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
SITE2015-63,IA2015-95(2016-03)

FRT-Skip Graph: 柔軟な経路表に基づくSkip Graph型構造化オーバレイ

北條　真史†　坂野　遼平††,†　首藤　一幸†

† 東京工業大学
〒 152-8552 東京都目黒区大岡山 2-12-1
†† 日本電信電話株式会社 NTT 未来ねっと研究所
〒 180-0012 東京都武蔵野市緑町 3-9-11
E-mail: hojo.m.aa@m.titech.ac.jp, banno.ryohei@lab.ntt.co.jp, 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構築や，他ノードやデータの検索を実現する構造化オーバレイ技術のひとつに，Skip Graph がある．Skip Graph は，データ構造のひとつである Skip List 構造に基づくネットワーク構築を行い，キーと呼ばれる属性に対する範囲検索を可能とする構造化オーバレイである．Skip Graph は乱数に基づく経路表構築を行うが，乱数の偏りによって経路表エントリを有効に活用できず，経路長が理想より増加する問題点がある．そこで我々は，Skip Graph の問題点を解決し，さらに Skip Graph にはない特長を備える新たな構造化オーバレイ FRT-Skip Graph を提案する．提案手法は，柔軟な経路表に基づき設計された構造化オーバレイであり，Skip Graph と同様に範囲検索を可能とする．また，柔軟な経路表に基づく設計に由来する，経路表サイズの動的調整可能性や高い拡張性といった特長を備える．

キーワード　Peer-to-Peer, 構造化オーバレイ, Skip Graph

FRT-Skip Graph: A Skip Graph-Style Strucutred Overlay based on Flexible Routing Tables

Masashi HOJO†, Ryohei BANNO††,†, and Kazuyuki SHUDO†

† Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552 Japan
†† NTT Network Innovation Laboratories, NTT Corporation
Midori-cho 3-9-11, Musashino-shi, Tokyo, 180-0012 Japan
E-mail: hojo.m.aa@m.titech.ac.jp, banno.ryohei@lab.ntt.co.jp, shudo@is.titech.ac.jp

Abstract　Structured overlays can perform to construct a network autonomously by a number of nodes and to search other nodes and data. Skip Graph, one of the structured overlays, constructs an overlay network based on Skip List structure and supports range queries for keys. Skip Graph manages routing tables based on random digits; therefore, the deviation of them disturbs effective utilization of the routing table entries and increases path length than the ideal value. We therefore propose FRT-Skip Graph, a novel structured overlay that solves the issues of Skip Graph and provides desirable features not in Skip Graph. FRT-Skip Graph is designed based on Flexible Routing Tables and supports range queries similarly to Skip Graph. Furthermore, it provides features derived from FRT, namely, dynamic routing table size and high extensibility.

Key words　Peer-to-Peer, Structured Overlay, Skip Graph

1. はじめに

構造化オーバレイとは，実ネットワーク上の多数のノードにより構築される，アプリケーション層の仮想ネットワークである．構造化オーバレイでは，ノードやデータに ID を割当て，数学的・論理的な構造によりネットワークトポロジが決定される．ノード間の通信やデータの検索は，指定した宛先 ID に向けて，各ノードがメッセージの転送を繰返すことで実現される．メッセージのルーティングを行う際，各ノードは隣接ノードの ID とアドレスを載せた表，すなわち経路表を参照し，宛先 ID

にメッセージを届けるために適切な転送先ノードを決定する．構造化オーバレイは，多数のノードにメッセージを拡散させることを回避し，ノード数の増加に対してメッセージの転送回数を線型未満のオーダに抑えるため，スケーラビリティに優れた技術である．

構造化オーバレイの活用例として，ノード群によるデータの格納と取得を実現する分散ハッシュ表（DHT）がある．DHT では，データを (key, value) の形式で格納する操作と，キーを指定しデータを取得する操作が提供される．データには，キーのハッシュ値等で決定したデータ ID が割当てられ，そのデータ ID を担当するノードに格納される．各ノードのデータ ID の担当範囲は，自身の ID と隣接ノードの ID から決定される．データの格納や取得を行う際は，この操作を行うノードが，構造化オーバレイ内でデータ ID に向けてメッセージを送信し，データ ID の担当ノードがその要求に応える．データの取得の際，キーは完全一致の形式で指定することしかできないため，キーの範囲で取得を行うと行った柔軟な検索は不可能である．

柔軟な検索が不可能であるという欠点を持つ DHT に対して，キー順序を保ったネットワーク構築を行う構造化オーバレイである Skip Graph [3] が提案されている．Skip Graph は，連結リストを拡張したデータ構造である Skip List [9] を着想とするネットワークトポロジを持ち，多くの DHT と同様に $O(\log N)$ の経路長でデータの検索を実現する．さらに，Skip Graph はキーの順序を保ったネットワークを構築するため，キーの範囲検索を可能とする．乱数に基づいてショートカットリンクを形成することによって，キーの順序を保ちつつ小さい経路長を実現するが，各ノードが持つ乱数の偏りによってこれが増大する可能性があることも，Skip Graph の持つ性質の 1 つである．

そこで我々は，Skip Graph と同様に乱数を用いることで範囲検索を実現しつつ，乱数の偏りによって経路長が増大することを抑える新たな構造化オーバレイ FRT-Skip Graph を提案する．FRT-Skip Graph は，Skip Graph 同様の Skip List をネットワークトポロジとしつつ，乱数の偏りにより経路表エントリに無駄が生じるという Skip Graph の問題を解決する．加えて，FRT-Skip Graph は柔軟な経路表（Flexible Routing Tables, FRT）[7] に基づいて設計されているため，経路表サイズの動的調整可能性や高い拡張性といった Skip Graph にない特長を備える．

本稿では提案アルゴリズム FRT-Skip Graph の詳細を述べ，これが経路長を効果的に削減することを示す．また，FRT に基づく設計によりもたらされる拡張性に関する考察や，FRT-Skip Graph の更なる改良についての検討も行う．

2. 関 連 研 究

本章では，まず提案手法のベースとなった構造化オーバレイである Skip Graph を解説する．次に，我々が提案手法を設計するにあたって利用した，構造化オーバレイの設計手法である FRT を解説する．

2.1 Skip Graph

Skip Graph [3] は，データ構造の 1 つである Skip List [9] を応用した構造化オーバレイである．Skip List は，連結リスト上で確率的なショートカットリンクを形成し，データの検索を効率的に行う乱択アルゴリズムである．Skip Graph は多数のノード群で自律分散的に Skip List 構造のネットワークを構築するものであり，Skip List と同様に，全順序関係を持つキー群をその順序を維持したまま管理することができる．キーをハッシュする DHT と異なり，Skip Graph はキーの範囲を指定した検索を行うことができる．

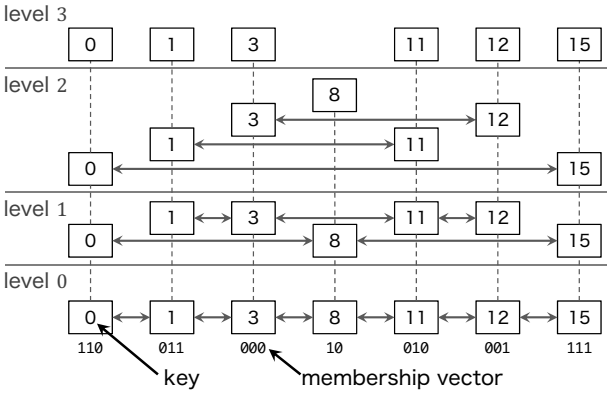

図 1　Skip Graph の構成例
Fig. 1　An example of Skip Graph.

Skip Graph のネットワーク構造の例を図 1 に示す．Skip Graph では，各ノードが 1 つのキーを担当し，これらのノード群がキーの順に対称なリンクのネットワークを形成する．その上で，複数階層（レベル）にわたって各ノードがショートカットリンクを形成する．ショートカットリンクの形成には，各ノードがキーとは別に持つ，membership vector (MV) と呼ばれる m 進数の乱数を用いる（本稿では $m = 2$ とする）．レベル i では，MV の接頭 i 桁までが等しいノード同士がショートカットリンクを形成する．このとき，各ノードが保持するリンクの数，すなわち経路表サイズは，ノード数が N のとき，$O(\log N)$ となる．

Skip Graph におけるルーティングは，Skip List と同様に，メッセージを発行するノードが自身の最高レベルから開始し，宛先ノードに最も近づくレベルのリンクを使用した転送を繰返すことによって行う．上位レベルで多くのノードをスキップしながら宛先ノードに近づくこのルーティング方式により，経路長は $O(\log N)$ となる．

Skip Graph は各ノードが持つランダムな MV に基づいて経路表構築を行うため，非効率的な経路表構築が行われてしまうことがある．例えば，近接する 2 ノードの MV の一致度が高い状況では，低レベルから高レベルまでのリンクが総てこの 2 ノードで張られてしまうことになる．換言すると，経路表内のレベル毎に用意された各エントリには，同一のノードが挿入されることになる．このような状況では，高レベルのリンクで多くのノードをスキップする狙いに適さない経路表構築となってしまう．

2.2 柔軟な経路表（FRT）

柔軟な経路表（Flexible Routing Tables, FRT）[7] は，構造

化オーバレイにおけるルーティングアルゴリズムの設計手法である．従来の構造化オーバレイの多くが，あらかじめ各ノードが構築すべき経路表をノード ID 等の組合せによって厳密に定義し，各ノードがこの定義に当てはまるノードを経路表に保持することで経路表構築を行っていたのに対して，FRT は全く異なった経路表構築手法を提案している．すなわち，経路表の完成形をあらかじめ定めるのではなく，経路表の優劣を示す順序関係 $\leqq_{\mathrm{RT}}$ を定義し，これに従って経路表の改良を繰返すというものである．構築すべき経路表を厳密に定義し，経路表に追加するノード候補を限定してしまうアルゴリズムに比べて，FRT に基づいて設計されたアルゴリズムは，2.2.1 項で挙げる様々な利点を備える．

FRT に基づく構造化オーバレイとして，これまでに幾つかのアルゴリズムが提案されている [1], [4], [6], [7].

2.2.1 FRT に基づく設計の利点

FRT に基づく構造化オーバレイは，以下の利点を備える．

- 経路表サイズ L の動的な調整が可能である．L はノード性能やネットワークの安定性などに応じて変更できる．
- シングルホップ動作とマルチホップ動作を一貫して扱う構造化オーバレイが設計できる．ネットワークに参加するノード数 N と経路表サイズ L に関して，$N \leqq L$ を充たす場合では経路表に全ノードを載せることが可能となり，シングルホップ動作を実現する．$N > L$ となる場合ではマルチホップ動作となり，経路表に載せるノードを選択しながら性能向上を図る．
- 様々な指標を考慮するための拡張性を備える．例えば，経路長削減に加えて通信遅延を考慮した Proximity-aware Flexible Routing Tables (PFRT) [5] や，グループ間通信の削減を考慮した Grouped Flexible Routing Tables (GFRT) [8]，メッセージの流量を考慮した Flow-based Flexible Routing Tables (FFRT) [2] が提案されている．

2.2.2 経路表管理操作

FRT では，経路表を構築・改良していく際に以下の 2 つの操作を伴う．

- **Entry learning**: Entry learning は，経路表にエントリを追加する操作である．この操作により，新たに得たノード情報は選別されることなく経路表に追加される．追加するノード情報は，様々な通信の結果から得ることができ，また，適当な ID に向けて能動的に探索することによって得ることも可能である．
- **Entry filtering**: Entry filtering は，経路表エントリ数が，あらかじめ設定された経路表サイズ L を越えたときに経路表エントリを 1 つ削除する操作である．削除の際には，sticky entry と呼ばれる削除候補から除外するエントリ集合を除くエントリ群から，2.2.3 項にて述べる経路表間の順序関係に従って，最も優れた経路表となるように削除するエントリを決定する．sticky entry は，到達性保証やアルゴリズムの拡張のために定めるものである．

FRT に基づく経路表構築では，経路表エントリ数が最大サイズ L を越えるまでは entry learning が繰返される．エントリ数が L を越えると，entry learning とともに entry filtering が実行され，経路表の改良が行われる．

2.2.3 経路表間の順序関係 $\leqq_{\mathrm{RT}}$

経路表間の順序関係 $\leqq_{\mathrm{RT}}$ とは，経路表に含まれるノードの組合せに注目し，その優劣を判定するために定義するものである．$\leqq_{\mathrm{RT}}$ を用いることによって，2 つの経路表 E と F について $E \leqq_{\mathrm{RT}} F$ が得られたとき，E は F よりも優れた経路表であると判断できる．構造化オーバレイの設計者は，採用するネットワークトポロジや ID 距離などの指標から，適切に $\leqq_{\mathrm{RT}}$ を定義することによって，FRT に基づくアルゴリズムの設計ができる．

3. FRT-Skip Graph

我々は，Skip Graph と同様のネットワーク構築によって範囲検索を可能としつつ，乱数の偏りによる問題点を解決する新たな構造化オーバレイ FRT-Skip Graph を提案する．FRT-Skip Graph は，Skip Graph と同様に各ノードが 1 つのキーと MV を持ち，Skip List 構造をネットワークトポロジとする．この時，Skip Graph と異なり，あるノードが他ノードとリンクを張る際に対称性を要求しない．つまり，このリンクを張る際に，当該の他ノードが自身に向けたリンクを張ることを要求しない．その代わりに，各ノードが独自に経路表を構築することを繰返すことによって，それぞれのリンクには対称性が認められるようになる．このような経路表構築手法は，柔軟な経路表（FRT）のフレームワークに基づいて設計している．FRT に基づいた設計により，FRT-Skip Graph は経路長を小さく保ちつつ，Skip Graph にはない様々な指標を導入する拡張性を持つ．例えば，範囲検索やブロードキャストを効率良く実行するために，自ノードの周辺ノードに対するリンクを多く保持する拡張ができる．また，通信遅延の小さいノードへのショートカットリンクを優先して持つことで，ルーティングの性能向上を図ることができる．

3.1 経路表構築

本節では，まず FRT-Skip Graph の構築する経路表の構造について述べる．その後，FRT に基づいて経路表を構築する際に必要な，sticky entry の指定や，経路表間の順序関係の定義について述べる．

3.1.1 経路表の構造

FRT-Skip Graph 上の各ノードはキーの順序関係に従って並び，自身のキーよりも小さいキー群の方向（前方）と，大きいキー群の方向（後方）の 2 方向に向けたリンクを管理する．そのため，各ノードは前方用と後方用の 2 つの経路表を独立に管理する．以下ではそれぞれを前方経路表（forward table, FT），後方経路表（backward table, BT）と呼ぶ．

FRT-Skip Graph の管理する経路表の構造を図 2 に示す．FT と BT は，それぞれ各ノードが独自に設定可能な L 個のエントリを保持する．各エントリはキーと IP アドレスの他に，Skip Graph と同様の MV を保持する．レベル毎にエントリを設け，複数エントリに同一ノードが挿入されることもあった Skip Graph と異なり，FRT-Skip Graph では，キー順にソートされたフラットな経路表構造となっている．このため 1 つのノード

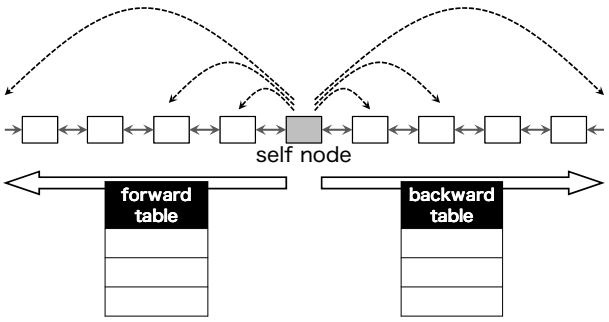

図 2　FRT-Skip Graph の経路表
Fig. 2　Routing tables of FRT-Skip Graph.

が複数エントリを占領することがない．このような設計により，レベル i のエントリで 2^i 個のノードをスキップする狙いが達成されないという Skip Graph の問題点を回避することができる．

3.1.2　Sticky Entry

FRT に基づく構造化オーバレイの設計時には，経路表の改良の過程で entry filtering で削除せず，経路表に必ず確保しておくエントリ，すなわち sticky entry を定義する．FRT-Skip Graph においては，sticky entry を，前方/後方のそれぞれで自身から最も近いキーを持つエントリと定義する．各ノードが sticky entry を確保することで，全ノードが 1 つの対称な連結リストを成し，任意の宛先キーへの到達性が保証される．ノードの churn への耐性を高めるために，自身から近い数エントリを sticky entry に指定し，前後数ノードが対称リンクを保つアルゴリズム設計とすることも可能である．

3.1.3　経路表間の順序関係 $\leqq_{\text{FRT-SG}}$

FRT-Skip Graph は，経路表内の各エントリについて，キーの順序に基づく自ノードからの遠近と MV の一致度に基づくレベルを計算することで，経路表の優劣を判断する．以下では，FT, BT 共に，単に $E = \{e_i\}_{i=1,\ldots,|E|}$ と書き，あるエントリ e のレベルを e.lv と書く．ここで，経路表の各エントリは，キーが自ノードから近いものほど小さいと見なす大小関係に基づき，昇順で添字が振られているとする．

FRT-Skip Graph における経路表間の順序関係 $\leqq_{\text{FRT-SG}}$ を定義するにあたって，Skip Graph の経路表を考察する．Skip Graph で定義されるレベルとは，全ノードが形成する連結リスト（すなわちレベル 0 のリスト）において，ノードを幾つ隔てたエントリなのかを確率的に推定するためのものである．例えば，レベルが i のエントリは，2^i 個離れたノードであることが期待される．Skip Graph では，各レベルで 1 つずつエントリを保持することにより，$2^0, 2^1, ..., 2^i$ 個離れたノードとのショートカットリンクを形成している．

FRT-Skip Graph においても，経路表内には各レベルのエントリが含まれていることで，Skip Graph と同等の経路長効率が期待できると考えられる．加えて，Skip Graph の問題点であった，ある高レベル i のエントリが 2^i 個よりも小さい個数のノードしかスキップできないことに注目し，高レベルのエントリを優先して複数持つことで，ショートカットリンクの有効性を高めることができると考えられる．

以上の考察から，我々は，FRT-Skip Graph の構築すべき理想的な経路表を，各レベルで同数程度のエントリを持ち，経路表サイズに応じて高レベルのエントリを優先して多く持つものと定義している．このような経路表を構築するために，以下で幾つかの記法を定義した後，FRT-Skip Graph における経路表間の順序関係 $\leqq_{\text{FRT-SG}}$ を定義する．

まず，レベルに基づく E の分割を以下で定める．

［定義 3.1］（経路表 E のレベルに基づく分割 E_l）

$$E_l = \{e \in E | e.\text{lv} = l\} \tag{1}$$

次に，各分割 E_l の要素数，すなわちレベル l となるエントリの個数に注目し，次の数列を定義する．

［定義 3.2］（経路表 E のレベル数列 $\text{seq}(E)$）

$$\begin{aligned} \text{seq}(E) \;&=\; (-l_1, -l_2, \ldots) \\ &\forall i < \forall j,\;\; (|E_{l_i}| > |E_{l_j}|) \vee (|E_{l_i}| = |E_{l_j}| \wedge l_i > l_j) \end{aligned} \tag{2}$$

例えば，$E = E_0 \coprod E_1 \coprod E_2 \coprod E_3$ かつ $|E_0| = 3$, $|E_1| = 4$, $|E_2| = 4$, $|E_3| = 1$ のとき，$\text{seq}(E) = (-2, -1, 0, -3)$ となる．

［定義 3.3］（個数に基づく最大レベル順列 $(E)_{\text{lv}}$）　E_{l_i} を i の昇順で並べ，かつ各 E_{l_i} 内の各要素を添字の降順に並べ繋げた列を，個数に基づく最大レベル順列 $(E)_{\text{lv}}$ と呼ぶ．

例えば，$E = E_0 \coprod E_1 \coprod E_2 \coprod E_3$ かつ $|E_0| = 3$, $|E_1| = 4$, $|E_2| = 4$, $|E_3| = 1$ のとき，$(E)_{\text{lv}}$ は E_2, E_1, E_0, E_3 の順に並び，かつ各 E_l 内でエントリは添字の降順に並ぶ．

以上を用いて，FRT-Skip Graph の経路表間の順序関係を次のように定める．

［定義 3.4］（経路表間の順序関係 $\leqq_{\text{FRT-SG}}$）

$$\begin{aligned} E \leqq_{\text{FRT-SG}} F \;\Leftrightarrow\; &(\text{seq}(E) <_{\text{dic}} \text{seq}(F)) \\ &\vee\{(\text{seq}(E) =_{\text{dic}} \text{seq}(F)) \\ &\quad \wedge((E)_{\text{lv}} \leqq_{\text{dic}} (F)_{\text{lv}})\} \end{aligned} \tag{3}$$

ここで，$\leqq_{\text{dic}}$ は辞書式順序であり，

$$\{a_i\} <_{\text{dic}} \{b_i\} \;\Leftrightarrow\; a_k < b_k \;\;(k = \min\{i \mid a_i \neq b_i\})$$

$$\{a_i\} =_{\text{dic}} \{b_i\} \;\Leftrightarrow\; a_i = b_i$$

$$\{a_i\} \leqq_{\text{dic}} \{b_i\} \;\Leftrightarrow\; (\{a_i\} <_{\text{dic}} \{b_i\}) \vee (\{a_i\} =_{\text{dic}} \{b_i\})$$

である．

3.2　ルーティング

Skip Graph におけるノードの探索は，高いレベルから順にレベルを下げながら，宛先キーに最も近づくレベルのエントリに転送を繰返すものである．これに対して FRT-Skip Graph では，経路表中で最も宛先キーに近づくエントリに転送するという greedy routing によってノードの探索を行う．レベルに基づき転送先エントリを決定する Skip Graph に比べて，全エントリから最適なものを選ぶ FRT-Skip Graph は，経路表エントリの無駄のない利用と相俟って，Skip Graph 以上に小さい経路長を達成する．

FRT-Skip Graph が Skip Graph と同等以上の経路長効率を達成することを示すために，幾つかの補題を用意する．

［補題 3.1］ ノード数 N で構築される FRT-Skip Graph のネットワークにおいて，経路表に含まれる Skip Graph 由来のエントリ数は高々$O(\log N)$ である．

［証明 ］ Skip Graph では，MV の最長一致桁数，すなわち自身が孤立する最大レベルの数だけ経路表エントリが設けられている．このとき経路表エントリ数は $O(\log N)$ であり，また，実際には複数のエントリに同一ノードが挿入されることがあるため，Skip Graph の経路表内に含まれるノードの種類は経路表エントリ数以下である． □

［補題 3.2］ ノード数 N で構築される FRT-Skip Graph のネットワークにおいて，経路表サイズを $L^* = O(\log N)$ としたとき，Skip Graph の経路表に含まれるノードは，$\leqq_{\mathrm{FRT-SG}}$ に基づく entry filtering で削除されない．

［証明 ］ 経路表サイズを $L^* = O(\log N)$ としたとき，補題 3.1 より，Skip Graph 由来のエントリを含むには充分である．よってここでは，$\leqq_{\mathrm{FRT-SG}}$ に基づく entry filtering で Skip Graph 由来のエントリが削除されないことを示す．定義 3.4 より，$\leqq_{\mathrm{FRT-SG}}$ に基づいてエントリを 1 つ削除するとき，エントリ数が最も多いレベル内で，かつ自ノードからより離れたエントリが選択される．Skip Graph 由来のエントリは，そのレベル内で最も自ノードに近いエントリであるため，削除対象に選ばれることはない． □

以上の補題を用いることで，次が示される．

［定理 3.1］ ノード数 N で構築される FRT-Skip Graph のネットワークにおいて，経路表サイズを $L^* = O(\log N)$ としたとき，充分な経路表の改良が行われた後の経路長は $O(\log N)$ である．

［証明 ］ 補題 3.2 より，entry learning によって一度経路表に挿入された Skip Graph 由来のエントリは，entry filtering で削除されることはない．ここで，充分な経路表の改良操作が行われ，各ノードの経路表に Skip Graph 由来のエントリが総て挿入された場合を考える．この時，FRT-Skip Graph の greedy routing 方式のノード探索では，Skip Graph でのノード探索で用いるエントリか，宛先にさらに近づくエントリに対して転送を行うことになる．各ノードがこの方針で転送を繰返すことにより，経路長は Skip Graph と同等以上に小さくなるため，FRT-Skip Graph の経路長は $O(\log N)$ である． □

定理 3.1 では，各ノードの経路表サイズが $O(\log N)$ のとき，経路長が $O(\log N)$ となることを示した．FRT-Skip Graph では，ネットワークの安定性やノード性能に応じて経路表サイズをさらに大きく設定することも可能であり，2.2.1 項で述べたシングルホップ動作も実現可能である．

3.3 Skip Graph エミュレーション

FRT-Skip Graph は，適切な経路表サイズを設定することで Skip Graph 由来のエントリを総て含むことができるため，ルーティングアルゴリズムを greedry routing から Skip Graph のアルゴリズムに変更することで，Skip Graph を模倣した動作を実現できる．Skip Graph のエミュレーションが可能なことにより，これまでに提案されてきた Skip Graph の応用に対して FRT-Skip Graph が適応することが期待できる．

4. 評　　価

オーバレイネットワーク構築ツールキット Overlay Weaver [10], [11] 上に FRT-Skip Graph を実装し，シミュレーションにより評価実験を行った．実験に用いた環境は以下の通りである．

- Overlay Weaver 0.10.4
- OS : 64 bit Windows 7
- CPU : Intel Core i7-3612QM 2.10 GHz
- Java SE 7 Update 21

以下の実験では，クエリの転送方式として，探索を開始したノードが次の転送先の問合せを繰返す反復方式を採用した．実験は，全ノードがオーバレイに参加したのち，各ノードがランダムなキーに向けて探索を繰返すことで経路表を構築し，各ノードの探索回数が 200 回の時点での経路長を測定した．

4.1 Skip Graph との比較

FRT-Skip Graph のルーティング効率を評価するために，比較対象として，3.3 節にて述べた FRT-Skip Graph による Skip Graph を模倣した構造化オーバレイ，Emulated Skip Graph を用意した．Emulated Skip Graph は，経路表構築に関しては FRT-Skip Graph と同様に行うが，ルーティング時には Skip Graph 由来のエントリのみを利用する．ただし転送先ノードの選択は，レベルを順に下っていく本来の Skip Graph のものではなく，宛先キーに最も近づくエントリを選ぶ greedy routing に基づいている．実験時には，各ノードが Skip Graph 由来のエントリを総て経路表に追加するまで探索を繰返すことにより，Skip Graph と同じ条件で動作させた．以下では，Emulated Skip Graph を単に Skip Graph と呼ぶ．

実験は，ノード数を $N = 100, 1000, 10000$ として行った．また，FRT-Skip Graph における経路表サイズを，$\log N$ を目安とし，それぞれ $L = 7, 10, 14$ とした．

実験結果を図 3，4 に示す．いずれの実験でも，FRT-Skip Graph は Skip Graph に比べて経路長を大幅に削減することができた．このような結果が得られた理由として，有効に利用された経路表エントリ数や，ショートカットリンクを形成するノードの選び方の違いによるものが考えられる．Skip Graph は 2.1 節で述べたように，経路表の複数エントリに同じノードが挿入されることがある．そのため，経路表内の有効なエントリ数が経路表サイズ $\log N$ 以下となる．それに対して FRT-Skip Graph は，設定した経路表サイズ L の総てのエントリに異なるノードを挿入することができる．また，Skip Graph では，自ノードの近くにレベルの高い他ノードが存在するとき，本来高いレベルで遠隔のノードへ転送する狙いのショートカットリンクが，そのノードに向かうことが起きていた．FRT-Skip Graph では，高レベルのショートカットリンクを複数持つことが可能であるため，遠隔に向けたショートカットリンクが形成でき，小さい経路長となったと考えられる．

4.2 経路表サイズと経路長の関係

FRT-Skip Graph の機能の 1 つである経路表サイズの調整を利用し，経路表サイズ L を変更しながら平均経路長を測定する実験を行った．ノード数を $N = 10000$ とし，経路表サイズを，

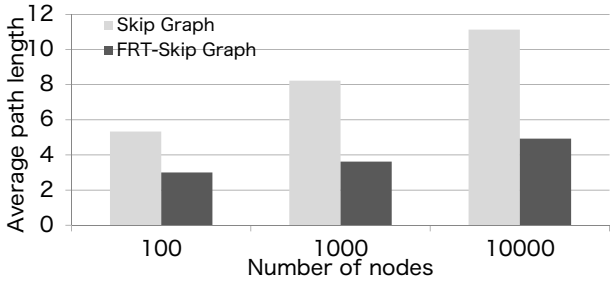

図 3　平均経路長の比較

Fig. 3　Average path length.

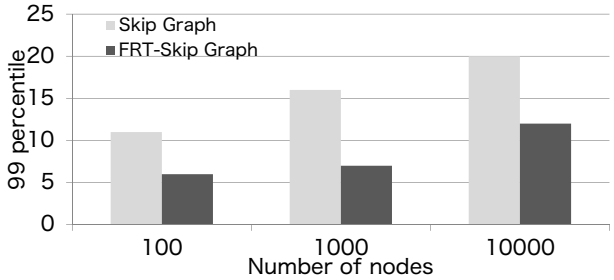

図 4　経路長の 99 パーセンタイル値の比較

Fig. 4　99 percentile of path length.

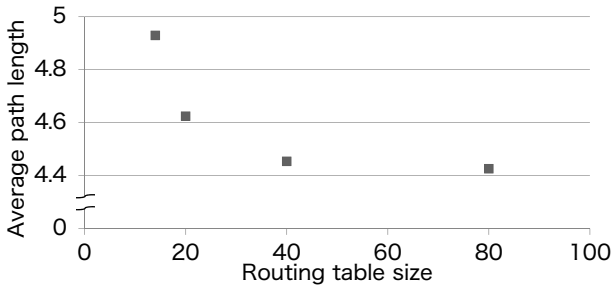

図 5　経路表サイズと平均経路長

Fig. 5　Routing table size and average path length.

前節の実験 $L = 14$ に加えて $L = 20, 40, 80$ と変化させながら，経路長を測定した．

実験結果を図 5 に示す．この結果から，経路表サイズを大きくするにつれて平均経路長が小さくなっていることがわかる．特に，$L = 14$ から $L = 20, 40$ と変化させたときに平均経路長は大きく削減されている．$L = 40$ と $L = 80$ のとき平均経路長にほとんど変化はなかったが，これは $N = 10000$ かつ 200 回の探索を行った後で測定を行ったためであり，さらに経路表の改良を進めることによって，$L = 80$ での経路長はさらに削減される．

実験では，$N = 10000$ という規模の大きい状況のシミュレーションを行った際の結果を示した．これに対して $N = 100$ といった小規模のとき，$L > N$ となるように経路表サイズを設定することによって，平均経路長を 1 まで削減することができた．すなわち，FRT の利点の 1 つである，シングルホップ動作とマルチホップ動作を一貫して扱えることを確認することができた．

5. まとめと今後の課題

本稿では，Skip Graph と同様にキーによる範囲検索を可能とする新たな構造化オーバレイ，FRT-Skip Graph を提案した．FRT-Skip Graph は，乱数の偏りにより経路表エントリが有効利用できないという Skip Graph の問題点を，FRT に基づく新たな経路表構築手法の設計により解決した．

FRT に基づく設計により，FRT-Skip Graph は経路長を小さく保ちながら様々な指標を導入する拡張性や，経路表サイズを動的に設定する機能を有する．FRT 由来のこれらの特長を活かすことで，ショートカットリンクを増やしてさらなる経路長効率を達成したり，アプリケーションの要請に応じて経路表エントリを有効活用する拡張が可能である．

本稿では，FRT-Skip Graph の経路長に関する性質を明らかにするとともに，評価実験により，FRT-Skip Graph の経路表構築が経路長削減に効果的であることを確認した．

今後の課題として，ノード位置に応じて経路表サイズのバランスを調整する手法を考案することや，経路表にとって望ましいノードを効率良く探索する手法を導入することがある．また，極めて小さい経路表サイズを設定した際に，どのような挙動を示すか考察し，検証していく必要がある．

謝辞　本研究の遂行にあたって議論して頂いた，大阪市立大学の安倍広多教授に感謝致します．本研究は JSPS 科研費 25700008, 26540161 の助成を受けたものです．

文　献

[1] Y. Ando, H. Nagao, T. Miyao, and K. Shudo. FRT-2-Chord: A DHT Supporting Seamless Transition between On-hop and Multi-hop Lookups with Symmetric Routing Table. In *Proc. ICOIN 2014*, pp. 170–175. IEEE, 2014.

[2] Y. Ando, H. Nagao, T. Miyao, and K. Shudo. Routing Table Construction Method Solely Based on Query Flows for Structured Overlays. In *Proc. IEEE P2P 2014*, pp. 1–5. IEEE, 2014.

[3] J. Aspnes and G. Shah. Skip Graphs. *ACM Transactions on Algorithms*, Vol. 3, No. 4, Article 37, 2007.

[4] M. Hojo, H. Nagao, T. Miyao, and K. Shudo. A Two-dimensional Structured Overlay Based on Flexible Routing Tables. In *Proc. IEEE ISCC 2015*, pp. 309–314. IEEE, 2015.

[5] T. Miyao, H. Nagao, and K. Shudo. A Method for Designing Proximity-aware Routing Algorithms for Structured Overlays. In *Proc. IEEE ISCC 2013*, pp. 508–514. IEEE, 2013.

[6] T. Miyao, H. Nagao, and K. Shudo. A Structured Overlay for Non-uniform Node Identifier Distribution Based on Flexible Routing Tables. In *Proc. IEEE ISCC 2014*. IEEE, 2014.

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

[8] H. Nagao and K. Shudo. GFRT-Chord: Flexible Structured Overlay Using Node Groups. In *Proc. IEEE UIC 2014*, pp. 187–195. IEEE, 2014.

[9] W. Pugh. Skip Lists: A Probabilistic Alternative to Balanced Trees. *Commun. ACM*, Vol. 33, No. 6, pp. 668–676, 1990.

[10] K. Shudo. Overlay Weaver. http://overlayweaver.sourceforge.net, 2006.

[11] K. Shudo, Y. Tanaka, and S. Sekiguchi. Overlay Weaver: An Overlay Construction Toolkit. *Computer Communications*, Vol. 31, No. 2, pp. 402–412, 2008.