柔軟な経路表：経路表空間上の順序関係に基づくオーバレイネットワークルーティング方式

長尾洋也[†]　首藤一幸[†]

オーバレイネットワークにおけるルーティングアルゴリズムの構成手法である“柔軟な経路表（FRT）”を提案する．FRT に基づくアルゴリズムは，経路表間の優劣を表す順序関係 $\leq_{ID}$ を定義し，それに従って経路表の改良を繰り返すことで，経路表構築の際に経路表の候補を限定することなくノード ID を考慮することが可能となる．

FRT に基づく DHT アルゴリズム FRT-Chord を設計し，十分経路表が更新されるとき，経路長がノード数 N に対して $O(\log N)$ になることを示した．また，FRT-Chord を実装し，FRT の経路表更新手続きが意図したとおりに動作し，経路表を短縮することを確認した．

Flexible Routing Tables: Routing Algorithms in Overlay Networks with Total Orders in Routing Table Space

Hiroya Nagao[†] and Kazuyuki Shudo[†]

We present “Flexible Routing Tables (FRT)”, a method of designing routing algorithm for structured overlay networks. The algorithms based on FRT are able to consider node identifiers without restriction of candidates for routing tables, because they define orders “$\leq_{ID}$” of preferativity for node identifiers in tables, and repeat updating own tables based on the order.

We design FRT-Chord, a DHT based on FRT and demonstate that it achieves $O(\log N)$-hop lookup performance. We also implement it and show that its method of constructing routing tables works well and make path lengthes short.

1. はじめに

オーバレイネットワークとは，実ネットワーク（アンダーレイネットワーク）上に構築した仮想的なネットワークである．この技術を用いてピュア Peer-to-Peer 型のネットワークを構成する場合，オーバレイネットワーク上でのルーティング方式は大きく二つに分類される．一つは非構造化オーバレイと呼ばれ，ノード探索クエリを再帰的に拡散させる方式である．この方式は Gnutella や Freenet 等で利用された．もう一方は構造化オーバレイと呼ばれ，ノード探索クエリの転送方式が構造的に定められており，目的とするノードに確実に到達することが保証された方式である．構造化オーバレイの方式に Distributed Hash Table（DHT）がある．

2000 年以降，DHT は盛んに研究され，Chord[1] をはじめとする多くの方式[2)~6)] が提案された．これらの方式は 100 万ノード超の大規模かつ不安定なアンダーレイネットワークを対象とし，オーバレイネットワーク上を短い経路長でルーティングすることを目標のひとつとして設計され，優れた負荷分散性能と耐故障性を持つ．このとき，各ノードが管理する経路表は非常に小さく，数エントリから数百エントリである．これは，大規模ネットワーク上において，全ノードが載った経路表の維持・管理が現実的ではないという判断に基づいている．しかし，適切なエントリの数は様々な条件次第である．OneHop[7] や EpiChord[8] は，経路表を大きくとることで全ノードを経路表に載せる方式であり，OneHop では，Gnutella ネットワークのノードの出入り頻度を仮定したとき，10 万ノード以下の OneHop ネットワークにおいて 99%の探索が正しくルーティングされる．このように，必ずしも経路表を小さく保つ必要はなく，ノードの総数や参加・離脱の頻度などの条件に応じて自由にかつ動的に変更可能であるべきである．そして，元々大規模で不安定なネットワークを対象に構成された DHT は，その対象領域を広げ，新たな利用方法へ拡張されつつある．

† 東京工業大学
Tokyo Institute of Technology

拡張の基礎的な例として，通信遅延の考慮があげられる．ランダムに決定されるノード ID のみを考慮して経路表を構成する従来の DHT に対して通信遅延の考慮を追加した方式に LPRS-Chord[9] や Coral[10] がある．また，ノードグループの考慮を追加した方式に Diminished Chord[11] や GTap[12] がある．しかし，より複雑で強力な拡張を行うためには，克服すべき本質的な問題がある．

経路表構築の基本は，ノード ID の考慮である．ここへ他の要素の考慮を追加することでアルゴリズムは拡張される．従来アルゴリズムは，この基本となるノード ID の考慮を，経路表に含めるノード ID の組合せを限定することで実現する．たとえば Chord[1] は経路表に載せるノード ID を，2^i 先の担当ノード ID に限定する．また，Kademlia[4] は，XOR に基づく距離が $[2^i, 2^{i+1})$ になるノード ID がそれぞれ一定個以下になる経路表に限定する．しかし，このようなノード ID の組合せを限定する方式には，ノード ID 以外の要素を考慮する機会を奪ってしまう問題や，拡張の際にノード ID に基づく限定をどのように緩和すべきかが明確でない問題，また，ノードが少ない場合に経路表に全ノードを載せることが出来ない問題などがある．これらはアルゴリズムの拡張を本質的に妨げる問題である．したがって，DHT の拡張性を向上させるためには，経路表を限定しない，あらゆるノード ID の組合せを対象とした，ノード ID を考慮する方式が必要不可欠である．そしてその新たな方式が，今後のオーバレイネットワークの応用を促し，多数のマシンを高度に連携する方式へ続いていくと我々は信じる．

我々は，オーバレイネットワークにおけるルーティングアルゴリズムの構成手法である**柔軟な経路表（Flexible Routing Table，FRT）**を提案する．FRT に基づくアルゴリズムは，経路表間の優劣を表す順序関係 $\leq_{ID}$ を定義し，それに従って経路表の改良を繰り返すことで，経路表の候補を限定することなくノード ID を考慮した経路表構築が可能となる．

FRT には次の特徴がある．

拡張性の向上　FRT は，経路表に載るノード ID の組合せを限定せずに経路表を構築する．これにより，ノード ID 以外に基づくあらゆる制限に対して柔軟に対応した経路表構成が出来る．

DHT の適用範囲の拡張　ネットワーク上のノード数と経路表のサイズに応じて，経路表サイズが大きい場合は経路長が $O(1)$ になるフルメッシュ型，小さい場合にはマルチホップ型のルーティングとなる．また，これらの間を連続的に推移することが出来る．

エントリ情報の有効利用　従来の DHT は経路表に入れたい都合の良いエントリ情報（ノード ID と IP アドレスの組）のみを経路表に入れる構造を持つが，FRT は，あらゆる経路表のパターンを許容するため，入手したあらゆるエントリ情報を無駄にせず，より良い経路表を構成できる．

高い状況適応性　常に経路表サイズを変更可能なため，ノード数やノードの安定性，要求の変化に応じた経路表を構成することが出来る．

具体的な FRT のアルゴリズムは，実際に FRT を利用して構成した DHT アルゴリズム **FRT-Chord** を用いて説明する．また，我々は FRT-Chord の実装・実験を行い，ノード ID の考慮が正しく行われ，従来同様の短い経路長が達成されることを確認した．

5 章において，将来の拡張を考慮した DHT に必要なアルゴリズムデザインについて論じ，FRT の拡張可能性に言及する．

2. 関 連 研 究

2.1 Chord

Chord は Chord リングと呼ばれるリング状の ID 空間 $[0, 2^m)$ を利用する DHT である．各ノードはランダムに m ビットのノード ID を持ち，key の m ビットのハッシュ値から時計回りに進んで最初に出くわすノードを，key に対応する**担当ノード**と呼び，key に対応する value を保存する．

Chord は 3 つの経路表（successor list，predecessor，finger table）を保持する．経路表の各エントリは，ノードの ID と IP アドレスの組である．successor list は，自身のノード ID から Chord リング上を時計回りに進んだ先にある定数個のノードを，predecessor は，反時計回りに進んだ先にある最初のノードをエントリとして保持する経路表である．この 2 つの経路表は，stabilize と呼ばれる通信を定期的に行うことで正しさが担保され，あらゆる ID の担当ノードへの到達性を保証する．この役割を担うエントリを **Short Distance Link(SDL)** と言う[6]．Finger table は，自身のノード ID から時計回りに $2^i (i = 0, ..., m-1)$ 進んだ ID の担当ノードのエントリを保持する．これにより，1 回のフォワーディングで目的 ID までの距離を大きく短縮出来るようになり，経路長が短縮される．この役割を担うエントリを **Long Distance Link(LDL)** と言う[6]．

Chord は ID x から ID y への距離 $d(x, y)$ を，時計回りに進んだ距離として次のように定義する．

定義 2.1 (ID 間距離 $d(x,y)$)

$$d(x,y) = \begin{cases} y-x, & (x<y) \\ 2^m, & (x=y) \\ y-x+2^m, & (x>y) \end{cases}$$

そして，担当ノードの predecessor に辿り着くまで探索対象の ID t への距離 $d(s',t)$ が最も小さくなるノード s' へフォワーディングを繰り返す．このように，ルーティング先のノードに最も近いノードへのフォワーディングを繰り返すルーティングを **greedy routing** と言う．このとき，ネットワーク上のノード数 N に対して $O(\log N)$ の経路長で到達する．

2.2 EpiChord

EpiChord[8] は，Chord リング上を両方向にフォワーディングする DHT であり，エントリ情報を次々にキャッシュとして経路表へ追加することで，経路表が大きいとき経路表に全ノードが載り，フルメッシュな経路表になる．

Chord リングを slice と呼ばれる領域に分割し，slice ごとにキャッシュエントリの数が一定以上になるように保つことで $O(\log N)$ の経路長を保証する．しかし，これは同時に経路表のパターンを限定してしまっている．FRT は EpiChord 同様，経路表に知り得たエントリ情報を次々追加することでフルメッシュな経路表を構成可能であるが，経路表の限定を必要としない．

2.3 Small-World Phenomenon

Small-World Phenomenon[13] は，2 次元空間上の格子点にノードがあり，それらの SDL が隣接する 4 ノードを指し，1 つの LDL が各ノードからのマンハッタン距離の 2 乗に反比例する確率密度分布に従って決まるノードを指すネットワークにおいて，経路長が $O(\log^2 N)$ になることを明らかにした研究である．1 次元の場合の Small-World Phenomenon を利用した DHT に Symphony[6] がある．Symphony は，Chord リングにおいて，自ノードからの距離に反比例する確率密度関数に従いランダムに ID を生成し，その ID の担当ノードを経路表エントリとする．これを k 回繰り返し k エントリの経路表を構築することで，$O((\log^2 N)/k)$ の経路長を実現する．確率分布を用いることで，経路表の制限を緩和しているが，ランダムに決定される ID に従ってエントリを選択するため，結果として経路表に入れるノード ID を限定してしまう．

FRT-Chord における優れた経路表の基準は，Symphony の確率密度分布と一致する．しかし，FRT に基づきその確率密度分布と経路表との関係を $\leq_{ID}$ へ変換することにより，経路表に入れるノード ID を限定しない方式を実現している．

3. FRT-Chord

FRT-Chord は，FRT に基づいて設計した DHT である．FRT-Chord は Chord リングを利用し，担当ノードの決定方法も Chord と同様であるが，経路表の構成手法に特徴がある．

FRT-Chord は，successor list，predecessor，finger table を区別せず，一つの経路表 E を保持する．経路表 E は経路表エントリ e_i の集合 $\{e_i\}$ である．各経路表エントリ e_i はノード ID e_i.id と，IP アドレスとポート番号の組 e_i.addr を含む．ただし，e_i.id を e_i と略記することがある．$\{e_i\}$ は自ノード ID s からの時計回りに整列されており，$i<j \Rightarrow d(s,e_i) < d(s,e_j)$ である．ここで，e_1 は successor（successor list の先頭）に相当し，$e_{|E|}$ は predecessor に相当する．このとき，これらは stabilize (3.2 節) によって正しい状態に保たれていると仮定する．

FRT-Chord は Chord と同様に，greedy routing を行うため，目的 ID t に対応するフォワーディング先 E.forward(t) は $d(e_i,t)$ が最小になる $e_i(\in E)$ となる．

3.1 $\leq_{ID}$：ID 集合間の順序関係

FRT は，経路表に含まれるノード ID の組合せに基づいて経路表空間上に順序関係 $\leq_{ID}$ を定義する．そして，$\leq_{ID}$ に従って経路表の改良を繰り返すことで，より良い経路表へ推移させるアルゴリズムである．

ここでは，FRT-Chord における $\leq_{ID}$ の設計について述べる．

3.1.1 最良経路表の定義

自ノード ID を s，目的 ID を t としたとき，あるフォワーディングに対する**短縮倍率**を，フォワーディング後の残り距離 $d(E.\text{forward}(t),t)$ をフォワーディング前の残り距離 $d(s,t)$ で割った値と定義する．つまり，値が小さいほど，残り距離を大きく短縮した優れたフォワーディングであることを表す．ここで，FRT-Chord の $\leq_{ID}$ の定義において，predecessor を除く e_i にフォワーディングする際の短縮倍率の最悪値（最大値）$r_i(E)$ に注目する．目的 ID が e_{i+1} のときに最悪値をとるので，$r_i(E)$ は次のようになる．

定義 3.1 (最悪短縮倍率 $r_i(E)$)

$$r_i(E) = \frac{d(e_i,e_{i+1})}{d(s,e_{i+1})}, (i=1,...,|E|-1)$$

ここで，任意の構成可能な経路表 E に対して $\tilde{E} \leq_{ID} E$ が成立する経路表 $\tilde{E}=\{\tilde{e}_i\}$ を**最良経路表**と呼び，次のように定める．

定義 3.2 (最良経路表 $\tilde{E}$) 最良経路表 $\tilde{E} = \{\tilde{e}_i\}$

は，$\max\{r_i(E)\}$ を最小にする経路表である．

補題 3.1

$$r_i(\tilde{E}) = 1 - \left(\frac{d(s,\tilde{e}_1)}{d(s,\tilde{e}_{|\tilde{E}|})}\right)^{\frac{1}{|\tilde{E}|-1}}$$

証明 $\Pi_{i=1}^{|E|-1}(1 - r_i(E)) = d(s,e_1)/d(s,e_{|E|})(=$一定) であるから，全項が等しいとき $\max\{r_i(E)\}$ が最小となる． □

定理 3.1 (最良経路表の経路長) $|E| = O(\log N)$ において，最良経路表を保持するとき，経路長は $O(\log N)$ となる．

証明 一般に N が大きいとき十分高い確率で $d(s,e_1) > 2^m/N^2$ が成立し[14]，距離の定義から $d(s,e_{|E|}) < 2^m$ が成立する．したがって，補題 3.1 より，任意の $i = 1,...,|\tilde{E}|-1$ について

$$r_i(\tilde{E}) < 1 - \left(\frac{d(s,\tilde{e}_1)}{2^m}\right)^{\frac{1}{|\tilde{E}|-1}} < 1 - \left(\frac{1}{N}\right)^{\frac{2}{|\tilde{E}|-1}}$$

が成立する．$|\tilde{E}| = 1 + 2\log N$ とすると，残り距離を $2^m/N$ 以下にするまでに必要な経路長は，最悪でも

$$\log_{r_i(\tilde{E})} \frac{1}{N} < \log_{1-\left(\frac{1}{N}\right)^{\frac{2}{|\tilde{E}|-1}}} \frac{1}{N} = \log N$$

となり，$O(\log N)$ である．このとき predecessor へのフォワーディングを考慮していないが，predecessor にフォワードした場合は自身が担当ノードでありフォワーディングが終了するからである．また，残り距離が $2^m/N$ 未満のとき，自ノードから $2^m/N$ 先までに存在するノードは，N が大きいとき十分高い確率で $O(\log N)$ 個となることが知られており[1]，その経路長は $O(\log N)$ となる．以上より，経路全体の経路長は $O(\log N)$． □

3.1.2 $r_i(E)$ による $\leq_{ID}$ の定義

最良経路表に “近い” 経路表を優れた経路表とみなし，その “近さ” を表す指標 $\leq_{ID}$ を次のように定義する．

定義 3.3 ($\leq_{ID}$) $\{r_i(E)\}$ を降順に並べた列を $\{r_{(i)}(E)\}$ としたとき，$E \leq_{ID} F \Leftrightarrow \{r_{(i)}(E)\} \leq_{\mathrm{dic}} \{r_{(i)}(F)\}$.

ここで辞書式順序 $\leq_{\mathrm{dic}}$ とは，

$\{a_i\} <_{\mathrm{dic}} \{b_i\} \Leftrightarrow a_k < b_k (k = \min\{i | a_i \neq b_i\})$

$\{a_i\} =_{\mathrm{dic}} \{b_i\} \Leftrightarrow a_i = b_i$

$\{a_i\} \leq_{\mathrm{dic}} \{b_i\} \Leftrightarrow (\{a_i\} <_{\mathrm{dic}} \{b_i\}) \cup (\{a_i\} =_{\mathrm{dic}} \{b_i\})$

である．

定理 3.2 ($\leq_{ID}$ の正当性) $\tilde{E} \leq_{ID} E$

証明 補題 3.1 より明らか． □

このようにして定義した $\leq_{ID}$ を用いた FRT-Chord のアルゴリズムを，FRT の重要な構成要素である到達性保証操作，エントリ情報学習操作，エントリ厳選操作に分けて説明する．

3.2 到達性保証操作

FRT は，到達性を保証する経路表エントリの更新操作を**到達性保証操作**と呼ぶ．

FRT-Chord は Chord と同様の stabilize 操作により，SDL を最新に保つ．これにより，経路表にいかなるエントリが載っていたとしても，Chord と同等の到達性が保証される．

3.3 エントリ情報学習操作

FRT はエントリ情報の学習を**エントリ情報学習操作**と呼ぶ．FRT はエントリ情報の具体的な学習方法を制限せず，知り得たあらゆるエントリ情報を有効活用できる．

学習方法には，たとえば，参加時に他のノードから経路表全体をコピーする方法や，ノードの探索時等に通信した相手のエントリ情報を取得する方法，エントリ情報を学習するために探索を行う方法（**能動学習探索**と呼ぶ）がある．

FRT-Chord における能動学習探索は Symphony に類似する方法であり，最良経路表に含まれるノード ID の分布を確率分布とみなし，その確率分布に従うように生成したノード ID への探索を行う．確率分布は自ノード s からの距離の逆数に比例する分布とし，その累積分布関数 $F(x)$ は次の通りである．

$$F(x) = \frac{\ln(\mathrm{d}(s,x)/\mathrm{d}(s,e_1))}{\ln(\mathrm{d}(s,e_{|E|})/\mathrm{d}(s,e_1))} \quad (1)$$

0 以上 1 未満の乱数 rnd を用いて探索対象 ID は次のようにして生成することが出来る．

$$s + \mathrm{d}(s,e_1)\left\{\frac{\mathrm{d}(s,e_{|E|})}{\mathrm{d}(s,e_1)}\right\}^{\mathrm{rnd}} \quad (2)$$

FRT-Chord では，参加時と探索時にエントリ情報の学習を行う．また，学習を促進する目的で必要があれば能動学習探索を行う．

FRT は新たに学習したエントリ情報を経路表 E へ次々と追加する．エントリ学習操作が繰り返されることで，より短縮倍率の優れたフォワーディングが可能となる．一方で追加を繰り返すことで経路表に含まれるエントリ数 $|E|$ は増加し続けるため，エントリ数に最大値 L を設け，無制限な経路表エントリの増加を防ぐ．ここで，L は遅延やマシン性能，ノードの離脱頻度などに応じて決めることが出来る，保持可能な経路表エントリ数に応じて柔軟にかつ動的に変更可能な値である．

ネットワーク内のノードが安定している場合など，

L がノード数 N より大きく設定されるとき，全ノードのエントリ情報を保持するフルメッシュな経路表が達成され，経路長は $O(1)$ となる．

3.4 エントリ厳選操作

FRT は，$L < N$ において，$|E|$ が L を越えようとするとき，現在の経路表に含まれるエントリ情報と，新たに取得したエントリ情報の合計 $|E|+1$ の中から 1 つを捨て，エントリを厳選し，$|E| \leq L$ に保つ．これを**エントリ厳選操作**と呼ぶ．FRT は順序関係 $\leq_{ID}$ に基づきエントリ厳選操作を行うことで経路表の改良を繰り返し，経路長を短縮する．こうして FRT は経路表の可能なパターンを限定することなくノード ID を考慮する．また，このエントリ厳選操作に ID 以外の要素を導入することでアルゴリズムを見通しよく拡張できる．

FRT-Chord の場合，新しく学習したエントリ情報を一時的に追加した経路表 $E = \{e_i\}$ において，$i = 2, ..., |E|-1$ から $S^E_{i-1} + S^E_i$ が最小の $i = i^*$ を探索し，e_{i^*} を削除する．このとき，$i \neq 1, |E|$ であるのは，それらが successor，predecessor に相当し，削除の対象としないからである．ただし，S^E_i は**正規化間隔**であり次のように定義される．

定義 3.4 (正規化間隔 S^E_i) $S^E_i = \log d(s,e_{i+1}) - \log d(s,e_i), (i=1,\ldots,|E|-1)$

降順に並べた正規化間隔の列を $\{S^E_{(i)}\}$ とすると，次が成立する．

補題 3.2 (正規化間隔と $\leq_{ID}$ の関係)
$E \leq_{ID} F \Leftrightarrow \{S^E_{(i)}\} \leq_{\mathrm{dic}} \{S^F_{(i)}\}$

証明

$$r_i(E) = \frac{d(s,e_{i+1}) - d(s,e_i)}{d(s,e_{i+1})} = 1 - \frac{d(s,e_i)}{d(s,e_{i+1})}$$
$$= 1 - 2^{-\log \frac{d(s,e_{i+1})}{d(s,e_i)}} = 1 - 2^{-S^E_i}$$

よって，$r_i(E)$ と S^E_i の大小が一致する． □

定理 3.3 (削除の最適性) FRT-Chord のエントリ厳選操作により作成される経路表を $E - \{e_{i^*}\}$ とするとき，任意の削除対象エントリ $e_i (i = 2, ..., |E|-1)$ について $E - \{e_{i^*}\} \leq_{ID} E - \{e_i\}$ が成立する．

証明 削除後の経路表に関して正規化間隔を考える．$k^* = \min\{j | S^E_{i^*-1} + S^E_{i^*} > S^E_{(j)}\}$ とするとき，$\{S^{E-\{e_{i^*}\}}_{(i)}\}$ の先頭から k^*+1 番目までは $S^E_{(1)}, ..., S^E_{(k^*-1)}, (S^E_{i^*-1} + S^E_{i^*}), S^E_{(k^*)}$ となる．このとき，k^*-1 番目までは削除前と変化しない．同様に，$k = \min\{j | S^E_{i-1} + S^E_i > S^E_{(j)}\}$ とするとき，$\{S^{E-\{e_i\}}_{(i)}\}$ の先頭から $k+1$ 番目までは $S^E_{(1)}, ..., S^E_{(k-1)}, (S^E_{i-1} + S^E_i), S^E_{(k)}$ となる．このとき，$k-1$ 番目までは削除前と変化しない．

ここで，i^* の決定方法から，$S^E_{i^*-1} + S^E_{i^*} \leq S^E_{i-1} + S^E_i$ が成立するので，$k^* \geq k$．したがって，$i = 1, ..., k-1$ において $S^{E-\{e_{i^*}\}}_{(i)} = S^{E-\{e_i\}}_{(i)}$ が成立し，かつ $S^{E-\{e_{i^*}\}}_{(k)} \leq S^{E-\{e_i\}}_{(k)}$ が成立する．したがって，補題 3.2 より $E - \{e_{i^*}\} \leq_{ID} E - \{e_i\}$ が成立する． □

ここで，$\{S^E_{i-1} + S^E_i\}$ を昇順にしておくことで i^* の探索は $O(1)$ で完了し，経路表の変更時に行われる更新も $O(\log N)$ で終了する．

FRT-Chord のエントリ情報学習操作とエントリ厳選操作を繰り返すことで，学習したエントリと削除するエントリが常に一致し，経路表の改良が停止することが考えられる．このときの経路表 E を**収束した経路表**と呼び，次が成立する．

定理 3.4 (厳選操作の非停止性) ノード数 N のネットワークにおいて，全ノードが収束した経路表 E を持つとき，$|E| = O(\log N)$ の条件下で経路長は $O(\log N)$ である．

証明 Chord リング上の領域 (e_i, e_{i+1}) にノードが存在する i の集合を J，存在しない i の集合を K と定義する．このとき，収束した経路表の定義から，$j \in J, i = 2, ..., |E|-1, S^E_j \leq S^E_{i-1} + S^E_i$ が成立するから，j を固定してこの不等式の和をとることにより，次が成立する．

$$S^E_j \leq \frac{\sum_{i=2}^{|E|-1}(S^E_{i-1} + S^E_i)}{|E|-2}$$
$$= \frac{2(\sum_{i=1}^{|E|-1} S^E_i) - S^E_1 - S^E_{|E|-1}}{|E|-2}$$
$$\leq \frac{2(\sum_{i=1}^{|E|-1} S^E_i)}{|E|-2} = \log \left(\frac{d(s,e_{|E|})}{d(s,e_1)} \right)^{\frac{2}{|E|-2}}$$

また，補題 3.2 より，定理 3.1 の証明と同様にして，

$$r_j(E) < 1 - \left(\frac{1}{N}\right)^{\frac{4}{|E|-2}} \quad (3)$$

である．ここで，残り距離を $2^m/N$ 以下にするために必要な経路長の上限を考える．フォワーディング先が $e_k (k \in K)$ のとき，e_k と e_{k+1} の間にノードが存在しないため，担当ノードは e_{k+1} であり，ルーティングが終了するため，フォワーディング先は $e_j (j \in J)$ の場合のみを考慮すればよい．このとき，式 1 が成立するので，定理 3.1 の証明と同様にすれば経路長は $O(\log N)$ である．また，残り距離が $2^m/N$ 以下についても同様に経路長は $O(\log N)$ であり，経路全体の経路長は $O(\log N)$ となる． □

削除の候補を $i = c+1, ..., |E|-1$ に変更すると，長

さ c の successor list が実現され，stabilize 操作が行われることで，正しい successor と predecessor が保持される．また，このときも定理 3.3 と同様，$\leq_{ID}$ に基づいて最も経路表を改善する選択となり，更新を繰り返すことで経路長が短縮される．このように，FRT は，経路表に載せるかどうかの条件に対して柔軟に対応することが可能である．

FRT では，エントリ厳選操作において削除の候補にしない successor list や predecessor などのエントリのことを *sticky entry* と呼ぶ．Sticky entry という例外を設けることで，順序関係 $\leq_{\mathrm{ID}}$ の構成をシンプルに行うことが出来る．こうするとき，FRT-Chord におけるエントリ厳選操作は次のように記述できる．

(1) $C \Leftarrow E$
(2) C から sticky entry を取り除く．
(3) C から $S^E_{i-1} + S^E_i$ が最小となるエントリを削除する．

4. 評　　価

オーバレイネットワーク構築ツールキット Overlay Weaver[15)16)] 上に FRT-Chord を実装し，エミュレーションにより評価を行った．実験には Overlay Weaver 0.9.11 を用い，OS は Windows 7 64bit，CPU は Core 2 Duo E8400 3.00 GHz，メインメモリは 8.00GB のマシン上で JRE 6 Update 20 を用いた．

4.1 エントリ厳選操作とエントリ情報学習操作方式

FRT-Chord のエントリ厳選操作を繰り返すことで，経路表が最良経路表に近づくことを明らかにする．また，各エントリ情報学習操作の特徴を調べる．

$N = 10^4$ のノードが参加を行い，ランダムに選択したノードからランダムに選択した ID への探索を 10 万回実行した後，新たに 1 ノードを参加させて，そのノードからの探索を 100 回行った．経路表エントリの最大数 L は 80 とした．

参加時に successor から経路表をコピーするかどうかおよび，ランダム ID への探索により学習するか能動学習操作により学習するかの計 4 通りのエントリ情報学習操作を行い，最後に参加したノードの経路表に含まれるノード ID を記録した．探索回数を 25 回および 100 回とし，条件ごとに自ノードから各エントリまでの距離の対数 $\log d(s, e_i)$ をプロットした（**図 1**, **図 2**）．また，最良経路表を参考のために表示しており，最良経路表に近いほど，より改良された経路表である．探索を繰り返した結果，どの条件においても最良経路表に近づき，エントリ厳選操作が有効にはたらいていることが分かる．

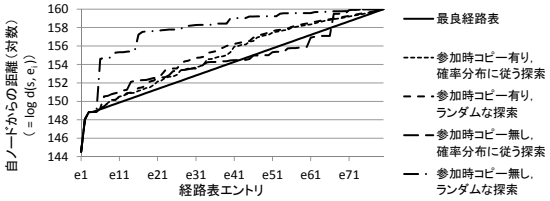

図 1 25 回探索後の経路表エントリ ID(縦軸対数)
Fig. 1 Entries after 25 lookups (semilog graph).

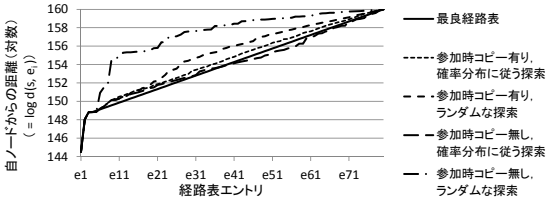

図 2 100 回探索後の経路表エントリ ID(縦軸対数)
Fig. 2 Entries after 100 lookups (semilog graph).

図 1 より，参加時にコピーせずにランダムに探索した場合が最も最良経路表から離れている．これは，ランダムの探索において，エントリ同士の間隔が狭い自身から近いエントリへのフォワード回数が相対的に少なくなり，自身から近い領域にあるエントリの学習回数が少なくなってしまうからである．一方，同じコピー無しであっても，確率分布に従った乱数による探索を行った場合は，そのようなフォワード回数の偏りが解消されており，最良経路表に大幅に近づいている．これらから，確率分布に従った探索は，エントリ情報の学習効果が高いと言える．また，ランダムに探索を行っていても，参加時に経路表をコピーした場合は最良経路表に近い経路表となっている．このことから，参加時の経路表コピーはエントリ情報の学習効果が高いと言える．

以上から，エントリ厳選操作は，参加時のコピーと確率分布に従って探索することで効率よく行われることが分かった．そのため FRT-Chord では，参加時のコピーを行う．一方で能動学習探索は，経路表を構成するための探索でありコストを伴うため，経路のエントリと直接通信しない再帰方式のフォワーディングを行う際など，経路表の学習頻度が少ない場合に行うこととする．

4.2 エントリ厳選操作と経路長の関係

全ノードを参加させた後，ランダムに選択したノードからランダムに選択した ID への探索を 50 万回行う実験を行った．ノード数 N を 10^2，10^3，10^4 とし，経路表サイズの上限 L は 20，40，80，160，successor

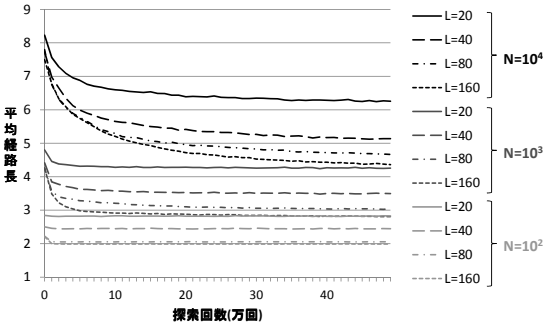

図 3 探索回数と平均経路長の変化
Fig. 3 Changes of average path length along with the number of lookups.

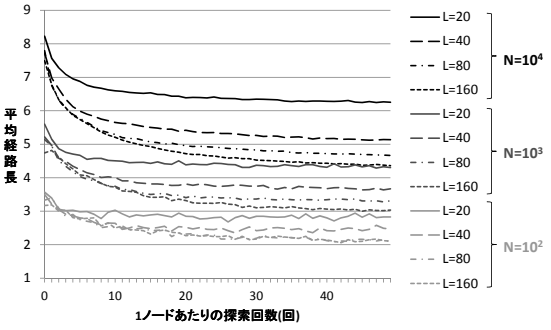

図 4 ノードあたりの探索回数と平均経路長の変化
Fig. 4 Changes of average path length along with the number of lookups per node.

list の長さは 4 である．フォワーディングは，探索を行うノードが次のフォワーディング先の問い合わせを繰り返す反復方式で行った．

探索時の経路長を 1 万回ごとに集計し平均を求めたものが**図 3** であり， どの条件でも，探索を繰り返すごとに平均経路長が短縮されていることが分かる．これは，探索によってエントリ情報を学習し，厳選操作が繰り返し行われることで，経路表が優れた状態へ推移した結果であると考えられる．

N 回ごとに平均経路長を求めることで横軸をノードあたりの探索回数としたものが**図 4** である． この図と図 3 を比較すると，1 ノードが行う探索回数が同じであれば，平均経路長が短縮される速さは同程度であると分かる．したがって，各ノードがそれぞれ一定回の探索を行い経路表を更新することは，ノード数 N に依らない経路表更新効果が得られると言える．このことから，FRT-Chord のエントリ学習操作はノード数に対してスケールすると言える．

4.3 ノード数，経路表サイズ，経路長の関係

十分に経路表厳選操作が行われた経路表を用いて

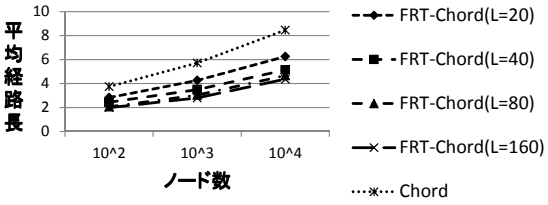

図 5 経路表サイズと平均経路長の関係
Fig. 5 Correlation between number of routing table size and average path lengths.

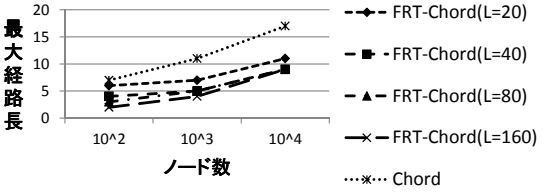

図 6 ノードあたりの探索回数と平均経路長の関係
Fig. 6 Correlation between number of routing table size and max path lengths.

ノード数と経路表サイズを変更しながら，経路長がどのように変化するかを計測する．ノード数 N を 10^2, 10^3, 10^4 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（**図 5**，**図 6**）．ここから，定理 3.4 で示した経路長の $O(\log N)$ 性が確認できる．また，L と経路長のトレードオフを自由に変更できることが確認できた．そして，経路表サイズがノード数より大きいとき（$N=10^2$，$L=160$），経路表がフルメッシュとなっており，経路長が $O(1)$ になることが確認できた．

5. 拡　　張

FRT は高い拡張性を提供する．これは，FRT が経路表空間のあらゆる部分集合に対して，ノード ID に関する優劣を定義するからである．

ノード ID 以外の要素を考慮するように DHT の経路表を拡張することは，すなわち，経路表に対して何らかの制限を加えることである．従来のアルゴリズムは，構築可能な経路表をたとえば経路長が $O(\log N)$ になるクラスの部分集合になるように限定し，ノード ID を考慮する．これを拡張することは，ノード ID に基づき限定した経路表の中からの選択を行うに過ぎない．その部分集合に含まれなかった経路表が $O(\log N)$ にならないとは限らず，また，そもそも経路長はルーティングの性質の一面しか表しておらず，ノード ID に基づく限定は，将来の拡張に対して強すぎる制限を与えている．これに対して FRT は，ノード ID 以外のあらゆる要素に関する条件を満たす経路表集合の中

で，どの経路表がより優れているかを表明することにより，ノード ID の考慮を行うことを可能にする．この点が従来 DHT と最も異なる点であり，DHT の適用範囲拡大に伴う，将来の拡張を見据えたアルゴリズムに必要な特徴であるといえる．

また，安定した環境や比較的ノードが少ない場合に経路表に全ノードを載せ，$O(1)$ の経路長を実現できることは，Web サービスやクラウドの構築などにおいて，DHT の特徴を活かした低い管理コストかつ可用性の高いシステムの構成に効果的であると考えている．しかし，実用にはアンダーレイを考慮した様々な最適化を施す必要がある．

アンダーレイを考慮する一例として，FRT の拡張性を利用し，FRT-Chord をノードグループを考慮するように拡張したアルゴリズムに LFRT-Chord[17] がある．LFRT-Chord では，経路表に同じグループに属するノードを優先的に載せるという制限のもとでノード ID を考慮する．これにより，経路長の伸びを抑えつつ，グループをまたぐ経路の削減を実現している．たとえば ISP やデータセンタなどをグループとすることで，それらをまたぐ通信量を削減でき，従来の DHT の持つデメリットを抑えることが出来る．

6. ま と め

本論文では，オーバレイネットワーク上のルーティングアルゴリズム構成手法である柔軟な経路表（FRT）および，FRT に基づいて構成した DHT である FRT-Chord を提案した．

従来の DHT は経路表の候補を限定することで経路表構築のために必要なノード ID の考慮を行う．FRT は，経路表間の順序関係 $\leq_{ID}$ を定義することで，経路表のパターンを限定しない，従来とは異なるノード ID の考慮方式を実現した．

FRT-Chord の諸性質を明らかにするとともに，FRT-Chord の実装・実験を行い，FRT に基づく経路表の更新が経路長を短縮することおよび，N に依らない回数の探索を各ノードが行うのみで経路表が十分更新されることを明らかにした．また，経路表の最大エントリ数がノード数より大きいとき経路長が $O(1)$ になる状態へ移行可能であることを示した．

本論文では，FRT-Chord を提案したが，今後は，FRT に基づく他のアルゴリズムや，それらを拡張した実用性の高い応用手法を構成していく．

謝辞 提案手法の分析方法について議論して頂いた大西真晶様に感謝致します．本研究は科研費（22680005）の助成を受けたものである．

参 考 文 献

1) Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. and Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *Proc. ACM SIGCOMM '01*, pp. 149–160 (2001).
2) Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D. and Kubiatowicz, J. D.: Tapestry: A Resilient Global-scale Overlay for Service Deployment, *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1, pp. 41–53 (2004).
3) Rowstron, A. I. T. and Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, *Proc. IFIP/ACM Middleware 2001*, pp. 329–350 (2001).
4) Maymounkov, P. and Mazières, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, *Proc. IPTPS '02 LNCS 2429*, pp. 53–65 (2002).
5) Kaashoek, M.F. and Karger, D.R.: Koorde: A Simple Degree-Optimal Distributed Hash Table, *IPTPS*, pp. 98–107 (2003).
6) Manku, G. S., Bawa, M. and Raghavan, P.: Symphony: distributed hashing in a small world, *Proc. USITS'03*, pp. 10–10 (2003).
7) Fonseca, P., Rodrigues, R., Gupta, A. and Liskov, B.: Full-Information Lookups for Peer-to-Peer Overlays, *IEEE Trans. Parallel Distrib. Syst.*, Vol. 20, pp. 1339–1351 (2009).
8) Leong, B., Liskov, B. and Demaine, E. D.: EpiChord: Parallelizing the Chord lookup algorithm with reactive routing state management, *Comput. Commun.*, Vol. 29, pp. 1243–1259 (2006).
9) Zhang, H., Goel, A. and Govindan, R.: Improving lookup latency in distributed hash table systems using random sampling, *IEEE/ACM Trans. Netw.*, Vol. 13, pp. 1121–1134 (2005).
10) Freedman, M. J. and Mazières, D.: Sloppy Hashing and Self-Organizing Clusters, *IPTPS*, pp. 45–55 (2003).
11) Karger, D. R. and Ruhl, M.: Diminished Chord: A Protocol for Heterogeneous Subgroup Formation in Peer-to-Peer Networks, *IPTPS*, pp. 288–297 (2004).
12) Zhang, Y., Li, D., 0002, L.C. and Lu, X.: Flexible Routing in Grouped DHTs, *Peer-to-Peer Computing*, pp. 109–118 (2008).
13) Kleinberg, J.: The Small-World Phenomenon:

An Algorithmic Perspective, *Proc. ACM STOC 2000*, pp. 163–170 (2000).
14) Cordasco, G. and Sala, A.: 2-Chord Halved, *Proc. HOT-P2P '05*, Washington, DC, USA, IEEE Computer Society, pp. 72–79 (2005).
15) Shudo, K., Tanaka, Y. and Sekiguchi, S.: Overlay Weaver: An overlay construction toolkit, *Computer Communications*, Vol. 31, No. 2, pp. 402–412 (2008).
16) 首藤一幸: Overlay Weaver: An Overlay Construction Toolkit, `http://overlayweaver.sourceforge.net/`.
17) 長尾洋也, 首藤一幸: オーバレイネットワークにおけるグループ間通信抑制手法, 情報処理学会研究報告，2010-OS-115 (2010).