DHT ネットワーク上の ID 変更による検索時間削減方式の提案

嶋野裕太[†]　　佐藤文明[††]

現在Ｐ２Ｐの検索方式として分散ハッシュテーブル(DHT)を利用した手法が盛んに研究開発されている。DHT の一つである Chord は、ノードを論理的なリング構造に構成し、リング状に検索要求が転送される。効率的な検索をするためには、検索を始めてから結果が返ってくるまでの時間を短くする必要がある。しかし、Chord のオーバーレイネットワークは、実ネットワークの状況を考慮せず構築されるため無駄の多い検索が含まれる問題点がある。本研究では、実ネットワークの状況をオーバーレイネットワークに反映させることで、検索要求の転送時間を削減する方式の提案をする。本提案方式では、検索要求もしくはフィンガーテーブルを用い遅延時間の計測を行い、それを元に Chord リングのノードを動的に再配置する。三つの提案方式のシミュレーションを行い、検索要求の転送時間を削減することを示した。

Lookup Performance Improvement Method of DHT by Dynamic Relocation of Node ID

YUTA SHIMANO[†]　FUMIAKI SATO[††]

Information lookup systems using Distributed Hash Table (DHT) have been actively studied. In the Chord DHT, nodes are configured as a ring structure and lookup requests are forwarded along with the ring. In order to get information efficiently, response time should be short. However, since Chord overlay network is constructed without considering the real network, the efficient lookup cannot be done. In this paper, we propose a method to reduce the response time by the rearrangement of the node ID of the Chord by the latency. From the simulation results, it is shown that the response time of the proposed method reduced

1. はじめに

近年、サーバを用いずエンドユーザ同士で検索・データ通信を行う Ｐ２Ｐ（Peer to Peer）が注目されている。Ｐ２Ｐは非構造化オーバーレイと構造化オーバーレイ[1]の二つに大別される。オーバーレイネットワークとは仮想的に個々のコンピューター同士が直接つながりネットワークを形成する。オーバーレイネットワーク上のノードは下位ネットワークのトポロジーを意識せずに通信することができる。

非構造化オーバーレイではオーバーレイネットワーク構成にルールが存在しないのでオーバーレイネットワークが自由に構成される。柔軟な検索が可能であるものの、存在するコンテンツが発見できない場合もあるので検索の効率は悪いという特徴を持つ。また、各ノードへ検索要求がネットワーク上に接続された送信可能なすべての端末に対して、送信される（フラッディングと呼ばれる）ため帯域を圧迫してしまう。一方、構造化オーバーレイとはオーバーレイネットワークがルールに従って構築される。代表的なものに DHT（Distributed Hash Table：分散ハッシュテーブル）があり、盛んに研究開発されている。DHT の代表的なアルゴリズムとして、CAN[2]、Chord[3]、Pastry[4]、Tapestry[5]、Kademlia[6]がある。完全一致探索しかできないものの、検索の効率がよい。また、非構造化オーバーレイに比べ、遥かに大量のピアがネットワークに参加することが可能となっている。ただし、その複雑さゆえ実装が難しいというデメリットもある。DHT はコンテンツやノードを識別するのにハッシュ値を用い、そのハッシュ値を一定の範囲から受け持つノードを特定し、検索の効率化を図っている。

しかし、DHT を利用したオーバーレイネットワークは、実ネットワークの状況を考慮されて構築されず、実ネットワークでのノード間の近さとオーバーレイネットワーク上でのノード間の近さは無関係になっている。結果、オーバーレイネットワーク上では無駄のない検索ホップに見えたとしても、実ネットワークでは効率の悪い検索ホップになっている可能性が含まれている。従来研究ではネットワークアドレスを利用した ID 生成法や、ランドマークノードを利用し ID を生成するなど、ID 生成時にネットワーク状況を考慮し ID を生成する。これらは実ネットワークの状況を考慮したオーバーレイネットワークの構築が行われる。だが近年、スマートフォンや WiMAX の利用者が増加したことにより移動端末が増えてきている。そのためネットワークの状況が時間と共に変化していく。従来研究ではネットワーク状況の変化には対応していない。

本研究では、DHT アルゴリズム Chord を基礎とし、一度決定したノード ID を動的

[†]東邦大学大学院理学研究科
Toho University, Graduate School of Science.
[††]東邦大学理学部情報科学科
Toho University, Faculty of Science, Department of Information Science.

に交換していくことで移動端末による実ネットワークの変化をオーバーレイネットワーク上に反映させる手法を提案する。

本論文における構成を以下に示す。第２章では、本研究の研究背景として、DHT アルゴリズム Chord の概要について説明する。第３章では提案方式について説明する。第４章では提案方式におけるシミュレーションを通して考察を行う。最後に第５章では結論を述べる。

2. DHT アルゴリズム Chord

2.1 Chord

代表的な DHT アルゴリズムの一つ Chord では SHA-１などのハッシュ関数を用いて自分のノードの ID を生成し、自ノードの ID と他ノードの ID をもとに管理領域を決定する。そのため完全に分散して動作することが可能となっている。従来の Pure P２P では検索クエリをフラッディングするため、検索可能ノード数 N の増加に伴い線形のパケット増加が起こってしまうが、Chord ではパケット数ホップ数共にO(logN)に抑えることができ、規模拡張性に優れている。

ハッシュ関数 SHA-１を用いる場合 160bit の空間に各ノード ID（ノード ID＝hash（IP address））及び各キーID（キーID＝hash（key））をマッピングする。160bit の ID 空間の場合、3.4×10^{38} の ID が存在することになり、コンテンツと IP アドレスをマッピングしても衝突しないことが前提とされる。ここでは説明しやすいように 5bit（ID＝0～31）の ID 空間を用いて具体的な説明をする。

Chord におけるオーバーレイネットワークは、ID をもった各ノードは仮想的なリングを形成し、自ノードの ID から時計回りに見て最初に存在するノードを successor ノードとして記憶する。また、反時計回りに最初に存在するノードを predecessor ノードとして記憶し、predecessor ノードと自ノードの間の ID 空間を自分の責任領域として管理を行う。また successor ノード、predecessor ノードの各ノードとの接続はルータ同士での接続ではなく、あくまでオーバーレイネットワーク上での接続である。そのためノード間には複数のルータが存在し、実ネットワークの近さは考慮されていない。図 2.1 で各ノードが管理する Key ID の一例を示す。

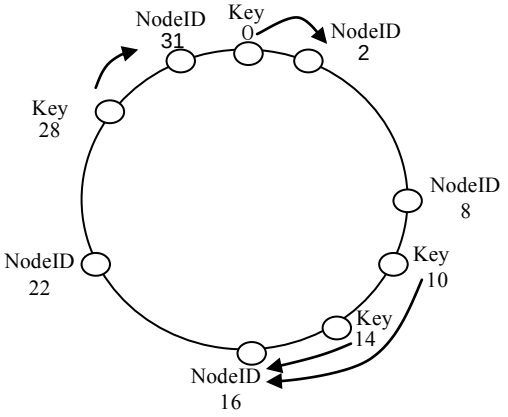


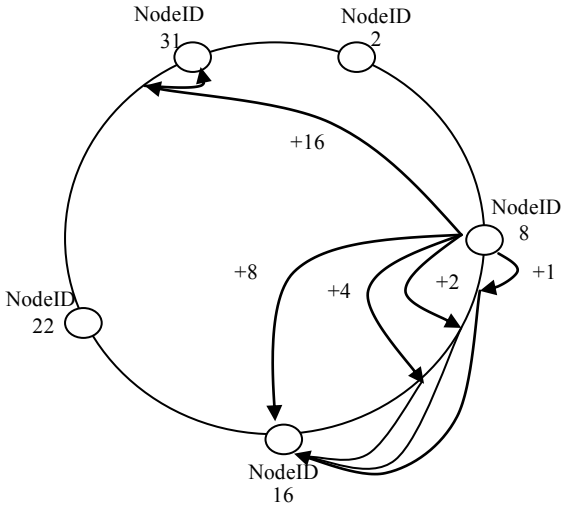
図 2.1　ノードの責任領域

ノード ID2、8、16、22、31 が存在する。ノード 2 は key0 を管理している。同様にノード 16 は key10、14、ノード 22 は key22、ノード 31 は key28 を管理している。

全てのノードは自分の前後にいるノードとしか接続されていないため、例えば検索者であるノード 8 が Key0 に対応するデータを取得したい場合、ノード 8 から時計回りで順番に検索していく。ノード 8→ノード 16→ノード 22→ノード 31→ノード 2 へと検索クエリが転送される。ノード 2 は Key0 を管理しているため、Key0 に対応するデータをノード 8 に送信する。

この方法では、N ノードで構成されているネットワークの検索に最悪 N 回、O(N)のホップ数がかかることになる。このままでは検索の効率が非常に悪い。そこで、Chord では検索効率の改善のためフィンガーテーブルを用いた検索手法を定義している。フィンガーテーブルとは、自ノードの ID より 2＾k（0＜＝k＜＝N）先のノードの IP アドレスを常に保持しているテーブルであり、このテーブルを参照してクエリの転送をショートカットしていく。自ノードの ID より 2＾k（0＜＝k＜＝N）先のノードが存在しない場合は、そのハッシュ値を管理するノードがフィンガーテーブルに書き込まれる。

よって logN のルーティングテーブルを保持することになり、O(logN)のホップ数に抑えることが可能となっている。図 2.2 にノード 8 が所持しているフィンガーテーブルを示す。



Finger Table（node8）

Node8+1	Node16
Node8+2	Node16
Node8+4	Node16
Node8+8	Node16
Node8+16	Node31

図 2.2　ノード 8 が所持するフィンガーテーブル

検索クエリのホップは、ノード 8 はフィンガーテーブルを参照しノード 31 へ Key0 の検索クエリを転送する。ノード 31 は successor を参照することでノード 2 が Key0 を管理していることがわかるので、ノード 2 へ検索クエリを転送する。ノード 2 は Key0 を管理しているので Key0 に対応するデータをノード 8 に転送する。仮に、ノード 31 が successor を参照してもノード 2 が目的の Key を管理していなかった場合は、ノード 31 が保持しているフィンガーテーブルを参照することでスキップしながら検索クエリを転送していく。

Chord では successor に対して Predecessor の情報を定期的に問い合わせることでリングを維持している。また、参加・脱退・離脱を検出している。これを stabilization という。stabilization によって、各ノードは参加・脱退・離脱を検出する。

2.2 関連研究

前節で述べた、オーバーレイネットワークの近さに実ネットワークの近さが考慮されていないという問題点に対して、改善を行った研究を紹介する。

・ネットワークアドレスを利用した ID 生成法[7]

インターネットにおける IP アドレス体系においては、一般的に、ネットワークアドレスが同じまたは近いノード同士は物理ネットワーク上でも近くに位置する可能性が高い。この特徴を利用し、ノードのネットワークアドレスとノード ID の上位ビットが対応するようノード ID を構成することで、識別子空間上でのノード間の距離に、物理的な距離をある程度反映させることが可能となる。具体的には、識別子ｍビットの上位ｋビット（１≦ｋ＜ｍ）を、ネットワークアドレスを元にハッシュ関数等で決定する。次に、残りの下位ｍ－ｋビットを、従来法と同様に当該装置の IP アドレスおよびポート番号の組からハッシュ関数で決定する。

・HIERAS [8]

HIERAS では、各ノードはランドマークノード（ネットワーク中のどのノードからも存在と IP アドレスなどが知られているノード）への遅延時間を基にした、物理ネットワークでの位置を表すリング ID と呼ばれる特別な識別子を持つ。同じリング ID を持つノード同士で生成した Chord のネットワークを通常の Chord でのネットワークとは別に保持することで階層化を行っている。このように複数の Chord ネットワークを使用することで先に挙げた問題点を改良している。

・LCLV (Layered Chord with Landmark Vector) [9]

LCLV は物理ネットワーク上での距離を通信に必要な遅延時間により近似可能であるという前提のもとに、遅延時間の短いノード同士は、分散ハッシュ上のネットワークにおいても近くなるような ID を生成する。ランドマークノードの 2 次元空間の座標をランドマーク間の実ネットワーク上での遅延時間を測定し、測定された遅延と 2 次元空間での距離との誤差の和が最小となる点にランドマークを配置する。次に同様の方法を用いて、ノード N から各ランドマークノードへの通信遅延時間から N の 2 次元空間での座標を求める。次に、2 次元空間をいくつかの空間に分け、そこに空間充填曲線（Space Filing Curve：SFC）を引く。ここでは Hilbert 曲線を空間充填曲線に用いている。そして、空間充填曲線が通った順に数字を割り当てる。このとき、ノード N が存在する座標上の SFC によって割り当てられた数字をノードの ID として新たに割り当てる。

これらの研究では、ノード ID 決定時にあらかじめ実ネットワークを考慮して ID の生成を行うことで、実ネットワーク上で近いノードがオーバーレイネットワーク上でもある程度近い位置に配置されるようになっている。

3. 提案手法

3.1 目的

従来研究では、オーバーレイネットワーク構成時に実ネットワークの状況を反映させていた。しかし、一度構築した構成は、ネットワークの状態が変化しても変更されない。

本研究ではネットワークの状況が変化した場合でも、動的に構成を変更し、最適な構成に近づけることを目的とする。

3.2 基本設計

Chord におけるオーバーレイネットワークは、実ネットワーク上で遅延時間が短いノードがオーバーレイネットワーク上で ID が近い位置に配置されることで、検索効率の向上が確認されている。これを参考にし、オーバーレイネットワークが利用されている最中に適宜ノード ID を変更していくことで、実ネットワーク上で遅延時間が短いノード同士をオーバーレイネットワーク上でも近い ID を付与することが可能となる。なお、提案手法には制御ノード等全体を管理するシステムは存在せず、各ノードが自律して処理を行う。

3.3 提案手法 1

提案手法 1 では検索要求の発生をきっかけに、検索するノードが自分に近いノード(Target ノード)を発見する方法をとる。発見後、Target ノードに隣接した場合 Chord リングが本当に改善されるのかの確認後、Target ノードの前後どちらかのノードとノード ID の交換を行う。以下にその手順を示す。

①検索するノードが、検索終了後にクエリが転送されたノードとの遅延時間を計測する。計測された遅延時間の中で、最も小さい遅延時間を選別する。(図 3.1)

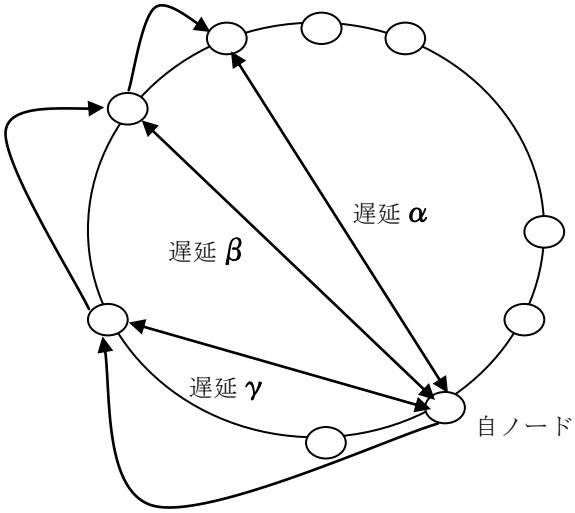


図 3.1　遅延計測及び最小遅延の選別
ここでは $\alpha < \beta < \gamma$ とし、選別される遅延時間は α とする。

②選別された遅延時間が自ノード前後の遅延時間よりも小さい場合、相手のノードを Target ノードとし、交換手続きを開始する。以降、自ノードを要求元ノードとする。

③Target ノードが決定した場合、自ノードは Target ノードに隣接するため交換候補先を Target ノードから教えてもらう。

④Target ノードは③の要求が来ると、要求元と Target ノードの前後ノードとの遅延を比較し、要求元ノードとの遅延が前後両方、または前後どちらかよりも小さい場合、前後ノードの遅延が大きいほうを交換候補ノードとし、要求元に IP アドレスを伝える。

⑤仮に要求元ノードが交換候補先ノードと交換をした場合、両ノードの前後に来るノード間の遅延計測をする(図 3.5)。

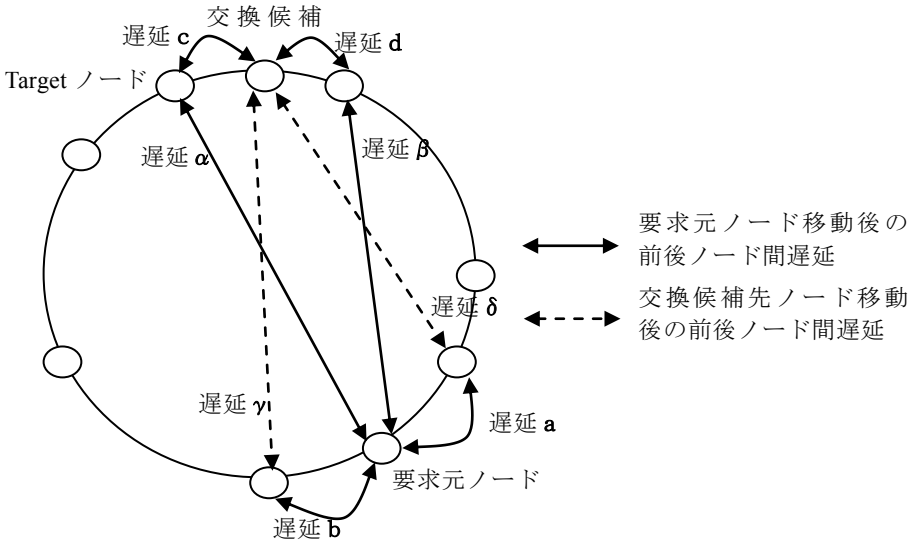


図 3.5　交換後の前後ノード間遅延計測

⑥交換前、交換後に前後に来るノード間の遅延の合計を計算し、交換後の総遅延が交換前より小さい場合、要求元ノード及び交換候補先ノードのノード ID 交換を行う。

図 3.5 では（$\alpha + \beta + \gamma + \delta$）＜（a ＋ b ＋ c ＋ d）となっていれば、ＩＤ交換を行う。

⑦交換後より交換前の総遅延が小さいのであれば交換は行わない。このとき、④で Target ノードの前後ノード間遅延が両方とも要求元ノード間の遅延より大きい場合、交換候補先をもう一方に変更し、手順⑤から再開する。

手順⑦で交換を行わない理由として、要求元ノードは隣接するノードが近いノードになる。しかし、要求元とノード ID を交換したノードに注目した場合、交換されたノードの隣接ノードは遠いノードになる可能性があるため。

3.4 提案手法 2

提案手法 2 では提案手法 1 と同様に検索要求を利用して､Target ノードを発見する。提案手法 1 と違う点はノード ID を Target ノードの隣接するノードと交換するのではなく､Target ノードと Target ノードの後方に隣接するノード(predecessor)との間に割って入るノード ID を新たに付与することで、Target ノードに隣接することを実現する。

以下にその手順を示す。

①～②提案手法 1 の①～②と同様

③Target ノードの ID とその predecessor ノードの ID の中間を計算し、その値を自分のノード ID として利用する。

3.5 提案手法 3

提案手法 3 はこれまでと違い検索要求ではなくフィンガーテーブルを利用して Target ノードを発見する。処理の発生条件は、ノードの移動が発生し再度参加処理が必要になった場合である。フィンガーテーブルを参照しテーブル全てのノードとの遅延を計測する。Target ノード発見後の処理は提案手法 2 と同様である。フィンガーテーブルを利用してノード ID の変更処理はネットワーク参加時と、携帯端末が移動することによって IP アドレスが変更された場合に発生する。

これらの手順を、オーバーレイネットワークが動作している状態で繰り返し行っていくことで、実ネットワーク上で近いノードを発見することができ、交換後にオーバーレイネットワーク上でも近いノード配置に変更されていくと考えられる。

次章で提案方式の有効性について検討するために行ったシミュレーション結果を示す。

4. 性能評価

4.1 シミュレーション条件

本研究では、DHT を代表するアルゴリズムの一つである Chord を基礎にし、構築されるオーバーレイネットワークを動的に改善する手法のシミュレーションを行った。

以下にシミュレーション条件を示す。

・オーバーレイネットワークに参加するノードは 1000 台。

・実ネットワークのトポロジーはトポロジー生成ツール GT-ITM[10]の TS モデルを使用した。

トランジットノードは 1600 台。

スタブノードは 16000 台。

ネットワーク図を図 4.1 に示す。

・移動するノードの割合を 10%～90%まで 10%刻みで増加させた。また、移動する頻度はシミュレーション中に約 1 回移動する。

・各ノード約 50 回の検索をし、検索 1 回の平均遅延時間を算出した。

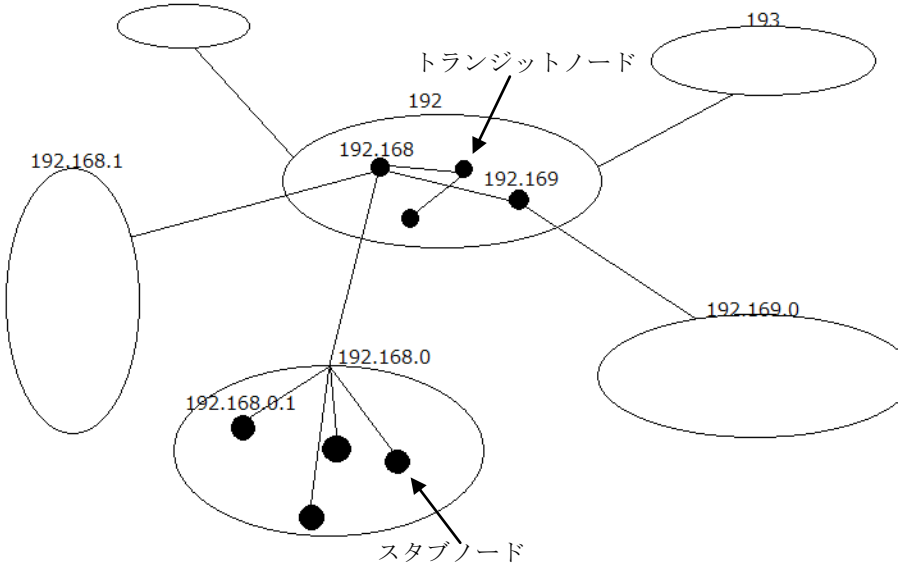


図 4.1　シミュレーションに利用したネットワーク図

4.2 シミュレーション結果

従来方式として、ネットワークアドレスを利用した ID 付与法を使用している。また、全ての提案手法においても同様にネットワークアドレスを利用して ID を付与している。

図 4.2 に従来方式と提案方式、それぞれ各ノード約 50 回の検索を繰り返し、検索一回の平均遅延時間を算出し比較したグラフを示す。

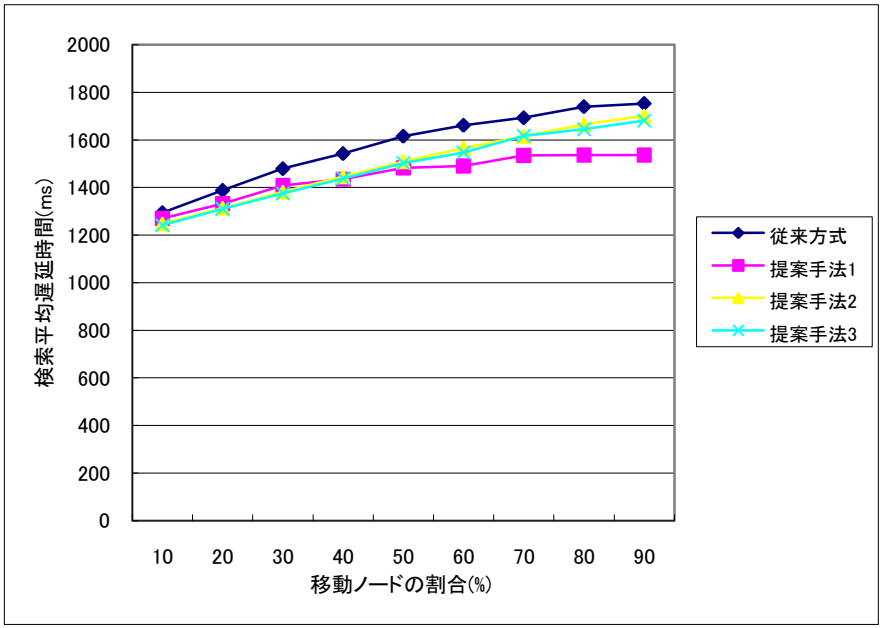


図 4.2　従来方式との平均遅延時間の比較

提案方式全てが従来方式の検索平均時間よりも少ない時間で検索が成功していることがわかる。提案手法 2 及び 3 の場合移動ノードの割合が 40～60%の時、特に削減効果が出ている。提案手法 1 の場合移動ノードの割合が 60%以降、一番検索平均遅延時間が少ない。

4.3 オーバーヘッド

オーバーヘッドメッセージ数については、表 4.1 に示す。

ID の再割り当てはノードが参加するのと同程度のコストがかかる。提案手法 1 では ID の交換回数が非常に多くなり、かなりのコストがかかることが予想される。提案手法 2 及び 3 では、移動ノード数とほぼ同じ回数の ID 再割り当てが発生していた。

表 4.1 処理の発生タイミングとオーバーヘッドメッセージ数

	提案方式 1	提案方式 2	提案方式 3
ID 再割り当てのタイミング	検索完了時	検索完了時	ノード移動時
オーバーヘッドメッセージ数	2×hop 数＋2＋データ交換メッセージ	2×hop 数＋データ交換メッセージ	2×finger table size＋データ交換メッセージ

5. まとめ

Chord のオーバーレイネットワークは実ネットワークの状況が考慮をされていない。そのためオーバーレイネットワーク上では理想的な検索要求の転送が行われているのに対して、実ネットワーク上では無駄が含まれる転送になる問題点があった。本研究ではこの問題点に対して、動的にその問題を解決する手法を提案し、その有効性をシミュレーションによって確認した。

提案方式としては、一度オーバーレイネットワークを構築した後に、検索要求を利用して近いノードを発見し、ノード ID を交換もしくは変更していくことで近いノードを隣接させる。このようにオーバーレイネットワークに実ネットワークの状況をある程度反映させる方式を提案した。

1000 台のノードを用意し従来方式、提案方式それぞれの検索時間を調査するシミュレーションを行った。その結果、全ての提案方式が従来方式の検索平均遅延時間よりも少ない時間で検索が可能となっていた。特に提案手法 1 では移動ノードの割合が高いほど削減率が高くなった。ただし、他の二手法とくらべコストが大きくなるというデメリットもある。

今後の課題として、

・ノード数の増加に対しての削減率の調査
・具体的に数値化したコストの算出

などが挙げられる。

参考文献

[1]構造化オーバレイ
http://www.shudo.net/publications/swopp2006/shudo-SWoPP2006-slides-overlay-routing.pdf

[２] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, “Chord:A Scalable Peer-to-peer Lookup Service for Internet Applications”, In the Proceedings of ACM SIGCOMM, 2001.

[３]Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, “A Scalable Content-Addressable Network”, In Proceedings of ACM SIGCOMM, 2001.

[４] A.Rowstron and P.Druschel, “Pastry:Scalable, distributed object location and routing for large-scale peer-to-peer systems”, IFIP/ACM International Conference on Distributed Systems Platforms(Middleware), Heidelberg, Germany, November 2001.

[５] B.Y.Zhao, J.Kubiatowicz, and A.D.Joseph, “Tapestry:An infrastructure for fault-tolerant wide-area location and routing,” Tech. Rep. UCB/CSD-01-1141, Computer Science Division, University of California, Berkeley, Apr 2001.

[６] “kademla.”http://dev.ariel-networks.com/modules/xfsection/article.php?articleid=29

[７]縣 亮、金子 豊、堀内 幸夫 "DHT を利用したデータ検索システムの高速化手法" 電子情報通信学会技術研究報告. IN, 情報ネットワーク 106(237) pp.121-126 20060907

[８]Zhiyong Xu、Rui min、Yiming Hu "HIERAS: A DHT Based Hierarchical P2P Routing Algorithm" icpp, pp.187, 2003 International Conference on Parallel Processing (ICPP'03), 2003

[９]羽場 裕介、 松尾 啓志 "物理ネットワークの状況を考慮した階層型分散ハッシュ法の提案" 情報処理学会研究報告. UBI, [ユビキタスコンピューティングシステム] 2006(14) pp.43-48 20060216

[10]Megan Thomas and Ellen W. Zegura. "Generation and Analysis of Random Graphs to Model Internetworks." Technical Report GIT-CC-94-46, College of Computing, Georgia Tech