　Nakanomachi 2665–1, Hachioji–shi, Tokyo, 192–0015 Japan
E-mail: {miki.y.ad,kaneko.t.ay}@m.titech.ac.jp, banno@computer.org, shudo@is.titech.ac.jp

Abstract　A structured overlay is a technique to construct an autonomous decentralized network of nodes and to achieve efficient routing. Skip Graph is a structured overlay and one of its advantages is capability for range queries. In this paper, we propose a new routing method for range queries in Skip Graph. In the proposed method, a node which receives a query, divides its target range into subranges by the key of the center of the subordinate node sequence, and delegates the subranges to its neighbor nodes. We confirmed that the average path length of the proposed method is about 20% shorter than that of the existing methods when the number of nodes is sufficiently large.

Key words　Skip Graph, Structured overlay, Peer-to-peer network, Routing algorithm, Range query

1. はじめに

オーバーレイネットワークは，既存のネットワーク上に構築される論理ネットワークである．下位ネットワークに依存しないトポロジを構築することで、メッセージ配信やノード・データ検索などの機能を提供する．特に構造化オーバーレイは，特定のデータ構造やプロトコルに従って，ノード群による自律分散的なネットワークを構築することで，高いスケーラビリティや効率的なルーティング等を実現する．その性能から，近年注目を集めているブロックチェーンの分野においても適用が検討されている [1].

構造化オーバーレイにおいて，分散ハッシュテーブル（Distributed Hash Table, DHT）ベースのもの [2], [3] は，ハッシュ化によってキーを一様な ID 空間に写像し，そのキーと値のペアでデータを分散管理することで，負荷分散を実現する．しかし，ハッシュ化によりキーの順序関係が保存されないため，キーの範囲を指定して検索するような複雑なクエリへの対応を困難にする．

一方で，範囲検索が可能な構造化オーバーレイの一つに Skip Graph [4] がある．Skip Graph では，キーをハッシュ化せずにデータを管理するため，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．Skip Graph における範囲検索クエリにおけるルーティングについては，複数の手法が提案されている [5]〜[7] が，それらの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．

そこで本稿では，Skip Graph における新たな範囲検索クエリ

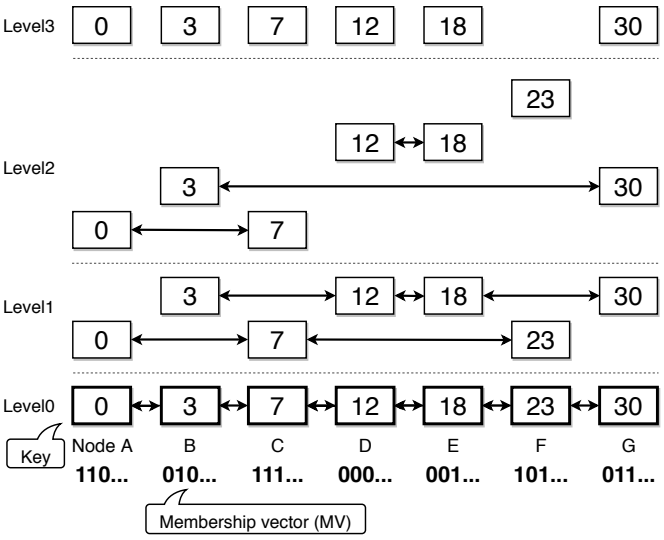


図 1: Skip Graph のトポロジ

Fig. 1 An example of Skip Graph.

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，下位ノード列の中央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，最小限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を可能にする．

本稿の構成は以下の通りである．**2.** で 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の概要について述べる．次に，**3.** で提案手法の詳細を述べ，**4.** で提案手法の評価実験の内容とその結果を述べる．最後に，**5.** で本研究のまとめと今後の課題について述べる．

2. 関 連 研 究

Skip Graph [4] は Skip List [8] を元に設計され，そのトポロジは各ノードが複数のソートされた双方向連結リストに属した構造，即ち Skip List を多重化した構造になっている．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ち，それらの値に基づいてトポロジを決定する．本稿では，メンバーシップベクタを構成する文字の集合を有限アルファベット Σ として表す．$\Sigma = \{0, 1\}$ である場合に，Skip Graph が形成するトポロジの例を図 1 に示す．

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連結リストを形成する．特にレベル 0 では全ノードが一つの双方向連結リストに属する．

2.1 完全一致検索クエリのルーティング

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，検索対象のキーを超えない範囲でホップし，レベルを一つ下げる，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になる．例えば図 1 において，ノード B がノード F のキーを検索する場合，ノード B からノード D（レベル 2），ノード D からノード E（レベル 1），ノード E からノード F（レベル 0）という経路にて，経路長 3 で目的ノードに到達できる．

Skip Graph のノード数を n とすると，完全一致検索クエリの経路長は $O(\log n)$ となる．また，各ノードが保持する経路表のサイズも $O(\log n)$ である．

キーの完全一致検索において，迂回経路を活用することで上述の手法よりも経路長を短縮する手法として，DSG（Detouring Skip Graph）が存在する [9]．DSG では，下位ノード列のキー空間における中央を推定し，その値を基準に迂回するかどうかの判定を行う．中央の推定において，キーの分布をあらかじめ得ることは難しいが，評価実験によって多くの場合で，2 つのキーの平均を用いることで十分に経路長を短縮可能であることがわかっている．例えば図 1 において，ノード B からノード F のキーを検索する場合，ノード B からノード G（レベル 2），ノード G からノード F（レベル 0）という経路にて，経路長 2 で目的ノードに到達できる．このように迂回経路を活用することで，上述の手法よりも経路長が短縮されている．

2.2 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．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おり，Beltran ら [5] は以下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

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．

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ため，独自の追加リンクを必要とする方式 [5] については，本稿のスコープ外とする．

上記の方式では，経路長とメッセージ数の両方を抑えること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 [6].

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．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例を示す．

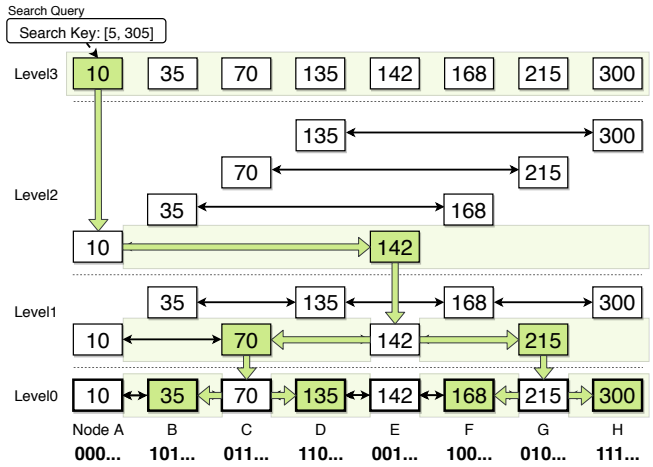


図 2: MRF によるルーティング例

Fig. 2 Example of routing with MRF.

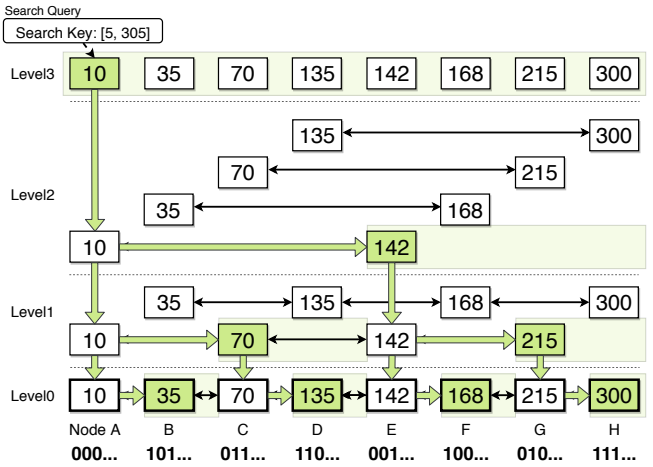


図 3: SFB によるルーティング例

Fig. 3 Example of routing with SFB.

半透明な緑の矩形はターゲット範囲あるいは部分領域を表しており，範囲内の各ノードはクエリを受け取ったレベルにて緑で塗られている．

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．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．その非効率性を解消する手法として，SFB（Split-Forward Broadcasting）が存在する [7].

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部分領域へ分割し，それらを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．図 3 は，SFB を用いたルーティングの例を示している．表記法は図 2 と同様である．

しかしながら，SFB のルーティングでは構築済みのリンクを十分に活用できていないため，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 1 において，ノード B が $R = [1, 31]$ の範囲検索クエリを受け取った場合，ノード F はノード B から経路長 1 かつ自身の隣接ノードであるノード G を経由せず，ノード C とノード E を経由して当該クエリを受信してしまうため，経路長が 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.** にて提案手法を交えてより詳細に比較を行う．

3. 提 案 手 法

2. 章で述べた既存手法 SFB に対し，その平均経路長を削減可能な手法を新たに提案する．提案手法では，SFB と同様に，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，SFB とは異なる分割ポリシーを用いる．具体的には，範囲検索クエリを受け取った各ノードは，下位ノード列の中央，即ち，自身と最も高いレベルで隣接しているノードと次に高いレベルで隣接しているノードの中央を推定し，その値を基準に部分領域への分割を行う．中央の推定方法については，2.1 節で述べた通り，キーの分布をあらかじめ得ることは難しいため，ここでは 2 つのキーの平均を用いるものとする．

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，下位ノード列の中央を基準に部分領域を分割し，それらを委譲していく．これら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．但しレベル 0 においては，下位ノード列が存在しないため，SFB と同様に隣接ノードのキーで分割し，それを委譲する．

アルゴリズム 1 は，提案手法の擬似コードである．uponReceivingAtStart は，ターゲット範囲内または委譲された範囲内で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，そのノードにおいては uponReceiving が呼び出される．

2 行目から 5 行目にかけて，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．6 行目から 7 行目にかけて，当該ノードはその部分領域それぞれについて，uponReceiving を呼び出す．uponReceiving では，9 行目から 15 行目にかけて，部分領域内の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 *delegationNode* を抽出する．16 行目から 30 行目にかけて，該当する隣接ノードが存在した場合，その次に高いレベルで隣接してい

Algorithm 1 Routing process of proposed method

```
1: procedure uponReceivingAtStart(localNode, range, query)
2:     leftSubRange ← clone of range
3:     leftSubRange.setRightClosedBound(localNode.key)
4:     rightSubRange ← clone of range
5:     rightSubRange.setLeftClosedBound(localNode.key)
6:     uponReceiving(leftSubRange, query, FALSE)
7:     uponReceiving(rightSubRange, query, TRUE)
8: end procedure
9: procedure uponReceiving(range, query, isRightSide)
10:    if isRightSide = TRUE then
11:        neighbors ← routingTable.getRightNeighbors()
12:    else
13:        neighbors ← routingTable.getLeftNeighbors()
14:    end if
15:    delegationNode ← neighbors.getDelegationNode(range)
16:    if delegationNode ≠ NULL then
17:        subRange ← clone of range
18:        lowerNode ← neighbors.getLowerNode()
19:        if lowerNode ≠ NULL then
20:            divideKey ← mid(delegationNode, lowerNode)
21:        else
22:            divideKey ← delegationNode.key
23:        end if
24:        if isRightSide = TRUE then
25:            subRange.setLeftClosedBound(divideKey)
26:            range.setRightOpenBound(divideKey)
27:        else
28:            subRange.setRightClosedBound(divideKey)
29:            range.setLeftOpenBound(divideKey)
30:        end if
31:        uponReceivingAtStart(delegationNode, subRange, query)
32:        uponReceiving(range, query, isRightSide)
33:    end if
34: end procedure
```

るノード *lowerNode* を抽出する．*lowerNode* が存在した場合，*delegationNode* と *lowerNode* の中央を推定し，その値を基準に部分領域への分割を行う．存在しない場合には，SFB と同様に *delegationNode* を基準に部分領域への分割を行う．その後，*delegationNode* には自身が含まれる部分領域が委譲され，uponReceivingAtStart が呼び出される．残った部分領域については，再帰的に uponReceiving が呼び出され，徐々にレベルを下げながら同様の処理が繰り返される．なお，擬似コードにおいて，setRightClosedBound と setLeftClosedBound は，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る．また，mid は引数で指定された 2 つのノードの中央を推定し，そのキーの値を返す関数である．

図 4 は提案手法を用いたルーティングの例を示している．表記法は図 2 と同様である．例として，ノード A が $R = [5, 305]$ 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，自身のキーの値 10 を基準にターゲット範囲 R を $[5, 10]$ と $[10, 305]$ という 2 つの部分領域に分割する．部分領域 $[10, 305]$ 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 であり，次に高いレベルで隣接するノードは C であるから，ノード A は $[10, 305]$ をノード E とノード C のキーの平均値 106 で $[10, 106)$ と $[106, 305]$ に分割する．このうち後者についてはクエリに付加してノード E へ転送し，前者についてはレベルを下げてレベル 1 まで同様の処理を繰り返す．即ち，レベル 1 でノード C に部分領域 $[52.5, 106)$ を委譲する．レベル 0 では，SFB と同様にノード B のキーの値 35 で分割し，ノード B に部分領域 $[35, 52.5)$ を委譲する．

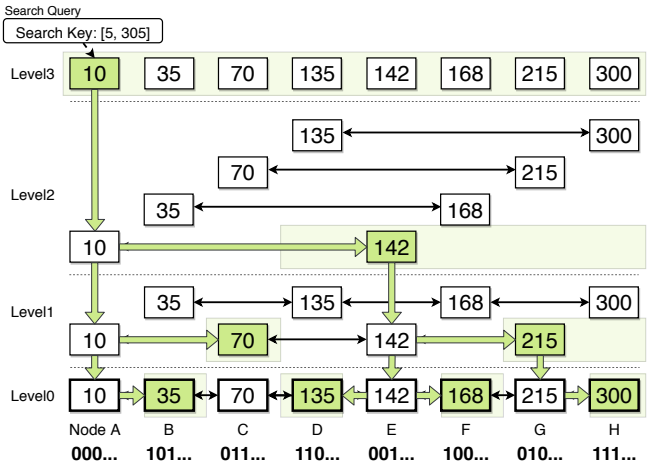


図 4: 提案手法によるルーティング例

Fig. 4 Example of routing with the proposed method.

4. 評　　価

シミュレーションによってターゲット範囲内における経路長に関する評価実験を行い，提案手法が経路長に与える影響を観察した．キーの生成方法として次の 2 通りを試した．

- 一様分布で生成した乱数
- べき分布で生成した乱数

各実験の詳細は以降の各節で述べる．なお，メンバーシップベクタのアルファベット集合は $\{0, 1\}$ で固定している．上述の方法で生成したキーを元に，10000 ノードにより Skip Graph を構築し，MRF および SFB，提案手法のそれぞれについて，ターゲット範囲内のノード数 N_R を変更して，平均経路長を測定した．測定の流れは以下の通りである．

（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する．

（2） 各ノードは，指定したルーティング方法によってクエリを転送する．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する．

（3） クエリ配送完了後，経路長情報を集計し平均値を算出する．

（4） 以上の手順を同じトポロジに対して 100 回繰り返し，100 回の試行の平均値を算出する．それを 5 つの異なるトポロジに対して行い，平均値を算出する．

4.1 一様分布で生成した乱数

参加ノードのキーを確率変数 v.key として，

$$P\{v.\text{key} = k\} = \frac{1}{2^{30}} (k \in \{0, 1, ..., 2^{30} - 1\})$$

の一様分布に従うように，疑似乱数を使用してキーを生成した．このキーの分布における中央の推定値 mid は

$$\text{mid}(k_1, k_2) := \frac{k_1 + k_2}{2}$$

である．

図 5(a) には，上述の手順で測定したときの平均経路長を示す．図 5(b) には，ターゲット範囲内のノード数 $N_R = 10000$ で固定したとき，ある経路長で当該クエリが到達するノードがいくつ存在するかの分布を示す．

結果として 3 つのルーティング手法は，N_R の変化に対して同様の傾向を示しているものの，提案手法が他の 2 つの手法より短い平均経路長となっていることがわかる．図 5(a) 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，SFB と比較して，3.15%（$N_R = 10$），13.05%（$N_R = 100$），17.46%（$N_R = 1000$），20.48%（$N_R = 10000$）となっている．

4.2 べき分布で生成した乱数

参加ノードのキーを確率変数 v.key として，確率密度関数

$$f(k) = ck^{10} (0 \le k \le 2^{30})$$

に対して，

$$P\{v.\text{key} \le k\} = \int_0^k f(k)dk (0 \le k \le 2^{30})$$

のべき分布に従うように，疑似乱数の値を変換してキーを生成した．ここで c は

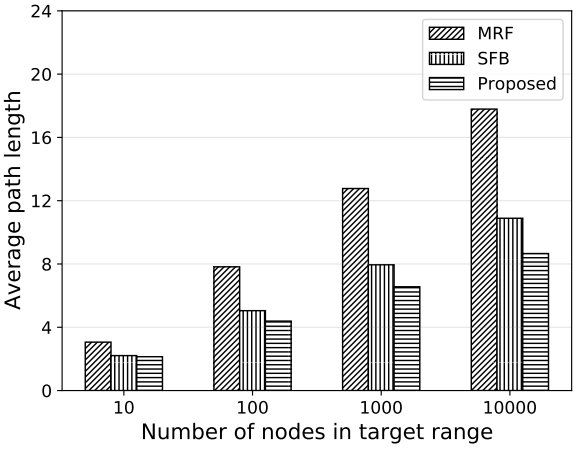
$$\int_0^k f(k)dk = 1$$

を満たす定数である．2.1 節で述べた通り，多くの場合で 2 つのキーの平均が中央の推定値として有効であるため，このキーの分布における中央の推定についても 4.1 節で用いた mid を使用する．べき分布を使用した目的は，偏ったキー分布に対する提案手法の効果を評価するためである．そのため，ここでは 4.1 節で用いたトポロジにおいて，キーの値のみを変更したトポロジを用いて，異なる 2 つのキー分布に対する平均経路長の変化を観察する．

表 1 には，上述の手順で測定した結果と 4.1 で測定した結果を示す．異なる 2 つのキー分布に対して，3 つのルーティング手法それぞれで同等の経路長が観察された．提案手法に関しても，2 つのルーティング手法に対して 4.1 節と同等の削減率となり，キー分布が偏った場合でも提案手法が有効であることがわかった．

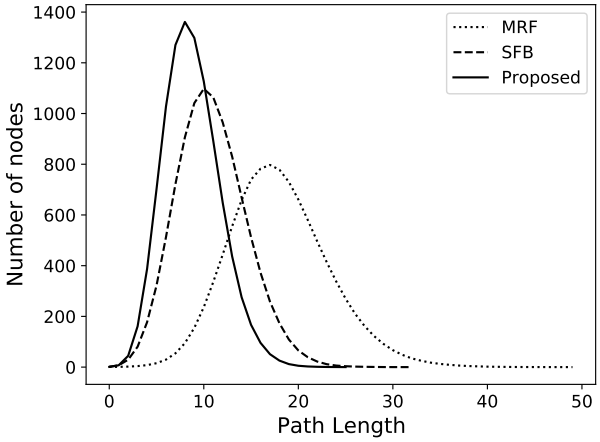
4.3 Detouring Skip Graph との比較

2.1 節で述べた単一キー検索手法である Detouring Skip Graph と提案手法の平均経路長を比較する．キーの生成には，一様分布で生成した乱数を用いるものとし，中央の推定には 4.1 節と同様の mid を用いるものとする．また Detouring Skip Graph を用いた検索では，範囲内の各ノードに対して単一キー検索を行い，記録した経路長の平均値を算出するものとする．但し，事前に範囲内のノードの各キーを取得することは困難で



(a) 平均経路長

Average path lengths.



(b) 経路長の分布，ターゲット範囲内ノード数 $N_R = 10000$

Distribution of path lengths where $N_R = 10000$.

図 5: 一様分布で生成した乱数をキーとしたトポロジ上での経路長の比較

Fig. 5 Comparison of average path lengths on a topology where keys were generated by uniform distribution.

表 1: 異なる分布で生成した乱数をキーとしたトポロジ上での経路長の比較

Table 1 Comparison of average path lengths on a topology where keys were generated by two different distributions.

		$N_R = 10$	$N_R = 100$	$N_R = 1000$	$N_R = 10000$
MRF	key: uniform	3.06	7.82	12.77	17.79
	key: power	3.06	7.82	12.77	17.79
SFB	key: uniform	2.22	5.06	7.95	10.90
	key: power	2.22	5.06	7.95	10.90
Proposed	key: uniform	2.14	4.40	6.56	8.67
	key: power	2.14	4.41	6.60	8.75

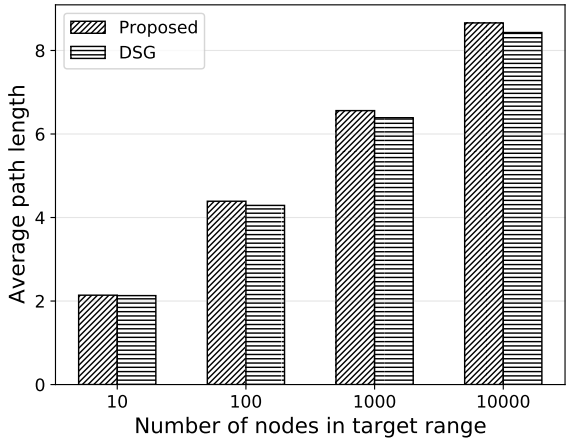


図 6: Detouring Skip Graph との経路長の比較

Fig. 6 Comparison of path lengths with Detouring Skip Graph

あるため，範囲検索を行う際に各ノードに対して単一キー検索を行うのは現実的ではない．Detouring Skip Graph と比較する目的は，単一キー検索で迂回経路を活用して当該クエリを配送する場合と比較することで，提案手法が十分に迂回経路を活用できているかを観察するためである．

図 6 には，上述の手順で測定した結果を示す．結果として 2 つのルーティング手法は，N_R の変化に対して同様の傾向を示しているものの，Detouring Skip Graph が提案手法よりわずかに短い平均経路長となっていることがわかる．その理由としては，提案手法において各ノードは受け取った部分領域の範囲内でしか迂回経路を活用できないことが原因だと考えられる．それに対して，範囲内の各ノードに対して Detouring Skip Graph を用いて単一キー検索した場合では，迂回するかを判定する際に部分領域の範囲内という制約がないため，提案手法よりもわずかに短い平均経路長になっていると考えられる．

5. おわりに

本稿では，Skip Graph における範囲検索クエリの新たなルーティング手法を提案した．提案手法は，Skip Graph のトポロジを効率的に使用することで，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．評価実験により，提案手法が既存手法 SFB と比べ，平均経路長を 20% 程度削減できることを示した．今後の課題として，経路長短縮に対する解析的な評価や中央の推定方法の検討等を行う予定である．

文　献

[1] Elias Rohrer and Florian Tschorsch. Kadcast: A structured approach to broadcast in blockchain networks. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies (AFT'19)*, pp. 199–213, 2019.

[2] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on networking*, Vol. 11, No. 1, pp. 17–32, 2003.

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

[4] James Aspnes and Gauri Shah. Skip graphs. *ACM Transactions on Algorithms (TALG)*, Vol. 3, No. 4, pp. 37–es, 2007.

[5] A González-Beltrán, Peter Milligan, and Paul Sage. Range queries over skip tree graphs. *Computer Communications*, Vol. 31, No. 2, pp. 358–374, 2008.

[6] Yoshimasa Ishi, Yuuichi Teranishi, Mikio Yoshida, Susumu Takeuchi, Shinji Shimojo, and Shojiro Nishio. Range-key extension of the skip graph. In *2010 IEEE Global Communications Conference (GLOBECOM 2010)*, pp. 1–6. IEEE, 2010.

[7] Ryohei Banno and Kazuyuki Shudo. An efficient routing method for range queries in skip graph. *IEICE TRANSACTIONS on Information and Systems*, Vol. E103-D, No. 03, pp. 516–525, 2020.

[8] William Pugh. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, Vol. 33, No. 6, pp. 668–676, 1990.

[9] Takeshi Kaneko, Ryohei Banno, Kazuyuki Shudo, Kota Abe, and Yuuichi Teranishi. Detouring skip graph: Efficient routing via detour routes on skip graph topology. *IEEE Open Journal of the Communications Society*, Vol. 1, pp. 1658–1673, 2020.