Vol.2011-DPS-147 No.23
Vol.2011-MBL-58 No.23
2011/6/3

構造化オーバレイにおける柔軟な経路表を活用したネットワーク近接性の考慮

宮尾武裕[†1]　長尾洋也[†1]　首藤一幸[†1]

Chord は経路表の管理をノードに割り当てられた ID によって厳密に行っているので，経路表に入るノードに制限がある．そのためルーティングの際に通信遅延を考慮することが難しい．しかし，Chord の改良版である FRT-Chord は経路表に入れるノードの ID による制限がないので，柔軟な経路表により経路表の管理を行うと，ネットワーク近接性を考慮できるように拡張することが容易である．本研究では通信遅延の閾値を設定し，その閾値より大きな通信遅延を持つノードを優先的に経路表から削除するように柔軟な経路表の削除アルゴリズムを拡張する手法を提案する．ある閾値で提案手法を用いた時に，Chord より平均遅延が約 42 % 改善され，提案手法の有効性を示すことができた．

Proximity-aware Structured Overlays with Flexible Routing Tables

Takehiro Miyao,[†1] Hiroya Nagao[†1]
and Kazuyuki Shudo[†1]

Existing DHT algorithms such as Chord prescribe which nodes are held in a routing table based on the nodes' ID. Because of it, network proximity can be incorporated to those algorithms under the ID-based restriction. In contrast to them, there is no such restriction in FRT-Chord, and it allows better consideration for network proximity. We incorporated network proximity into FRT-Chord and an experiment result showed that average routing latency was reduced to about 58% of Chord.

1. はじめに

現在 Gnutella や BitTorrent などのさまざまな P2P アプリケーションが存在する．P2P システムでは，ノード同士が対等に接続しネットワークを構築しているため，従来のクライアントサーバシステムに比べて特定のサーバに負荷が集中しにくい．そのため，P2P システムはスケーラビリティや耐故障性などの点で優れている．P2P ネットワークのように，物理的に接続された実ネットワーク上に独自のトポロジで構築されたアプリケーションレベルのネットワークをオーバレイネットワークと呼ぶ．オーバレイネットワークによって，非集中であっても多数のノードをうまく連携させることができる．

オーバレイネットワークは二つに分類することができる．一つは非構造化オーバレイネットワークであり，もう一つは構造化オーバレイネットワークである．非構造化オーバレイネットワークでは，オーバレイネットワーク構築時のルールが一定の数学的ルールに従わずにネットワークを構築されており，その技術は Gnutella や BitTorrent などのアプリケーションに用いられている．逆に，構造化オーバレイでは一定の数学的ルールに従いネットワークを構築する．

構造化オーバレイネットワークには Chord[1)]，Kademlia[2)]，Pastry[3)] などのアルゴリズムが存在し，近年多くの研究がなされている．これらのアルゴリズムは分散ハッシュテーブル（DHT）と呼ばれる．DHT とは連想配列を複数のノードで管理する技術である．非構造化オーバレイでは発見のためにフラッディング等が必要となるが，これらの DHT では，ノード数を N とすると $O(\log N)$ の経路長で効率のよいルーティングを行うことが可能である．大規模な P2P システムではノードが世界中に存在するため，一対一のノード間の通信遅延はさまざまなものとなる．しかし，従来の DHT の多くではハッシュ関数で割り当てられたノード ID によってルーティングを行うため，ネットワーク近接性が考慮されない．その結果ルーティングにおいて通信遅延が大きなノードと通信を行うことが頻繁に起こり，ルーティングにおける通信遅延を小さくすることができない．

ネットワーク近接性を考慮するためには，Proximity Neighbor Selection（PNS），Proximity Route Selection（PRS），Proximity Identifier Selection（PIS）[4)] といった方針が提案されてきた．PNS は経路表構築時に，PRS は経路選択時に，PIS はノード ID 決定時

†1 東京工業大学
Tokyo Institute of Technology

Vol.2011-DPS-147 No.23
Vol.2011-MBL-58 No.23
2011/6/3

にそれぞれネットワーク近接性を考慮する方法である．本研究では，PNS を用いてネットワーク近接性を考慮する．

Chord，Kademlia といった多くの DHT では，ノード ID のみによって経路表に追加するノードを厳密に選択しているので，PNS を用いてネットワーク近接性を考慮することが難しい．そこで，従来の DHT とは異なる柔軟な経路表を持つ FRT-Chord[5] に着目する．柔軟な経路表では，経路表にノード ID 以外の制限を加えることが可能であるという，容易な拡張性を持っている．そこで本研究ではネットワーク近接性を考慮するために，通信遅延の閾値を設定し，その閾値より大きな通信遅延を持つノードを優先的に経路表から削除することで，経路表にノード間の通信遅延による制限を加える柔軟な経路表の削除アルゴリズムを拡張する手法を提案する．

2. 関連研究

2.1 Chord

Chord[1] は DHT アルゴリズムの一つである．ハッシュ関数を利用して各ノードそれぞれに m ビットの ID を設定する．時計回りに ID の値が増加していくリング状の ID 空間に各ノードそれぞれを配置する．key にもハッシュ関数を利用して m ビットの ID を設定することで，リング状の ID 空間での位置が決まる．keyID から時計回りに進んで最初に出会うノードを key の担当ノードと呼び，担当ノードへ key と対応する value を保存する．

各ノードは successor list，predecessor，finger table と呼ばれる 3 種類の経路表を持っている．ノードがリング状の ID 空間を時計回りに進んで最初に出会ったノードを successor と呼び，successor list とはリング状の ID 空間を時計回りに進んで出会うノードのノード情報を順番に一定数保持する経路表である．リング状の ID 空間を反時計回りに進んで最初に出会ったのノードを predecessor と呼び，各ノードは predecessor のノード情報を保持する．finger table とはノードから 2^i（$i = 0, 1, ..., m-1$）の距離だけ離れた ID の担当ノードのノード情報を保持する経路表である．ID 間の距離とは，リング状の ID 空間における時計回りに測った長さとする．

Chord のルーティングでは，keyID の担当ノードへルーティングするために，経路表エントリ内で keyID への距離が最も小さいノードへフォワーディングを行う．フォワーディングを繰り返すことにより担当ノードの predecessor へ到達し，そのノードの successor へホップすることで keyID の担当ノードへ到達することができる．ルーティングにおけるフォワーディングの方法は iterative と recursive の 2 種類がある．iterative は，DHT に対してクエリを発行したノードがすべてのフォワーディング先ノードと直接通信を行うことを繰り返す方法である．recursive は，クエリを受け取ったノードが次ホップへクエリを転送していく再帰的な方法である．iterative では，フォワーディング先ノードの故障を検知したり，1 つのクエリを複数のノードに送ることでルーティングの遅延を短くすることができる．recursive では，フォワーディングの際のメッセージが片道なのでトラフィックが小さくなったり，ルーティングの遅延が短くなる．

Chord は，successor list と predecessor によってルーティングにおける到達性を保証し，finger table によって効率の良いルーティングを行うことを可能とする．これにより，Chord はノード数を N とすると，ルーティングの経路長が $O(\log N)$ となる．

2.2 FRT-Chord

Chord，Kademlia といった従来の DHT のようにノード ID の距離によって厳密に管理された経路表とは異なり，FRT-Chord[5] は柔軟な経路表を用いる．従来の DHT は経路表に載せるノードを ID に基づいて限定する．一方，柔軟な経路表には，ID によらず任意のノードを載せることができる．ただし，自ノードからの ID 距離の分布を調整することで経路長 $O(\log N)$ を達成する．この分布関数は既存 DHT にならって次の通りに定める．Chord，Kademlia では，m ビット ID 空間では経路表エントリの自ノードからの ID 距離の分布は，分布関数を $F(x)$ とすると $F(2x) - F(x) =$（一定）（x は自ノードからの距離）となっている．つまり，経路表エントリの自ノードからの ID 距離の分布は分布関数 $F(x) = \frac{1}{m} \log_2 x$ に従っている．FRT-Chord もこの分布関数を採用する．

柔軟な経路表での経路表の管理アルゴリズムは追加アルゴリズムと削除アルゴリズムで構成されている．追加アルゴリズムはオーバレイネットワークへの参加時やルーティング時など他ノードと通信を行ったときに手に入れたエントリ情報を経路表に追加する．そのため，経路表追加時にノード ID を考慮する必要がなく，容易に拡張が可能となる．削除アルゴリズムは次の通りである．大きさが L の経路表エントリ i のノードを n_i とし，その自ノードからの ID 距離を d_i（$d_0 = 0$，$d_{L+1} = 2^{160}$，$d_i < d_{i+1}$）とする．経路表エントリのうち successor や predecessor などを除いた削除対象候補の中で，$\log_2 d_{k+1} - log_2 d_{k-1}$ が最小となるノード n_k を経路表から削除する．このアルゴリズムにより，経路表に入っているノードの自ノードからの ID 距離の分布を目的の分布関数 $F(x) = \frac{1}{m} \log_2 x$ に近づけている．

柔軟な経路表にはノード情報の有効利用や容易な拡張性，ノード数の増減に対応可能といった優れた特徴がある．

Vol.2011-DPS-147 No.23
Vol.2011-MBL-58 No.23
2011/6/3

ノード情報の有効利用　従来の DHT では，経路表の管理を ID により厳密に行っていたので ID が一定の条件を満たすノード情報しか経路表に入れることができなかった．しかし，経路表を柔軟に管理することで，入手したノード情報をむやみに捨てることを防ぐことが可能である．

容易な拡張性　経路表に載せるノードに制約がないので，ネットワーク近接性等の ID 以外の要素に基づいて経路表に載せるノードを決めることができる．提案手法はこの特長を活用している．

ノード数の増減に対応可能　経路表の大きさよりノード数が少ない場合，全ノードを経路表に入れることができる．また，経路表の大きさよりノード数が多い場合，$O(\log N)$ の経路長であるマルチホップルーティングを行う．2 つの状態をパラメータを変更することなく連続的に推移させることができる．

3. LPRS-Chord

LPRS-Chord[6] は PNS によりネットワーク近接性を考慮した DHT の一つである．

Chord では，finger table において自ノードからの ID 距離が 2^i（$i = 0, 1, ..., m-1$）である ID の担当ノードのノード情報を保持していた．しかし，LPRS-Chord では他のノードと通信を行った際に自ノードとの通信遅延を測定し，自ノードからの ID 距離が $[2^i, 2^{i+1})$ であるノードの中で自ノードとの通信遅延がより小さいノードを経路表に保持するように，経路表を更新していく．これにより，ルーティングにおける経路長が $O(\log N)$ を維持しつつ，経路表に入っているノードの自ノードとの通信遅延が小さくなるので，ルーティングにおける平均遅延が小さくなる．

4. 提 案 手 法

提案手法の目的は、経路長を長くせずに，自ノードとの通信遅延の小さいノードのみが載った経路表を構築することである．その目的を達成するために，提案手法では柔軟な経路表の削除アルゴリズムを拡張した．拡張した削除アルゴリズムは次の通りである．

(1) 経路表エントリの内、通信遅延が事前に設定した閾値より大きいノードを削除対象候補として、FRT-Chord の削除アルゴリズムを実行

(2) 1 において削除対象候補が存在しないとき，経路表エントリのすべてのノードを削除対象候補として、FRT-Chord の削除アルゴリズムを実行

拡張した削除アルゴリズムにより，自ノードとの通信遅延が閾値以下のノードのみが載った経路表を構築することができる．しかし，通信遅延に基づいてノード情報を選別するため，経路表エントリの自ノードとの ID 距離の分布が目的の分布関数 $F(x) = \frac{1}{m}\log_2 x$ に近づけなくなることが考えられる．目的の分布関数に近づけていない経路表では，経路長が長くなり，それが理由でルーティングの遅延が大きくなりかねない．そこで，実験により、閾値と、経路長およびルーティング遅延の関係を調べる。

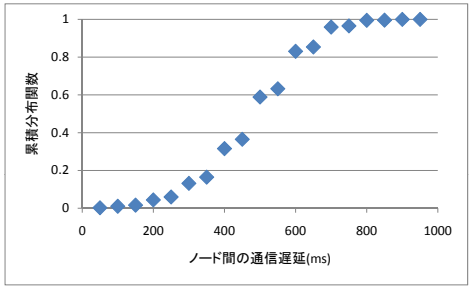

図 1　ノード間の通信遅延の分布
Fig. 1　CDF of communication latency

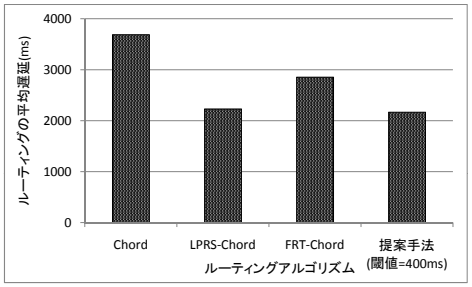

図 2　各アルゴリズムのルーティングの平均遅延
Fig. 2　Average routing latency of Chord, LPRS-Chord, FRT-Chord and our method

5. 評　　価

マシン 1 台上でエミュレーションによる実験を行った結果を示す．

5.1 実 験 内 容

Transit-Stub モデル（TS モデル）[7] により実ネットワークのトポロジをシミュレートし，ノード間の通信遅延を設定した．TS モデルはトランジットノードとスタブドノードの 2 種類のノードで構成されている．トランジットノード間，トランジットスタブノード間，スタブノード間の遅延をそれぞれ 100 ms，20 ms，5 ms と設定した．その結果，ノード間の通信遅延の分布は図 **1** となった．最大値は 1000 ms であり，平均値は 470 ms である．

オーバーレイ構築ツールキットである Overlay Weaver[8] 上に提案手法により提案手法を実装し，一台のマシン上でエミュレーションによる実験・評価を行った．ノード数は 10000，経路表の大きさを 16 とし，経路表が構築するための十分な回数の通信を行った後，10000 回のクエリを実行し，その際の経路長とルーティングの遅延を測定した．実験環境は以下の通りである．

- Overlay Weaver 0.10.1

Vol.2011-DPS-147 No.23
Vol.2011-MBL-58 No.23
2011/6/3

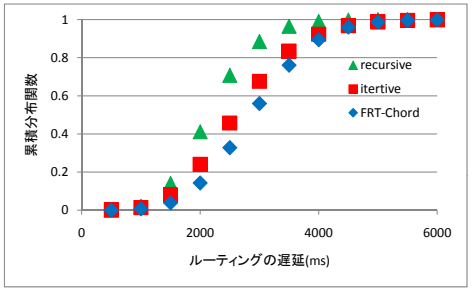

図 3　iterative，recursive のルーティングに使用したノードの分布
Fig. 3　CDF of communication latency in iterative and recursive lookup

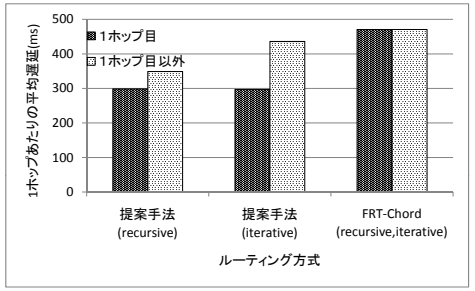

図 4　iterative，recursive の 1 ホップ目および 2 ホップ目以降の 1 ホップあたりの平均遅延
Fig. 4　Average latency per hop in every hop

- OS: Windows 7 Professional 32 bit
- CPU: Intel 2.67 GHz Core 2 Quad Q9400
- メモリ: 4 GB

5.2　実 験 結 果

5.2.1　提案手法の有効性

図 **2** は，Chord および LPRS-Chord，FRT-Chord，提案手法のアルゴリズムそれぞれにおけるルーティングの平均遅延を示している．ノード数が 10000 なので，Chord および LPRS-Chord の finger table に入っているノード数は約 13 である．さらにこれらのアルゴリズムの経路表は finger table の他に successor list と predecessor を持っているので，経路表の大きさが 16 の FRT-Chord とほぼ同じ数のノードを経路表に保持しているといえる．そのため，これら 4 種のアルゴリズムのルーティングの結果を比較することが可能である．また，提案手法では、閾値 400 ms において最も良い結果が得られたので、この閾値での結果を採用した。

提案手法によって、ルーティングの遅延が Chord より約 42 %，FRT-Chord より約 25 % 減少した．これにより提案手法の有効性を示すことができた．また，提案手法は LPRS-Chord の遅延とほぼ同じであった．つまり，LPRS-Chord はネットワーク近接性を考慮したアルゴリズムなので提案手法も充分ネットワーク近接性を考慮することができているといえる．さらに，提案手法は柔軟な経路表を用いているので，経路表の大きさの変更可能などの柔軟な経路表の優れた特長を持っている．

5.2.2　recursive と iterative の比較

図 **3** は，ルーティングにおけるフォワーディング方法が iterative および recursive であるときの提案手法と FRT-Chord それぞれのルーティングに使用したノードの自ノードとの通信遅延の分布を示している．ただし，FRT-Chord ではルーティングに使用するノードは自ノードとの通信遅延が考慮されていないので，recursive および iterative のルーティングに使用したノードの自ノードとの通信遅延の分布は同じである．

提案手法における recursive および iterative のルーティングの結果を比較する．図 3 より，recursive のときのほうが iterative に比べてルーティングに使用したノードの自ノードとの通信遅延が小さいことがわかる．なぜなら，提案手法では自ノードとの通信遅延が閾値 400 ms 以下であるノードを用いて経路表を構築しており，recursive はルーティング時に自ノードの経路表に入っている通信遅延の小さいノードと通信を行うからである．しかし，iterative はクエリ発行元ノードがフォワーディング先ノードと通信を行うため，最初の 1 ホップ目を除き自ノードの経路表に入っている通信遅延の小さいノードと通信を行わないからである．よって，提案手法では iterative より recursive でルーティングを行うほうが効率よくルーティングを行うことができる．

次に iterateive であるときの提案手法および FRT-Chord のルーティングの結果を比較する．図 3 より，提案手法のほうが FRT-Chord に比べてルーティングに使用したノードの自ノードとの通信遅延が小さいことがわかる．ここで、iterative では最初の 1 ホップ目だけが自ノードの経路表に入っている通信遅延の小さいノードと通信を行うと考えていたが，2 ホップ目以降との通信遅延も短縮された。図 **4** にそれが表れている．

図 4 はそれぞれのルーティングにおいて 1 ホップ目および 2 ホップ目以降の 1 ホップあたりの平均遅延を示している．提案手法の iterative では，recursive と 1 ホップ目の遅延が同じであることがわかる．また，FRT-Chord では経路表構築時に遅延を考慮していないため，1 ホップ目と 2 ホップ目以降での 1 ホップあたりの平均遅延は同じである．提案手法の iterative および FRT-Chord の 2 ホップ目以降の 1 ホップあたりの平均遅延を比較する．提案手法のほうが 2 ホップ目以降での 1 ホップあたりの平均遅延が小さいことがわかる．つまり，提案手法においてルーティング方式が iterative であっても，2 ホップ目以降のルーティング効率がよくなっていることがわかる．これは，実ネットワークにおいて自ノードと近いノードと近いノードは，自ノードと近いからである．たとえば，ノード A がノード B 経由してノード C にルーティングを行ったとする．このときノード A の経路表にはノード B が，ノード B の経路表にはノード C が入っていることから，ノード AB 間およ

Vol.2011-DPS-147 No.23
Vol.2011-MBL-58 No.23
2011/6/3

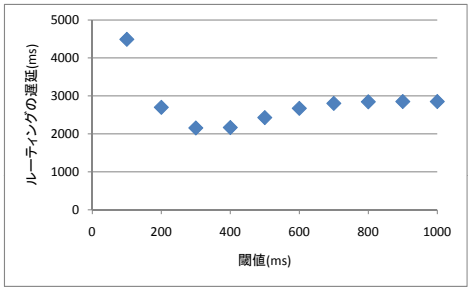

図 5　各閾値でのルーティングの平均遅延
Fig. 5　Average routing latency

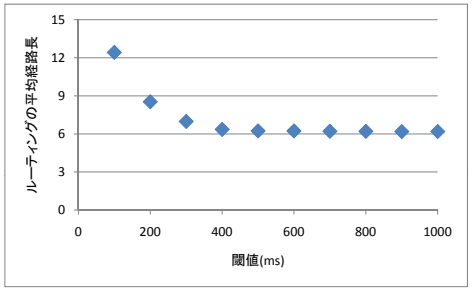

図 6　各閾値でのルーティングの平均経路長
Fig. 6　Average route length

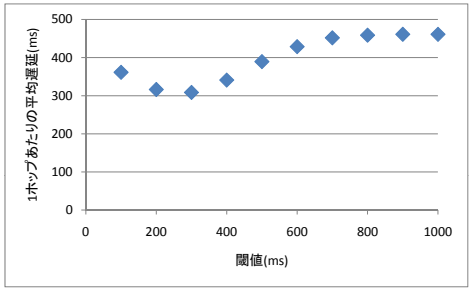

図 7　各閾値での 1 ホップあたりの平均遅延
Fig. 7　Average latency per hop

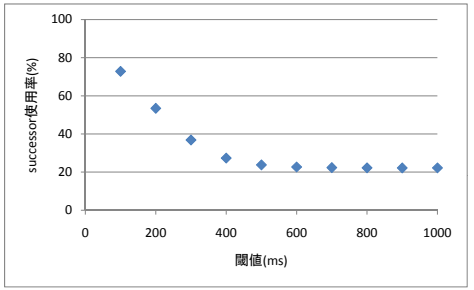

図 8　ルーティングで successor を使用したホップの割合
Fig. 8　Rate of successor is followed in a routing

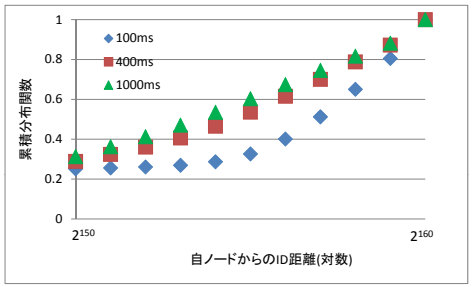

図 9　経路表エントリの ID 距離の分布
Fig. 9　CDF of ID distance in a routing table

びノード BC 間が実ネットワーク上で地理的に近いといえる．このとき，ノード A にとってノード C は他にランダムに選んだノード D 比べると実ネットワーク上で近いといえる．

5.2.3　各閾値での経路長とルーティング遅延

図 5 および**図 6** は，閾値を 100 ms から 1000 ms まで 100 ms 毎に変化させたときの，ルーティングの平均遅延と平均経路長を示している．閾値が 300 ms および 400 ms のとき，ルーティングの遅延が最小になっていることがわかる．閾値が 400 ms 以上のとき，閾値が増加しても経路長がほぼ一定であり，閾値を小さくすることで経路表エントリの各ノードと自ノードとの通信遅延が小さくなるので，ルーティングの遅延が小さくなる．次に，閾値が 400 ms 以下では閾値が小さくなるにつれて経路長が増加している．そのため，閾値 100 ms，200 ms では遅延が大きくなっている．しかし，閾値 300 ms では経路長が増加しているにも関わらず，遅延が増加していない．これは経路長が増加したが，閾値が小さくなったことで経路表エントリの自ノードとの通信遅延が減少したからであると考えられる．

図 7 は各閾値においてルーティングの 1 ホップあたりの平均遅延を示している．閾値 300 ms 以上において閾値が減少するごとに 1 ホップあたりの遅延が減少しているのがわかる．閾値が小さくなるにつれて経路表エントリの自ノードとの通信遅延が小さくなるからである．また，閾値が 300 ms より小さいとき，閾値が減少するごとに 1 ホップあたりの遅延が増加している．この理由は**図 8** から説明できる．

図 8 は各閾値においてルーティングで successor を使用したホップの割合を示している．提案手法において successor は遅延を考慮せずに例外として経路表に維持しているノードなので，図 1 の分布のノードであり，ノード間の通信遅延の平均が 470 ms である．閾値 300 ms より小さいとき，閾値が小さくなるにつれてルーティングで successor を使用する割合が増加している．つまり，通信遅延の平均が 470 ms であるノードと通信をする割合が増加し，反対に通信遅延が閾値以下のノードと通信をする割合が減少する．その結果，ルーティングの 1 ホップあたりの平均遅延が増加した．

閾値が適切な値をとるとき，経路長が長くならずに 1 ホップあたりの遅延が小さくなり，ルーティングの遅延が減少する．しかし，閾値が適切な値より小さいとき，経路長が長くなるだけでなくルーティングの successor の使用率が増加することで，ルーティングの遅延が大きくなる．

5.2.4　経路表エントリの分布

2.2 節より FRT-Chord は経路表エントリの自ノードからの ID 距離の分布を分布関数 $F(x) = \frac{1}{m}\log_2 x$ に従わせようとする．本実験では 160 ビットの ID 空間を用いているので，$m = 160$ である．この節では，閾値が 400 ms 以上のとき経路表エントリの自ノードか

Vol.2011-DPS-147 No.23
Vol.2011-MBL-58 No.23
2011/6/3

らの ID 距離の分布が目的の分布関数 $F(x) = \frac{1}{160}\log_2 x$ に近づき，閾値が 400 ms 未満のときは目的の分布に近づけていないことを確認する．図 **9** は，閾値 100 ms，400 ms，1000 ms において経路表エントリの自ノードからの ID 距離の分布を示している．ただし，横軸は 2^{150} から 2^{160} までの距離の対数をとったものである．

閾値 400 ms および 1000 ms において近似を行った結果，それぞれの近似式は $y = 16035\log_2 x - 2\times10^6$，$y = 15560\log_2 x + 2\times10^6$ となり，それぞれの R^2 誤差は $R^2 = 0.972$，$R^2 = 0.9992$ となった．この近似式は目的の分布関数 $F(x) = \frac{1}{160}\log_2 x$ と比べて，$\log_2 x$ の係数や y 切片の値が大きく異なる．これは，実験に使用したノード数が 10000 であったために，ノード間の ID 距離が大きくなり，自ノードからの ID 距離が小さいノードが存在しなかった．その結果，目的の分布関数 $F(x) = \frac{1}{160}\log_2 x$ に近付けなかったと考えられる．しかし，閾値 400 ms，1000 ms のときの R^2 誤差は $R^2 = 0.972$，$R^2 = 0.9992$ であることから，もともとの目的である $F(2x) - F(x) =$（一定）つまり $F(x) = \log_2 x$ に比例するという目的に近付けていることがわかる．

閾値 100 ms において近似を行った結果，近似式は $y = 16170\log_2 x + 2\times10^6$ となり，R^2 誤差は $R = 0.816$ となった．閾値 100 ms の R^2 誤差は閾値 400 ms，1000 ms に比べると小さい値となっている．よって，閾値 100 ms では目的の分布に近付けていない．しかし，経路表エントリにおいて自ノードからの ID 距離が 2^{157} 以上のノードの分布の近似式は，$y = 34403\log_2 x + 4\times10^6$ であり，R^2 誤差は $R = 0.989$ となった．このことから閾値 100 ms でも，経路表の自ノードからの ID 距離が大きいエントリにおいて目的の分布に近付けていることがわかる．つまり，閾値を小さくしても自ノードとの ID 距離の小さいノードを経路表に入るようアルゴリズムを改良することで，ルーティングの遅延が減少すると考えられる．

6. まとめ・今後の課題

提案手法を用いることで，Chord よりルーティングの遅延が約 42 % 改善された．また，ネットワーク近接性を考慮するために Chord を改良した LPRS-Chord とルーティングの遅延が同程度であるが，提案手法は柔軟な経路表の特長を持っているため，優れているといえる．

閾値が適切な値より小さいとき，ルーティングの経路長が長くなり，successor の使用頻度が増加し、結果としてルーティングの遅延が増加した．しかし，閾値が適切な値のとき，ルーティングの経路長が長くならず，1 ホップあたりの平均遅延が増加しないので，ルーティングの遅延が減少した．

ルーティングにおけるフォワーディングの方法は，recursive のほうが iterative に比べて効率よくルーティングを行うことができる．しかし，iterative であっても FRT-Chord より提案手法のほうがルーティングの遅延が小さくすることができる．

適切な値より小さな閾値を用いた時，自ノードからの ID 距離が小さいところで経路表の分布が目的の分布関数に近付けていないが，ID 距離が大きいところでは近付けていた．そのため，提案手法はさらに改良することが可能であると考えている．また，ノードが知ることのできる情報から適切な閾値を計算する方法を考える必要がある．

謝辞　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, F. and Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *Proc. SIGCOMM 2001* (2001).
2) Maymounkov, P. and Mazieres, D.: Kademlia: A Peer-to-peer Information Systems Based on the XOR Metric, *Proc. IPTPS 2002* (2002).
3) Rowstron, A. and Druschel, P.: Pastry:Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, *Proc. Middleware 2001* (2001).
4) Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S. and Stoica, I.: The Impact of DHT Routing Geometry on Resilience and Proximity, *Proc. SIGCOMM 2003* (2003).
5) 長尾洋也，首藤一幸：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，*Proc. SACSIS 2011* (2011)(採択決定).
6) Zhang, H., Goel, A. and Govindan, R.: Incrementally Improving Lookup Latency in Distributed Hash Table Systems, *Proc. SIGMETRICS 2003* (2003).
7) W.Zegura, E., L.Calvert, K. and Bhattacarjee, S.: How to Model an Internetwork, *Proc. INFOCOM 1996* (1996).
8) 首藤 一幸：Overlay Weaver, http://overlayweaver.sourceforge.net/.