Vol.2010-IOT-8 No.9
2010/3/1

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

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

本稿では，多数の2次元位置情報をP2Pネットワークを用いて分散管理する一手法を提案する．提案手法では，2次元平面をZ曲線を用いて分割し，各領域を一つのピアが管理する．管理領域内のデータ数が一定数を越えると領域を分割することでピアの負荷を一定に保つ．範囲検索のためには構造化オーバレイネットワークの一種であるSkip graphを用いる．範囲検索に要するホップ数を削減するために，領域の分割方法を工夫している．提案手法はシミュレーションにより範囲検索に要するホップ数と管理に必要なノード数を評価した．

An Efficient Distributed Management Scheme for 2D Location Information using Z-curve

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

In this paper, we propose a distributed management scheme for 2D location information using Peer-to-Peer network. In the proposed scheme, a 2D plane is divided into fragments with Z-curve and each fragments is managed by a peer. Skip graphs, a kind of structured overlay network, are used for range queries. To reduce hop counts required for range queries, we devised a method to choose appropriate points on dividing. We have evaluated the method with regard to number of hops required for range queries and number of requried peers.

†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

1. はじめに

位置情報を取得できる携帯端末の普及により，端末の位置情報を利用したサービスの需要が高まっている．位置情報を扱うサービスでは，指定された範囲内の位置情報を効率良く検索できる必要がある。また，管理すべき位置情報が増加しても，位置情報の検索や登録などの要求にスムーズに対応できることが重要である．

位置情報管理をサーバなどで集中して管理する手法は，データ構造やアルゴリズムが比較的単純で，サービス利用者の要求に伴う通信回数が少ないなどの利点があるが，単一故障点の存在や，管理する位置情報の数に比例してサービス提供者にかかる負荷が増大するといった問題がある．

そこで本稿では，P2Pネットワークを用いた範囲検索可能な位置情報の分散管理手法の提案とその評価を行う．提案手法は，地域や国，世界のような地理的範囲を対象とし，対象とする全領域を，位置情報数の増加に伴い分割することで，P2Pネットワークに参加するピアで分散管理する．各ピアは担当領域に含まれる位置情報を保持する．領域の分割方法を工夫することで，範囲検索性能の向上や，効率的な負荷分散を実現した．

本稿では，P2Pネットワークを用いて効率的に位置情報を管理する手法の提案を行うとともに，シミュレータにより範囲検索の性能を評価した．

2. 提案手法

提案手法は，大きさ $d \times d$ の矩形領域内の位置情報を管理する．登録する位置情報の座標を (x, y) とするとき，x, y は $0 \leq x < d, 0 \leq y < d$ を満たす整数とする．

提案手法について述べる前に，2.1節と2.2節で提案手法で利用するZ曲線とSkip graphについて簡単に説明する．提案手法の詳細は2.3節以降で述べる．

2.1 Z曲線

Z曲線は空間充填曲線の一種である．**図1**に例を示す．Z曲線を用いると $d \times d$ の平面上の座標 (x, y) は0から $d^2 - 1$ までの整数値（Z値と呼ぶ）にマップすることができる．座標値とZ値は簡単なビット演算により互いに変換できる．平面上で2点が近い距離にあるとき，多くの場合Z値の差も小さくなるという特徴がある．

2.2 Skip graph

Skip graph[1)] は構造化オーバレイネットワークの一種で，P2Pネットワークに適した分散データ構造である．Skip graphの構造を**図2**に示す．図中の正方形はノードを表し，中

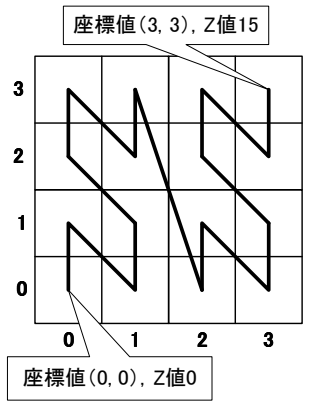

図 1 Z 曲線

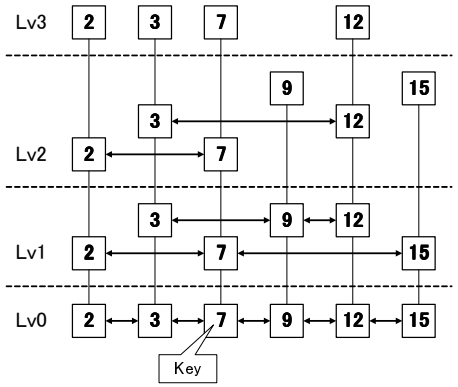

図 2 Skip graph の構造とルーティング

の数字はノードが保持するキーを示している．

Skip graph の大きな特徴に，キーがソートされて格納されているという点が挙げられる．このため，指定したキー値と正確に一致するノードを検索する（メッセージを転送する）ほかに，指定したキー値より等しいか小さい最も大きいノードを検索するといったことが可能である．

Skip graph では，ノードの登録，検索は平均 $O(\log n)$ ホップで実行できる（n は Skip graph に登録したデータ数）．

2.3 データ構造

提案方式では，位置情報を管理するピアは，Z 値がある範囲 [minZ, maxZ) に含まれる全ての位置情報を保持する（minZ，maxZ は各ピアが保持する変数）．初期状態では唯一のピア（初期ピア）が全ての位置情報を管理している（minZ=0, maxZ=d^2）．ピアが保持する位置情報のエントリ数が，格納できる最大エントリ数 (maxEnt) を越えると，管理領域を分割し，新たなピアを割り当てる（詳しくは 2.5 節で述べる）．

位置情報を登録・検索する場合は，その Z 値から担当するピアを見つける必要がある．このために，各ピアは自身の minZ の値をキーとして Skip graph に登録する．任意の Z 値 z を担当するピアを見つけるには，Skip graph から z と等しいか小さい最も大きいピアを検索すればよい．Skip graph 上で検索を行おうとするノードは，Skip graph にデータを登録している任意のピアを知る必要があるが，このためには，例えば全てのノードは Skip graph にダミーの値をキーとして登録しておけばよい．

データ構造の例を**図 3** に示す．図では，4×4 の平面を Z 曲線により Z 値 0 から 63 までにマッピングしている．丸印で示している 9 つの位置情報を 3 つのピア A，B，C で管理している（線で囲まれている範囲がそれぞれの担当する領域を示している）．それぞれが Z 曲線上で連続する範囲を担当していることに注目されたい．

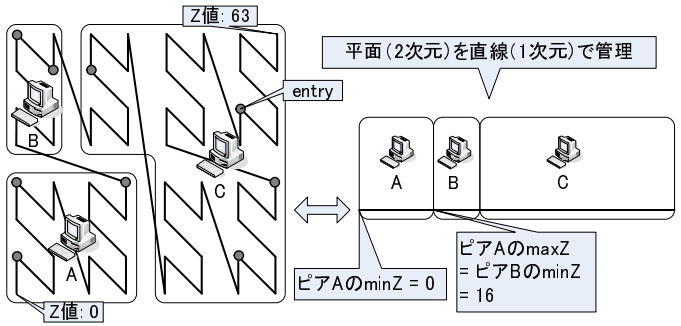

図 3 基本データ構造

2.4 位置情報の登録と削除

位置情報を登録したいノード（クライアントと呼ぶ）は，位置情報に対応する Z 値 z を求め，次に 2.3 節で述べた方法により z を担当するピアに登録要求を転送する．

登録要求を受信したピアは，自身が保持するエントリ数をチェックし，空きがあれば（maxEnt 未満ならば）登録する．空きがなければ次に述べる領域の分割を行う．

クライアントが位置情報を削除する場合は，登録処理と同様に対象ピアに削除要求を転送し，削除要求を受信したピアがエントリを削除すればよい．

2.5 領域の分割

2.5.1 領域分割の概要

登録要求を受信したピア u においてエントリの空きがない場合，ある Z 値 s を選び，自分の管理する領域を [u.minZ, s) と [s, u.maxZ) の 2 つに分割する．u は前者の区間を管理し，後者の区間は新たなピア v を割り当て，管理する．ピアの割り当て方法については 2.7 節で述べる．

領域を分割する際，s の選び方が問題となる．本研究では，u と v のエントリ数を均等に

分割する SplitEven と，範囲検索の範囲の形状を考慮した SplitSmart の 2 つの分割手法を考えた．

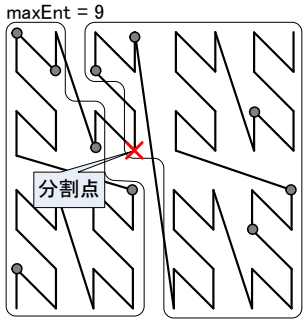

図 4 SplitEven による分割例

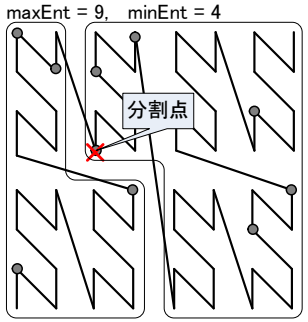

図 5 SplitSmart による分割例 1

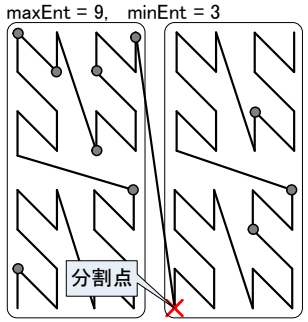

図 6 SplitSmart による分割例 2

2.5.2 エントリ数を均等に分割 (SplitEven)

SplitEven では分割後のピアの負荷を均等にするため，2 つのピア u, v が保持するエントリ数が $(maxEnt + 1)/2$（u は端数切り捨て，v は端数切り上げ）となるように分割する．

SplitEven を用いて分割した例を**図 4** に示す．この図は 1 つのピアで領域全体を管理していたところに maxEnt(=9) を超える位置情報が登録され，SplitEven により 2 つのピアで管理するようになった状態を表している．図中の×印が分割点を表す．分割後は u と v の管理する位置情報の数は 5 となり，また分割点は，10 個ある位置情報を Z 値の小さい方から数えて 5 番目と 6 番目の位置情報の中間にある．

ところで，範囲検索を行う際は，矩形で与えられる検索領域に対し，担当する領域が重なる全てのピアに対して検索要求メッセージを送信し，応答を受け取る必要がある．SplitEven による分割では，図 4 からもわかるように領域の形の凸凹の数が多くなるため，検索領域と重なる担当領域を持つピアの数が多くなると考えられる．そこで，分割後の領域の凸凹が少なくなるように分割する方法 SplitSmart を考えた．

2.5.3 形状を考慮した分割 (SplitSmart)

SplitSmart ではピアの担当領域をなるべく凸凹が少なくなるように分割する．

この場合，SplitEven とは異なり，分割後の 2 つのピアが保持する情報の数に偏りができる．どの程度の偏りを許すのかを決めるために，パラメータ minEnt を導入する ($minEnt \leq maxEnt/2$)．minEnt は分割後のピアが保持すべき最小の位置情報の数を表す．

SplitSmart を用いた例を**図 5**(minEnt=4 の場合)，**図 6**(minEnt=3 の場合) に示す．管理する位置情報は図 4 の場合と同一であるが，分割点が異なる．SplitSmart を用いた場合，分割後の各領域の凹凸の数が SplitEven を用いた場合よりも少なくなっていることがわかる．

SplitSmart のアイデアは単純である．Z 曲線の形状から，分割後の凸凹の数を少なくするためには，2 進表記したときに末尾 (LSB 側) の 0 の数がなるべく多い Z 値を境界として分割すればよい．分割点を選ぶ際は，このような値を [u.minZ, u.maxZ) の範囲から選ぶ．ただし，分割後の u と v のエントリ数のいずれかが minEnt 未満となる場合はその分割点はスキップする．図からもわかるように，minEnt が小さいほど凸凹の数は少なくなる．

2.6 範 囲 検 索

本節では，範囲検索のアルゴリズムについて説明する．

2.6.1 範囲検索の概要

範囲検索の基本的な考え方は単純である．まず，検索範囲（矩形）の最も小さい Z 値 zL と最も大きい Z 値 zH を矩形の座標から算出する．次に，Skip graph を用いて zL を担当するピアへ検索要求メッセージを転送する．メッセージを受信したピアは，検索領域に含まれるエントリを抽出し，次に，Skip graph 上でそのピアの右ノード（Z 値が大きい方向に隣接するピア）にメッセージを転送する（このときメッセージには検索結果を追記する）．これを繰り返し，メッセージが zH を担当するピアまで到達すると検索が終了する．

図 7 に例を示す．図では，全領域を A から I の 9 個のピアで分散管理している．検索範囲を B, C, F, G にまたがる矩形（灰色）とすると，zL は B が担当し，zH は G が担当している．このため，B からメッセージを順番に C, D, E, F, G と転送すれば検索範囲内を検索できる．

しかし，この方法では検索範囲を担当しないピアまで検索メッセージを転送してしまい，非効率的であるという問題がある（図の例では D と E）．このため，このような検索領域を担当しないピアをスキップする方法を以下に述べる．

2.6.2 検索領域を担当しないピアをスキップするアルゴリズム

検索要求メッセージを受信したピアを q とする．$q.maxZ$ が検索範囲に含まれていれば，明らかに q の右ノードは検索領域を担当するピアであるため，検索メッセージを転送する．

$q.maxZ$ が検索範囲に含まれない場合，$q.maxZ$ から Z 曲線を Z 値の大きい方向に辿って検索範囲内に戻ってくる点 P を求め，P を担当するピアに検索メッセージを転送する．

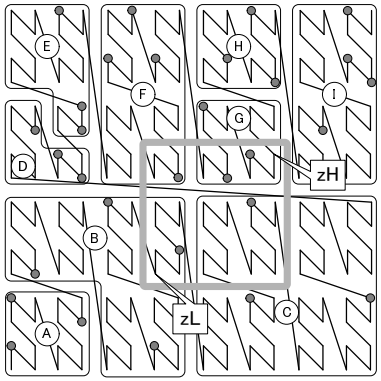

図 7 範囲検索 1

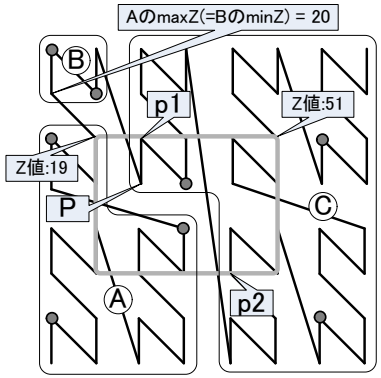

図 8 範囲検索 2

図 8 を用いて説明する．灰色の矩形で示した検索領域の左下が zL であり，この点はピア A が担当している．A.maxZ は検索範囲外であるため，Z 曲線が再度検索領域内に戻ってくる点 P を求め，P を担当するピア C にメッセージを転送する（ピア B はスキップする）．P は以下の方法で求めることができる．

ピア q は，$q.maxZ$ が検索範囲外の場合，次の処理を行う．検索範囲の矩形の各辺のそれぞれについて，辺の両端の座標の Z 値 (a, b) を求め，$a \leq q.maxZ < b$ となる辺を探す．図 8 の場合，検索範囲の上辺と下辺が該当する．次に，そのような全ての辺に対し，$q.maxZ$ よりも大きい最小の Z 値を持つ辺上の点を探す（辺上の座標に対して 2 分探索を行えばよい）．図では p1 と p2 が該当する．求まった点で最も Z 値が小さいものを m とする．

m が下辺か左辺上の点である場合，m が求める P である．そうではない場合，m から下方向（m が上辺上の点の場合）あるいは左方向（m が右辺上の点の場合）に進むと，いずれ q の担当する領域に入るはずである．入る直前の点が求める P になる（担当領域に入ったかどうかは点の Z 値と $q.maxZ$ を比較することで判定できる．また，この処理は実際は 2 分探索で実装可能である）．図では，p1 が m であるが，そこから下方向に進んで q の担当領域に入る直前の点が P となる．

P を求めたら，P の Z 値から Skip graph を用いて P を担当するピアに検索メッセージを転送する．この方法でスキップしたピア数を s とするとき，$O(\log s)$ ホップで次の担当ピアにメッセージを転送できる．

2.7 ピアの割り当て方法と耐故障性の確保

2.5.1 節で述べたように，提案手法では領域が分割される毎に担当領域に新たなピアを割り当てる必要がある．ただし，単純に 1 つの物理ピアを 1 つの領域に割り当てるだけでは，ピアが離脱した場合にそのピアが担当していた領域の情報が失われてしまうため，何らかの冗長性が必要である．

この問題は，P2P 基盤ソフトウェア musasabi[2),3)] 上の仮想ピアを用いることで解決できる．仮想ピアは複数の物理ピアを集め，その上でアプリケーションを一貫性を保ちながら動作させることで耐故障性を確保する仕組みである．物理ピアが離脱した場合は，別の物理ピアを仮想ピアに組み入れることによって動作を継続する．

提案方式で用いるピアを仮想ピアを使って実現することで，それぞれのピアは高い可用性を得ることができる．仮想ピアは必要に応じて生成できる（方法は文献 3) 参照）．

3. 実験と評価

提案手法を評価するための実験を行った．本章では実験内容とその結果について述べる．

3.1 2 つの分割手法の評価

2.5.1 節で SplitEven と SplitSmart という 2 つの分割手法を述べた．SplitSmart による効果を調べるために以下のシミュレーション実験を行った．

全体で管理する矩形領域の大きさを 1024×1024 とした．位置情報を一様乱数により生成し，複数のピアで分散管理させた．さらに，100×100 の大きさの検索範囲を乱数で決定し，範囲検索に必要なホップ数を計測した（ホップ数には，Skip graph によるホップ数を含んでいる）．範囲検索は 200 回行い平均値を求めた．なお，位置情報の数は 10000 から 1000000 まで変化させた．また，maxEnt=600，minEnt=180 (SplitSmart の場合）とした．結果を**図 9** に示す．

グラフの X 軸が位置情報の数，Y 軸が平均検索ホップ数である．全体的に SplitSmart を用いた方が検索ホップ数が少なくなることがグラフから確認できる．

3.2 SplitSmart のパラメータの評価

次に，SplitSmart を用いた際のパラメータ，maxEnt と minEnt の適切な値を求めるために次の実験を行った．

先ほどと同様，全体で管理する矩形領域の大きさを 1024×1024 とし，位置情報を一様乱数により 100000 個生成して複数のピアで分散管理させた．領域分割には SplitSmart を用いる．maxEnt を 200 に固定し，minEnt を maxEnt の 5%から 50% まで変化させたと

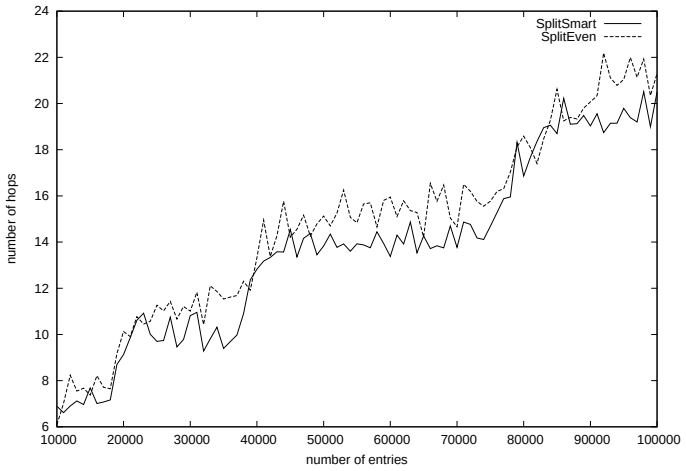

図 9 SplitEven と SplitSmart の比較

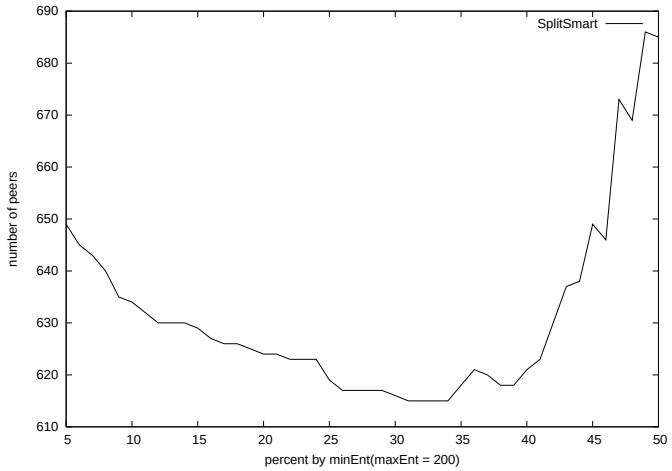

図 10 maxEnt と minEnt の関係

きの，範囲検索に必要なホップ数と領域管理を担当するピアの数を計測した（範囲検索の条件は上の実験と同様である）．

検索ホップ数に関しては，minEnt の割合を変化させてもそれほど差は出なかった（紙面の都合上結果は省略する）．ピア数に関する結果を**図 10** に示す．X 軸が minEnt の maxEnt に対する割合，Y 軸が管理に必要なピア数を示している．この結果，minEnt は maxEnt の 25〜40%程度が適当と思われる．

4. 関 連 研 究

P2P ネットワークにより位置情報を管理する手法として ZNet[4]，SONAR[5]，P2PRtree[6] が挙げられる．

ZNet は提案手法と同様に Z 曲線と Skip graph を用いている．ZNet と提案手法の大きな違いは，領域の分割手法にある．提案手法では，領域の分割はデータ数が一定数 (maxEnt) を越えたときに行うのに対し，ZNet では P2P ネットワークに新たなピアが参加するたびに領域を分割する（つまり，ZNet ではすべてのピアで位置情報を分散管理する）．また，ZNet では各ピアが保持する領域の形はすべて正方形であり，範囲検索のアルゴリズムはこのことを利用したものとなっている．ZNet では位置情報管理に参加するピアの数が位置情報の数と無関係に増加するため，検索に必要なホップ数は提案手法よりも多くなると思われる．

SONAR は構造化 P2P ネットワークの一種である CAN[7] を用いた手法である．位置情報の数が増加すると矩形領域を CAN の手法によって分割する．各矩形領域からは，縦方向と横方向に関して 2^n 個離れた矩形領域にリンクを張ることで効率的な検索を可能にしている．この方法は，他の矩形領域へのリンクを維持する処理が煩雑である．

P2PRtree は，位置情報を管理するための集中型データ構造である R-tree を P2P ネットワークで利用できるように改良したものである．R-tree の MBR (Minimum Bounding Rectangle) とピアを対応させることにより木構造である R-tree を P2P ネットワーク上に実現している．基本的に木構造であるため，木の上位に位置するピアほど大きな負荷がかかるという問題がある．

5. ま と め

本稿では P2P ネットワークを用いた効率的な位置情報管理手法について述べた．提案手法では Z 曲線を用いて 2 次元座標を Z 曲線上の位置（Z 値）に変換し，Z 曲線上の領域をピアに割り当て，管理する．ピアが管理する位置情報が増加すると，管理する領域を分割して別のピアを割り当てることで負荷分散を図っている．ある座標を担当するピアは Skip

Vol.2010-IOT-8 No.9
2010/3/1

graph を用いて検索する．区間を分割する際に領域の形を考慮することで，検索に要するホップ数を改善できることを示した．

今後の課題としては位置情報の削除によってピアが管理するエントリ数が減少した際の再構築方法の検討，関連研究との性能比較などが挙げられる．

謝辞 本研究の一部は科研費 (18700069) 及び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) James Aspnes and Gauri Shah. Skip graphs. *ACM Trans. on Algorithms*, Vol.3, No.4, pp. 1–25, 2007.
2) 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, pp. 131–136, 2009.
3) 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.
4) Yanfeng Shu Beng, etal. 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.
5) Thorsten Schütt, Florian Schintke, and Alexander Reinefeld. A structured overlay for multi-dimensional range queries. *Euro-Par 2007 Parallel Processing*, pp. 503–513, 2007.
6) Anirban Mondal, Yilifu, and Masaru Kitsuregawa. P2PR-tree: An R-tree-based spatial index for peer-to-peer environments. In *Proc. of the Intl. Workshop on Peer-to-Peer Computing and Databases*, pp. 516–525. Springer-Verlag, 2004.
7) Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pp. 161–172, 2001.