キャッシュを用いたChordの検索負荷分散方式

磯田賢志 [1,a)]　　佐藤文明 [1,b)]

P2P の形態でコンテンツを検索する技術に分散ハッシュテーブル（DHT）がある。DHT は、ネットワークに大量の問い合わせメッセージが拡散することがなく、また検索負荷が集中することがないことから、IP 電話やファイル共有などに利用されている。Chord は、DHT の一つであり、ピアによって論理的なリングを構成する検索アルゴリズムである。DHT においても、人気のあるコンテンツは検索が偏り負荷の集中を招く問題が生じる。本研究では、キャッシュを用いることで負荷の分散を検討する。キャッシュの管理の方式は、負荷の状況を監視する必要があったり、削除の手間がかかるものであった。本研究ではクエリの到着頻度に応じて、キャッシュの作成確率を高くすることで、人気が高いノードにはキャッシュが作成されやすくなる方式を提案する。また、キャッシュに有効期限を設けることで、アクセスがなくなったキャッシュは自動的に消去される方式にすることで、管理の手間を掛けない方式を提案する。この提案方式を、シミュレーションによって評価し、従来方式と比べて検索ホップ数を改善するとともに、保持すべきキャッシュの量を削減できることを明らかにした。

1. はじめに

多人数でのファイル共有や、ファイル配信、IP 電話のアドレス検索といったサービスにおいて、従来のクライアント・サーバモデルに代わり P2P（Peer to peer）技術が注目されている[1-4]。P2P はサーバを用いず、エンドユーザ同士で直接通信を行うという特徴を持つ。P2P におけるコンテンツの情報検索では、コンテンツのインデックス情報のみをサーバで集中管理するハイブリッド型、完全にサーバを持たず検索メッセージをネットワーク上にフラッディングするピュア型がある。それぞれ、サーバがボトルネックになる点、ネットワークに過大な負荷がかかる点が欠点として挙げられている。これらの問題を解決する方式として、分散ハッシュテーブル（DHT）が提案されている。

DHT はコンテンツやノードの識別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自動的に決定して、検索対象の探索を高速化する技術である。ノードは DHT によりフラットなオーバレイネットワークにまとめられている。この DHT をアルゴリズムとしたネットワークは、構造化（structured）オーバレイ[5]と呼ばれる。

一般に、P2P システムの負荷は特定の人気のある情報を管理しているノードに集中する特徴がある。従って、P2P のアクセス効率を高めるためには、人気のある情報に対する検索に対して効率的に応答するシステムが必要となる。DHT の一つであるＣＡＮに対して、検索要求の通過する経路が集中するノードに対して、キャッシュノードを設置して負荷を分散する方式が提案されている。しかし、この方式ではどのノードに負荷が集中しているかの情報を集めるコストや、キャッシュノードを起動停止する管理コストがかかる問題がある。一方、DHT の一つの Chord[7]では、隣接するノードに情報をキャッシュしておくことで、負荷を分散する方法が提案されている。しかし、コンテンツの人気に応じてキャッシュが配置されていないため、無駄なキャッシュが配置されるという問題があった。

本研究では、DHT の一つである Chord において、コンテンツの人気に応じた確率でキャッシュを作成し、無駄なキャッシュがない効率的なキャッシュ配置方式を提案する。また、キャッシュに対して有効期限を設けることで、キャッシュが使われなくなった時点でキャッシュが削除されるようにした。その結果、キャッシュを管理する手間がかからないという利点がある。シミュレーションによって、その有効性を評価した。

2. Chord の概要

2.1 分散ハッシュテーブル DHT

DHT は、P2P においてサーバを必要としないデータ検索システムを構築するのに用いられる。基本的な仕組みは、コンテンツとノードの両方に同じハッシュ関数のハッシュ値を割り当て、コンテンツのハッシュ値を受け持つピアへ分散配置する。検索する際には、検索したいコンテンツのハッシュ値を計算し、そのハッシュ値を受け持つノードに順々に問い合わせることで検索を行う。その後、ピア間で直接通信を行う。この DHT は、前節で述べた HybridP2P 型や PureP2P 型よりもスケーラビリティが高く、全てのノードに対して検索でき、負荷の分散などの点で優れている。

DHT を利用した検索アルゴリズムとして CAN[6]、Chord[7]、Pastry[8]、Tapestry[9]、Kademlia[10]などがある。本研究では、他の検索アルゴリズムに比べて比較的単純かつ、耐故障性に優れている Chord を対象とした。

2.2 Chord

Chord は、Stoica らが提案した DHT 検索アルゴリズムの一つである。ノードの参照やネットワークへの参加・脱退にサーバを必要とせず、各ノードが完全に分散して処理を行う。Gnutella 等の検索クエリをフラッティングするネットワークにおいては、検索可能ノード数Ｎの増加に伴い線

1 東邦大学理学部
Toho University, Faculty of Science,
2-2-1 Miyama, Funabashi, Chiba, 274-8510, Japan.

a) 5513010i@nc.toho-u.ac.jp
b) fsato@is.sci.toho-u.ac.jp

形のパケットの増加が起こるが、Chord ではパケット数の増加はO(logN)であり、規模拡張性に優れている。

各キーIDとノードIDはSHA-1というハッシュ関数を用いて作成され、同じ空間にマッピングされる(キーID＝hash(key)、ノードID＝hash(IP address))。160bitのID空間の場合、１０の４８乗もの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ほぼ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6bit(0～63)のID空間を用いて説明を行う。

Chordネットワークにおいて、IDを持った各ノードは仮想的なリングを形成し、時計方向に接続を行っている。図1ではN1→N8→N15→N22→N31→N36→N43→N47→N52→N56→N1のように接続されている。この接続はルータ同士の接続ではなく、ノード同士で作られる接続である。そのため、ノード間には複数のルータが存在し、ネットワークの近さをまったく考慮しない仮想的なネットワークである。

各キーの値とデータのペアは、時計回りに最も近いノードに格納され、このノードを代表ノードと呼ぶ。Key10は時計回りに最も近いノード15に格納され、Key24、Key30はノード31、Key36はノード36に、Key54はノード56に格納される。

図2にKey54を検索する際のクエリの転送状況を示す。Chordにおけるクエリの転送方法における特徴から、人気のある情報へのクエリが特定のノードへのクエリの集中を招き、同時にそのノードの直前のノードにそのクエリの転送要求が集中することが分かる。従って、クエリの集中するノードの負荷を分散することで、応答時間を改善できる可能性がある。しかも、その負荷を分散すべきノードは直前のノードであることがわかる。

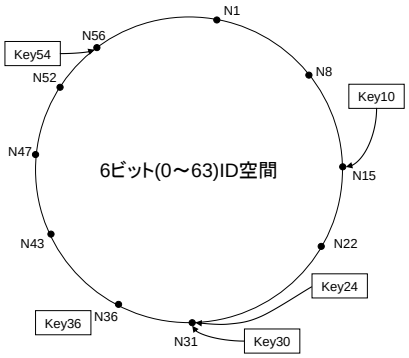


図1　キーIDのマッピング

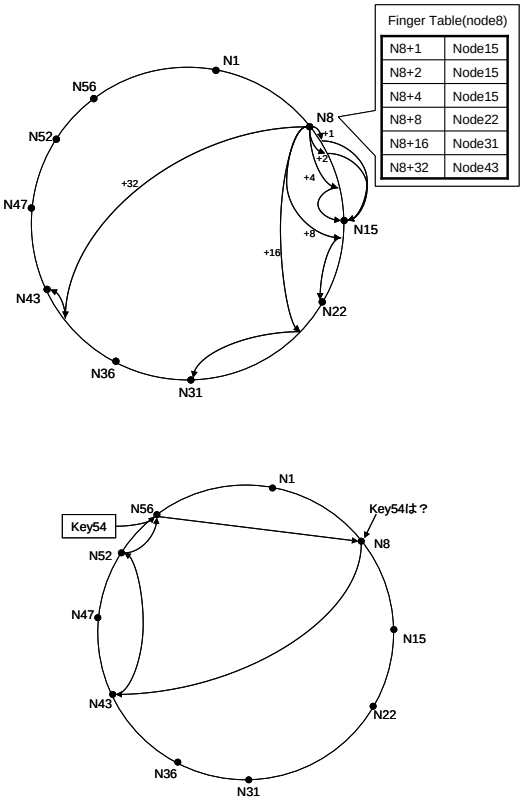


図2　高速な参照方法(key54を検索)

2.3 DHTへのキャッシュの導入

Chordにキャッシュを導入することで検索負荷を分散し、検索時間を短縮する方式が提案されている。Chordのリングで隣接するノードに､インデックス情報をキャッシュ(複製）する方法が提案されている。特に直前ノードインデックス情報をキャッシュしておくことで、オリジナルのノードにアクセスするクエリと、直前ノードにアクセスするクエリとに負荷を分散することができる。また、直前ノードが検索結果を返すことから、検索の転送回数（ホップ数）が削減されて、応答時間が短くなる利点もある。

ただし、インデックス情報は事前にアクセス頻度が分からないため、キャッシュはすべてのノードについて行われる。従って、アクセスセス頻度が低くキャッシュが有効でない場合でもキャッシュが作られる問題がある。

CANにキャッシュを導入することで検索負荷を分散し、検索時間を短縮する方式が提案されている。特にP2Pネットワーク上でのキャッシュの配置方法が考察されている。

CC方式では、各ノードが一度検索したインデックスをローカルの記憶領域上にキャッシュし、同一の検索を行う場合にはキャッシュを利用する。またキャッシュを他のノードと共有することで､キャッシュの利用性を高めている。

例えば、他のノードからの検索要求を転送する前に、自身のキャッシュ内に対応するインデックスを保持しているかを確認し、保持している場合は検索要求を転送せずにキャッシュを利用して応答する。

CCPR 方式は CC 方式を拡張した方法である。CCPR ではキャッシュの効果を高めるために、中継ノードと呼ぶノードを設定し、中継ノードにもそのインデックスをキャッシュする。検索時には、中継ノードを経由する経路を採用することで、他のノードが検索した同じ要求に対するキャッシュを利用することができる。

中継ノードは、検索要求が多数経由する経路に配置され、検索要求を効率的に処理することができる。しかし、検索要求が少なくなったときには、中継ノードはオーバーヘッドになるために、削除される。これらの制御を行うために、各ノードの負荷の情報を定期的に収集し、負荷に応じて中継ノードを設置したり削除するための管理コストが必要となる。

3. Chord キャッシュ管理方式の提案

3.1 目的

人気のあるコンテンツに対してクエリ（問い合わせ）が集中する問題に対して、予めキャッシュ（または複製）を配置して、負荷を分散する方式が提案されてきた。しかし、予めどのコンテンツに人気がでるのかは分からないため、すべてのコンテンツに対してキャッシュが作成されることで無駄なキャッシュが作られることがあった。また、人気が出ることで負荷が高くなったノードにキャッシュを配置する方式が提案されているが、ノードの負荷情報を定期的に収集する手間や、人気がなくなった時点でキャッシュを削除するなどの管理上のコストが大きくなる問題があった。

本研究ではクエリに対して、確率的にキャッシュを作成することで、人気があるノードにはキャッシュが作成されやすく、人気のないコンテンツにはキャッシュが作成されにくい方式を提案する。また、キャッシュに有効期限を設けることで、人気がなくなりアクセスが少なくなったキャッシュは自動的に消去される方式にすることで、管理の手間が少ない方式を提案する。また、キャッシュ作成ではオリジナルのノードから１回だけコピーされる単一モードと、キャッシュからさらにキャッシュが作成される反復モードとがあり、それぞれ評価を行う。

3.2 キャッシュの作成と削除

キャッシュは、検索が成功したノードの直前のノードに作成されるが、常に作成されるわけではない。なぜなら、利用頻度の低いキャッシュを作っても無駄になるからである。本研究では、キャッシュをある確率（キャッシュ確率）で作成することとした。この結果、検索頻度が高いノードでは１回１回のキャッシュ確率が小さくても、全体としてキャッシュが作成される確率が高くなる。一方、検索頻度が低いノードではキャッシュが作成されにくくなる。

また、キャッシュが一度作成されると、それをどのように削除するかが難しいが、我々は有効期限方式をとった。これは、有効期限をキャッシュに設定しておき、有効期限が切れた場合キャッシュを自動的に削除する方式であり、集中的な管理や負荷状態の通知などの手間が不要である。

図３に基づき、キャッシュ作成の流れを説明する。図３ではノード１がノード１４を検索している。ノード１４は、検索結果をノード１に返すとともに、最終ホップの直前ノードであるノード１３に対して、キャッシュ作成依頼を行う。ノード１３は、ノード１４から依頼されたキャッシュを持っていない場合、システムに設定されたキャッシュ作成確率に基づき、確率的にキャッシュを作成する。もし、すでにキャッシュができている場合は、作成しない。また、図３のノード６からノード１４へのクエリのように、もし自分が持っているキャッシュにヒットするクエリが来た場合、キャッシュにある情報を検索元に返す。また、キャッシュの有効期限を延長する。

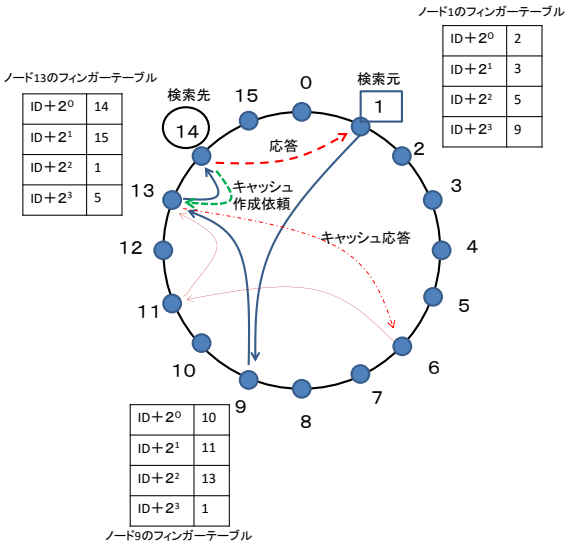


図３　キャッシュ作成手順と、キャッシュからの応答

3.3 反復キャッシュ

キャッシュは、検索が成功したノードの１ホップ前のノードに作成される。つまり、本方式は Chord の検索ルート上にキャッシュを配置することになる。この考えを拡張して、キャッシュにヒットしたノードの１ホップ前のノードに更にキャッシュを配置することを考える。これを本研究では反復キャッシュ（キャッシュのキャッシュ）と呼ぶ。反復キャッシュは、キャッシュの増大につながるが、キャッシュ確率と有効期限の調整によって適切に管理すれば、

検索対象と検索元との間に適切にキャッシュが配置されると考えている。

4. 評価

4.1 シミュレーションの条件

提案方式の有効性を評価するために、シミュレーションを実施した。シミュレーションの条件は、以下の通りである。

表１　シミュレーション条件

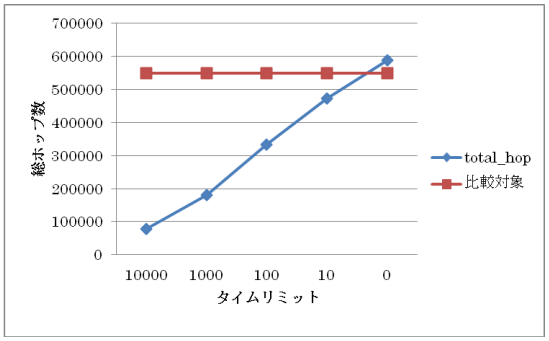
項目	値
ノード数	128, 256, 512, 1024
クエリの回数	100,000 回
クエリ頻度	0.005 回/秒/ノード
キャッシュ確率	0, 0.1, 0.5, 1
キャッシュの有効期限	0, 10, 100, 1000, 5000, 10000
コンテンツの数	100
検索要求発生確率	Zipf の法則に基づく
ハッシュ値の最大値	32768（2 の 15 乗）
通信遅延	10ms/ホップ

実験は、条件を変えて総検索ホップ数、ノードに保持された平均総キャッシュ数を測定した。図４は、反復キャッシュを導入した提案方式と、隣接ノードに固定でキャッシュを配置する比較対象方式とで、総ホップ数を比較した結果である。なお、キャッシュ確率は１である。

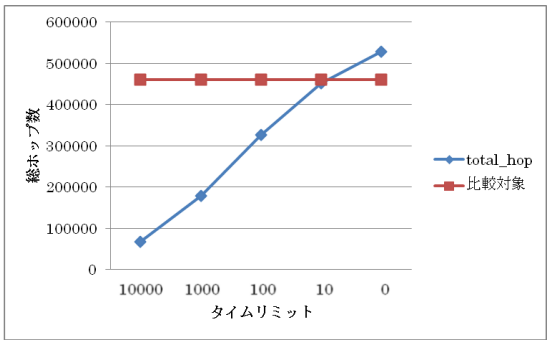
図４より、提案方式は、キャッシュの有効期限を長くするほど、キャッシュの配置量が増えることから、総ホップ数が減少することが分かる。

また、比較対象方式とキャッシュ数を比較すると、図５のようになった。比較対象は、コンテンツ数である 100 個のキャッシュで済むのに対して、提案方式は有効期限を長くすると急激にキャッシュ量が増大する傾向にある。

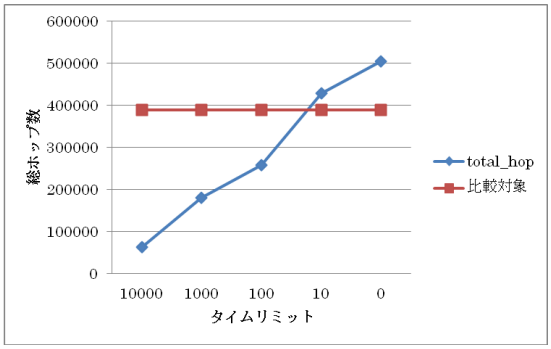
しかしながら、例えばノード数が 128 の場合、キャッシュの有効期限を 100 に設定すれば、総ホップ数とキャッシュ数ともに比較対象を下回ることがわかる。また、キャッシュの作成においても、検索対象から遠く離れた位置にキャッシュを配置してもヒット率は見込めないことから、検索対象から遠いノードでのキャッシュを制限する方法などが考えられる。



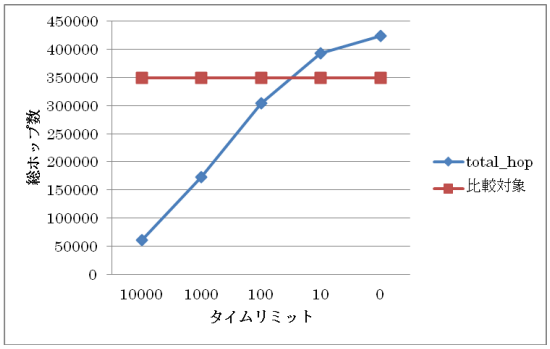
（ａ）ノード数 1024



（ｂ）ノード数 512



（ｃ）ノード数 256



（ｄ）ノード数 128

図４　キャッシュの有効期限に応じた総ホップ数

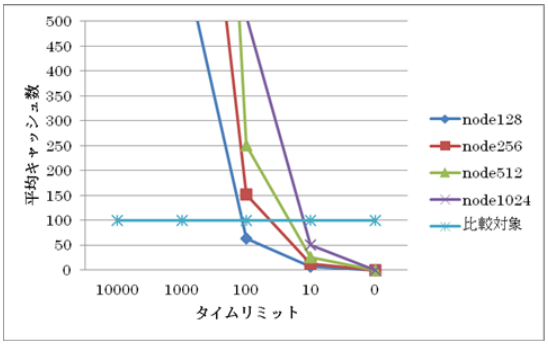


図 5　キャッシュの有効期限に応じた作成キャッシュ数

キャッシュの確率を変化させた場合の比較を行った。図 6 にキャッシュ確率を変化させたときの総ホップ数を示す。また、図 7 にキャッシュ作成回数を示す。キャッシュ確率が高くなると、キャッシュの作成される割合が高まるため、キャッシュへのヒット率が増えることで総ホップ数が削減される。しかし、キャッシュが大きくなると、その分キャッシュを保持するためのコストがかかる。

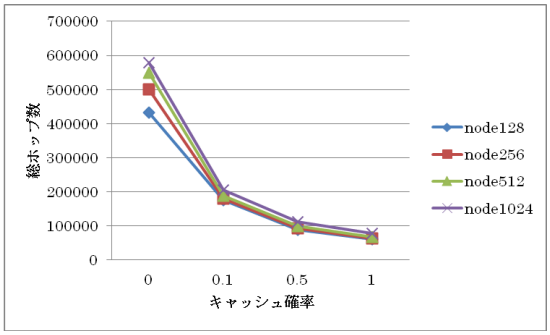


図 6　キャッシュ確率に応じた総ホップ数

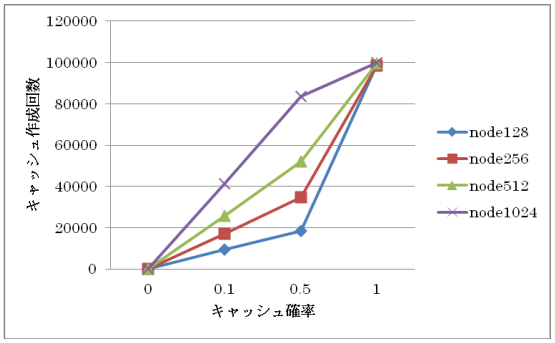


図 7　キャッシュ確率に応じた、キャッシュ作成回数

また、反復キャッシュの効果を見るために、反復キャッシュと単独キャッシュとの比較を行った。反復キャッシュを使うことで、図 8 に示すように総ホップ数は大きく削減することが可能である。一方で、図 9 に示すように、平均キャッシュ数は非常に大きくなる。従って、キャッシュ数に制限があるような環境では、反復キャッシュは利用しにくい。そのため、適切なキャッシュの保持時間（タイムリミット）や、キャッシュ確率を設定することで、平均キャッシュ数を少なく制限して利用することが必要となる。

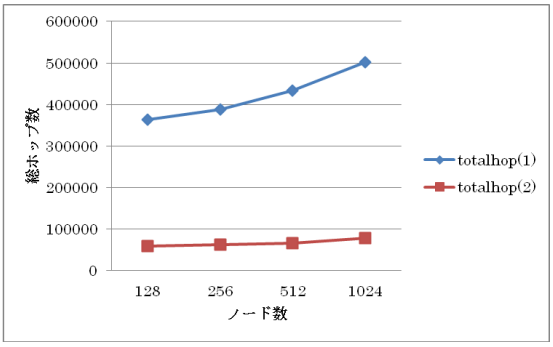


図 8　反復キャッシュを利用する場合と利用しない場合の総ホップ数比較（Totalhop(1) 単独キャッシュ、Totalhop(2) 反復キャッシュ）

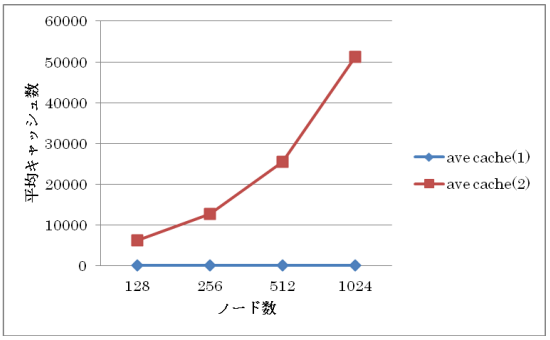


図 9　反復キャッシュを利用する場合と利用しない場合の平均キャッシュ数比較（Ave cache(1) 単独キャッシュ、Ave cache(2) 反復キャッシュ）

5. まとめと今後の予定

本研究では、P2P ネットワークシステムにおける検索負荷が人気のあるコンテンツに対して集中する問題に対し、適切なキャッシュを動的に配置することで性能を改善する方式を提案した。P2P ネットワークとしては、分散ハッシュテーブルの一つである Chord アルゴリズムを対象とした。提案方式をシミュレーションによって評価した。検索要求の発生確率は、Zipf の法則と呼ばれる確率分布を用いて作成した。その結果、従来の Chord において隣接ノードに全情報をキャッシュする方式にくらべて、提案方式はキャッシュ効率が良く、ホップ数も少なくてすむ設定が可能であることを明らかにした。また、キャッシュのキャッシュ（反復キャッシュ）を入れる場合と入れない場合についてもその性能を明らかにした。反復キャッシュを入れると、単一

のキャッシュでは得られないホップ数削減効果が得られる。ただし、キャッシュの量は非常に多くなる。キャッシュ量について、制約が少ない状況であれば、反復キャッシュは非常に有効な方式であることを明らかにした。

今後は、より詳細な評価を行い、キャッシュの配置場所、配置・削除のタイミングの制御方法を検討していく。また、参加脱退が多いモバイルノードで構築された場合の信頼性を向上させるためのキャッシュ（または複製）に関する研究を進めていく。

参考文献

1) ”Napster.” http://www.napster.com/
2)”Gnutella.” http://www.gnutelliums.com/
3) 金子 勇，アスキー出版社，“Winny の技術”，2005
4) ”Skype.” http://web.skype.com/intl/ja/
5)首藤一幸，田中良夫，関口智嗣:"オーバレイ構築ツールキット Overlay Weaver",情報処理学会論文誌: コンピューティングシステム, Vol.47, No.SIG12 (ACS 15), pp.358-367, 2006.
6)Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, “A Scalable Content-Addressable Network”, In Proceedings of ACM SIGCOMM, 2001.
7)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.
8)Antony Rowstron; Peter Druschel: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, Lecture Notes in Computer Science 2218/2001. Springer Berlin, p. 329, (2001).
9)Ben Y. Zhao, John D. Kubiatowicz, Anthony D. Joseph:“Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing”. Technical Report: CSD-01-1141. Berkeley, CA, USA: University of California at Berkeley, (2001).
10) Petar Maymounkov and David Mazières, Kademlia: A Peer - to - peer Information System Based on the XOR Metric. In 1st International Workshop on Peer-to-peer Systems (IPTPS'02), 2002.