Vol.2010-ARC-189 No.11
Vol.2010-OS-114 No.11
2010/4/22

柔軟な経路表によるオーバレイネットワークの設計

長尾洋也[†1]　首藤一幸[†2]

DHT アルゴリズムにおいて経路表を柔軟に管理する概念，柔軟な経路表を示す．柔軟な経路表は従来の DHT アルゴリズムの利点を保つ一方で，高い拡張性を提供する．また，サーバサイドシステムのような数ノードから数百ノードの小さなネットワークから，Peer-to-Peer のような百万ノード超の大規模ネットワークまでに対して一貫して利用可能な DHT アルゴリズムを提供する．一方で，本手法がその拡張性により分断されたネットワークやネットワーク近接性等を考慮するアルゴリズムの基礎になる可能性を示す．

柔軟な経路表を Chord に適用したアルゴリズム *FRT-Chord* を構成した．実験により，FRT-Chord がノード数の小さいネットワークにおいてゼロホップ DHT を実現し，ノード数が大きいネットワークにおいて $O(\log N)$ の経路長を実現していることを確認した．

Designing Overlay Networks with Flexible Routing Tables

Hiroya Nagao[†1] and Kazuyuki Shudo[†2]

In this paper, we propose *flexible routing tables*, a concept to flexibly maintain routing tables in DHT-algorithms. Flexible routing tables provide DHT-algorithms good expandability while keeping features of existing DHT-algorithms. Flexible routing tables also provide DHT-algorithms available in networks with from a few nodes in server-side systems to millions nodes in Peer-to-Peer networks. In addition, we show possibilities that the expandability forms the foundation for algorithms on a network-proximity and algorithms for segmented networks.

We construct a DHT-algorithm called *FRT-Chord* with the application of flexible routing tables to Chord which is a typical DHT-algorithm. Experiments on FRT-Chord demonstrated that FRT-Chord works as zero-hop-DHT with a small number of nodes and makes path lengths $O(\log N)$ with 10,000 nodes.

1. はじめに

ネットワークの性能向上と共にネットワークの利用方法が多様化し，その実現方法としてオーバレイネットワークが注目されている．オーバレイネットワークは，既存の IP ネットワーク上で新たに構築される論理的なネットワークである．オーバレイネットワークの例としては，Peer-to-Peer ネットワークや，アプリケーション層マルチキャスト（ALM），CDN などがあげられる．特に Peer-to-Peer ネットワークの構築に利用される Distributed Hash Table（DHT）は数百万台規模のオーバレイネットワーク上に連想配列を構築する技術であり，スケーラビリティや耐故障性，負荷分散などに優れている．

DHT のルーティングアルゴリズムは Chord[1)] や Kademlia[2)]，Tapestry[3)]，Pastry[4)] など，多数提案されてきた．これらのアルゴリズムの多くは，経路表サイズに対する経路長短縮を突き詰めた結果，経路表の構成が厳格に行われ，経路表の構成に関する制約が強い．この強い制約により効率的なルーティングが達成されているが，一方で，その制約に当てはまらないノード情報がむやみに捨ててしまう問題や，経路表構築時に ID 以外の情報を考慮しにくいといった問題がある．また，多くのアルゴリズムはノード数が大きい場合に対して最適化されている．そのため，ネットワーク上のノード数が小さい場合には，ノード数より最大経路表サイズを大きくとったとしても全ノードを経路表に保持しておくことが出来ない．これにより転送を避けられず，通信遅延が大きくなる問題が発生している．

本稿では，これらの問題を解決すべく “柔軟な経路表” という概念を提案する．柔軟な経路表では，ホップ数を小さく抑えるための経路表管理を経路表エントリ分布の管理ととらえ，従来手法における厳格な経路表の制約を緩和する．このような管理により，ノードの地理的な情報や通信遅延などの情報を経路表管理時に考慮しやすくなり，ルーティングにおける遅延を短縮出来る．また，数ノードの小規模ネットワークから数百万ノード超の大規模ネットワークまでをシームレスにかつ効率よく扱うことが可能である．本研究では，柔軟な経路表を既存の DHT アルゴリズム Chord に適用した FRT-Chord を用いて実験を行った．その結果，FRT-Chord がノード数の小さいネットワーク上では全ノードを経路表に保持す

†1 東京工業大学 理学部 情報科学科
Dept. of Information Science, Faculty of Science, Tokyo Institute of Technology
†2 東京工業大学 大学院情報理工学研究科 数理・計算科学専攻
Dept. of Mathematical and Computing Sciences, Graduate School of Information Science and Engineering, Tokyo Institute of Technology

Vol.2010-ARC-189 No.11
Vol.2010-OS-114 No.11
2010/4/22

るゼロホップ DHT になり，ノード数が大きいネットワーク上では Chord 同様に経路長がノード数 N の場合に $O(\log N)$ になることが確認できた．

2. 柔軟な経路表

2.1 柔軟な経路表の利点

経路表を厳格に管理することでルーティング効率を向上させる従来手法とは対照的に，柔軟な経路表は経路表管理の柔軟性とルーティング効率の両立を図る．それに伴い，次の利点が得られる．

- **ノード数に応じたゼロホップ DHT への移行**
  柔軟な経路表はノード数が大きいとき従来アルゴリズムと同等の経路長を実現するだけでなく，ノード数が小さいときにネットワーク上のノードすべてを経路表に保持するゼロホップ DHT となる．また，ゼロホップ DHT になる際にノード数などのパラメータの指定は不要であり，ノード数の減少に応じて，シームレスにゼロホップ DHT へ移行することが可能である．同様に，ノード数の増加に応じて従来アルゴリズム同様の経路長特性，つまり Chord における $O(\log N)$ の性質を示す．そしてそれらの中間域においても適切に経路表エントリが管理され，経路長を短く保つ．
- **ノード情報の有効利用**
  ノード情報とは，ノードの ID と IP アドレスのペアおよびノードに関するその他の付加的な情報である．柔軟な経路表は，このノード情報をむやみに捨てることを防ぐ．従来アルゴリズムでは，経路表の各スペースに保存されるエントリノードにそのノードが満たすべき ID の数値上の制約条件があり，その制限によってむやみにノード情報が捨てられてしまっていた．柔軟な経路表にはそのような制限が無く，経路表の総エントリ数が固定されるのみである．そのため，通信のたびに得たノードの情報を次々に経路表へ反映することが可能である．
- **高い拡張性**
  柔軟な経路表に従って構成された DHT アルゴリズムは，ネットワーク近接性や各ノードの負荷といった付加的な情報を柔軟に採り入れることが可能である．従来の DHT アルゴリズムに対してノードの地理的情報や通信遅延等の付加的な情報を考慮する拡張を行う研究が盛んになされているが[5]，そのような拡張は従来の DHT アルゴリズムでは考慮されていないことが多く，本来の良い性質を保とうとすると，拡張時に発生する制約が強い．しかし，柔軟な経路表ではその制約が緩和されているため，経路表管理手法を様々に拡張することが可能である．
- **フォワーディング先ノードを柔軟に選択可能**
  フォワーディングとはメッセージの転送のことである．また，メッセージの転送先ノードをフォワーディング先ノードと呼ぶ．柔軟な経路表では，経路表サイズを大きくすることで，フォワーディング先ノードとして採りうるエントリを複数保持できる．これを利用すると，複数のフォワーディング先ノードの候補から，ID 以外の付加的な情報，たとえばノードの地理的情報や通信遅延等（ネットワーク近接性）を考慮してフォワーディング先ノードを選択し，ルーティング処理全体の通信遅延を短縮することが可能となる．従来の DHT アルゴリズム，たとえば Chord では，フォワーディング先ノードは ID に従って一つに決定されてしまい，選択の余地が存在しない．
  柔軟な経路表は，このようなフォワーディング先ノードの決定時にネットワーク近接性を考慮する Proximity Route Selection（PRS）[6] だけでなく，経路表構成時にネットワーク近接性を考慮する Proximity Neighbor Selection（PNS）[6] も同様に行いやすく設計されているため，両者を組み合わせることでより自由度の高い拡張が可能である（5.2 節）．
- **最大経路表サイズの活用と動的な変更**
  既存の DHT アルゴリズムでは，経路表エントリ数がその場のノード数などの環境に依存したり（例：Chord），最大経路表サイズがネットワーク上のノード間で事前に決定されている（例：Kademlia）．しかし，柔軟な経路表にはそのような制限が存在しない．柔軟な経路表に基づく経路表管理は，各ノードに与えられた経路表の最大サイズを埋めるように働く．たとえば，Chord の finger table サイズは ID のビット長であるが，実際の運用において埋まることはまずない．そのため，finger table エントリがその最大数まで増加することを考慮しているのにも関わらず，実際に保持するエントリ数がきわめて少ないという状況が発生してしまう．これでは最大経路表サイズを十分に活かすことが出来ない．
  柔軟な経路表はメモリの容量等のノード情報を反映した動的な最大経路表サイズの変更が可能である．たとえば，Chord の finger table の最大エントリ数は ID のビット長に固定されており，変更することが出来ない．また，柔軟な経路表はネットワーク上の各ノードがそれぞれ互いに異なる経路表最大サイズを指定したとしても正しく動作する．そのため，それぞれのノードが独自に最大経路表サイズの変更を行うことも可能である．

Vol.2010-ARC-189 No.11
Vol.2010-OS-114 No.11
2010/4/22

2.2 柔軟な経路表の特徴

“柔軟な経路表” は，DHT アルゴリズムの構成方法である．柔軟な経路表は従来の DHT アルゴリズムに適用することが出来る．本稿では代表的な DHT アルゴリズムである Chord への適用結果を報告する．柔軟な経路表には，前節で述べた利点を得るための特徴が存在する．

- **目標分布に基づく経路表管理**

  柔軟な経路表は，従来アルゴリズムにおける詳細な経路表エントリ管理を経路表エントリの分布管理ととらえる．

  従来の DHT アルゴリズムに基づいて管理される経路表を分析すると，ノードが ID 空間上に密に分布しているとき，その密度関数が自ノードからの距離の逆数に比例する分布となっている場合がある（例：Chord，Kademlia）．自ノードから ID x までの距離を $d_s(x)$ とすると，密度関数 $f(x)$ は $f(x) \sim d_s(x)^{-1}$ である．

  しかし，実際にはノードは ID 空間上に密に分布していないため，単純にその分布に近づくように経路表管理を行うことは出来ない．そこで，柔軟な経路表では，経路表エントリの理想的な分布を目標分布と呼び，従来手法の経路表管理を分布に基づいて行うために，“目標分布が満たす条件” を次のように設定する．

  ノード ID 間距離の最大値を $d_{\max}$ とし，経路表に含まれるノード ID 集合を X とする．このとき，自ノードからの ID 的距離に関する分布関数 $F(x)$ を式（2）のように定義する．このとき，目標分布が満たす条件を式（3）と設定する．

$$f(x) = \begin{cases} 1 & (\exists e \in X, d_s(e) = x) \\ 0 & (\text{otherwise}) \end{cases} \tag{1}$$

$$F(x) = \frac{1}{|X|} \sum_{d=1}^{\lfloor x \rfloor} f(d) \quad (1 \leq x \leq d_{\max}) \tag{2}$$

$$F(2x) - F(x) = (\text{一定}) \tag{3}$$

  そして，柔軟な経路表では，経路表分布がこの条件を満たす目標分布に近づくように経路表を管理する．このように管理することで，従来手法のような経路長を維持したまま，経路表エントリに関する制限が緩和され，経路表管理に柔軟性を与える．

- **ルーティングアルゴリズムの枠組み**

  柔軟な経路表では，ルーティングアルゴリズムを経路表エントリの追加アルゴリズム・経路表エントリの削除アルゴリズム・フォワーディングアルゴリズムの 3 つの組ととらえる．それぞれのアルゴリズムは次の特徴を持つ．

  – **追加アルゴリズム**

    経路表へのエントリの追加方式を追加アルゴリズムと呼ぶ．経路表への経路表エントリの追加は，他のノード情報を入手したとき行われる．また，他のノードからノード情報を受け取り追加することも可能である．

    多くのアルゴリズムは，ノード ID が制約条件を満たし追加すべきかどうかを追加時に判断する．したがって，制約条件を満たさないノードは追加しない．それに対して柔軟な経路表は，そのような取捨選択を追加時には行わない．

  – **削除アルゴリズム**

    経路表からのエントリの削除方式を削除アルゴリズムと呼ぶ．経路表エントリの削除は，経路表の追加により経路表内のエントリ数がその最大サイズに到達した時点で行う．削除対象の経路表エントリは，経路表エントリの分布や，その他のネットワークやノードに関する情報を総合して決定する．この削除アルゴリズムにより，経路表エントリの分布が理想的な分布に近づくように管理する．

    また，削除アルゴリズムはノード ID だけに基づくのではなく，ノードの地理的情報や通信遅延，ネットワーク参加時間などに基づいた構成が可能である．このようにして高い拡張性が実現される．

    この枠組みの設定により，ノードが十分少ない場合に全ノードを経路表に保持することが可能となる．

  – **フォワーディングアルゴリズム**

    フォワーディング先ノードの決定は目的 ID に近いノードを選択するだけでなく，その他のネットワークやノードに関する情報を総合して決定することが可能である．削除アルゴリズムと連携させることで，複雑な機能を持つことも可能である．

  柔軟な経路表では，このようにアルゴリズムを構成することで，前述の利点を実現する．

3. FRT-Chord：柔軟な経路表の Chord への適用

FRT-Chord は，柔軟な経路表の概念に基づいて構成した DHT アルゴリズムである．FRT-Chord は，代表的な DHT アルゴリズムである Chord がベースである．また，FRT-Chord は，successor list と finger table という二つの経路表を区別せず，単一の経路表を用いる．FRT-Chord では，削除アルゴリズムおよびフォワーディングアルゴリズムにおいて ID 以外の情報，つまりネットワーク近接性等の情報を考慮していない．ただし，削除アルゴリズ

Vol.2010-ARC-189 No.11
Vol.2010-OS-114 No.11
2010/4/22

ムやフォワーディングアルゴリズムを変更することで，ID 以外の情報を組み込むことが可能である．

3.1 Chord

FRT-Chord は Chord をベースにしており，多くの部分が Chord と共通している．

柔軟な経路表の適用例のベースアルゴリズムとして Chord を選択したのにはいくつかの理由がある．まず先行研究が多く，経路長が高い確率で $O(\log N)$ になるなど，いくつかの事項に関して証明されていることがあげられる．また，DHT 実装に広く利用されており，利用実績が豊富であることがあげられる．そして，Chord の ID 空間や，フォワーディング方法がシンプルであり，柔軟な経路表を評価するためのアルゴリズムとして適切であると判断したためである．

3.2 追加アルゴリズム

FRT-Chord は，あらゆる通信をトリガとして経路表へ通信相手を追加する．具体的には，ルーティング時および Join 時の通信相手を経路表へ追加する．

3.3 削除アルゴリズム

FRT-Chord において，経路表エントリの削除は通信の失敗時およびエントリ追加時に経路表サイズ l がその最大サイズ L を越える場合に実行される．経路表サイズが大きくなりすぎたとき，経路表に残すエントリの分布が目標分布に近づくように削除する経路表エントリを選択する．ここで FRT-Chord では，経路表エントリ ID 間の間隔に注目する．この削除エントリの選択は，具体的に次のように実行される．

ノード ID のビット長を $m = 160$，経路表エントリのノード ID 集合を X とし，$l = |X|$ とおく．ここで，経路表エントリ ID を自ノードから近い経路表エントリから順に $x_0, x_1, ..., x_{l-1}$ とする．自ノードから ID x までの距離を $d_s(x)$ とするとき，正規化ノード間隔 C_i を式（4）のように定義する．

$$C_i = \bar{F}(d_s(x_i)) - \bar{F}(d_s(x_{i-1})) \tag{4}$$

$$\bar{F}(x) = \frac{1}{m}\log_2 x \tag{5}$$

FRT-Chord は，この正規化間隔 C_i が均一に近づくように削除エントリとして $x_j (C_j = min_i C_i)$ を選択する（**図 1**）．つまり，正規化間隔が最小となるエントリを取り除き拡大することで正規化間隔の均一化を図る．ここでは $\bar{F}(x)$ を式（5）のように定義しているが，これはノードが密に分布しているときの目標分布の分布関数を表している．正規化間隔の大小が変わらない範囲であれば $\bar{F}(x)$ は自由に変えて構わない．

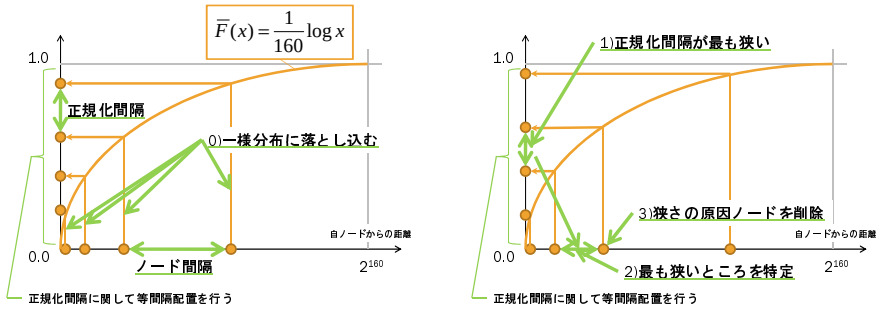

図 1 FRT-Chord における経路表からのノード削除アルゴリズム
Fig. 1 Node removal algorithm in FRT-Chord.

また，successor ノードとなる x_0 が取り除かれないよう，$C_0 = \infty$ とする．さらに，Chord の successor list による安定性を維持するために，経路表の先頭から k 個のエントリも同様に取り除かれないようにする．結果，C_i の最終的な定義は式（6）のようになる．

$$C_i = \begin{cases} \infty & (0 \le i < k) \\ \bar{F}(d_s(x_i)) - \bar{F}(d_s(x_{i-1})) & (\text{otherwise}) \end{cases} \tag{6}$$

経路表へのエントリの追加や削除が行われるとき，C_i の変更は局所的であり，また，C_i を昇順で管理することで，削除対象エントリの探索も効率よく行うことが可能である．従って，この削除アルゴリズムは時間効率よく実行されるといえる．つまり，このように経路表を管理したとしても管理コストのオーダーが大きく増すことはないといえる．

3.4 フォワーディングアルゴリズム

FRT-Chord では，目的 ID までの ID 的距離が最小になる経路表エントリノードをフォワーディング先ノードとして選択する．

3.5 Join 方式

Join 時には，他のルーティングアルゴリズムと同様に，自 ID の担当ノードを探索する．この探索中の通信相手はすべて経路表に追加される．自 ID の担当ノードの探索完了後，担当ノードの経路表エントリのうち，ノード近くのエントリを定数個取得し，経路表へ追加する．

Vol.2010-ARC-189 No.11
Vol.2010-OS-114 No.11
2010/4/22

4. 評　　価

4.1 実　　装

柔軟な経路表に基づく DHT アルゴリズム FRT-Chord をオーバレイ構築ツールキットである Overlay Weaver[7)8)] 上に実装した．

4.2 実 験 環 境

実験は，Overlay Weaver 上に実装した FRT-Chord アルゴリズムを利用し，1 台のマシン上におけるシミュレーションにより行う．各ノードの挙動管理は，Overlay Weaver 用のシナリオファイルにより行う．

- Overlay Weaver 0.9.7
- JRE build 1.6.0_17-b04
- OS：Windows Vista SP2 32bit
- CPU：Intel Core 2 Duo E8400 3.00GHz
- メインメモリ：4.00GB

4.3 経路表構成手法の検証

FRT-Chord の経路表管理アルゴリズムが式 3 に近づくように正しく作用しているかを次の実験を行い検証する．

4.3.1 実 験 内 容

ノード数 $N = 10000$ とし，経路表サイズ $L = 5, 10, 20, 30, 40, 50, 100$ それぞれに関して，ランダムに選択した 1 ノードからランダムに選択した ID への lookup を 10000 回実行する．実行後，ノードの経路表の分布を取得し，目標分布の満たすべき条件にどれだけ近づいているかを検討する．

4.3.2 実験結果と考察

それぞれの経路表サイズにおける経路表エントリの分布関数は**図 2** の通りである．この図においては，分布関数のうち，エントリノードが存在する部分に限定しプロットを行った．つまり，それぞれのプロットされた点が経路表エントリを示しており，左から順番に $x_0, x_1, ...$ のエントリの位置を示している．

このグラフにおいて，自ノードからの相対距離が小さい領域において，経路表エントリが疎になっており，その他の領域と比較して傾向が異なることが分かる．これは，ノードが密に分布しておらず，自ノードの近くに分布に従うノードが存在していないことによる．それぞれの経路表サイズにおいて，一番左および左から二番目のエントリノードが共通であるが，これは successor ノードおよび，その次のノードがどの経路表サイズに関しても経路表に載っていることを意味している．

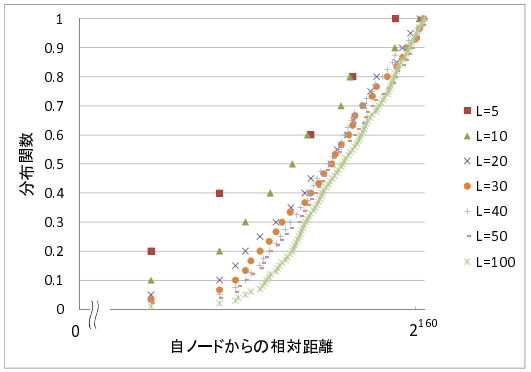

図 2　経路表エントリの分布関数（x 軸は log スケール）
Fig. 2　Distribution function of routing tables(log. scale).

このことを踏まえると，自ノードに近い部分に関しては，経路表エントリ間に別のノードが存在せず，直接担当ノードへ到達できるため，経路表エントリ間隔が広いことは問題にならない．

自ノードからある程度離れた部分に関しては，log スケールのグラフ上で，直線的に分布していると考えられる．つまり，式（3）に示した目標分布の条件に正しく近づいていると言える．

これらの結果から，FRT-Chord の経路表管理アルゴリズムが経路表エントリの分布を目標分布に近づけるように意図通り作用していることが確認できた．

4.4 ノード数と経路長の関係

FRT-Chord が従来アルゴリズムと同等の経路長を実現できていることを次の実験により検証する．

4.4.1 実 験 内 容

ノード数 N に対する経路長の変化を調べるため，N を 10，100，1000，10000 と変化させ，それぞれについて実験を行う．ここでは，FRT-Chord のベースとなった Chord（ID ビット長 $m = 160$）を比較対象とする．このとき，Chord の経路表サイズは 160 である．そ

Vol.2010-ARC-189 No.11
Vol.2010-OS-114 No.11
2010/4/22

表 1 ノード数と平均経路長の関係
Table 1 Correlation between number of nodes and average path lengths.

N	Chord	FRT-Chord($L = 20$)	FRT-Chord($L = 160$)
10	1.99	1.89	1.89
100	3.74	2.95	1.99
1000	5.72	4.41	3.00
10000	8.46	6.78	5.06

表 2 ノード数と最大経路長の関係
Table 2 Correlation between number of nodes and maximum path lengths.

N	Chord	FRT-Chord($L = 20$)	FRT-Chord($L = 160$)
10	3	2	2
100	7	5	2
1000	11	8	6
10000	17	14	11

のため，FRT-Chord も同様に $L = 160$ として実験を行う．また，比較対象として，$L = 20$ の場合の実験も行った．

4.4.2 実験結果と考察

実験結果は**表 1** および**表 2** である．また，それぞれを図示したものが**図 3** および**図 4** である．

図 3 および図 3 より，それぞれの N に関して，特に N の大きいときに関して，FRT-Chord が従来の Chord と比較して優れた経路長を示していることが分かる．ここで，Chord の経路長が $O(\log N)$ であることを考慮すると，FRT-Chord も同様に $O(\log N)$ であると考えられる．

図 4 より，FRT-Chord の最大経路長が $(N, L) = (10, 20), (10, 160), (100, 160)$ において 2 になっていることが分かる．この経路長 2 というのは，担当ノードの predecessor ノードへの到達に経路長 1，実際の担当ノードへの到達に経路長 1 の合計経路長 2 である．これは経路表にネットワーク上の全ノードが載っているからであり，最初のフォワーディングで担当ノードを特定できていることになる．また，この FRT-Chord において目的 ID の predecessor ノードを担当ノードとすると，ノードが少ない場合に担当ノードに 1 回のフォワーディングで到達可能となり，ゼロホップ DHT となる．

従来の Chord では，最大経路長が 3 以上となっていることが確認できる．これは，経路

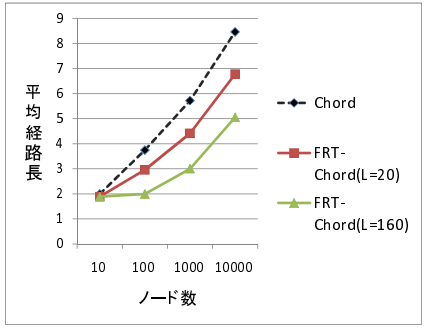

図 3 ノード数と平均経路長の関係
Fig. 3 Correlation between number of nodes and average path lengths.

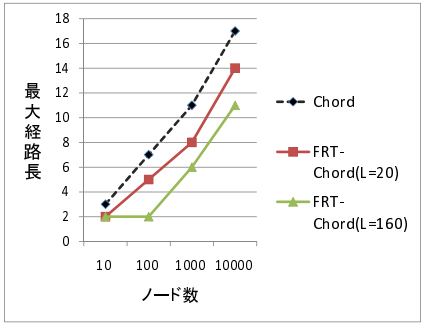

図 4 ノード数と最大経路長の関係
Fig. 4 Correlation between number of nodes and maximum path lengths.

表が十分なサイズ（160）を持つにも関わらず，経路表の構成手法が厳格であることによってノードが経路表サイズより小さいにもかかわらず，全ノードを経路表へ載せられていないことに起因する．つまり，ノード数が小さい場合に効率が悪くなっていることが確認できる．以上から，FRT-Chord はノード数が大きいとき Chord と同等の優れた経路長 $O(\log N)$ を実現している一方で，ノード数が少ない場合に Chord では達成できなかった経路表の全ノード保持が実現されていると言える．

5. 柔軟な経路表の応用可能性

我々は，柔軟な経路表が DHT の様々な可能性の基礎になると考える．

5.1 サーバサイドシステムと Peer-to-Peer の統合

従来の DHT アルゴリズムは，Peer-to-Peer に代表される，ノード数が数百万に達する大規模ネットワークに関して最適化されている．そのため，サーバサイドシステムに代表されるノード数の少ないネットワークで利用する際には，担当ノードの決定法のみを利用し，ルーティングに関しては全ノードの情報をあらかじめ知っている全く別の経路表管理が行われている．たとえば，代表的なサーバサイドシステムである Amazon の Dynamo[9] は分散 key-value store を構成する際に全ノードが全ノードを知っていることを前提とする方式をとっている．

本稿で提案した柔軟な経路表は，これらの対象ノード数の異なる全く別の経路表管理手法を統合する．これにより，従来手法では想定されていないノード数の大きな変化に対応可能

Vol.2010-ARC-189 No.11
Vol.2010-OS-114 No.11
2010/4/22

になるだけでなく，柔軟な経路表に基づき構成したアルゴリズムを利用したソフトウェアを小さいネットワークと大きなネットワークどちらであってもパラメータの変更なしに利用可能となる．我々は，これらの特徴を持つ柔軟な経路表が，幅広いノード数を対象としたソフトウェア開発を容易にするアルゴリズムとして利用されることを期待する．

5.2 様々なノード情報を考慮したルーティング

柔軟な経路表は，経路表構築時およびフォワーディング時に ID 以外の情報，たとえば通信遅延を柔軟に考慮することが出来る．文献 6) によれば，通信遅延に代表されるネットワーク近接性の考慮はそのタイミングに応じて，ID 割り当て時に近接性を考慮する Proximity Identifier Selection（PIS），経路表構成時に近接性を考慮する Proximity Neighbor Selection（PNS），フォワーディング先ノードの選択時に近接性を考慮する Proximity Route Selection（PRS）の 3 つに分類することが出来る．柔軟な経路表は，PNS と PRS の両方のタイミングで近接性の考慮を柔軟に行うことが可能であり，PNS のみや PRS のみでは出来ない近接性の考慮を行うことが可能である．

また，通信遅延だけではなく，ノードのネットワーク参加時間や，通信コスト，ノードのハードウェアなどを考慮したルーティングを行う DHT アルゴリズムの基盤として柔軟な経路表が適していると我々は考えている．

5.3 特定部分ネットワークにおける部分 DHT の構築

柔軟な経路表は，ID 空間上における経路表エントリの分布に注目して経路表管理を行う．ここで，ネットワークから特定の部分ネットワークを切り出すとする．これは，IP 的な切り分けではなく，任意の切り分けであり，特定の社内ネットワークや特定の IPS 内のネットワーク等が考えられる．

このとき，部分ネットワーク内における経路表の理想的な分布は，ネットワーク全体を考慮したときの経路表の理想的な分布と同一である．つまり，それぞれの部分ネットワークと全体のネットワークに共通する単一の共有経路表を目的分布に従って構築することが可能である．特に，それぞれの部分ネットワークに属するノードを経路表エントリへ優先的に載せることにより，特定部分ネットワークに特化した経路表を考える．このとき，特定部分ネットワークに特化させたとしても，全体のネットワークに関しても優れた経路表エントリ分布であることに注意されたい．

この手法を利用することで，全体のネットワークで構成された DHT の効率をそのままに，部分ネットワークに閉じたルーティングを実行したり，部分ネットワークで構築した部分 DHT を構築することが可能である．この部分 DHT の構築はさらなる応用可能性を含んでいる．

代表的な応用例として，社内に閉じた部分的な DHT の構築によるノードの使い分けや，同一 ISP に閉じたルーティングによる ISP をまたぐ通信の抑制などが考えられる．

5.4 将来の広域ネットワークでのルーティングへの適用

現在のインターネットに換わる新しいネットワークの形が検討されている．有線通信と移動体通信を統合する Fixed Mobile Convergence（FMC）でのルーティングを考えた場合，全対全で通信可能であるという前提を必要としないルーティングアルゴリズムが求められる．また，異なる通信網間の通信を最小限に抑える経路を利用するようにしなければならない．この複雑な判断が要求されるルーティングを実現するためには，経路表エントリやフォワーディング先ノードを様々な要素を考慮して選択する必要がある．我々は，そのルーティングに柔軟な経路表に基づく DHT アルゴリズムが利用できる可能性を検討している．

従来の DHT アルゴリズムは，ルーティングするための位置情報としてノード ID のみを割り当てていた．しかし，柔軟な経路表では，その他の情報を割り当てることで，ルーティングに特性を与えることが可能である．これを利用することで，従来の単純な全対全通信可能なネットワーク前提のアルゴリズムでは対応しきれない複雑な判断を行うことが出来る．

6. まとめと今後の課題

本論文では，柔軟な経路表を提案した．柔軟な経路表では，経路表の管理を経路表エントリ分布の管理ととらえ，経路表エントリ分布が目的分布に近づくように経路表エントリ管理を行う．

また，柔軟な経路表では DHT アルゴリズムを追加アルゴリズム・削除アルゴリズム・フォワーディングアルゴリズムに分割して考え，削除アルゴリズムにおいて，経路表エントリ分布が目的分布に近づくように削除対象エントリの選択を行う．このようにすることで，従来の DHT アルゴリズムの厳格な経路表管理の制限を緩和し，柔軟に管理可能となる．

実際に柔軟な経路表の概念を Chord に適用した FRT-Chord を Overlay Weaver 上に実装した．また，実験を行い，FRT-Chord の経路表管理が意図通りに動作していることを確認した．それに加え，ノードの多い場合には Chord と同等の経路長でのルーティングが行われていることを確認し，ノードが少ない場合には全ノードが経路表に載っていることを確認した．

今後，本稿で提案したアルゴリズムおよび諸性質の定量的評価を検討していく予定である．また，様々な既存の DHT アルゴリズムに対する柔軟な経路表の適用を行う予定である．

Vol.2010-ARC-189 No.11
Vol.2010-OS-114 No.11
2010/4/22

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, *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM, pp.149–160 (2001).
2) Maymounkov, P. and Mazières, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pp.53–65 (2002).
3) 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, pp.41–53 (2004).
4) Rowstron, A. I.T. and Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp.329–350 (2001).
5) Zhang, H., Goel, A. and Govindan, R.: Incrementally Improving Lookup Latency in Distributed Hash Table Systems, *SIGMETRICS Perform. Eval. Rev.*, Vol.31, No.1, pp.114–125 (2003).
6) 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 SIGCOMM 2003* (2003).
7) 首藤一幸：Overlay Weaver, `http://overlayweaver.sourceforge.net/`.
8) Shudo, K., Tanaka, Y. and Sekiguchi, S.: Overlay Weaver: An overlay construction toolkit, *Computer Communications*, Vol.31, No.2, pp.402–412 (2008).
9) DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. and Vogels, W.: Dynamo: Amazon's Highly Available Key-value Store, *Proc. SOSP 2007* (2007).