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[†] 長尾洋也[†] 首藤一幸[†]

1. はじめに

分散ハッシュテーブル（DHT）を実現するためのアルゴリズムの例として Chord[1] があげられる．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．そのためネットワーク近接性を考慮することが難しい．しかし，柔軟な経路表[2] は経路表に入るノードの 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．そこで本研究では，通信遅延の閾値を設定し，その閾値より通信遅延が大きな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．

2. FRT-Chord

ここでは Chord に柔軟な経路表を持たせるように改良した FRT-Chord について説明する．FRT-Chord の経路表では，自ノードからの ID 距離の分布が分布関数 $\log_2 x$（x は自ノードからの ID 距離）に比例させることを目的としている．経路表の管理は削除アルゴリズムにより行っている．削除アルゴリズムは次の通りである．大きさが l の経路表においてエントリ i のノードを n_i とし，そのノードの自ノードからの ID 距離を d_i（$d_0 = 0$，$d_{l+1} = 1$，$d_i < d_{i+1}$）とする．$\log_2 d_{k+1} - log_2 d_{k-1}$ が最小となるノード n_k を経路表から削除する．ただし，successor や predecessor などは削除しない．

3. 提案手法

提案手法の考え方は，自ノードとの通信遅延の小さいノードのみを使用した経路表を構築することでネットワーク近接性を考慮し，そして柔軟な経路表による経路表の管理を行うことで，ルーティングの経路長が長くならないようにすることである．そのために，提案手法では柔軟な経路表の削除アルゴリズムの拡張を行っている．拡張手法は次の通りである．まず自ノードとの通信遅延の閾値を設定する．そして，その閾値より大きな通信遅延を持つノードが経路表内に存在するとき，それらのノードを対象に柔軟な経路表の削除アルゴリズムを実行し，経路表からノードを削除する．それらのノードが存在しないときは，経路表内のすべてのノードを対象に削除アルゴリズムを実行する．

4. 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．

4.1 実験環境

TS モデル[3] 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最大値が 1000 ms となり，平均が 470 ms となった．

オーバーレイ構築ツールキットである Overlay Weaver[4] 上に提案手法により拡張した FRT-Chord 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，そのルーティングの結果を測定した．実験マシンは以下の通りである．

- Overlay Weaver 0.10.1
- OS: Windows 7 Professional 32 bit
- CPU: Intel 2.67 GHz Core 2 Quad Q9400
- メモリ: 4 GB

† 東京工業大学

4.2 実験結果

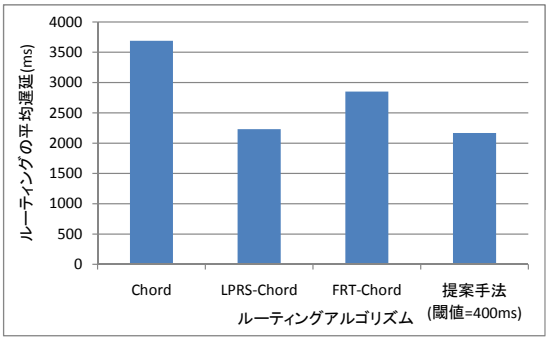

図 1 各アルゴリズムのルーティングの平均遅延

図 1 は，Chord ，LPRS-Chord[5] ，FRT-Chord ，提案手法，それぞれのアルゴリズムにおけるルーティングの平均遅延を示している．LPRS-Chord とは，Chord においてネットワーク近接性を考慮するように改良したアルゴリズムである．ノード数が 10000 なので，Chord ，LPRS-Chord の finger table に入っているノード数は約 13 である．さらに Chord ，LPRS-Chord の経路表は finger table の他に successor 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，提案手法において閾値 400 ms の結果は，今回の実験環境で最も優れた閾値であるので，他のアルゴリズムと比較するためにこの閾値の結果を用いた．

提案手法は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せた．また，提案手法は LPRS-Chord の遅延とほぼ同じ大きさであった．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．

図 2 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 と 400 ms において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 以上のとき経路長がほぼ一定であるので，閾値を小さくすることで経路表内のノードと自ノードとの通信遅延が小さくなり，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内のノード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．

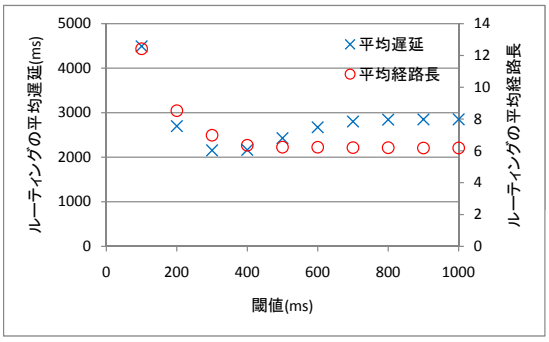

図 2 各閾値でのルーティングの平均遅延

5. まとめ・今後の課題

提案手法により Chord よりルーティングの遅延が約 42 %改善された．適切な閾値を用いることで，経路長を増加させずに経路表に入っているノードの自ノードからの通信遅延を小さくすることができた．

提案手法は適切な閾値がとても重要である．今後は，ノードが知ることのできる情報だけで適切な閾値の計算方法を考える．

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) 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).
3) W.Zegura, E., L.Calvert, K. and Bhattacarjee, S.: How to Model an Internetwork, *Proc. INFOCOM 1996* (1996).
4) 首藤 一幸: Overlay Weaver, http://overlayweaver.sourceforge.net/.
5) Zhang, H., Goel, A. and Govindan, R.: Incrementally Improving Lookup Latency in Distributed Hash Table Systems, *Proc. SIGMETRICS 2003* (2003).