平成 23 年度　東邦大学大学院理学研究科情報科学専攻　修士論文

DHT ネットワーク上の ID 再割り当てによる検索時間削減方式

学籍番号　6510009　氏名　嶋野裕太

要旨

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 アルゴリズムは、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 アルゴリズム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。本稿では、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。二つの方式のシミュレーションを行い、ID の動的再割り当ての有効性を確認した。

目 次

第１章 序論

近年、IT の技術は目覚ましい速度で発展している。特に物理的なネットワークに依存せずアプリケーションレベルで論理的なネットワークを構成するオーバーレイネットワークに注目が集まっている。Ｐ２Ｐ（Peer to Peer）ネットワークもその一つである。従来のネットワークではサーバを用いて行ってきた通信を、サーバを用いずエンドユーザ同士で検索・データ通信を行う手法がＰ２Ｐネットワークである。特に Pure Ｐ２Ｐモデルのネットワークはオーバーレイネットワークの構成の仕方から、非構造化オーバーレイネットワークと構造化オーバーレイネットワーク[1]の二つに分類される。

非構造化オーバーレイネットワーク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場当たり的に構成される特徴がある。そのため、柔軟な検索が可能であるものの、オーバーレイネットワークが大きくなるにつれ存在するコンテンツが発見できない可能性が高まっていくので、検索の効率は悪いという特徴を持つ。また、検索方法が P2P ネットワーク内の各ノードへ検索要求をフラッディング（接続しているノード全てに検索要求を転送する）していく方法をとっているので、帯域を圧迫してしまう。

一方、構造化オーバーレイネットワーク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、現在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]等がある。DHT の特性上、完全一致探索しかできないものの、非構造化オーバーレイネットワークと比較すると検索の効率が非常によい。また、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して構築せず、実ネットワークでのノード間の近さ(通信遅延の大小)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なく検索要求が転送されていった様に見えたとしても、実ネットワークでは効率の悪い検索要求の転送が発生してしまっている可能性が多く含まれている。従来研究では、ネットワークアドレ

スを利用したID生成、ランドマークノードを利用したID生成等、ID生成時に実ネットワークの状況を考慮する手法が提案されている。だが、近年スマートフォンやWiMAXの利用者が増え、移動端末が増加の傾向にある。これにより、ネットワークの状況は時間と共に変化することとなる。そのため、ID生成後にも実ネットワークの状況を考慮する必要が出てきた。

本研究では、DHTを利用したChordアルゴリズムを基礎とし、一度決定したノードIDを実ネットワークの状況によって動的にノードIDの再割り当てしていくことで、実ネットワーク上での検索効率を改善する手法を提案する。

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Ｐ２Ｐネットワークの概要、Chordアルゴリズムの概要について説明する。また、関連研究の紹介をする。第３章では提案方式であるノードIDの変更方法について説明する。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。

第 2 章 研究背景

2.1 P2P ネットワーク

P2P ネットワークとは、データの送り手と受け手が完全に固定されているクライアント・サーバ方式などと違い、ネットワーク上で対等な関係にある端末（ピア）間を相互に直接接続し、データを送受信するネットワーク形態の一つである。端末には一般家庭にある様な PC が使用されることが多い。

2.1.1 クライアント・サーバモデル

クライアント・サーバモデルとは、サービスを利用するクライアント、サービスを提供するサーバによって構成されるネットワーク形態である（図 2.1）。クライアントはサーバに要求を出し、サーバは要求を受け取りその結果をクライアントに返す。

このように、クライアントとサーバでそれぞれ役割が固定されている。クライアントからの要求は全てサーバへと送られサーバで処理されるため、クライアントの負担は小さいがサーバの負担が大きい。サービスの利用者が多くなれば当然サーバへのアクセスが増加することになり、アクセスが集中しすぎた結果サーバが故障してしまうということは珍しいことではない。サーバが故障してしまえば処理が行えなくなるので、当然クライアントはサービスを利用できなくなる。そのためサービスを提供する側は耐故障性に力を入れる必要が出てくる。

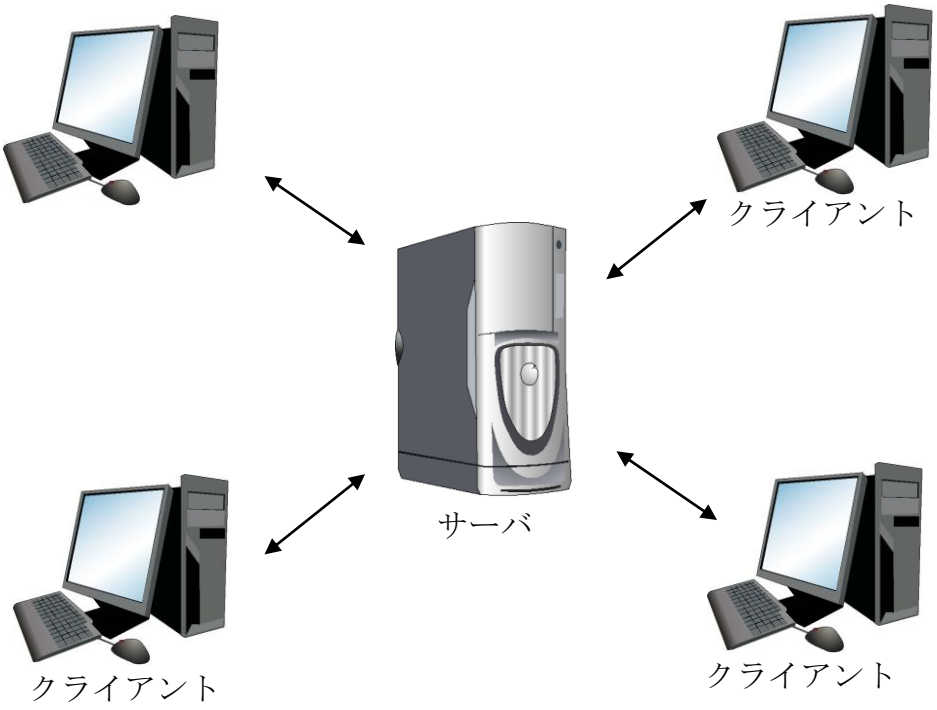


図 2.1　クライアント・サーバモデル

2.1.2　P２P（Peer to Peer）モデル

P２P モデルは、クライアント・サーバモデルとは異なり最初からクライアントとサーバの役割は決まっていない。処理を行う際要求を出すピアがクライアント、受け取るピアがサーバとなるネットワーク形態である。つまり全てのノードがクライアントでありサーバでもある。

クライアント･サーバモデルと比較した場合の P２P のメリットとデメリットは次のようなものがある。

メリット
- ・スケーラビリティが高い
- ・コストが安い
- ・耐障害性に優れる

デメリット
- ・実装が困難である
- ・動作確認が困難である
- ・インターネットへの負荷

クライアント・サーバモデルの場合、前述した通りクライアントが増加するとサーバの負荷も増加する。そのためクライアント数が限界を超えるとサービスが停止してしまう。一方Ｐ２Ｐモデルはサーバを用いない（一部に利用される場合もある）為、ユーザが増加してもそれぞれのピアに処理が分散されるのでスケーラビリティが高いといえる。またサービス提供側はサーバが必要ない、もしくはクライアント・サーバモデルに比べて高性能なサーバを必要としないためコストが安価で済む。そして特にPure Ｐ２Ｐの場合、処理にサーバを利用しないのでシステム全体が停止する心配がない。

一方で、端末はPCが一般的だがそのPCは千差万別である。そのため実装に非常に注意深くなる必要がある。また、動作確認は最終的に実機で行う必要があるが、これに膨大な手間とコストがかかってしまう。最後に以降で詳しく説明するが、Pure Ｐ２Ｐの場合、検索要求がネットワーク上に溢れかえり帯域を圧迫してしまう問題点がある。

2.1.3　Ｐ２Ｐの種類

Ｐ２Ｐモデルで重要なポイントはどのように目的のピアのIPアドレスを知るか、という部分である。現在のＰ２Ｐではコンテンツのタイトル(Key)とデータ(Value)を結び付けたインデックス情報(Key-Value ペア)と呼ばれる情報を利用し検索を行う。一般的にデータとはコンテンツを保持しているピアのIPアドレスである。従ってKeyをもとに、Keyに対応するValueを保持するノードを発見しその相手と通信を行う、という流れになる。よって、インデックス情報をどのように管理するかが重要になってくる。このインデックス情報の管理の仕方でＰ２Ｐは二つに大別されている。それぞれHybrid Ｐ２Ｐ、Pure Ｐ２Ｐと呼ばれている。

・Hybrid Ｐ２Ｐモデル

Hybrid Ｐ２Ｐモデルは、インデックス情報を管理するインデックスサーバを用いるネットワークである。ピアはインデクッスサーバに検索要求を送信し、検索対象を保持するピアのIPアドレスをサーバに調べてもらい、コンテンツの所在が判明すれば、その後ピア同士で情報の交換を行う（図 2.2)。対象コンテンツを保持するピアのIPアドレスが管理されていない場合、そのネットワークには対象コンテンツが存在しないことがわかる。このように役割がある程度クライアント・サーバモデルの様にそれぞれ決定されている。

この方式では各ピアの負担が少なく、検索要求はインデックスサーバでのみ処理されるので検索の効率が非常に良い。特に、後述する Pure P２P と違い対象コンテンツが P２P ネットワーク内に存在するのかそれとも存在しないのか、インデックス情報がサーバで一元管理されているため確実にわかる。また、セキュリティ管理はサーバ側で行えるので容易であるというメリットもある。

しかし、前述したクライアント・サーバモデルが一部で利用されているため、インデックスサーバが故障してしまうと、コンテンツは分散配置されているものの、コンテンツを保持しているピアの IP アドレスを知ることができなくなる。そのため、各ピアはサービスを受けられなくなる。またピアの増大に伴いスケーラビリティが減少していく点が後述の Pure P２P に劣る。

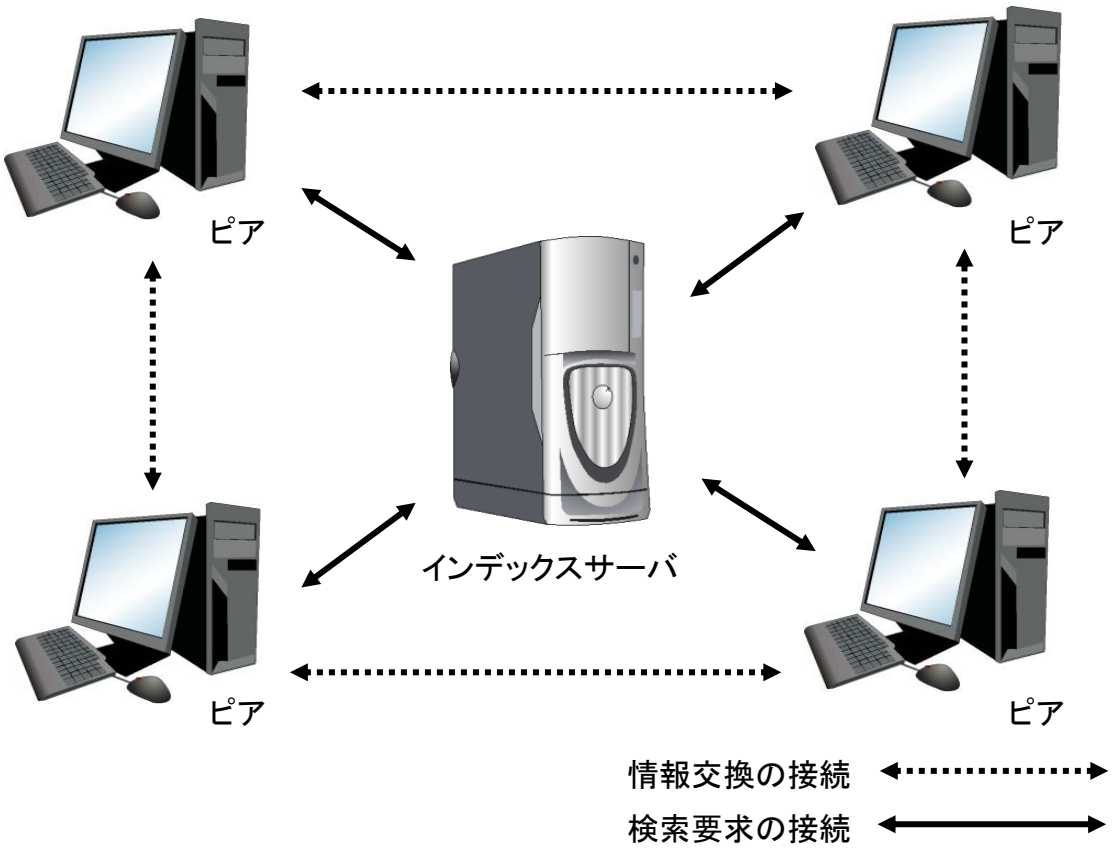


図 2.2　Hybrid P２P モデル

・Pure P２P モデル

Pure P２P モデルは Hybrid P２P モデルのようなインデックスサーバは存在し

ない。全てのピアが役割を決定されておらず、ピアの検索、情報の交換も全てピア同士で行う方式である。検索要求は接続されたピアにフラッディングされていき、ヒットしたピアと通信を行う（図 2.3）。

このモデルでは、サーバがないため、サーバの故障と共にサービスが停止するといったことがない。ピアの役割は決定されていないのでいくつかのピアが故障することがあったとしてもネットワークサービスは受けられる。よって Hybrid P２P モデルよりも耐故障性に優れていると言える。

一方、アカウント管理やユーザ認証がサーバで行われないため、セキュリティ管理が非常に難しくなっている。また、検索対象がネットワーク上にあるにもかかわらず、発見できない場合がある。これは、検索要求が有効である時間が設定されているためである。この制限がない場合検索要求がネットワーク上に永遠に残ってしまうことになる。そのためネットワーク内にコンテンツが存在していたとしても、対象のピアに検索要求が届かない場合がある。さらに、検索要求はフラッディングされて各ピアへ広がっていくので、ネットワーク内に検索要求があふれてしまい、帯域を無駄に使用し輻輳が起こりやすい。

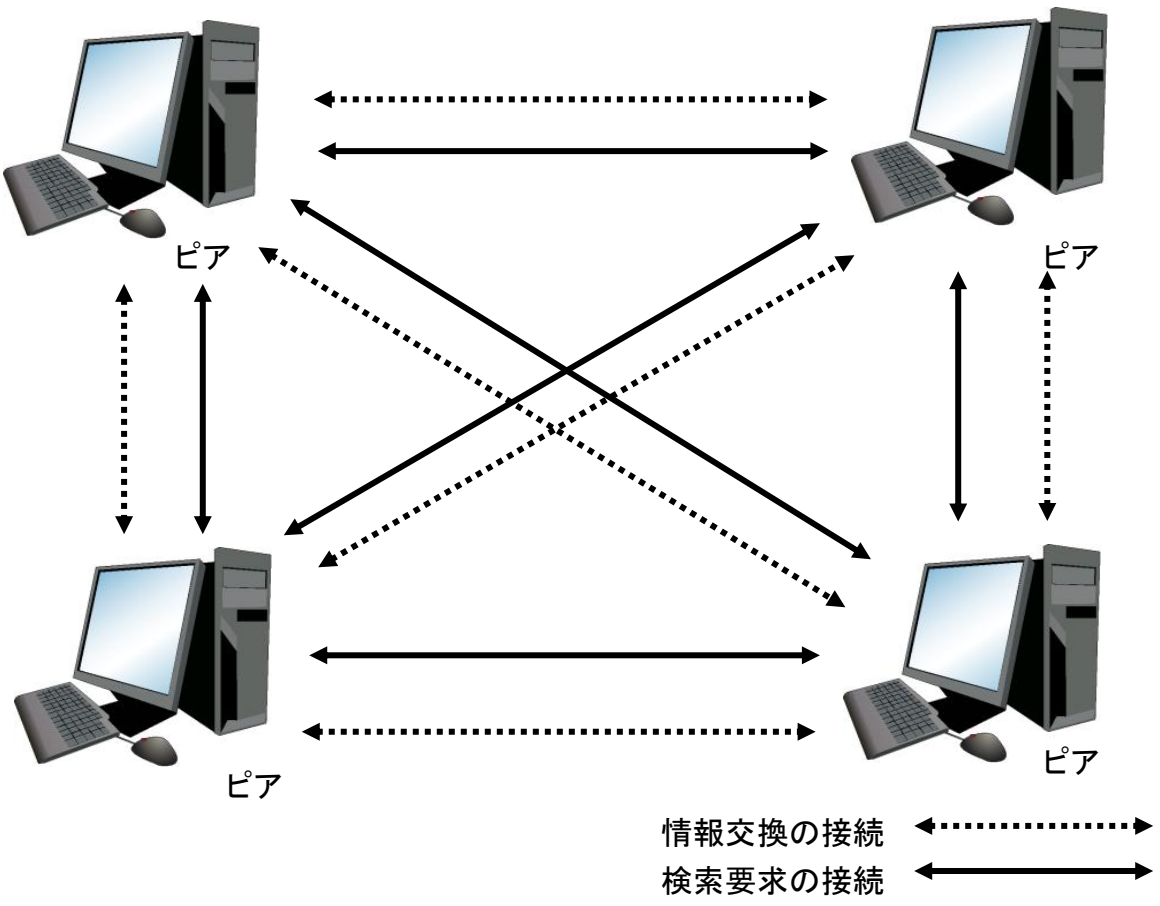


図 2.3　Pure P２P モデル

上記で述べたモデルは、サーバがダウンしてしまうことでのサービス停止や、検索要求がネットワーク内に溢れてしまうことで帯域を圧迫してしまい輻輳を引き起こしてしまう等の問題点がある。また Hybrid Ｐ２Ｐ モデル、Pure Ｐ２Ｐ モデルの両者共にピアの増大に伴い効率が落ちていく。

そこで、近年研究が盛んになっている検索方式として分散ハッシュテーブル（DHT）がある。

2.2　分散ハッシュテーブル（ＤＨＴ）

DHT とは、Pure Ｐ２Ｐ におけるデータ検索方式である。

コンテンツとノード（ピア）に同じハッシュ関数用いてハッシュ値を空間に写像し、その空間を複数のノードが分割して管理することで特定のピアに負担をかけない手法である。また、Pure Ｐ２Ｐ ではあるがオーバーレイネットワークの構成にルールが存在し、各ノードが管理するハッシュ値が決められている。そして検索の効率化のために、各ノードは他ノードへのハッシュテーブルも管理している。これを利用して検索を効率的に行うため、フラッディングを用いる必要がなく、検索要求が溢れかえる心配もない。一般に、2.1.3 のような Pure Ｐ２Ｐ のオーバーレイネットワークを非構造化オーバーレイネットワーク、DHT を利用したオーバーレイネットワークの様にルールが存在するものを構造化オーバーレイネットワークと呼ぶ。

具体的なノードの管理領域や検索要求の転送方法はアルゴリズムによって変わってくる。共通する部分は、まず取得したいコンテンツのハッシュ値の計算を行う。得られたハッシュ値を検索要求として、検索用のハッシュテーブルを参考に他のノードへ送る。検索要求を受け取ったノードも、そのハッシュ値を管理するノードを知らない場合、同様にハッシュテーブルを参考にして検索要求を転送する。これを繰り返すことで、最終的に目的のハッシュ値を管理しているノードに到達する。その後ピア間で直接通信を行う。この流れはどのアルゴリズムでも同じである。

非構造化オーバーレイネットワークと比較すると、構造化オーバーレイネットワークは非常にスケーラビリティが高い。スケーラビリティの高さは、DHT に参加するノード数 N に対して、どのノードから検索を開始しても、全てのノードへ O(N)よりも小さいオーダーのホップで到達可能であることが確率的に保証されているためである。DHT を利用した代表的なアルゴリズムでは O(logN)のホップで全ノードに到達可能となっている。

本稿では複数ある DHT アルゴリズムの中の Chord アルゴリズムを基礎とした。

2.3 Chord アルゴリズム

Chord は SHA-1(Secure Hash Algorithm 1)などのハッシュ関数を用いて自分のノード 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、スケーラビリティに優れている。

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 Key ID（Key ID＝hash（Key））をマッピングする。160bit の ID 空間の場合、約 1.46×10^{48} 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論理的な接続である。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(通信遅延の大小)は考慮されていない。図 2.4 で参加ノードが管理する Key ID の一例を示す。オーバーレイネットワークに参加しているノードは ID 2、8、16、22、31 である。また、put された Key は ID 0、10、14、28 である。図 2.4 では、ノード 2 は Key 0 を管理している。同様にノード 16 は Key 10、14、ノード 22 は Key 22、ノード 31 は Key 28 を管理している。

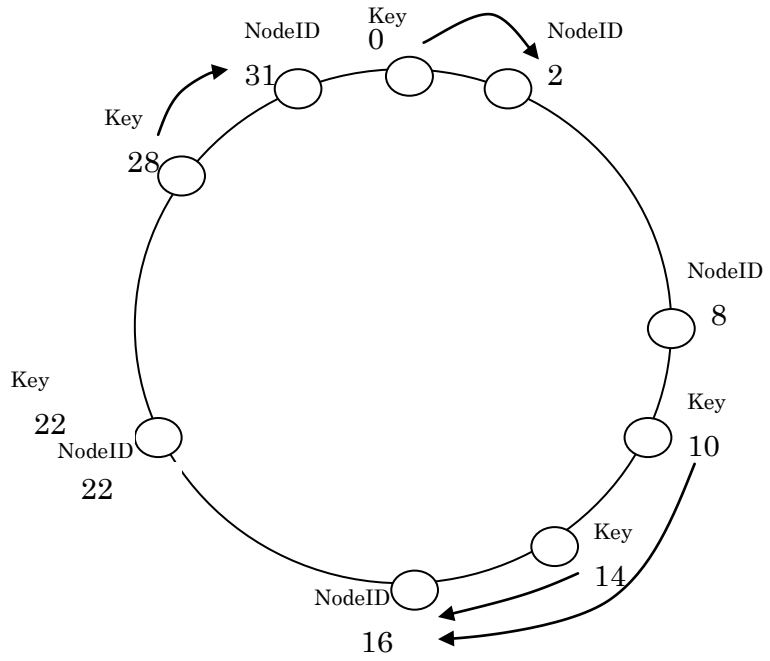


図 2.4　ノードの責任領域

2.3.1　ルーティング

図 2.5 で基本的な参照方法を示す。

検索者であるノード 8 がコンテンツ「アップル」を検索する場合、まず Key ID を計算する（Key ID＝hash（アップル））。仮に Key ID が 0 であれば、Key 0 を検索要求として他のノードへ転送する。ただし Chord には検索要求は時計回りに転送される特徴があり、ノード 8 から時計回りで各ノードを順番に検索していくことになる。図 2.5 ではノード 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 Key 0 を保持しているため、Key 0 に対応するデータをノード 8 に送信する。

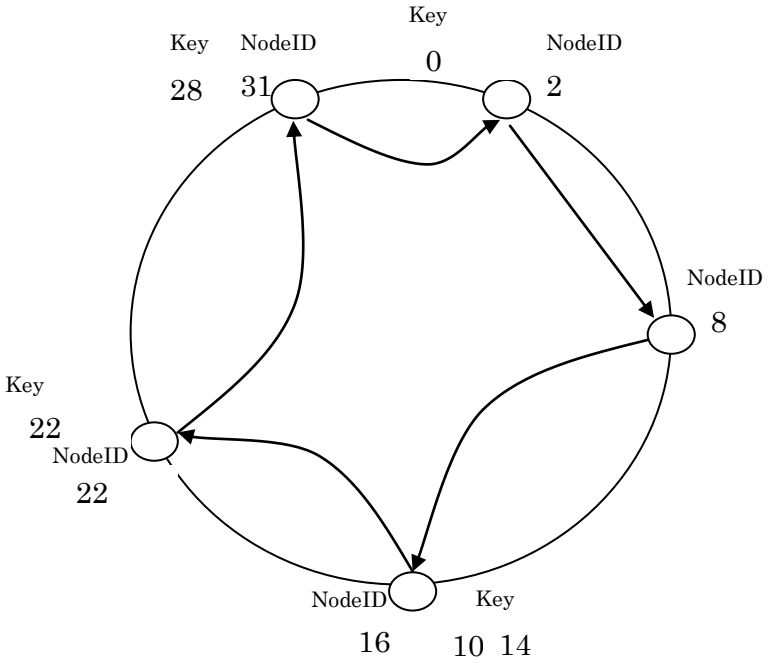


図 2.5　基本的な参照方法

この方法では、参加ノード数 N 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 アルゴリズム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 を X とした場合、n bit ハッシュ空間であれば $X+2^k \pmod{2^{160}}$ の ID を担当領域とするリストになる（0＜＝k＜n)。よって n bit ハッシュ空間であれば n 個のルーティングテーブルを保持することになる。このテーブルを参照して検索要求の転送をショートカットしていく。

具体的な検索要求の転送手順を説明する。まず、先ほどと同様に検索対象コンテンツのハッシュ値(Key ID)を計算し、それを検索要求とする。検索要求を受け取ったノードは、まず自身の管理領域に目的の Key ID が含まれているか確認する。含まれている場合は管理表を参照し回答する。含まれていない場合、目的の Key ID が自身の ID と Successor ノードの ID との間に位置するかの確認を行う。この範囲に目的の Key ID が位置している場合、Successor ノードが目的の Key ID を管理していることになる。その場合は Successor ノードへ検索要求を転送する。一方、目的の Key ID が自身の ID と Successor ノードの ID の間に位置しない場合、フィンガーテーブルを参照する。目的の Key ID から見て、反時計回りに最初に存在するノードを選択し検索要求の転送を行う。このようにして検索要求を転送されたノードが上記の処理を繰り返すことで、目的

の Key ID を管理するノードへ到達することが可能となる。このルーティングアルゴリズムを利用することで、Chord アルゴリズムは O(logN)のホップ数で任意のノードへの到達を可能にしている。

図 2.6 に図 2.5 と同様の検索を行う場合の検索ホップを示す。

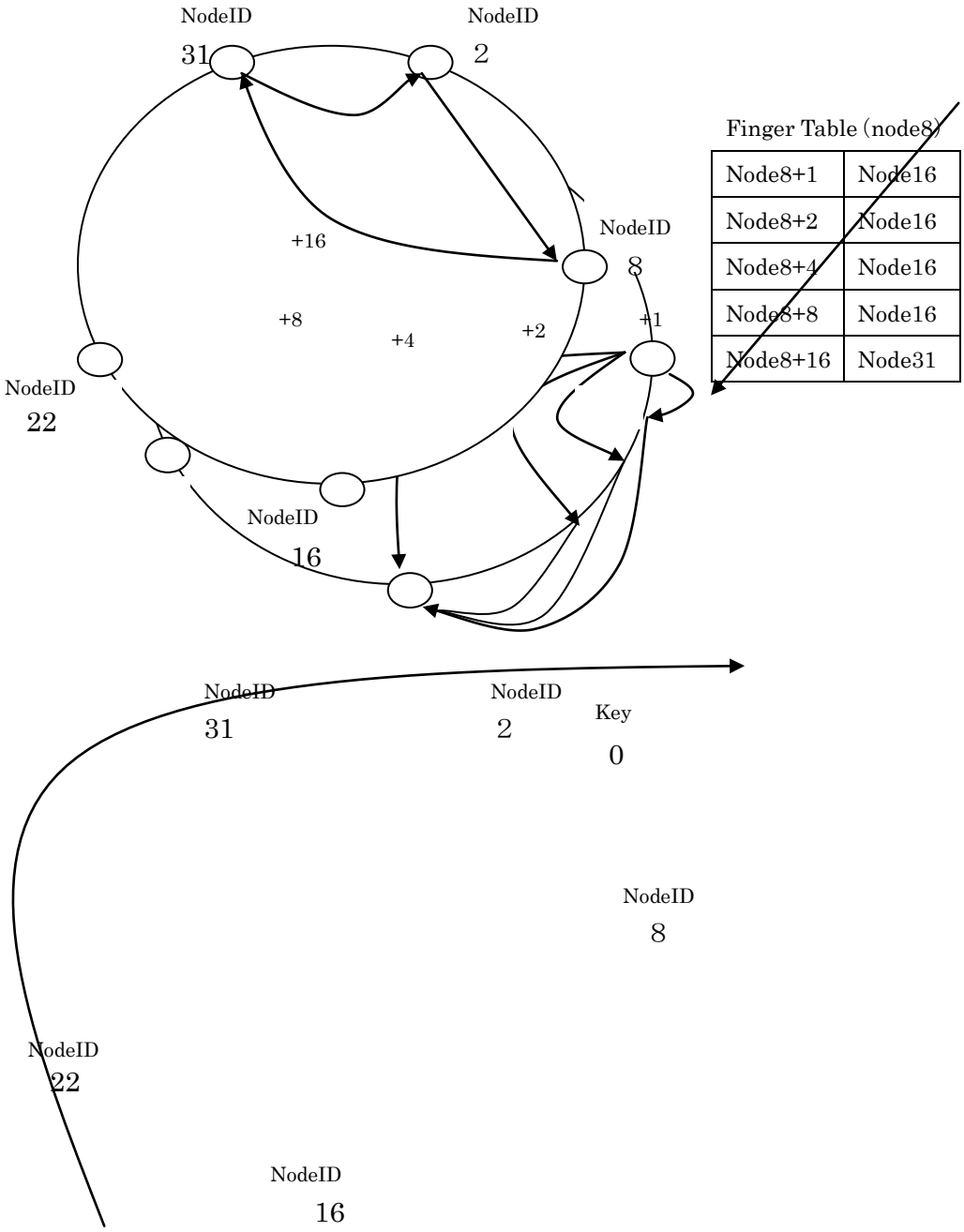


図 2.6　高速な参照方法(key 0 を検索)

2.3.2　Chord リングの保持

Chord では Successor ノードに対して 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。

以下に、参加・脱退・離脱の際の手順を示す。

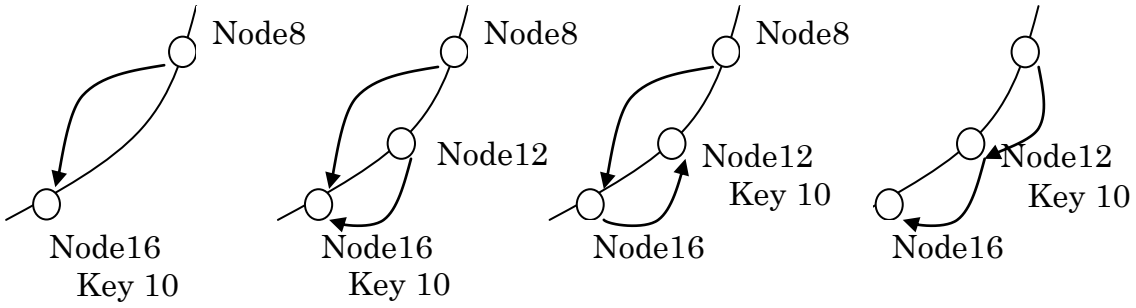
・参加時(図 2.7)

（１） Node12 がネットワークに参加しようとしている。(Successor は Node16 Predecessor は Node8。Node16 は key10 を管理しているとする)

（２） Node12 は自分の挿入する場所を確保するために、Successor となる Node16 を検索し、Node16 にリンクの接続を行う。

（３） Node12 は、Node16 から管理対象となる全てのキー（Key10）を受け継ぐ

（４） その後､各ノードの stabilize 処理によって Node8 は Node16 から Node12 にリンクを更新する。(各ノードのフィンガーテーブルもこの時更新される)

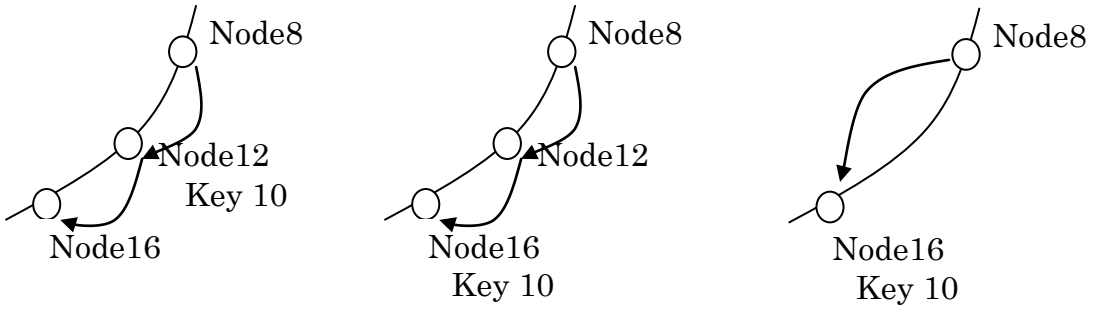


(1)初期状態　(2)Node12 初期化　(3)key の移動　(4)Node8 更新

図 2.7　参加処理

・脱退時(図 2.8)

（１）Node12、がネットワークから脱退しようとしている。（Successor は Node16、Predecessor は Node8。Node12 は Key10 を管理しているとする）
（２）Node12 は管理対象となる全てのキー（key10）を Node16 へ渡す。
（３）Node12 は Node16、Node8 それぞれに脱退後のリンク先を伝える。
（４）その後、各ノードのフィンガーテーブルの更新が行われる



(1)初期状態　(2)Key の移動　(3)Node 12 脱退

図 2.8　脱退処理

・離脱時

理想は上記の脱退処理にそって脱退ノードが管理領域を Successor ノードへ受け渡してから、ネットワークから脱退するのが望ましい。しかし、何らかの理由で脱退処理が行われずあるノードが離脱する可能性もある。その場合、stabilize 処理によってリングが復元されるので、サービスが停止してしまうような問題は発生しない。ただし、離脱したノードが管理していた情報は Successor ノードへの受け渡しがされていないため、離脱したノードと共に管理領域の情報が欠落してしまう可能性がある。そのため、Successor へあらかじめ情報の複製がされていることが望ましい。

2.3.3 問題点

ここで、本稿で解決する問題点について述べておく。

前述した通り、従来の Chord アルゴリズムではすべてがハッシュ値で管理されることになる。ノード ID は一般的に IP アドレス及びポート番号を利用して生成される。そのため、オーバーレイネットワーク上のノード配置は、実ネットワークとは無関係である。つまり、オーバーレイネットワーク上のノード同士の距離と、実ネットワーク上でのノード同士の距離(遅延の大小)は無関係であった。このことは、検索処理時にオーバーレイネットワーク上では理想的なホップであったとしても、実ネットワークでは不要なホップが多数発生する原因となっている。

そこで、本稿ではノード ID を実ネットワークの状態を考慮したものに変更し、検索処理にかかる遅延時間を削減する手法を提案する。

2.4　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(図 2.9)。

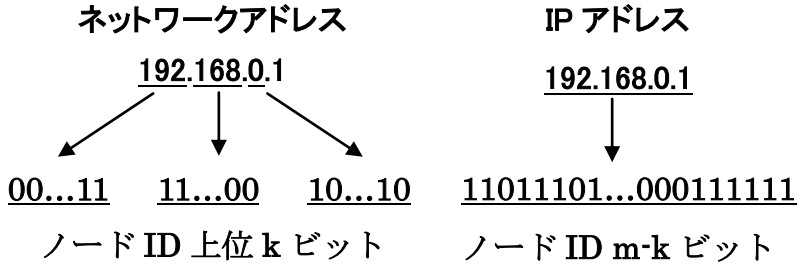


図 2.9　ネットワークアドレスを利用した ID 生成

・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、そこに図 2.10 のように空間充填曲線（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。

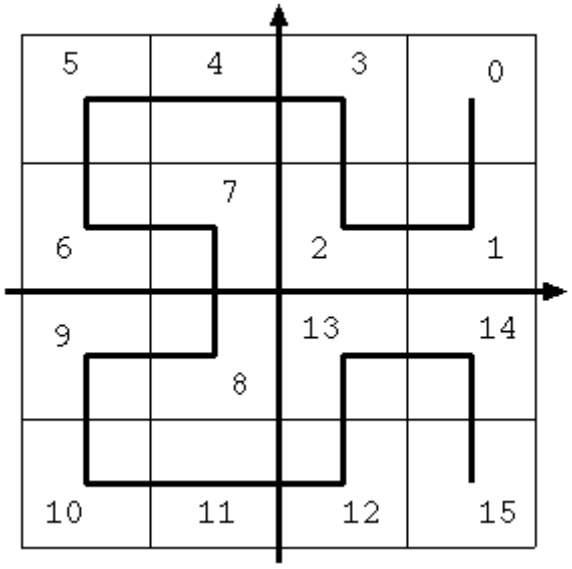


図 2.10　Space Filing Curve を用いた ID の割り当て

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いる。そのため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。ただし、これらの手法はノードがネットワークに参加する時点での実ネットワークは考慮されているが、参加後のネットワークの変化にまでは対応できていない。

第 3 章　提案方式

3.1　目的

Pure P２P は当初、非構造化オーバーレイネットワークがよく知られていた。

最近では、DHT を利用した構造化オーバーレイネットワークが注目され研究が盛んになっている。

以下に両者の特徴をまとめる。

非構造化オーバーレイネットワーク

- 隣接ノードに制約がない
- 検索に柔軟性がある
- スケーラビリティが低い
- 検索効率が非常に悪く、帯域を圧迫する

構造化オーバーレイネットワーク

- 隣接ノードに制約がある
- 検索は完全一致探索でしか行えない
- スケーラビリティが高い
- 検索効率は良い

前章の関連研究で紹介したが、構造化オーバーレイネットワークを効率的に運用するために、実ネットワークの状況を考慮しオーバーレイネットワークを構築する手法がいくつか提案されている。だが、近年スマートフォンや WiMAX の利用者が増え、移動端末が増加傾向にある。そのため、ネットワークの状況が時間と共に変化する。しかし、従来研究ではオーバーレイネットワーク参加後のネットワークの変化にまでは対応していなかった。

本稿では関連研究では対応していなかった、ネットワークの変化(移動端末を考慮)に対応する手法を提案する。

3.2　基本設計

本稿では DHT を利用したアルゴリズム Chord を基礎とする。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近いノードがオーバーレイネットワーク上で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。そして、近年増加してきた移動端末を考慮し、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 を交換していく手法を提案する。これにより、従来研究では対応できなかったネットワークの変化に柔軟に対応し、実ネットワーク上で近いノードをオーバーレイネットワーク上でも近い位置に修正することが可能となる。なお、提案する手法には制御ノード等の全体を管理するシステムは存在しない。処理は全て各ノードが自律して行う。

3.2.1　提案手法 1

まず、どのようにして実ネットワーク上で近い（遅延が小さい）ノードを見つけるかである。一番の方法は全てのノードとの遅延時間を計測することであるが、参加ノード数 N に対して N 回の通信が必要になるのは現実的ではない。そこで、提案手法 1 では、検索要求の転送を利用する。検索を行うノードからみて、検索要求が目的のノードに到達するまでに経由したノードを候補とする。その中から隣接するノードよりも近いノード（以下 Target ノード）を発見する方法をとる。以下に詳細な手順を示す。

① 検索を行ったノード（要求元ノード）は検索要求が転送されたノードに対して遅延を計測する（図 3.1）。

② 計測した遅延が最小値だったノードを選択する。図 3.1 では遅延 α が最小だったとする。

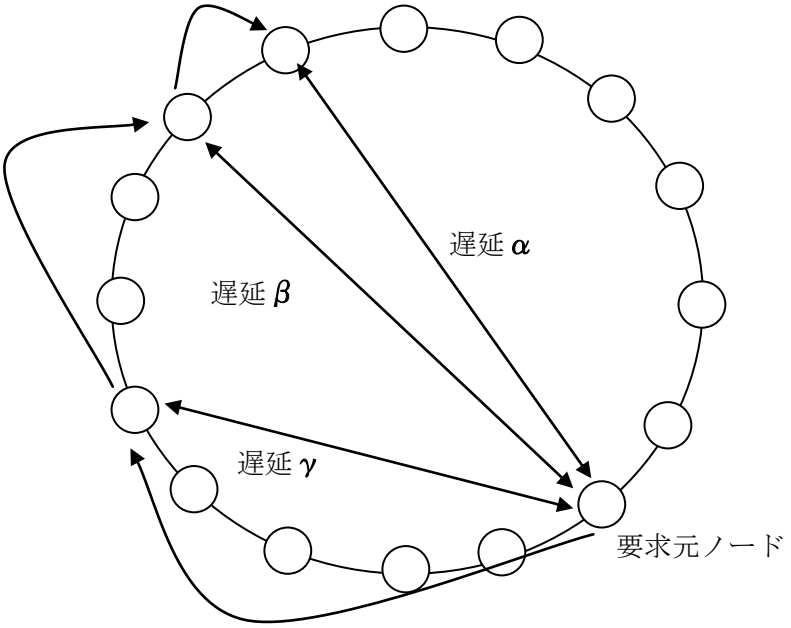


図 3.1　遅延計測

③ 要求元ノードに隣接しているノード間との遅延よりも、選択したノードとの遅延が小さい場合、選択したノードを Target ノードとする（図 3.2）。

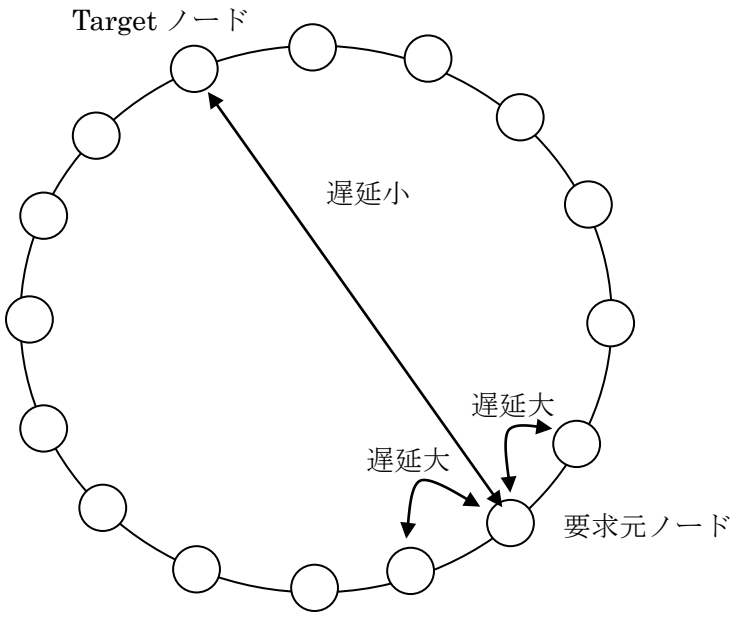


図 3.2　提案手法 1 における Target ノードの決定

Target ノードが決定した場合、要求元ノードから見て Target ノードは隣接ノードよりも実ネットワークで近いノードであることがわかる。そこで要求元ノードは Target ノードに隣接している前後どちらかのノードとの ID 交換を目標とする。

④ 要求元ノードは Target ノードに対して、前後どちらのノードと ID を交換するべきかどうかを尋ねる。

⑤ Target ノードは④の要求が来ると、要求元ノードと Target ノード、そして Target ノードと Target ノードに隣接しているノードとの遅延を比較する。要求元ノードとの遅延が前後の隣接ノード両方、または前後の隣接ノードどちらかよりも小さい場合（図 3.3)、前後の隣接ノードの遅延が大きいほうを交換候補ノードとし、要求元ノードに IP アドレスを伝える（図 3.4)。

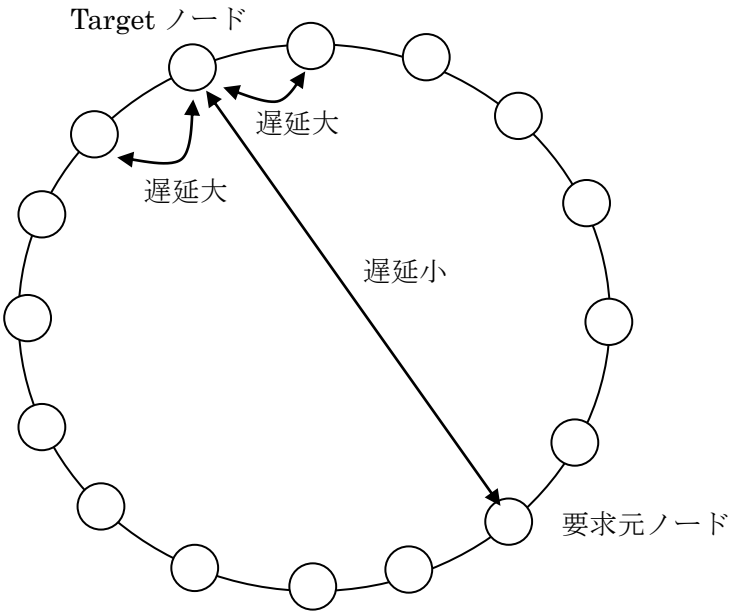


図 3.3　Target ノードの前後ノードと、要求元ノードの遅延比較

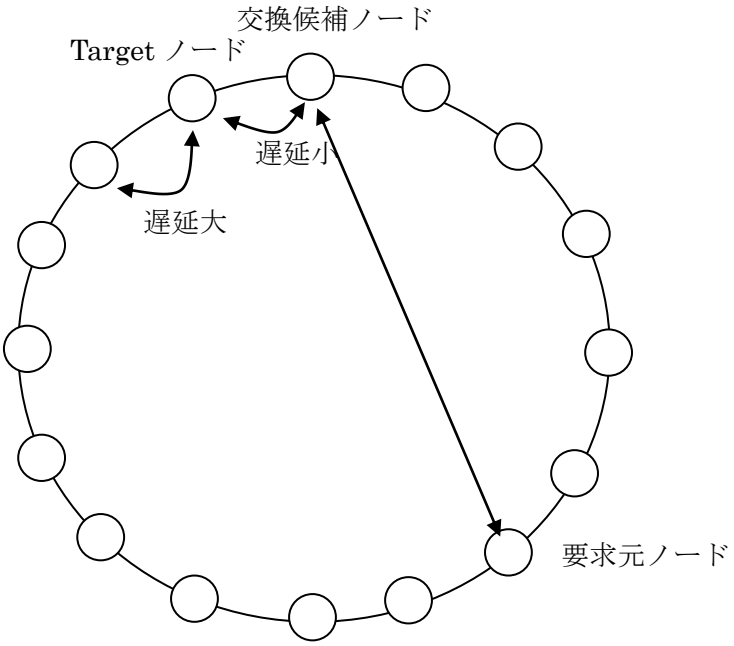


図 3.4　交換候補ノード決定

交換候補ノード決定後、すぐに ID の交換を行うと、要求元ノードが Target ノードに隣接することは実現する。しかし、要求元ノードから見て Target ノードとは反対側に隣接しているノード間の遅延、交換候補ノードが新たに隣接するノード間の遅延について現時点では判明しておらず、結果的に総遅延が ID 交換前に比べ大きくなってしまう可能性がある。そのため手順⑥、⑦で最終確認を行う。

⑥ 仮に要求元ノードが交換候補ノードと交換をした場合、両ノードに隣接するノードとの遅延計測をする（図 3.5）。

⑦ ID 交換前、交換後に隣接す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くなる場合、要求元ノード及び交換候補ノードのノード ID 交換を行う。

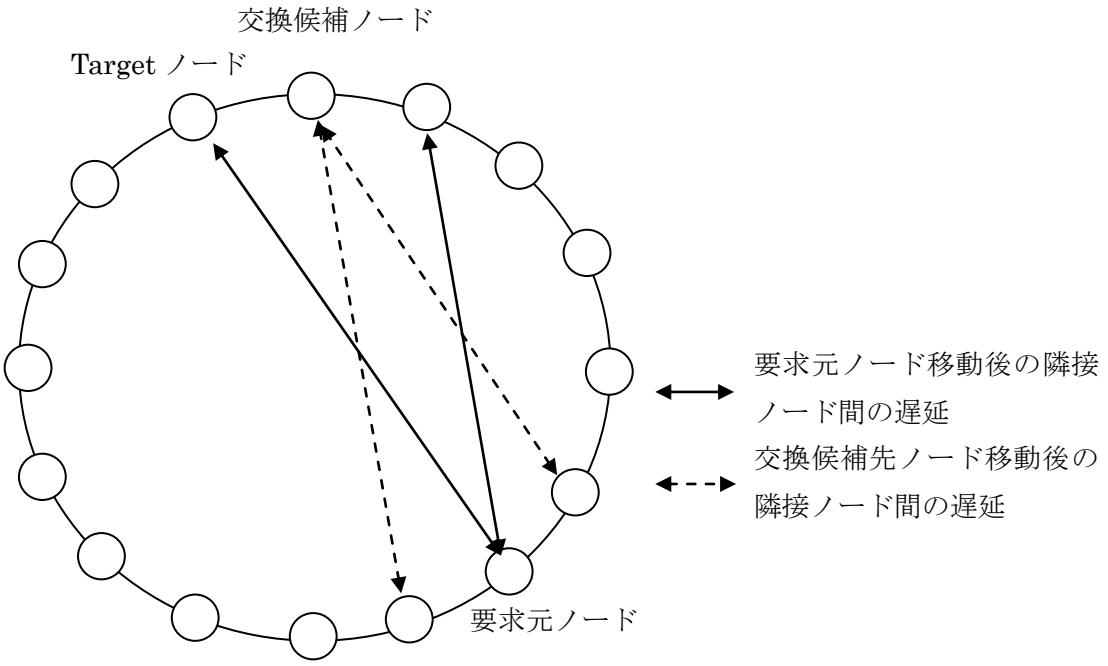


図 3.5　交換後の前後ノード間遅延計測

⑧ 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。

⑨ 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。

この手順を、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。

3.2.2　提案手法２

提案手法１では検索要求をもとに、要求元ノードに隣接するノードよりも遅延の小さなノードを発見した。そして ID 交換前、交換後を比較し、遅延が改善されると判断した場合 ID 交換を行った。しかし、検索要求発生が処理を開始する条件であること、また処理が細かくなるため、通信メッセージが多く発生することが予想される。

そこで、提案手法２では移動端末が移動し、IP アドレスの変更が発生した場合に処理を開始する。これは IP アドレスの変更が起きるとオーバーレイネットワークへ参加処理が再度必要となるためである。そこで、再度参加する際に実ネットワーク上で近いノードをあらかじめ調べる。実ネットワーク上での近いノードを発見する方法として保持しているフィンガーテーブルを利用する。さらに ID は Target ノードに隣接しているノードとの交換ではなく、Target ノードと Target ノードの Predecessor ノードとの間に割り込む手法を提案する。これにより、通信メッセージ数の削減及び ID 交換時に発生するコストの削減が可能になると考えられる。以下に詳細な手順を示す。

① 移動端末が移動することによって IP アドレスの変更が発生し再度参加する際、自ノードが保持しているフィンガーテーブル内から遅延が最小の端末を選択し、そのノードを Target ノードとして決定する（図 3.6）。

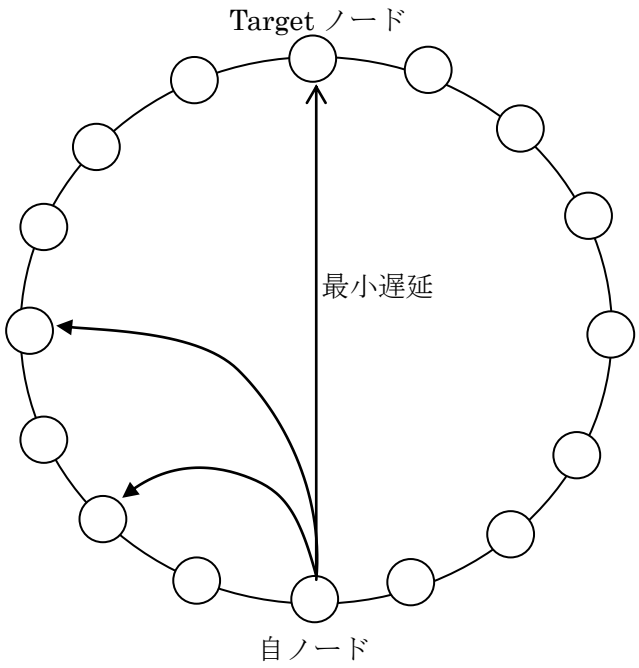


図 3.6　提案手法２における Target ノード決定

② Target ノードへ、Target ノードの Predecessor ノードの情報（ノード ID、IP アドレス）を要求する。

③ Target ノードと、Target ノードの Predecessor ノード間の中間に位置する ID を計算しそれを自ノードの ID とし、ネットワークに再度参加する（図 3.7）。

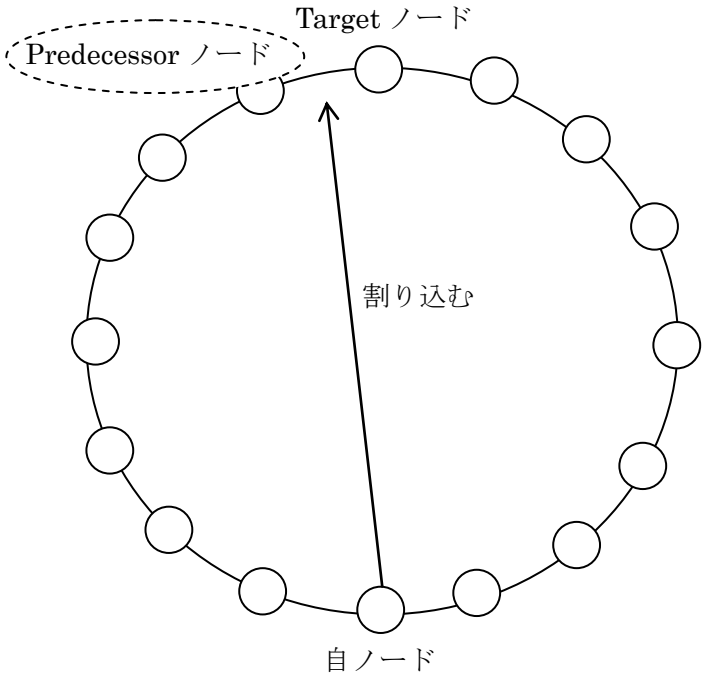


図 3.7　新ノード ID 決定

提案方式２の特徴は対象を移動するノードだけとし、通常の再参加処理の変わりに上記の手法で新しいノード ID を決定する。また、Target ノード決定に必要なフィンガーテーブルは保持しているので、Target ノードを決定するのに新たな通信は必要ない。

3.2.3　提案手法２の拡張

上記の提案手法２では、Target ノードを探す際に使用するフィンガーテーブルは自ノードが保持するフィンガーテーブルのみであった。これに加え他のノードが保持するフィンガーテーブルを取得し Target ノードに使用する。つまり、実ネットワークでより自ノードに近いノードを発見する確率が上がり、理想的なオーバーレイネットワークに近づくので、より検索効率が良くなると予想できる。一方で、自分が保持するフィンガーテーブルに載っているノードとは通信の必要がないのに対して、他ノードから取得するフィンガーテーブルに載っているノードは一度通信し遅延時間を計測する必要がある。ただし、重複しているノードはこの限りではない。

具体的に、どのノードのフィンガーテーブルを使用するかは図 3.8 に示す。

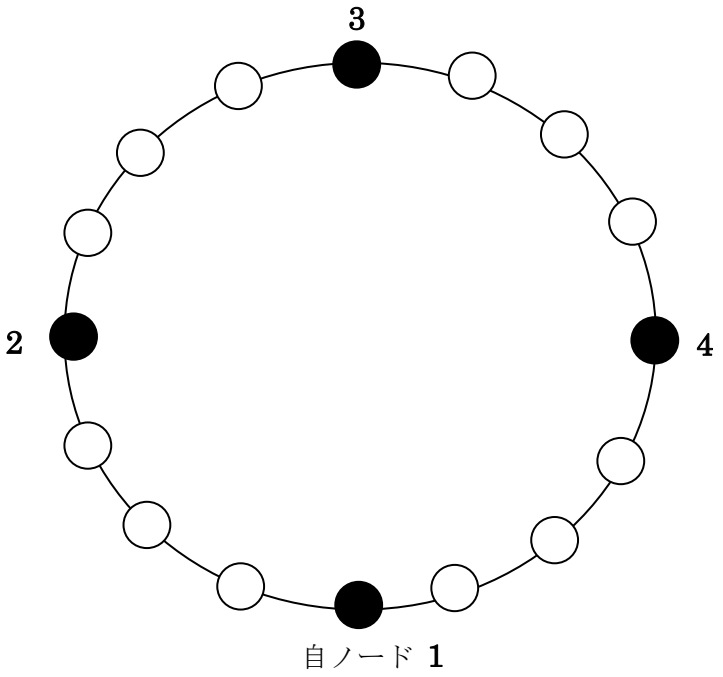


図 3.8　利用するフィンガーテーブルを持つノード

黒く塗りつぶされているノードの（前後のノードになる可能性が高い）フィンガーテーブルを Target ノードに利用する。番号 1 が自ノードになる。番号が 2 と 3 のノードは自ノードのフィンガーテーブルから選択される。ハッシュ空間が n bit であり自ノードの ID を X とすると、それぞれ $X+2^{n-2}(\mathrm{mod}2^{160})$、$X+2^{n-1}(\mathrm{mod}2^{160})$の位置に管理されているノードが選択される。番号 4 のノードについては自ノードのフィンガーテーブルでは管理していない。そこで番号 2 のノードから取得したフィンガーテーブルを参照し $X+2^{n-1}(\mathrm{mod}2^{160})$の位置か

ら番号 4 の情報を取得し、フィンガーテーブルを受け取る。または、番号 3 のノードが管理するフィンガーテーブルの $X+2^{n-2}(\mathrm{mod}2^{160})$に位置する情報でも同様、もしくは近い位置に存在するノードの情報が得られると考えられる。

次章で提案方式の有効性について検討するためにシミュレーションを行った。

第 4 章　評価

4.1　シミュレーション条件

本稿では、DHT を利用した代表的なアルゴリズム Chord を基礎とし、構築されるオーバーレイネットワークが実ネットワークを考慮しない問題点について、ID の再割り当てを行い、動的にオーバーレイネットワークを改善する手法を提案し、提案手法について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。

・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]を使用し構築した。トランジットノード 1600 台、スタブノード 16000 台で構成されている。ネットワーク図を図 4.1 に示す。
・オーバーレイネットワークに参加するノードはランダムに 1000 台選ばれる。
・オーバーレイネットワークに参加するノードのうち、移動するノードが参加する。割合は 10%から 90%まで 10%刻みで増加する。
・各ノードは検索を 50 回行い、検索要求の平均遅延時間を計測する。
・提案方式 1 では、各移動ノードの移動頻度はシミュレーション中に約 1 回及び 5 回の計 2 パターン、提案方式 2 では、1 回、2 回、3 回、4 回、5 回移動する、計 5 パターンを用意した。
・提案方式 2 で使用するフィンガーテーブルの数は最大で四つまで増える。
・関連研究で紹介したネットワークアドレスを利用した ID 生成を行う手法を従来方式とし、比較を行う。なお、本提案方式にも適用されている。

以上を条件にシミュレーションを行った。次節で結果を示す。

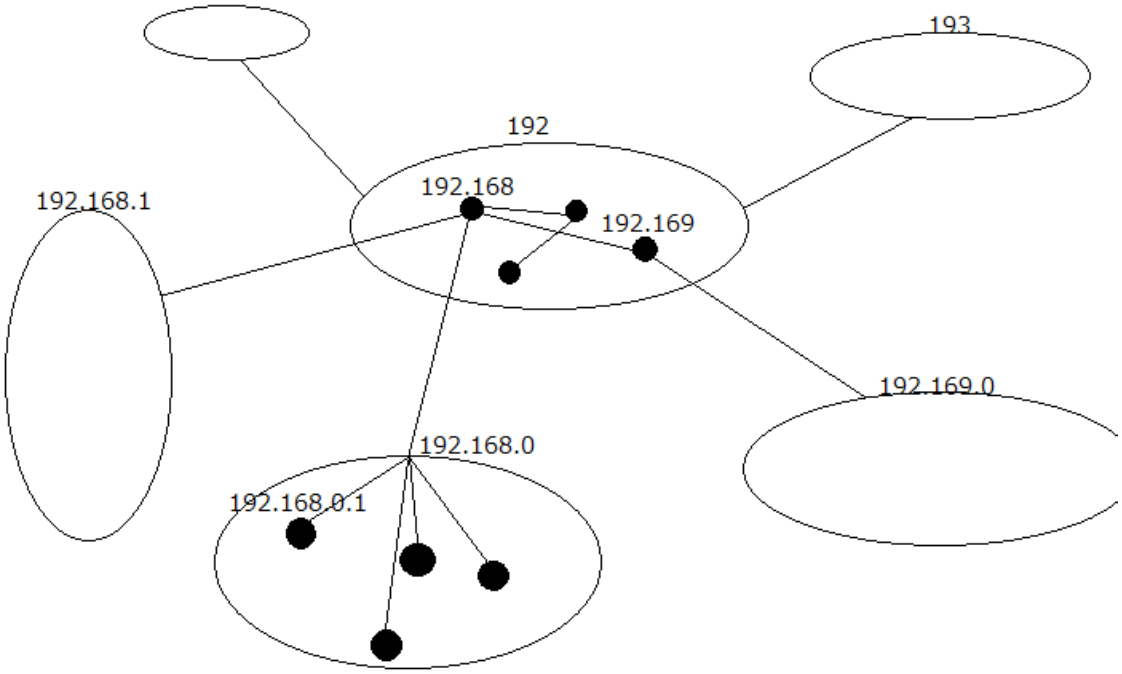


図 4.1　シミュレーションに利用したネットワーク図

4.2　シミュレーション結果

図 4.2 は提案方式 1 における移動ノードの割合が 40%時の検索 5000 回ごとの検索の平均遅延時間である。同様に図 4.3 は提案方式 2 の検索平均時間である。なお、移動するノードの移動回数は各ノード 1 回である。

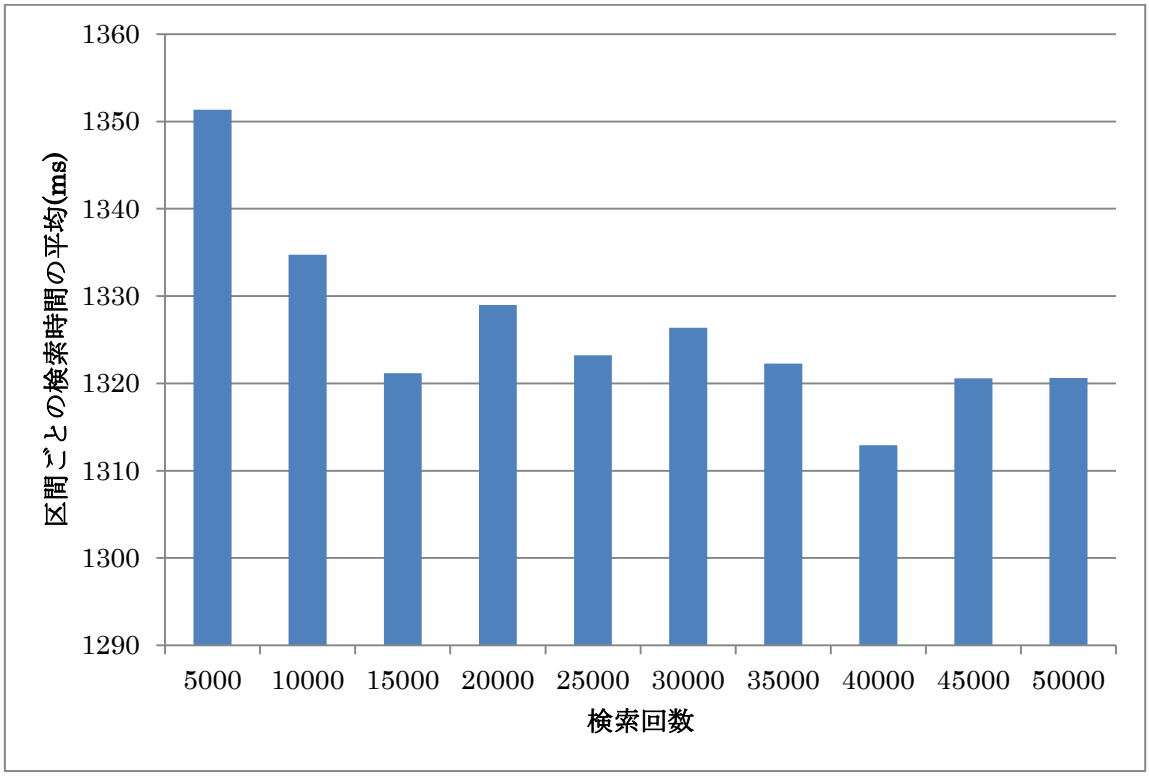


図 4.2　提案手法 1 の区間ごとの平均検索時間

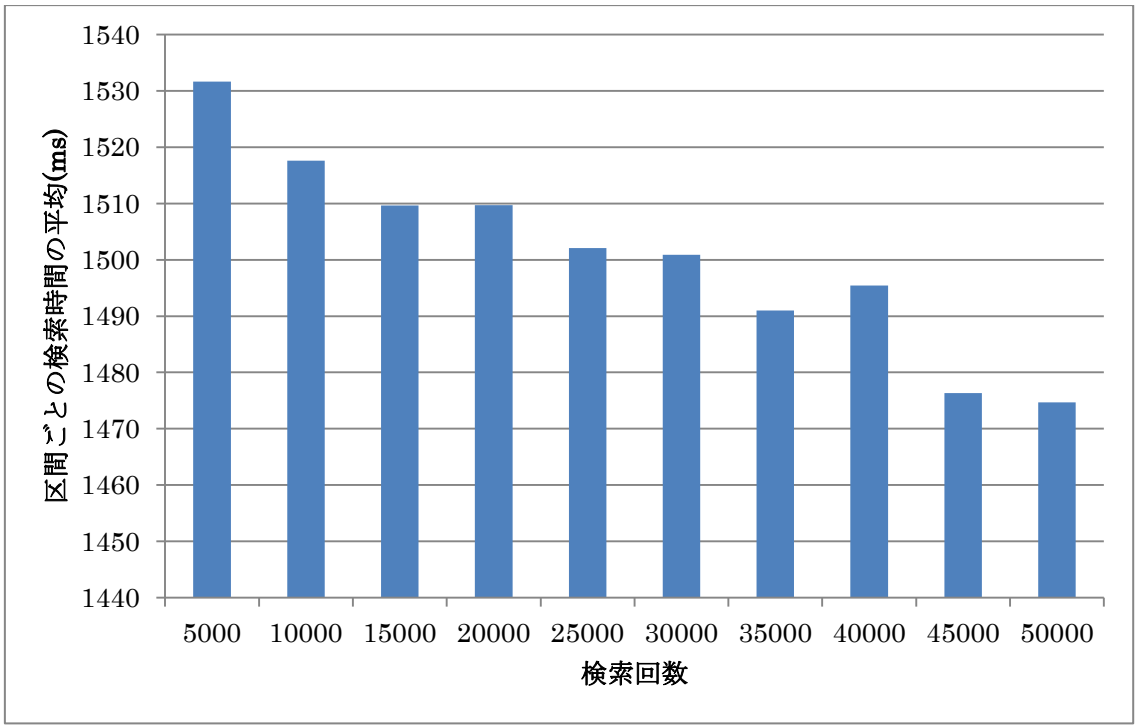


図 4.3　提案方式 2 の区間ごとの平均検索時間

図 4.2、図 4.3 共に最初の検索 5000 回の区間よりも、次以降の検索 5000 回の区間を比較すると、平均検索時間が削減されていることがわかる。よって ID の再割り当ては、オーバーレイネットワークの改善に有効であることが示された。図 4.2 において平均検索時間が上下している理由はノードの移動が発生しているからだと考えられる。

図 4.4 は従来方式と提案方式 1 の比較を行ったものであり、図 4.5、図 4.6 は従来方式と提案方式 2 の比較を行ったものである。なお、比較したデータは区間の最後であり、移動するノードの移動回数は各ノード 1 回である。また、提案方式 2 において、使用するフィンガーテーブル数は図 4.5 では 1 つである。

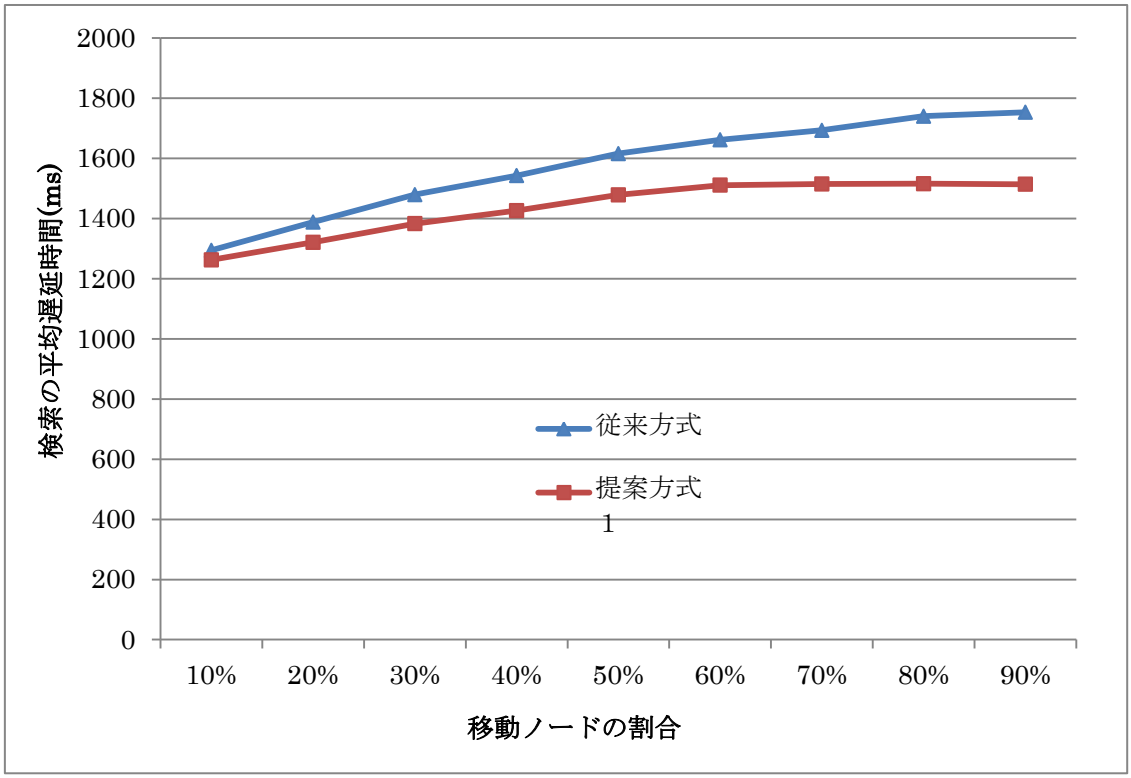


図 4.4　従来方式と提案方式 1 の検索の平均遅延時間の比較

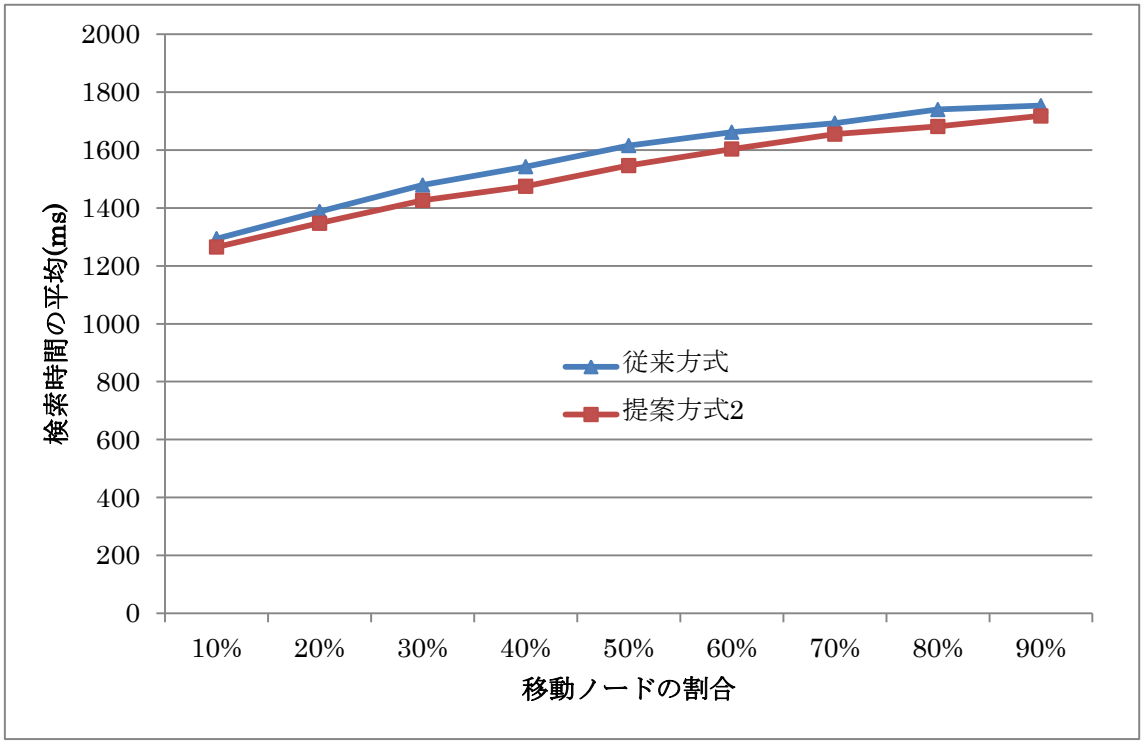


図 4.5　従来方式と提案方式 2 の検索の平均遅延時間の比較

従来方式と比べ提案方式 1 及び提案方式 2 は、移動するノードの割合全てにおいて、検索にかかる時間が削減されているのがわかる。提案手法 1 では特に移動するノードの割合が 90%の時に一番削減効果がでている。提案方式 2 は提案方式 1 ほど効果が出ていない。

図 4.6 は提案手法 2 において、使用するフィンガーテーブルの数を変えて評価した図である。なお、移動するノードの移動回数は各ノード 1 回である。

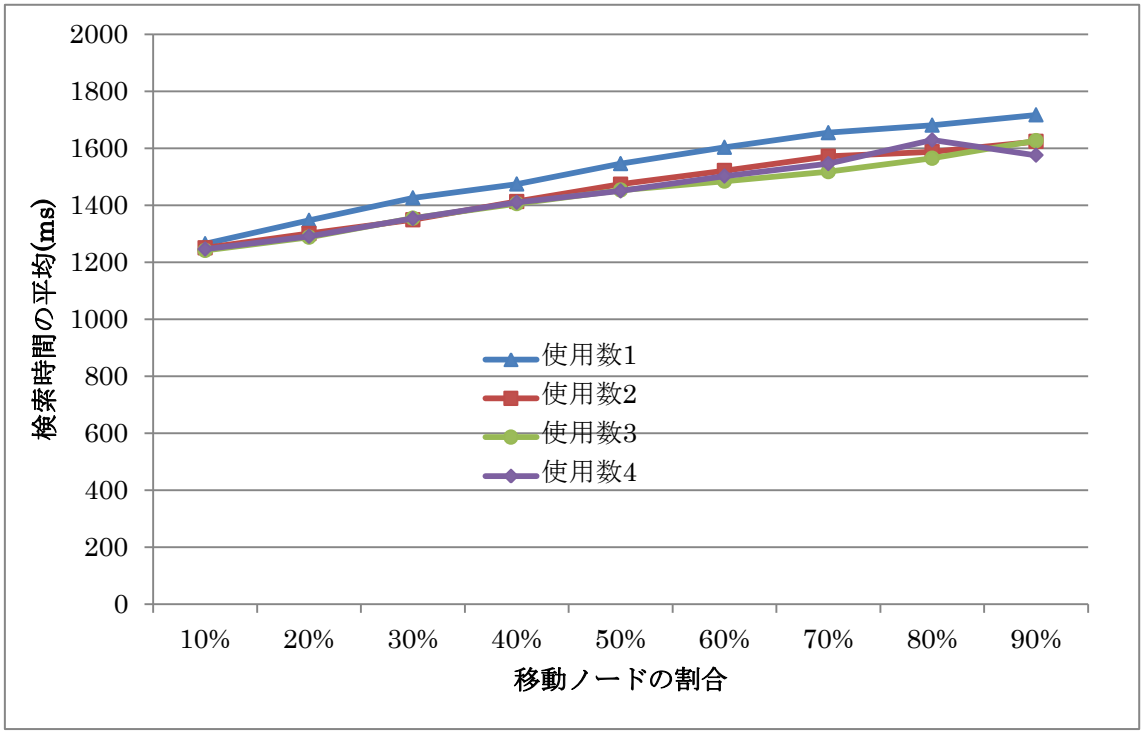


図 4.6　使用するフィンガーテーブル数変更の比較

図 4.6 から使用するフィンガーテーブルが 1 つの場合に比べ、使用数 2 以上は検索時間が削減されていることがわかる。一方で、使用数 2 以上ではあまり違いがない。

図 4.7 は提案手法 2 において、フィンガーテーブルの使用数を 1 とし、移動するノードが移動する回数を変えて比較したものである。また、図 4.8 はフィンガーテーブルの使用数を 4 にしてある。

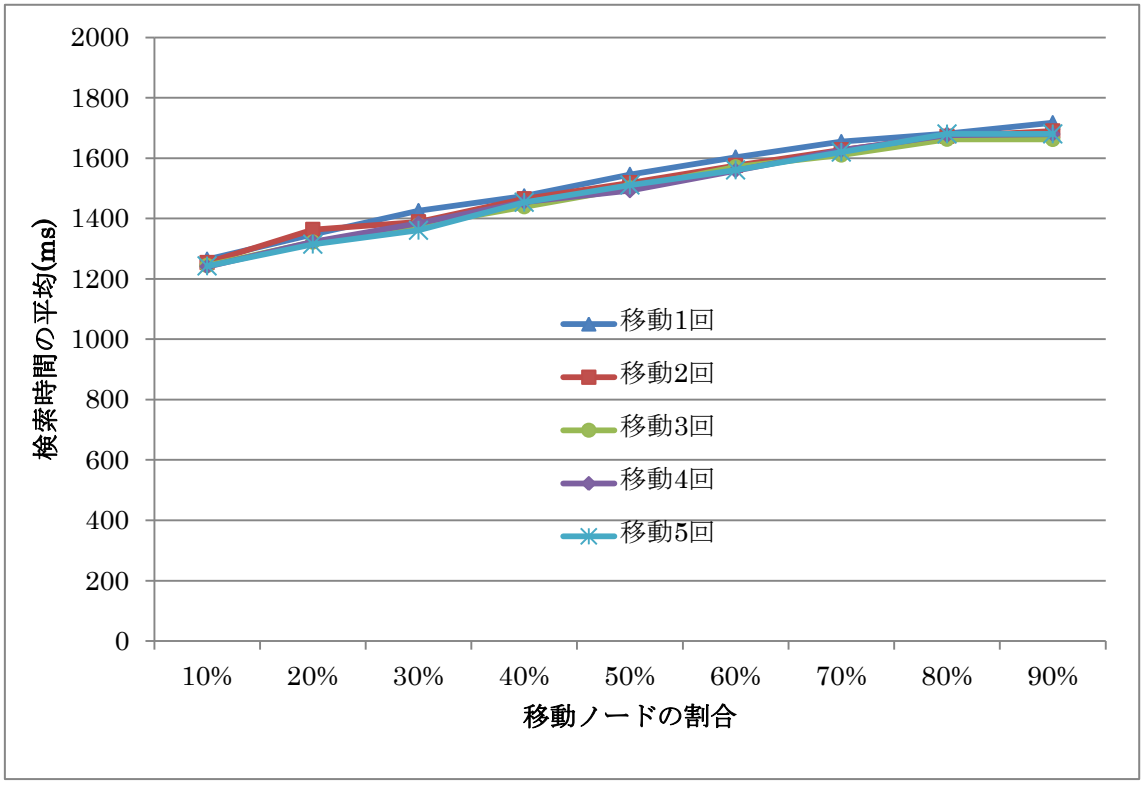


図 4.7　移動ノードの移動頻度変更時の比較(使用フィンガーテーブル 1)

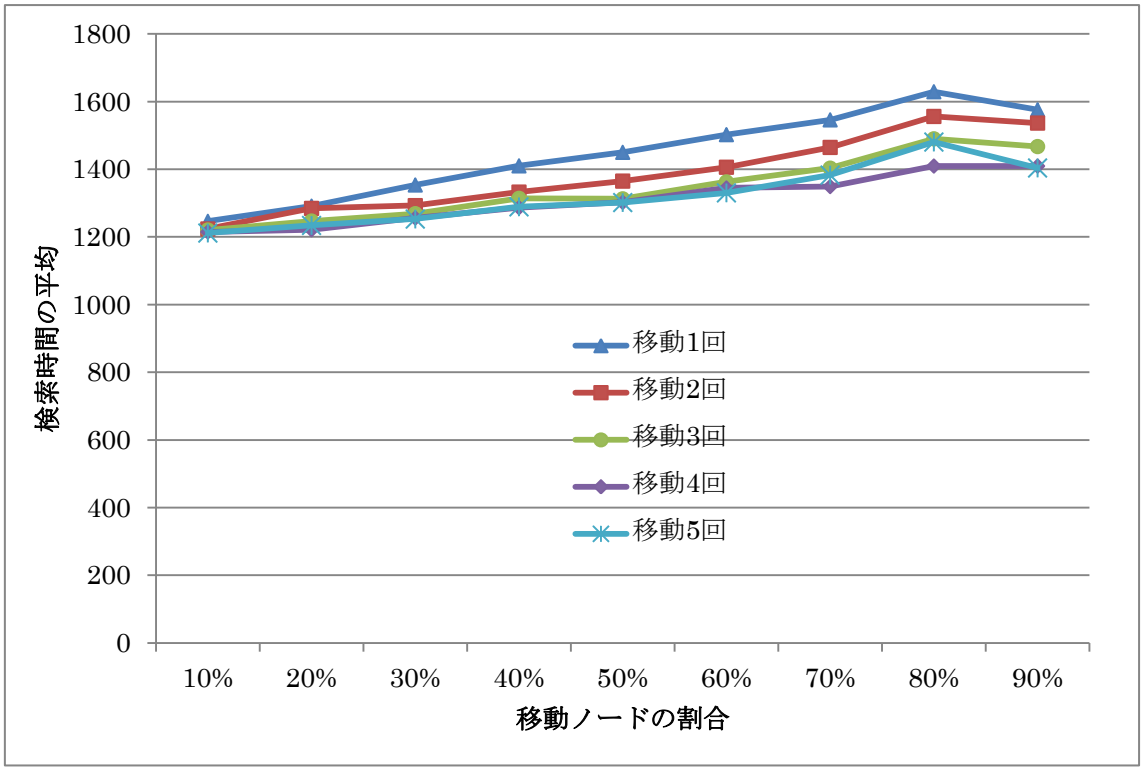


図 4.8　移動ノードの移動頻度変更時の比較(使用フィンガーテーブル 4)

図 4.7 では移動するノードの回数が変わっても、効果が出ていない。しかし、図 4.8 では移動回数によって変化が出ていることがわかる。これはノードの移動時にノード ID 再割り当ての処理が発生するため、移動回数が増えれば再割り当て処理も増え、オーバーレイネットワークが改善される可能性が高まるためだと考えられる。

最後に図 4.9 では従来方式、提案方式 1、提案方式 2 の比較を行う。条件は、移動するノードの移動回数を 5 回、提案方式 2 ではフィンガーテーブルの使用数を 4 つとした。

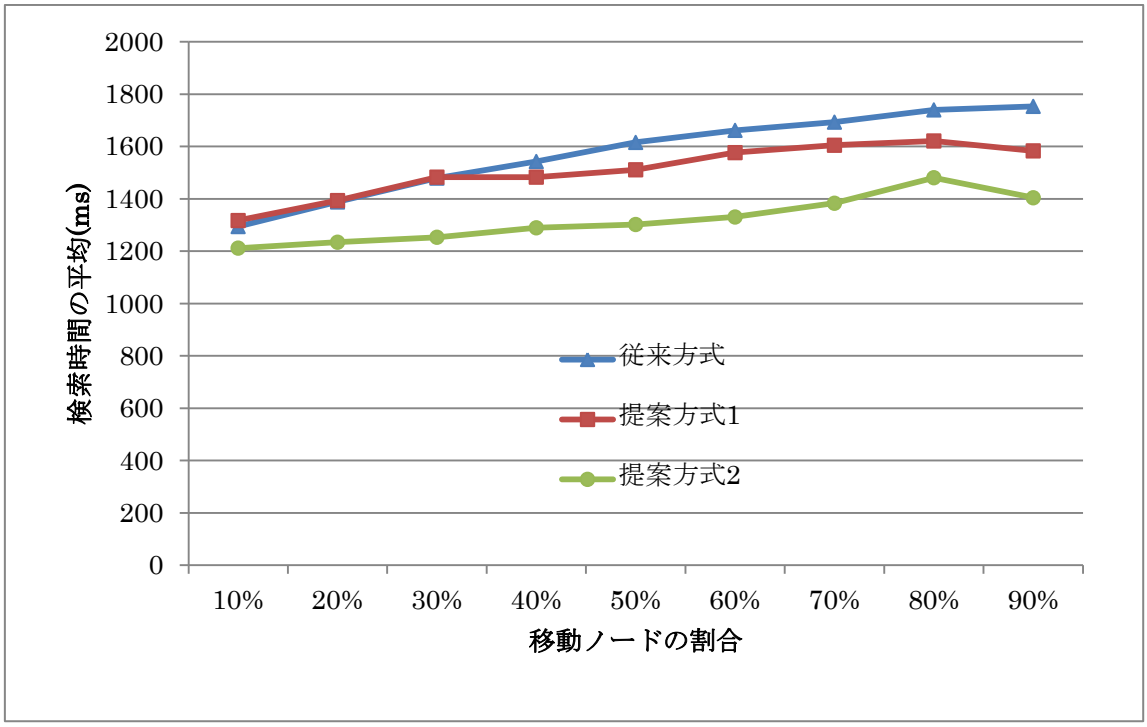


図 4.9　従来方式と両提案方式の比較

提案方式 1 は移動回数が増えることで、性能の劣化が見られた。これはノード ID の交換が移動ノードの移動に追いついていないことが原因だと考えられる。一方提案手法 2 は大きな改善が見られる。これは図 4.8 同様、移動回数が増えれば再割り当て処理も増えるからだと考えられる。従来方式、提案方式 1 と比較し、それぞれ平均で 16%と 12%、最大で 20%と 16%程検索時間が削減された。

4.3 考察

シミュレーション結果より、ID を動的に再割り当てすることによりオーバーレイネットワークを改善していく手法は有効であることが確認された。ただし、従来手法にはないオーバーヘッドが発生している。表 4.1 にオーバーヘッドについて示す。

表 4.1 処理のタイミングとオーバーヘッド

	提案手法 1	提案手法 2
タイミング	検索完了時	ノードの移動時
オーバーヘッド	検索ホップ数(O(logN))×2+ ID 交換処理に必要なメッセージ数(最大で 14)	最大で 30×3×2 のメッセージ
交換コスト	管理領域+ フィンガーテーブル の交換	交換ではないのでなし

提案手法 2 より、提案手法 1 のオーバーヘッドが大きいことがわかる。また、提案方式 1 の場合ノード ID の再割り当てが非常に多い。今回のシミュレーション中に、移動ノードの割合×35 回程度の ID 再割り当てが発生していた。そしてノード ID を交換するに至るまでの処理が複雑であること。またノード ID の交換には、対象ノードの管理領域及びフィンガーテーブルの交換が必要になる。

一方、提案手法 2 ではオーバーレイネットワークに再度参加が必要になった際に処理が発生する。よって提案手法 1 のように ID 交換のコストが必要ない。また、提案手法 2 において必要とするコストは、フィンガーテーブルの使用数に応じて増加する。しかし、160bit のハッシュ空間に 1 億台のノードが参加していた場合、フィンガーテーブルの 131 エントリが重複している。よって、必要となる通信は最大で 30×3×2（フィンガーテーブルを 4 つ使用した場合）である。この数字は Chord が行うフィンガーテーブルの更新に必要なメッセージの数より非常にすくない。

また、提案手法 1 は検索の完了を持って ID 再割り当ての処理がスタートするが、提案手法 2 ではノードの移動が発生すると（IP アドレスが変更される）ノード ID 再割り当ての処理がスタートする。よって、図 4.9 のように提案手法 1

は移動に対応しきれなくなるが、提案手法 2 はある程度の移動には耐えられると考えられる。

第 5 章 結論

DHT を利用した Chord アルゴリズム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していない。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。関連研究ではその問題点に対し、ノードがオーバーレイネットワークに参加する際、実ネットワークの状況を考慮し ID を付与する手法をとっていた。しかし、近年携帯端末が増加していることから、ネットワークの状況が時間と共に変化する。そのため関連研究のように、オーバーレイネットワーク参加時のみの実ネットワークの情報では対応できなくなってきた。そこで本稿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、検索要求が通過したノードの中から近いノードを発見し、オーバーレイネットワークが改善されると判断された場合ノード ID を交換する。この処理を続けていくことで、実ネットワーク上で近いノードをオーバーレイネットワーク上でも近い位置に配置することを可能にする。もしくは、ノードの移動と同時に再参加するための ID を、保持していたフィンガーテーブルの情報から、自ノードに近いノードを選択し新たな ID を付与することで、前述した方式同様オーバーレイネットワークに実ネットワークの状況を反映させることを可能にする。このようにオーバーレイネットワークを構築した後にもなんらかのタイミングで実ネットワークの情報を取得し ID 再割り当ての処理行いオーバーレイネットワーク参加時だけでなく、その後の実ネットワークの状況を動的に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。従来方式と比較した場合、少ないオーバーヘッドで最大 20%の検索時間が削減されるという結果が得られた。最終的に、提案手法 1 と 2 を比較し、動的に ID の再割り当てを行う際に発生するオーバーヘッド、また移動するノードのスピードに対応する能力から、提案手法 2 がより望ましいと結論づけた。

謝辞

本研究を進めるにあたって、研究全般にわたり適切なご指導、ならびに助言を賜りました、本学理学部情報科学科の佐藤文明教授に心から感謝いたします。また様々なことにおいてご協力、アドバイスをいただいた佐藤研究室のメンバーならびに友人たちに感謝いたします。

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