2007－EVA－22 (2)
2007／8／2

センシングデータを考慮したP2Pネットワーク上での並列ダウンロード性能に関する一考察

洞井晋一　松浦知史　藤川和利　砂原秀樹

奈良先端科学技術大学院大学 情報科学研究科

概要

これまでP2Pやオーバーレイネットワークと呼ばれる仮想的なネットワーク上では，大きなサイズのデータを共有することが主な目的とされてきた．例えば映像データや音楽データ，ISOのイメージなどである．しかし，データに対するメタ情報や利用者からのコメント，またセンサから得られるセンシングデータのように，サイズの小さなデータに対する注目が高まっている．このような小さなデータは数多く集めることで有意義な情報になり得る．そこで，サイズの小さなデータをオーバーレイネットワーク上で共有したときにデータのダウンロードに必要な時間がどのように変化するかを本論文では実験を通して考察した．性能の変化を測るためにオーバーレイネットワーク構築ツールキット「Overlay Weaver」を用いて実験を行い，ダウンロードに必要となる時間を計測した．この実験結果から，小さなサイズのデータをオーバーレイネットワーク上で共有するには多くの計算機に分散させるよりも，ある程度の大きさのデータを少数の計算機に集約する方ダウンロードに必要な時間を短縮できることが分かった．

A Study about Performance of Parallel Downloading on P2P Network Considering Sensing Data

Shin'ichi Doui, Satoshi Matsuura, Kazutoshi Fujikawa, Hideki Sunahara

Nara Institute of Science and Technology Graduate School of Information Science

Abstract

P2Ps or Overlay Networks have aimed for sharing large data such as music data, video data, or ISO image data. On the other hand, small data such as comments, metadata, or sensing data have gotten a lot of attention recently. These data become useful information in the case that data are aggregated. We study performance of downloading small data on P2P networks. To time downloading small data, we use "Overlay Weaver" which constructs overlay networks easily. We experiment with overlay networks constracted 10 computers to time downloading small data. The result of experiment shows that the performance of downloading aggregated data is better than the performance of downloading small data respectively.

1　はじめに

P2Pやオーバーレイネットワークと呼ばれる，既存のネットワーク上に構築された仮想的なネットワークが注目されている．オーバーレイネットワークはNapstar[3]やGnutella[2]，BitTorrent[1]といったものを始めとして，日本ではWinnyのようにコンテンツを共有するものが有名である．ここで共有するコンテンツの主な対象はサイズの大きなデータであるといえる．例えば，Linuxのディストリビューションの1つであるFedoraでは，インストールに必要なディスクイメージをBitTorrentを用いて配布している．また，Winnyでは映画のDVDや音楽のCDなどが余りにも容易に共有できるため，著作権上の問題が多いことでも知られている．これは，サイズの大きなデータを効率的に共有することが従来のサーバクライアントなシステムでは困難であるという背景がある．ユーザがサーバからサイズの大きなデータをダウンロードする場合，ユーザ側の帯域に関らずサーバ側の性能によってダウンロードに長い時間を要することになる．また，多くのユーザがサーバからダウンロードすることで，ダウンロードの性能

を劣化させる要因になり得る．オーバーレイネットワークではこうした問題を解決するために，サイズの大きなデータを分割して複数の計算機からダウンロードすることなどが行われている．

このようなオーバーレイネットワークの技術は構築するネットワークのトポロジが規定されているかどうかによって「構造化」と「非構造化」に大別される．前述のファイル共有ソフトなどは非構造化オーバーレイネットワークに分類されるが，ここ数年の研究は構造化オーバーレイネットワークに関するものが多い．特に分散ハッシュテーブル (DHT:Distributed Hash Table) を用いてオーバーレイネットワークに参加している計算機（ノード）やデータの管理をする方式が，検索効率や公平性に優れていると注目されている．

こうした背景の一方で，サイズの小さなデータの利活用が増えてきている．例えば Web2.0 という名称で取り上げられるようなブログの日記やコメントも，データのサイズは非常に小さい．また，データに関するメタデータや，センサから得られる現実世界のデータ（センシングデータ）も非常にサイズが小さい．このような小さなサイズのデータは単体では価値のあるデータになることが難しいが，多くのデータが集まると意味のあるデータになることができる．また，小さなデータは様々な人やデバイスから提供されることが考えられる．これまでは少数の提供者がサイズの大きなデータを提供してきたのに対し，今後は多数の利用者がサイズの小さなデータを提供し，利用者間で利活用するような場面が考えられる．

本論文ではこのようなサイズの小さなデータをオーバーレイネットワークを用いて共有したとき，サイズの大きなファイルを共有する場合と比べて性能劣化する部分を考察する．また，特にセンサから提供されるようなセンシングデータを対象とし，データをある程度の大きさに集約して共有することを考慮する．以下に本論文の構成を示す．2 章ではオーバーレイネットワークと小サイズのデータの関係について述べる．3 章ではオーバーレイネットワーク上で実際に行った実験について述べ，4 章で実験結果を示す．この実験結果から，5 章でオーバーレイネットワークと小サイズのデータについて考察し，6 章にてまとめる．

2 オーバーレイネットワークと小サイズのデータ

ここでは DHT を用いたオーバーレイネットワークによって，小さなサイズのデータを集めるときの弊害について議論する．

2.1 DHT を用いたデータのダウンロード

DHT では任意の (key,value) の組を用いてデータの分散保持と高速な検索を実現する．ハッシュ関数 H を用いて H(key) と value を対応させるため，key を用いた検索に対して一意に value を特定することができる．また，H を適切に選ぶことでノードの保持データ量に偏りを発生させることなく，高速に value を発見することができる．このような DHT を用いたオーバーレイネットワークとしては，Chord[11], CAN[8], Pastry[9], Tapestry[12], Kademlia[7], Koorde[4] などがある．これらの方式はオーバーレイネットワークのトポロジや，メッセージの転送方式などに差異がある．

DHT を用いてデータの共有を行う場合として，例えばファイル名を key とし，そのファイルを保持しているノードの IP アドレスを value にすることが考えられる．ノードは H(key) を計算し，value を保持しているノードを発見し，そのノードから value を得る．value の値は IP アドレスであるから，ノードはその IP アドレスのノードにアクセスし，データをダウンロードする．

また，データを分散して保持し，key と value を複数登録することで並列にダウンロードを行うことができる．現在，ISP の提供しているサービスは回線の帯域が非対称のものが多いため，並列にダウンロードすることでダウンロード側の帯域を有効に利用することができる．そのため，並列ダウンロードすることでダウンロードに必要な時間を短縮することができる．

次に小さなサイズのデータを共有する場合を考える．データサイズの大きさに関らず，単一のデータを取得するのであれば前述の通り key と value の解決を行うことでデータをダウンロードできる．またサイズが小さいデータであれば，key に対応する value には IP アドレスではなく，直接データを記述することが考えられる．そのため，利用者は key から value を保持しているノードを発見し，value を得るだけで

データを取得可能である．このように単一のデータを取得する場合はDHTを用いても問題にはならない．しかし，小さなサイズのデータは集合体となって意味のあるデータになる場合が多い．そのため，上記のようなデータの取得を数多く繰り返すことでデータを収集しなければならない．また，ハッシュ関数を用いるとkey間に関連性があってもハッシュ値には関連性が無くなる．小さなサイズのデータを共有したとき，そのデータ同士に関連性があっても全てのハッシュ値は全く異なる値になるため，多くのノードによって共有されることになる．1つの例としてセンサから提供される気温のようなデータが考えられる．ある1日の気温のデータが100個のデータによって表されるとき，利用者が1日の気温を得るためには100のノードから情報を得る必要がある．このとき，DHTを用いた方法では100のデータ全てについてkeyを担当しているノードの探索を行い，valueを得る必要がある．このkeyの探索を行う時間はダウンロード時間に比べて長くなることが予想される．

また，データを並列にダウンロードすることによってこのようなkeyを検索する時間が短縮できると考えられる．しかし，サイズの小さなデータをダウンロードするためにノード間のコネクションを生成することもダウンロード時間が長くなる要因となり得る．また，オーバーレイネットワークに対して多量の検索を行うことも，検索にかかる時間が長くなる要因となりうる．

これを解決する1つの方法としては，小さなサイズのデータをある程度の大きさの集合体として管理する方法である．文献[10]ではデータを集合体にするほか，ハッシュを使わないことでkeyに対して連続量の割り当てを可能とした．例えば気温のデータであれば，100個のデータを時間を用いて10個のデータの集合体とし，それぞれの集合体についてkeyを割り当て，オーバーレイネットワーク上で共有する．利用者はその10のノードから集合体のデータを得ることで，100のノードからデータを得るよりも効率よくデータを取得できると予想される．特にセンシングデータは取得した位置や時間に密接に関係しているため，この位置や時間を用いて集合体に集約することができる．例えば奈良県で取得された12時の気温情報は1つの集合体に集約して扱うことができる．データを集合体に集約することで，1つのノードからダウンロードするデータは大きくなるが，検索の回数が減るためにダウンロード時間を減らすことが期待できる．

以上のことから，データの取得時に以下のような違いがダウンロードに必要な時間に影響することと考えられる．

- データが集合体となっているかどうかの違い
  - サイズの小さいデータがそれぞれvalueとなり個々に存在している
  - サイズの小さいデータが集合体となり，その集合体がvalueとなっている
- 並列ダウンロード数の違い
  - データを逐次的にダウンロードしていく
  - データを複数のコネクションを使い並列にダウンロードする

それぞれの場合についてダウンロードに必要な時間を考察する．

2.2 ダウンロードに必要となる時間

データが集合体となっているかどうかについて注目し，ダウンロードに必要な時間を考察する．データが集合体になっている場合，集合体になっていない場合に比べると検索しなければいけないkeyが減るため，ダウンロードにかかる時間を短縮することが期待できる．ただし，多くのデータを集合体とした場合，余分なデータをダウンロードする割合も多くなるため好ましくない．

次に並列ダウンロード数の違いについて考える．例えば同じサイズのデータであれば回線が輻輳しない限り，100個のデータを100個のコネクションを使って並列にダウンロードしても，1個のデータを1個のコネクションを使って並列にダウンロードするときと同じ時間でダウンロードできるはずである．逆にサイズが大きくなるとダウンロードに必要な時間は長くなるはずである．集約されたデータはサイズが大きくなるため並列にダウンロードしたとしても，集約されていないサイズの小さなデータを並列にダウンロードする時間の方が短くなるはずである．しかし，データのサイズは非常に小さいため，ある程度のデータサイズの変化はダウンロードに必要な時間に大きな影響を及ぼさないことが予想される．更に，ノード間でコネクションを確立するために必要な時間や，オーバーレイネットワーク上で検索クエ

リが多くなることを考えると，必ずしも並列数を増やすことがダウンロード時間の短縮にはならないことも予想される．

3 小サイズデータのダウンロード実験

ここではデータサイズや並列数によってオーバーレイネットワーク上でのダウンロード性能がどのように変化するかをダウンロードに必要な時間を計測することで評価する．ここではダウンロードに必要な時間を計測を行う実験について述べる．

3.1 実験概要

オーバーレイネットワークを用いて実験を行うために，OverlayWeaver[13] を用いて実験を行った．OverlayWeaver では DHT を用いてオーバーレイネットワークを構築することができる．また，複数台の計算機を用いてオーバーレイネットワークをエミュレーションすることや，DHT を利用している様々な実装方式を切り替えてエミュレーションすることができる．この OverlayWeaver を利用したプログラムを作成し，小サイズのデータを大量に生成・取得を行う実験を行った．その際に，以下の項目を変化させることでダウンロードに必要な時間がどのように変化するかを記録した．

- DHT の実装方式
- 1 つのデータサイズ
- ダウンロード時の並列化数

DHT の実装方式を変えることで，メッセージのルーティングアルゴリズムなどによる性能変化を測定する．また，ダウンロードする総量を一定にし，1 つのデータサイズを変化させた．例えばダウンロードの総量が 100K バイトとしたとき，1 つのデータサイズが 1K バイトなら 100 個のデータをダウンロードし，データサイズが 10K バイトなら 10 個のデータをダウンロードする．そして，全てのデータをダウンロードし終えたときにダウンロードが完了したと見なし，ダウンロードにかかった時間を記録する．

表 1: 実験に用いた計算機

Sun Fire V40z 1 台	
CPU	AMD Opteron 852 (2.6GHz) x4
Memory	16GB
OS	Solaris 10
Sun Java Workstation W2100z 9 台	
CPU	AMD Opteron 252 (2.6GHz) x2
Memory	2GB
OS	Solaris 10

3.2 実験内容

実験は表 1 のように計算機を 10 台用いて行った．それぞれの計算機はオーバーレイネットワークの 100 台の仮想計算機をエミュレーションし，合計 1000 台の仮想計算機でオーバーレイネットワークを構築した．このオーバーレイネットワーク上で 1 台の仮想計算機がデータを生成し，取得を行う．データはサイズを変えながら，データの個数を調整することで各サイズ毎に 300K バイトのデータを生成する．データサイズは 10K バイトから 60K バイトまで 1K バイト刻みで変化させた．このデータ 1 つ 1 つに対して適切な key を割り当て，オーバーレイネットワーク上で共有した．データを生成した仮想計算機は全てのデータを共有できた後にデータの取得を行う．このときに同時にダウンロードを行う並列数を 1 から 30 まで変化させながらデータの取得を行った．データをダウンロードする仮想計算機は並列数に応じたスレッドを生成し，そのスレッドを用いてデータのダウンロードを行う．全てのスレッドがデータのダウンロードを終えた段階でダウンロードにかかった時間を記録する．

この実験を DHT の実装方式を変化させながら行った．DHT の実装方式は Chord, Kademlia, Koorde, LinearWalker の 4 つを用いた．最後の LinearWalker は Chord から Finger Table を取り除いたものである．また，データを生成・取得する仮想計算機を 10 台に増やした場合についても実験を行った．生成・取得を複数の仮想計算機が同時に行うことで，より現実に近い環境でのダウンロード時間を得ることが期待できる．

4 実験結果

実験を行った結果を，2つのグラフを用いて示す．まず，1つのデータサイズの大きさ・並列ダウンロード数・ダウンロードにかかった時間の3軸を用いて3次元のグラフを示す．図1にChordを用いた実験の結果を，図2にKademliaを用いた実験の結果を，図3にKoordeを用いた実験の結果を，図4にLinerWalkerを用いた実験の結果を示す．なお，並列数の数がデータの数よりも大きくなった場合，データの数しか並列ダウンロードができないため，並列数の最大値はデータの数と同じである．

実験の結果から，全ての実装においてデータのサイズを大きくしてダウンロードした方が，ダウンロード時間が短いことが分かった．また，並列化した方がダウンロード時間は短縮されるが，並列数を多くしてもそれほど変化が無いことも分かった．

この実験結果から，並列数を最大にしたときのダウンロード時間をグラフに抽出した．図5にChordを用いた実験の結果を，図6にKademliaを用いた実験の結果を，図7にKoordeを用いた実験の結果を，図8にLinerWalkerを用いた実験の結果を示す．このグラフから並列数を最大にしてダウンロードした場合でも，ファイルのサイズが大きい方がダウンロードに必要な時間は短いということが分かる．

最後に，データの生成・取得を行うノードを10台に増やした場合について，同様のグラフを図9，10に示す．なお，DHTの実装方式はChordを用いた．このグラフから，特にサイズの小さいデータをダウンロードしたときにダウンロード時間が長くなっていることが分かる．一方でサイズの大きなデータをダウンロードする場合はそれほど影響がないことも分かる．

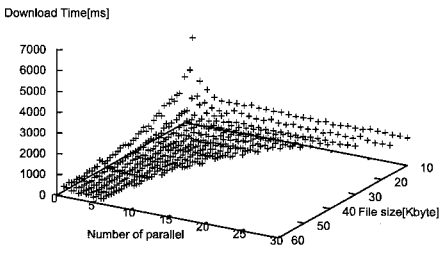

図 1: Chordを用いたときのダウンロード時間

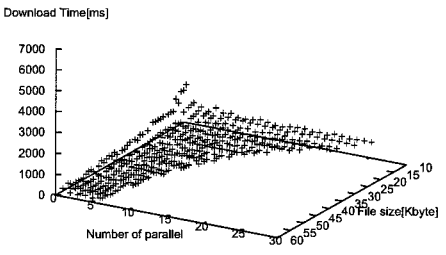

図 2: Kademliaを用いたときのダウンロード時間

5 考察

実験結果について考察する．まず，DHTの全ての方式において図1から図4に見られるように，データサイズが小さく，かつ並列数が少ない方がダウンロード時間が長くなることが分かった．データサイズは非常に小さいため，データのダウンロードにかかった時間をほとんど無視できるとすると，データの検索にかかった時間が長かったことが原因であると推測できる．つまり，300Kバイトのデータをダウンロードする時にデータの検索する回数が増えるため，検索に時間を浪費し，ダウンロード時間が長くなったと考えられる．また，同じサイズのデータをダウンロードするときでも，並列数を増やすことでダウンロード時間が短くなっていることを確認できた．しかし，5〜10以上に並列数を増やしてもダウンロード時間の大幅な短縮は見られない．そのため，並列数を一定数以上に増やしてもダウンロードに必要な時間は変化しないということが考えられる．また，データのサイズを大きくしてダウンロードした場合，サイズの小さなデータをダウンロードする場合よりもダウンロードにかかる時間が短いという結果も得られた．これはデータのサイズが大きくなり，データの個数が減ったためにデータの検索する回数が少なくなり，その結果ダウンロードにかかった時間を短縮できたと考えられる．特にLinerWalker（Finger Tableの無いChord）は全体的に長い時間がかかっているため，ChordからFinger Tableを除いた場合に

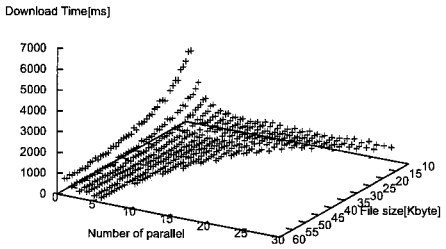

図 3: Koorde を用いたときのダウンロード時間

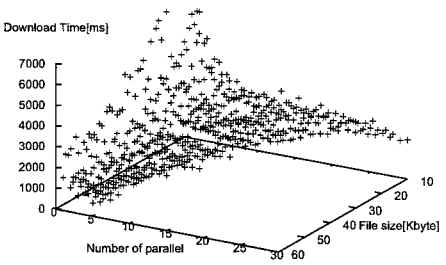

図 4: LinerWalker を用いたときのダウンロード時間

検索時間が長くなり，その影響がダウンロード時間に影響していると推測できる．

次に，図 5 から図 8 から得られた結果について考察する．ダウンロード時に並列数を最大にすれば，ダウンロードに必要な時間とデータの個数は関係なくなるはずである．しかし，実験の結果からデータサイズの小さいとき，すなわちデータの個数が多いときほどダウンロードに必要な時間は長くなっている．これは計算機に対してコネクションを確立するために必要な時間や，オーバーレイネットワーク上でデータを同時検索する性能による影響と考えられる．同時に複数のクエリが発生した場合でも，仮想計算機は逐次的に検索を解決する必要があるため，検索のクエリが多いほどダウンロードに必要な時間は長くなっていると考えられる．またこれは，図 9，10 の結果からも顕著に見てとれる．同時に複数の仮想計算機からデータの取得を行っているため，オーバーレ

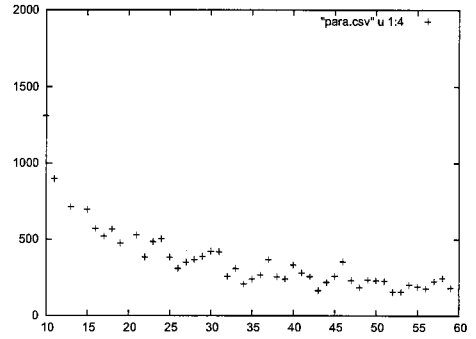

図 5: Chord を用いたときの最大並列時におけるダウンロード時間

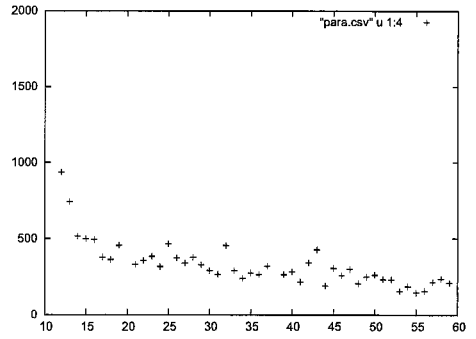

図 6: Kademlia を用いたときの最大並列時におけるダウンロード時間

イネットワーク上での検索クエリは非常に多くなる．そのため，特にサイズの小さいデータを取得するときに検索クエリが大量に発生し，ダウンロード時間が長くなっている．しかし，サイズの大きなデータを取得するときは検索クエリがそれほど多くならないため，ダウンロードに必要な時間も低い値に抑えられているのが分かる．

以上のようなことから，サイズの小さなファイルを DHT を用いて共有する場合，サイズの小さなデータをそのまま共有するのではなく，データの集合体へと集約し，ある程度の大きさにする方がダウンロードに必要な時間を短縮できると考えられる．センシングデータのような例では，文献 [10] のように地理位置や時間で集合体へと集約する手法が有効であるといえる．また，実験では理想的なネットワーク環

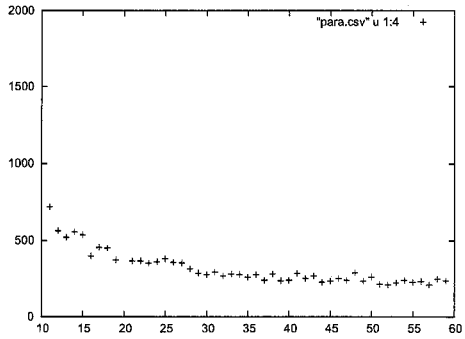

図 7: Koorde を用いたときの最大並列時におけるダウンロード時間

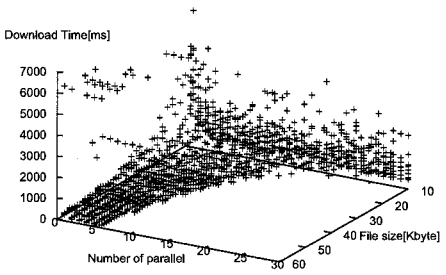

図 9: Chord を用いたときのダウンロード時間（データ取得ノード：10）

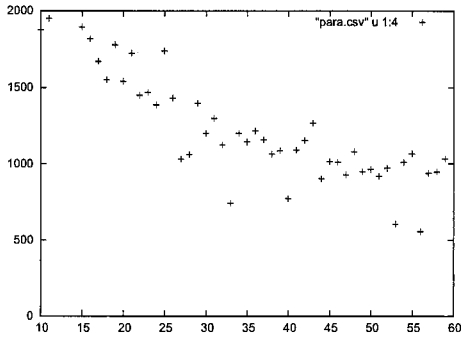

図 8: LinerWalker を用いたときの最大並列時におけるダウンロード時間

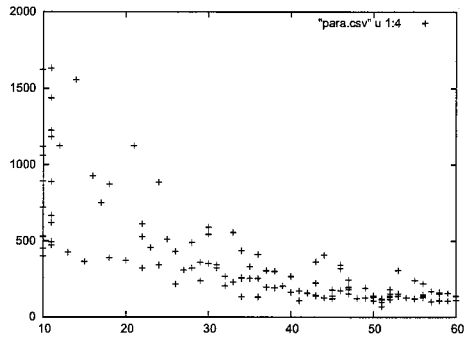

図 10: Chord を用いたときの最大並列時におけるダウンロード時間（データ取得ノード：10）

境・計算機性能での実験であったにも関らず，データサイズや並列化数による違いが大きく現れたため，現実の利用ではより大きな違いが現れることが予想できる．

一方で，ハッシュはデータ間の関連性を破壊するため効率的なダウンロードが行えないため，ハッシュを用いずにセンシングデータを共有する手法も見られる [6][5]．こうしたオーバーレイネットワーク上において，小さなサイズのデータに対して並列ダウンロードした場合，ハッシュを用いた方式と比べてどの程度の優位性があるのかも今後は検討する必要がある．

6 まとめ

DHT を用いたオーバーレイネットワーク上から，サイズの小さなデータをダウンロードしたときにその性能の変化を実験から考察した．実験の結果から，サイズの小さなデータを多くダウンロードした場合，オーバーレイネットワーク上からデータを検索するときに時間がかかり，結果としてダウンロードに必要な時間が長くなることが分かった．また，同じサイズのデータをダウンロードするのであれば小さなサイズのデータに分割するのではなく，ある程度の大きさにまとめておく方がダウンロードの時間を短縮できることも分かった．特に，複数の計算機がデータの生成・取得を行うような環境ではその影響が顕著に現れることも確認できた．今後はセンシングデータなどを考慮したオーバーレイネットワークをハッ

シュを用いた方式と比べてどの程度優位性があるかを検討する必要がある.

参考文献

[1] B. Cohen. Incentives Build Robustness in BitTorrent. *Workshop on Economics of Peer-to-Peer Systems*, 6, 2003.

[2] GNUTELLA.WEGO.COM. Gnutella: Distributed Information Sharing. *Online: http://gnutella. wego. com*, 2000.

[3] N. Inc. The napster homepage. *Online: http://www. napster. com*, 2000.

[4] M.F. Kaashoek and D.R. Karger. Koorde: A simple degree-optimal distributed hash table. *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS' 03)*, pages 98–107, 2003.

[5] Yu Kaneko, Kaname Harumoto, Shinya Fukumura, Shinji Shimojo, and Shojiro Nishio. A location-based peer-to-peer network for context-aware services in a ubiquitous environment. In *SAINT-W '05: Proceedings of the 2005 Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops)*, pages 208–211, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Satoshi Matsuura, Kazutoshi Fujikawa, and Hideki Sunahara. Mill: an information management and retrieval method considering geographical location on ubiquitous environment. In *Applications and the Internet Workshops, 2006. SAINT Workshops 2006 International Symposium on Applications and the Internet*, page 4, 2006.

[7] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS*, pages 53–65. Springer Berlin / Heidelberg, 2002.

[8] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.

[9] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*, page 329. Springer Berlin / Heidelberg, 2001.

[10] Kazutoshi Fujikawa Shinichi Doi, Satoshi Matsuura and Hideki Sunahara. Overlay network considering the time and location of data generation. In *International Symposium on Applications and the Internet Workshops (SAINTW'07)*, January 2007.

[11] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[12] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, Berkeley, CA, USA, 2001.

[13] 首藤一幸, 田中良夫, and 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**, 47:358–367.