Vol.2010-IOT-8 No.10
2010/3/1

P2P ネットワークを用いた時刻付き位置情報管理方式の提案

細川和宏[†1] 秋山大輔[†1] 安倍広多[†1]
石橋勇人[†1] 松浦敏雄[†1]

本稿では，P2P ネットワークを用いた時刻付き位置情報の分散管理方式について提案する．検討を行った方式は，2 次元 Z 曲線を用いた方式と 3 次元 Z 曲線を用いた方式および 3 次元空間を直方体に分割する方式である．いずれの場合も，曲線もしくは空間を分割し，分割された各セグメント内の位置情報を P2P ネットワーク上の 1 つのピアが管理する．これらのシミュレータを作成し，時刻付き位置情報データを管理するのに要するピア数と検索に要するホップ数より各方式の評価を行った結果，3 次元空間を直方体に分割する方式が時刻付き位置情報を管理する上で効率が良いことがわかった．

Proposal of Time-Attached Geoinformation Management Scheme using P2P Network

Kazuhiro Hosokawa,[†1] Daisuke Akiyama,[†1]
Kota Abe,[†1] Hayato Ishibashi[†1]
and Toshio Matsuura [†1]

This paper proposes time-attached geoinformation management schemes using P2P network. We considered three schemes: using two-dimensional Z-curve, using three-dimensional Z-curve, and using rectangular parallelepipeds on three-dimensional space. In each scheme, curves or spaces are split into fragments and each fragment is managed by a peer in a peer-to-peer network. Experiments for evaluating number of required peers to manage and required hop counts for range queries demonstrate that the scheme based on rectangular parallelepipeds is the most efficient.

1. はじめに

近年，カメラや GPS 機能などが搭載されている高性能な携帯端末が普及し，ネットワーク環境も整いつつある．さらに，携帯端末を利用してあらゆる場所において様々なサービスを受けることが可能となった．一例を挙げると，GPS 機能を搭載した携帯端末では現在位置と時刻を把握して目的地までの経路と到着時刻を提供するといったサービス[1)] がある．このような時刻情報が付与された位置情報 (以後，時刻付き位置情報とする) を扱うサービスでは，時刻付き位置情報データを効率よく管理し，検索できる必要がある．検索は，位置のみを指定した検索と位置と時刻を指定した検索が考えられるので，位置と時刻を効率的に管理するには地理的に近いデータと時刻に関して近いデータがそれぞれ取り出しやすいことが重要である．さらに，提供するサービスが多くの利用者に利用されることを想定すると多量の時刻付き位置情報データに対しても効率よく扱える必要がある．

本研究では，時刻付き位置情報を P2P ネットワーク上で管理する 3 方式を検討し，時刻付き位置情報データを管理するのに要するピア数と検索の際に要するホップ数より評価を行った．

2. 関連研究

2.1 ZNet

ZNet[2)] は，多次元空間を空間充填曲線の 1 つである Z 曲線を用いて 1 次元に変換した上で管理する．新しいノードが参加するに従って管理する領域を分割し，空間全体を分散管理する．

ZNet では，ネットワーク構造として Skip graph[3)] を利用している．Skip graph は key-value ストアの一種であり，key によってソートされた形でデータを格納している．このため，key の範囲を指定した検索 (範囲検索) を実行できるという特徴を持つ．Skip graph におけるノードの挿入や削除，探索にかかるコストは $O(\log N)$(N:ノード数) である．

2.2 SONAR

SONAR[4)] は，データ数が増加するに従って管理する多次元空間を分割し，分散管理する．管理する領域の分割は CAN[5)] の方式を利用する（CAN では，平面を矩形領域に分割して

†1 大阪市立大学大学院創造都市研究科
Graduate School for Creative Cities, Osaka City University

Vol.2010-IOT-8 No.10
2010/3/1

管理しており，領域の分割に際しては矩形の短辺に平行な直線で 2 等分する）．

分割された領域の各々を各ピアが管理する．個々のピアは隣接ピアの情報を持つとともに，座標軸に平行な方向に並ぶ空間のうち，$2^n (n = 1, 2, \ldots)$ 個分だけ離れた空間を管理するピアへのポインタを保持しており，これらを利用して検索を行う．検索にかかるコストは，N をピア数とすると $O(\log N)$ である．

3. 方式の検討

P2P ネットワーク上で管理する時刻付き位置データの検索にあたっては，位置のみを指定して検索する場合と，位置および時刻を指定して検索する場合が考えられ，これらの検索が効率よく行える必要がある．このためには，(1) 時刻付き位置データを位置を表す 2 次元空間上で管理し，各データに時刻が付随した形式で取り扱う方式と，(2) 位置と時間をあわせた 3 次元空間上でデータを管理する方式とが考えられる．

データを格納する 2 次元あるいは 3 次元の空間を分散管理するためには，(a) ZNet のように空間充填曲線を利用して空間を 1 次元に変換した後，その曲線を分割して管理する方式と，(b) SONAR のように元の多次元空間自体を分割する方式が考えられる．

以上より，本稿では，

(方式 1) 2 次元 Z 曲線を用いて位置平面を充填する方式 ((1) と (a) の組み合わせ)

(方式 2) 3 次元 Z 曲線を用いる方式 ((2) と (a) の組み合わせ)

(方式 3) 3 次元空間を直方体で分割する方式 ((2) と (b) の組み合わせ)

の 3 つの方式について検討を行う．

なお，本稿でいう検索は，一般にある範囲の位置やある範囲の時刻を対象としているが，これらは直線ないし平面状に切り取られることを前提としている．すなわち，位置の 2 次元平面を矩形状に切り取った範囲や，位置と時間からなる 3 次元空間を直方体状に切り取った範囲に含まれるデータが検索の対象となる．

3.1 2 次元 Z 曲線を用いた方式 (方式 1)

ここでは，2 次元 Z 曲線を用いた方式の概要を述べる[⋆1]．

本方式では，位置を表す 2 次元空間に時刻付き位置データを格納し，その空間を分割して P2P ネットワーク上のピアに割り当てることによって分散管理する．具体的には，位置の 2 次元空間を 2 次元 Z 曲線を利用して 1 次元の空間に変換し，変換された 1 次元空間を分割して各ピアに割り当てる．この方式では，1 つのピアが管理するのはある範囲の位置に存在するすべてのデータであり，その時刻は問わない．

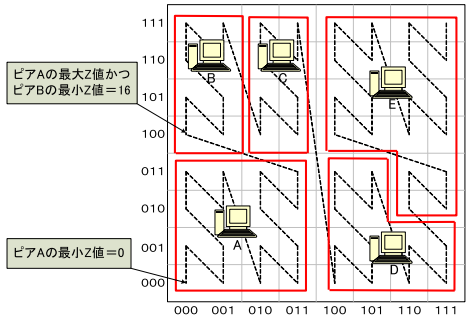

図 1 2 次元 Z 曲線を用いた方式の基本構造

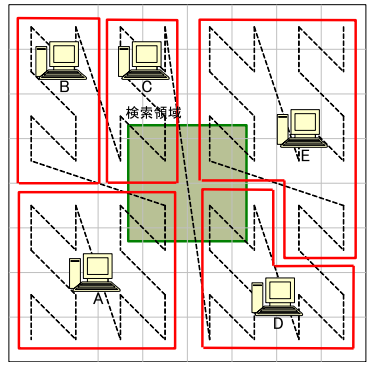

図 2 2 次元 Z 曲線を用いた方式の検索

空間のピアへの割当の例を**図 1** に示す．図において点線で示されているのが Z 曲線である．このように，与えられた空間を一定の数 (ここでは $8 \times 8 = 64$ 個の正方形 (3 次元ならば立方体)) に分割し，その中心を一度ずつ通るように Z の書き順で再帰的に結ぶことによって空間を充填する．

Z 曲線を 1 次元の数直線と考えた時に，原点からの距離を Z 値と呼ぶ．Z 値を用いることによって，多次元空間上の任意の位置を 1 次元の値に変換することができる（ただし，その分解能は分割の細かさによって決まる）．

Z 曲線によって与えられた 1 次元の空間において，初期状態では，1 つのピアが全体を管理している．登録された時刻付き位置情報データが増加するに従って新たなピアを参加させ，1 次元空間を分割することによって分担してデータを管理させる．各ピアは Z 曲線の連続した一部を管理しているため，1 つのピアが管理するデータは，基本的には地理的に近いものの集まりとなる (Z 曲線の性質によって，元の空間において近い距離にある点は Z 曲線上においても近い位置に写像されることが多い)．

ここで，あるピアが管理する (Z 曲線上の) 範囲における Z 値の最大値をその範囲の最大 Z 値，最小値をその範囲の最小 Z 値と呼ぶことにする．

ネットワークの構造には Skip graph を用い，管理する範囲の最小 Z 値を key としてピアを Skip graph に登録する．Skip graph では key でソートされた形でピアが並んでいるの

⋆1 本方式は文献 6) に基づいているため，詳細については文献 6) を参照されたい．

で，最小 Z 値を key として登録しておけば，任意の Z 値を管理するピアを検索できる．

3.1.1 登　　録

位置情報データの登録にあたっては，登録しようとする位置を Z 値に変換した値を key として Skip graph を検索し，そのデータを管理すべきピアを発見する．その後，見つかったピアにデータを登録する．

3.1.2 分　　割

各ピアに登録できるデータ数には，上限がある．そこで，データを登録しようとした際にそのピアの管理するデータ数が上限に達していた場合には，空間を分割して新たに参加させたピアとの間でその空間の管理を分担する．

空間の分割にあたっては，分割後に各ピアの管理領域が元の平面上でなるべく矩形になるように分割位置を決定する．管理するデータ数が均等になるように分割位置を決定する方法も考えられるが，検索領域が矩形であるため，ピアの管理する領域も矩形にしておいたほうが検索の際に効率が良くなる．

なお，分割の際に，特定のピアが管理するデータの数が極端に少なくならないように，各ピアが格納するデータ数には下限を定めている．

3.1.3 検　　索

データの検索は，位置の範囲のみを指定して，あるいは，位置と時刻の両方の範囲を指定して行う．

図 2 を用いて検索の方法を説明する．データを検索しようとするピアは，まず検索領域の最小 Z 値を算出し，それを key として Skip graph を検索することによって，その位置を管理するピア (図の A) を発見し，A の持つデータから指定された位置や時刻に合致するデータを抽出する．次に，Z 曲線の上を Z 値が大きくなる方向にたどりながらそこを管理するピアのデータを順に検索することによって，すべてのデータを探し出すことができる．

この際，図のピア B のように，検索領域と管理領域が共通領域を持たないようなピアを通過することがあるが，このようなピアの持つデータを検索することは無駄である．そこで，検索領域と管理領域が共通領域を持たないようなピアについては検索をスキップする[6]．

3.2 3 次元 Z 曲線を用いた方式 (方式 2)

方式 2 は，位置と時間からなる 3 次元空間を，3 次元 Z 曲線を利用して 1 次元に変換し，変換された 1 次元空間の一部をそれぞれのピアが管理する方式である．したがって，各々のピアは，地理的に近く，かつ，時刻に関しても近いデータの集まりを管理することになる．ネットワーク構造には Skip graph を用いる．

空間の取り扱い方という観点では，方式 2 は方式 1 を 3 次元に拡張したものと考えることができるため，データの登録や検索などの基本的な操作は方式 1 と同様である．

3.3 3 次元空間を直方体に分割する方式 (方式 3)

方式 3 では，位置と時刻からなる 3 次元空間を，直方体の形状をなす部分空間に分割し，それぞれの部分空間を 1 つのピアが管理する．このため，1 つのピアが保持するデータは，お互いに地理的に近くかつ時刻に関しても近い．

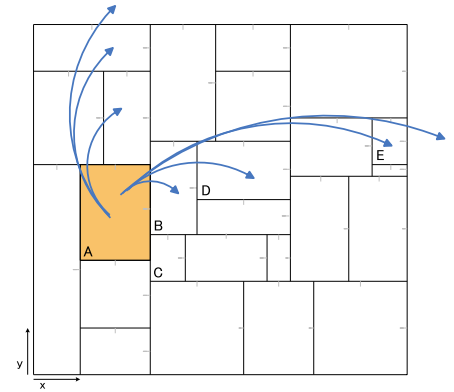

図 3　3 次元空間を直方体で分割する方式 (例:2D の場合)

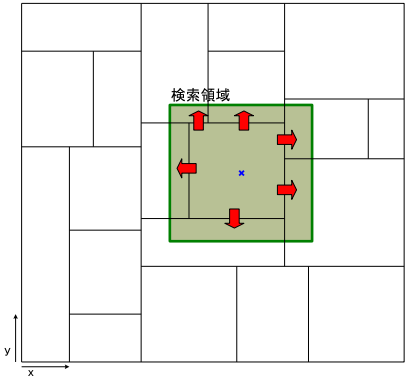

図 4　3 次元空間を直方体で分割する方式の検索 (例:2D の場合)

方式 3 においても，方式 1 および 2 と同様に当初は 1 つのピアだけが存在し，すべての空間を管理している．その後，ピアに登録された時刻付き位置情報の数が一定の基準以上に増加すると，そのピアが管理している空間を 2 つに分割し，一方の空間の管理を新たに参加させたピアに委譲する．以降，必要に応じてこの操作を繰り返す．

空間分割の様子を**図 3** に示す．ただし，3 次元空間をわかりやすく図示することは困難であるため，図では 2 次元空間において同様の操作を行った例を示している．

各ピアは，全体として 1 つの空間を管理するために，隣接ピアリストおよび座標軸ごとのルーティングテーブルを持つ．隣接ピアリストは，自身の管理する空間に隣接する空間を管理するピアのリストである．ルーティングテーブルは，座標軸に平行な方向に並ぶ空間のうち，距離が $2^n (n = 1, 2, \ldots)$ であるような空間を管理するピアへのポインタを格納している (図 3 の矢印がポインタを表している)．ただし，本方式では，図の左端と右端，上端と下端

はそれぞれ隣接するもの (トーラス構造) として取り扱っており，座標軸の正の方向へのポインタのみを保持している．

3.3.1 登　　録

時刻付き位置情報を登録しようとするピアは，ルーティングテーブルと隣接ピアリストを利用してその位置・時刻を管理すべきピアを発見し，登録を行う．

3.3.2 分　　割

登録の際にピアの保持するデータが許容量を超えた場合には，そのピアが管理する空間を 2 つに分割し，一方の管理を新たに参加させたピアに任せる．

以降では，空間の座標軸を，x 軸，y 軸および t 軸と呼ぶことにする．x 軸および y 軸は，地理的な平面を表し，t 軸は時間軸である．

空間を分割する際には，いずれかの座標軸に垂直な面で直方体を 2 つに分割する．この際，基準となる座標軸を $x \rightarrow y \rightarrow t$ の順にサイクリックに変更する．すなわち，最初は x 軸に垂直な面で空間を分割し，次に分割が必要になった場合には y 軸に垂直な面で，3 度目に分割が必要になった場合には t 軸に垂直な面で，と繰り返す．ここで，先に時間軸に垂直な面で空間を分割してしまうと，管理対象の空間が地理的に広く時間の範囲が短い空間となるため，時刻を特定せずにある位置に関する検索を行った場合に，検索対象のピアの数が増えてしまう．

空間を分割する位置は，元のピアと新しいピアの間で管理するデータの数が均等になるように決定する．2.2 の方式で採用している CAN では，管理対象の領域が均等になるように分割を行っているが，この場合，データの数が極端に不均等になる可能性があり，負荷分散の効果が低くなる．また，多数のデータを含む側の空間はすぐに再分割の必要が生じる可能性が高い．

空間を分割した場合には，その後，各ピアの隣接ピアリストとルーティングテーブルを更新する．

隣接ピアリストの更新

分割を行ったピアと新たに参加したピアは，各々の隣接ピアリストにお互いの情報を登録する．次に，各々の隣接ピアリストに登録されているピアが実際に隣接しているかどうかをチェックし，隣接していなければリストから削除する．隣接ピアリストに登録されているピアが保持する隣接ピアリストに，チェックしたピアの情報がない場合には，新たに登録する．

ルーティングテーブルの更新

ここでは，図 3 の空間 A の x 軸方向のルーティングテーブルの更新を例として説明する．

まず，隣接ピアリストを利用して，空間 A に対して x 軸の正の方向 (右辺側) に隣接する空間 (B および C) のうち A の右辺中央に隣接する空間を探す．ここでは空間 B がそれにあたるので，空間 B を管理するピアをルーティングテーブルに登録する．次に，空間 B のルーティングテーブルから B に隣接するピアの情報を得ることによって，空間 A からみて 2^1 個分離れた空間 D を管理するピアを知ることができる．さらに，空間 D の持つルーティングテーブルを利用して，空間 D から 2^1 個離れた空間 E の情報を獲得すれば，それは A からみて 2^2 個離れていることになる．

以下，同様に繰り返すことによって，ルーティングテーブルを構成することが可能である．

3.3.3 検　　索

検索アルゴリズムは，2.2 の検索アルゴリズムを改良したものである．

2.2 では，まず検索領域の中央を担当するピアを探し出し，そこから外側へ向かって検索領域外になるまで検索を進める (**図 4**)．この際，検索領域の中央を担当するピアをルーティングテーブルだけを利用して検索するので，斜め方向に隣接するピアが目的のピアである場合には，たどり着くまでに 2 ホップ以上必要となる可能性がある．

これを改善するために，提案方式では，ルーティングテーブルに加えて隣接ピアリストを利用して検索領域の中央を担当するピアを検索する．隣接ピアリストを利用することによって，検索領域の中央を担当するピアを探し出すのに要するホップ数が減少する．

検索領域の中央を担当するピアは，自分自身および隣接ピアリストに登録されているピア，座標軸ごとのルーティングテーブルに登録されているピアの中から検索領域の中心座標とピアが管理する領域の中心座標との距離が最小となるピアを探し出す．自分よりも近いピアが見つかった場合は，そのピアにおいて同様の操作を行い，これを繰り返して最終的に自身が管理する領域の中心座標と検索領域の中心座標との距離が最小となるピア，すなわち，検索領域の中央を担当するピアを見つける．

検索領域の中央を担当するピアが見つかれば，そこから隣接ピアリストを利用して検索領域外になるまで検索領域内のピアを順に辿ることによって検索を行う．

4. 評　　価

3 つの提案方式を評価するために，シミュレーションを行った．シミュレーションでは，空間の大きさを $2^{10} \times 2^{10} \times 2^{10}$ とし，時刻付き位置情報を順次登録した場合に必要となるピア数と，検索を行った際に要するホップ数を計測した．なお，1 つのピアが管理できる最大のデータ数は 200 としている．

位置情報は人間が発生させるデータであると想定すると，地球上に分布する人口の偏りによって，データの存在する位置にも偏りが存在すると考えることが自然である．そこで，人口の分布の偏りをある程度反映するために，Zipf の法則に従うよう乱数で発生させた位置データを使用した．時刻については一様乱数で決定している．

4.1 登録時に要するピア数

ここでは，時刻付き位置情報を登録するために必要となるピア数について評価する．

シミュレーションでは，一定数の時刻付き位置情報を登録し，その際に必要となるピアの数を調べた．登録した位置情報の数は 10,000 個から 100,000 個までの範囲であり，この間を 2,000 個刻みで増加させつつ，それぞれ 10 回ずつシミュレーションを行ってピア数の平均値を求めた．登録したデータの数とピア数の関係グラフに示したのが**図 5** である．

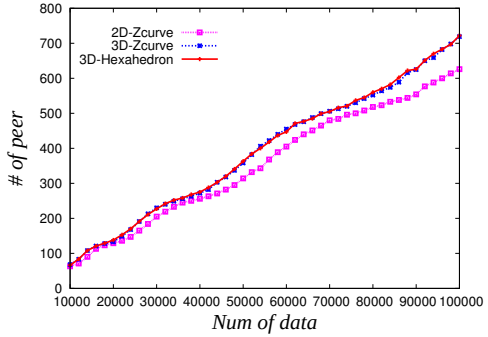

図 5 時刻付き位置情報データ追加時に要するピア数

10 万個のデータを追加した場合，方式 1 では 621 のピアが必要となり，方式 2 では 720，方式 3 では 722 のピアが必要となることがわかる．すなわち，方式 1 が全体のピア数がもっとも少なく，効率がよい．

グラフにおいて，増加率が一様ではなく振動しているのは，各ピアに平均的にデータが登録されると，どのピアでも同じようなタイミングで分割が生じるためであると考えられる．そのタイミングにはピア数の増加が続くが，一通り分割が終了すると，ピアのデータ格納数には余裕ができるため，しばらくは増加の程度が下がる．

4.2 検索に要するホップ数

ここでは，時刻付き位置情報データについて，位置の範囲と時刻を指定して検索した場合と，位置の範囲のみを指定して検索した場合の検索ホップ数について評価する．

シミュレーションでは，30,000 個の時刻付き位置情報データを登録した後に様々な大きさの検索範囲について検索を行った．

検索する範囲 (空間) の大きさは，$10 \times 10 \times 10$ から $200 \times 200 \times 200$ までの間とし，この間を 10 刻みで増加させつつそれぞれ 10 回ずつ検索を行って，検索に要したホップ数の平均値を算出した．

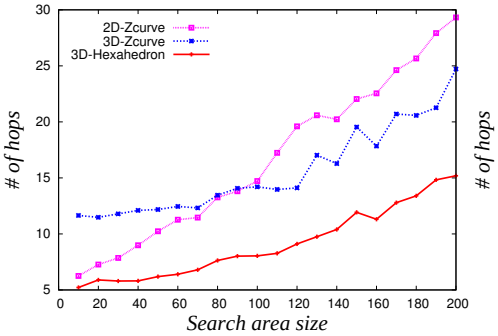

図 6 位置と時刻で検索した際にかかる hop 数

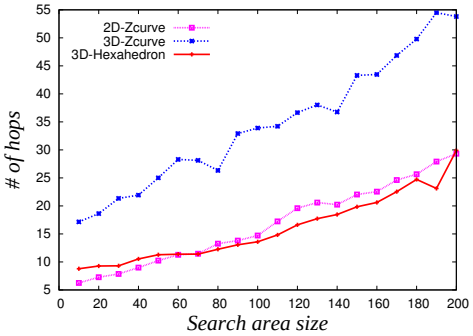

図 7 位置のみで検索した際にかかる hop 数

4.2.1 位置と時刻で検索した場合

シミュレーションによって得られた検索範囲の大きさとホップ数の関係を**図 6** に示す．

方式 1 では，検索領域が小さい場合は比較的検索ホップ数は少ないが，検索領域が大きくなるにつれて検索ホップ数も大幅に増加している．方式 2 では，検索領域が大きくなっても検索ホップ数の増加は方式 1 に比べて緩やかである．この理由として，方式 1 では時刻と格納先のピアには関連がないため，検索領域と位置は一致するものの時刻は一致しないようなデータを持つピアも検索しなければならない (実際に検索してみなければ時刻が一致しないことがわからない) ことが考えられる．

方式 3 については，検索領域の大きさに対するホップ数の増加は他の 2 方式に比べて緩やかである．これは，Z 曲線を利用する 2 方式に比べて，検索領域外のみを担当するピアをスキップする必要がないためであると考えられる．

これらより，位置と時刻の範囲を指定して検索した場合には，方式 3 が最も検索効率が良

いと考えられる．

4.2.2 位置で検索した場合

シミュレーションによって得られた検索範囲の大きさとホップ数の関係を**図 7** に示す．

方式 1 については，4.2.1 の結果と同一である．4.2.1 では，位置は合致するが時刻が合致しないケースが存在したが，位置の範囲のみでの検索では，すべての時刻が対象となるので，不要なピアを検索することはない．

一方，方式 2 については，検索領域が大きくなるにつれて検索ホップ数が大幅に増加している．これは，方式 2 では位置と時刻からなる空間を 3 次元 Z 曲線で 1 次元に変換しているため，位置のみで検索を行うと，元の空間では連続した 1 つの空間が Z 曲線上では多数の断片に分かれてしまうことが原因である．断片の個数が多いと，3.1.3 で述べた検索領域外を担当するピアの数が多くなるため，これらのピアをスキップする回数も多くなり，ホップ数が大幅に増加する．

方式 3 については，方式 1 と同程度であるが，4.2.1 の場合と比較して全体的にホップ数が増加している．これは位置のみで検索を行っているために，位置と時刻を指定した検索に比べて検索対象の空間が時間軸方向に大きくなり，検索領域内を担当しているピア数が多くなるためである．

以上より，位置の範囲のみで検索した場合は，方式 1 と方式 3 がほぼ同程度の効率であることがわかった．

5. おわりに

本稿では，時刻付き位置情報を管理する 3 方式についてシミュレーションを行い，評価を行った．時刻付き位置情報挿入時に要するピアや時刻付き位置情報検索時にかかるコストを総合して考えると，方式 3 の直方体で分割する方式が時刻付き位置情報を管理するのに適していることがわかった．

一般に，P2P ネットワークではピアの予期せぬ故障や離脱への対応が困難であるが，本研究では，我々のグループで別途研究を行っている，耐障害性の高い P2P 基盤ソフトウェア musasabi[7)–9)] の利用を前提としているため，この点については言及していない．

今後の課題としては，参照が少ないと考えられる過去のデータについては 1 ピアあたりのデータ数を増加させてピア数を減らす（これによって検索ホップ数が減少する）ことや，実環境において実装して評価を行うことがあげられる．

謝辞 本研究の一部は独立行政法人情報通信機構「高度通信・放送研究開発委託研究」の助成を受けている．

参 考 文 献

1) 株式会社交通新聞社. ぴたのりオートナビ. `http://pitanori.com`, (2010 年 1 月 24 日確認).
2) Yanfeng Shu Beng, et al. Supporting multi-dimensional range queries in peer-to-peer systems. In *Fifth IEEE Intl. Conf. on Peer-to-Peer Computing (P2P 2005)*, pp. 173–180, 2005.
3) James Aspnes, et al. Skip graphs. *ACM Trans. on Algorithms*, Vol.3, No.4, pp. 1–25, 2007.
4) Thorsten Schütt, Florian Schintke, and Alexander Reinefeld. A structured overlay for multi-dimensional range queries. In *13th Intl. Euro-Par Conf.*, pp. 503–513, 2007.
5) Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proc. of the 2001 ACM SIGCOMM Conf.*, pp. 161–172, 2001.
6) 秋山大輔, 細川和宏, 安倍広多, 石橋勇人, 松浦敏雄. Z 曲線を用いた効率的な 2 次元位置情報の分散管理手法の提案とその評価. 情報処理学会研究報告, Vol. 2009-IOT-8, No.9, 2010 (掲載予定).
7) 安倍広多. P2P システム上での安定したサービス提供基盤 musasabi. 情報処理学会研究報告, Vol. 2009-IOT-4, No.24, pp. 131–136, 2009.
8) Kota Abe, Tatsuya Ueda, Masanori Shikano, Hayato Ishibashi, and Toshio Matsuura. Toward fault-tolerant P2P systems: Constructing a stable virtual peer from multiple unstable peers. In *Proc. of Intl. Conf. on Advances in P2P Systems (AP2PS 2009)*, pp. 104–110, 2009.
9) 鹿野将典, 上田達也, 安倍広多, 石橋勇人, 松浦敏雄. P2P 基盤ソフトウェア musasabi の仮想ピアにおける通信方式. 情報処理学会研究報告, Vol. 2009-DPS-139, pp. 1–8, 2009.