# 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

羽場 裕介 † 松尾 啓志 †

† 名古屋工業大学大学院情報工学専攻

分散ハッシュ法 (Distributed Hashing) は，ピアツーピア環境において高速にファイルやノードを検索するための手法として注目されている．しかし，分散ハッシュ上のオーバレイネットワークの構築には，物理ネットワークを一切考慮しない．そのため，検索ホップ中に無駄が含まれる可能性がある．そこで本稿では，物理ネットワークにおける遅延時間を物理的な近さであると位置づけ，それを分散ハッシュ上の ID に反映させる．また，それに加え階層化を行うことでより効率的に検索を行える新しい階層型の分散ハッシュ法を提案する．

# Hierarchical Distributed Hashing in consideration of physical network

Yuusuke HABA † Hiroshi MATSUO †

† Department of Computer Science and Engineering

Graduate School of Engineering Nagoya Institute of Technology

Distributed Hashing attracts attention of method which fast lookup a file or a node in peer-to-peer environment. However, to construction of overlay-network based on DHT is not consider of physical network, so waste may be had within a lookup hop. Therefore, in this paper, physical closeness cast delay time in a physical network. And we let ID on DHT reflect physical closeness. Additionally, to hierarchized by using the ID. We suggest a new hierarchy Distributed Hashing to lokkup more effectively.

## 1 はじめに

近年，インスタントメッセンジャーなと多くのピアツーピア (P2P) アプリケーションが使用されており，インターネット上でオーバレイネットワークが構築されている．しかし，ピュアピアツーピアネットワークでは，中央サーバが存在しないという利点は存在するものの，目的のファイルやノードを高速に検索することが困難である．そのため，ピュアピアツーピアネットワーク上で高速に検索することを可能にする分散ハッシュ法 (Distributed Hashing) や分散ハッシュテーブル (Distributed Hash Table : DHT) と呼ばれる手法が注目されている．分散ハッシュテーブルを使用し，その上でオーバレイネットワークを構築することで高速な検索が可能となる．

しかし，分散ハッシュテーブルを使用したオーバレイネットワークは，実際の物理ネットワークの近さが分散ハッシュ上での近さとは無関係である．そのため，オーバレイネットワーク上では無駄の無い検索ホップであっても，実際の物理ネットワークでは，一度遠くのネットワークに存在するノードまで検索が進んだ後に，検索を開始したノードの近くのノードに検索が戻って来る無駄が含まれる可能性がある．そこで本研究では，物理ネットワークでのノード間の遅延時間を物理ネットワークでの近さであると定義し，物理ネットワークでの近さを分散ハッシュ上での ID に反映させ，その ID を使用して階層化を行うことで，先に挙げたような問題点を改良する手法を提案する．

本稿は，2 節で分散ハッシュテーブルの関連研究，本研究の提案手法のもととなる Chord，提案手法の比較となる HIERAS の説明を行う．3 節で提案手法の説明をし，4 節で提案手法の効果を確認するため評価実験を行い，最後に 5 節でまとめる．

## 2 関連研究

分散ハッシュテーブルを用いる手法の代表的なものとして Pastry[2], Tapestry[3], CAN[4] などがあり，少数のノード情報をテーブルに保存することで，目的のノードを $O(\log N)$(N:ノードの数) で検索を行うことが可能である． Pastry は N 分木を使用することで，検索にかかるホップ数は $O(\log_b N)$ (b は ID の基数) である．

本研究で提案手法のもととした Chord[1] では，各ノードは SHA-1 などのハッシュ関数により自分の ID を生成する．自分の ID と他のノードの ID をもとに管理すべき領域や保存すべきノードの情報を決定することができるため，完全に分散して動作することが可能である． Chord では，ID 空間上で自分の ID から見て時計周りの方向に最初に存在する

ノードを successor として記憶する．また，同様に反時計周りに最初に存在するノードを predecessor として記憶し，その predecessor と自分との間の ID 空間を責任領域として管理を行う．また，各ノードは，ルーティングに用いる $m$ 行 ( $m : ID$ のビット数) のテーブル (フィンガーテーブル) を保持し，その $i$ 行目には $ID + 2^{i-1}$ の successor の情報が入る．つまり，各ノードは自分の 1,2,4,...,$2^{m-1}$ 先のノード情報を保持していることになる．Chord はこれらの情報のみを用い，検索にかかるホップ数を $O(\log N)$ に抑えている．

Chord を使用し，先に挙げたような分散ハッシュを使用したオーバレイネットワークの問題点を改良した手法として HIERAS[5] が提案されている．HIERAS 上の各ノードは，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) への遅延時間をもとにした，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． HIERAS ではこのように複数の Chord ネットワークを使用することで先に挙げた問題を改良している．しかし， HIERAS では各階層で Chord と同じ情報を保持しなければならないため，維持コストが増加する．また，階層化を行うことにより，物理層での遅延時間は減少する反面，アプリケーション層でのホップ数が増加することが報告されている [5].

## 3 提案手法

本研究では，Chord で使用される ID に物理ネットワークの近さを反映させ，その ID を利用し階層化を行う LCLV (Layerd Chord with Landmark Vector) を提案する．HIERAS では，リング ID と呼ばれる分散ハッシュ上の ID とは別の識別子を新たに生成し，それをもとに階層化を行っていた．しかし，LCLV では分散ハッシュ上の ID そのものに物理ネットワークでの近さを反映させる．この点が HIERAS との違いである．

### 3.1 ランドマークベクトルによる ID の生成法

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．そのような手法として，近傍の情報を使用して生成されるランドマーククラスタリングが広

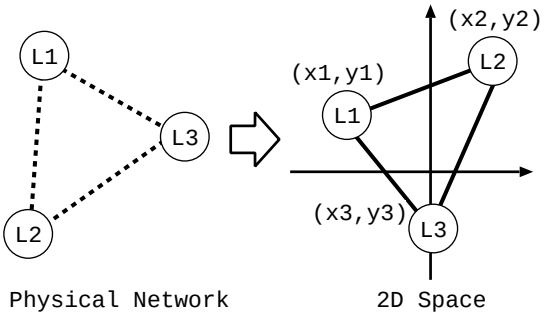

Fig. 1 Landmark

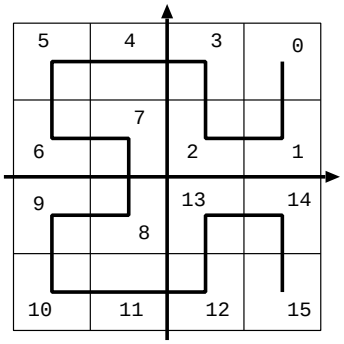

Fig. 2 Space Filling Curve

く使用されている [7][8][9]．また，近年，近隣情報とランドマークの情報のみを利用して負荷分散を行う DHT ベースの P2P システムが提案されている [6].

LCLV における新しいノード $N$ の ID の生成の流れを以下に示す.

1. 各ランドマークの 2 次元空間での位置 (座標) を求める.
2. 各ランドマークの座標と， $N$ から各ランドマークへの遅延時間から，$N$ の 2 次元空間での座標を求める.
3. 決定された座標から ID を生成.

まず，ランドマークノードを図 1 のように 2 次元空間にマッピングする (1.)．その方法として，各ランドマーク間の物理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，今 ID を決定したいノード $N$ の 2 次元空間での座標を求める (2.)．以上の方法で，物理ネットワークで近いノード同士は，2 次元空間においても近似的に近い座標を得ることが可能となる．

次に，2 次元空間をいくつかの空間に分け，そこに図 2 のように空間充填曲線 (Space Filling Curve : SFC) を引く．ここでは，空間充填曲線として Hilbert 曲線を用いた．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 (3.).

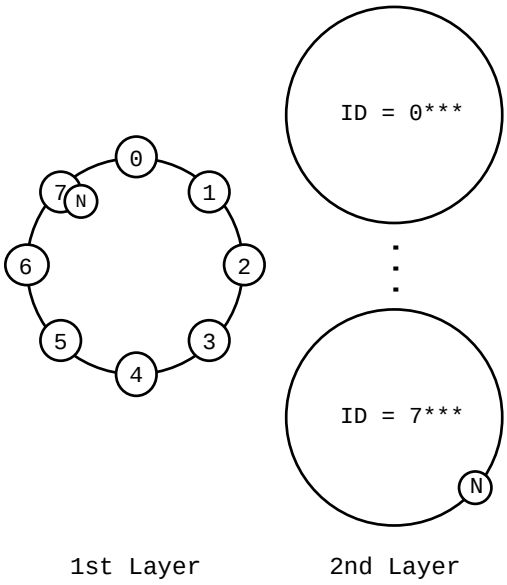

Fig. 3　Hierarchical

この方法を用いることにより，物理ネットワークで近いノード同士を，分散ハッシュ上での ID においてもおおよそ近い ID に割り当てることが可能となる．

### 3.2　階層化

LCLV における階層化は，Chord で用いる ID を数ビット毎に区切ることで行う．例として，ID の大きさが 12 ビット，階層数が 2，第一階層の ID が 3 ビット，第二階層の ID が 9 ビットとし，あるノード $N$ の ID が 8 進数表示で $ID = 7049$ と与えられた場合を考える．この場合，第一階層の ID は 7 となり，第二階層の ID は 049 となる．つまり，図 3 のように階層を分けることになり，ノード $N$ は図中の位置に配置される．このような階層化において，それぞれの階層は ID が小さいため，もとの Chord よりも小さい規模のネットワークとなる．このように LCLV は，小さな規模のネットワークに分けることでの階層化である．HIERAS は各階層で Chord と同じ大きさのネットワークを生成するため，LCLV の方が維持コスト，テーブル量において優位である．

LCLV では上位の階層の ID のみを 3.1 節で説明したランドマークベクトルによる ID を使用し，最下層の ID は通常の Chord と同様にハッシュ関数を使用して割り当てる．

このように ID を用いることで，図 3 においては，第一階層でのネットワークは物理的に遠いノード同士のネットワークとなるので，この階層における通信遅延は大きくなる．しかし，第二階層でのネットワークは物理的に近いノード同士で作られているので，通信遅延は小さくなる．

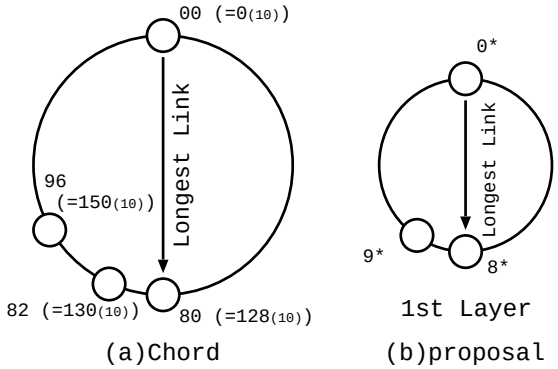

Fig. 4　Compare

LCLV では，次の理由によりホップ数減少の効果も期待できる．例として，図 4 に ID の大きさを 8 ビットとしたときの Chord と LCLV を比較した図を示す．図 4 の (a) に Chord の全体を，(b) に提案手法の第一階層のみを示す．なお，ノードの ID は 16 進数で表す．Chord では，最も遠方のリンクとして $80(= 128_{(10)})$ 先のノードの successor の情報を有する．つまり図 4 で，$ID = 00$ であるノードは $ID = 82(= 130_{(10)})$ であるノードの情報を持つ．LCLV においても，同様に最も遠方のリンクとして 80 先のノードの successor を知っていることになる．しかし，LCLV における第一階層に注目すると，ID が 8 から始まるノードの successor は ID が 9 から始まる (もしくは，それ以降) ノードになる．つまり，図 4 中で $ID = 00$ であるノードは $ID = 8*$ の successor である $ID = 9*$ のノード情報を持つ．これは，Chord では $ID = 96(= 150_{(10)})$ のノード情報を有することと同じ効果がある．このように，LCLV はより遠方のノード情報を持つ可能性が高いため検索にかかるホップ数が減少する効果が期待できる．

### 3.3　LCLV における検索

LCLV において $key$ を検索する手順を以下に示す．

1. 第一階層において $key$ に最も近い predecessor まで，Chord と同様の手法で検索を進め，次の階層に進む．
2. 第二階層においても同様に，$key$ に最も近い predecessor まで，Chord と同様の手法で検索を進め，次の階層に進む．
3. 以下同様に，最下層まで検索を進める．
4. 最下層において，$key$ を管理するノードを Chord と同様の手法で検索する．

以上のような検索の手順により，検索初期のホップは，上位の階層で検索が進む．3.1，3.2 節により，上位の階層は物理的に遠いネットワークとなるので，検索初期のホップでは 1 ホップあたりの通信にかか

Table 1　コスト比較

| | テーブル量 | 維持コスト |
|---|---|---|
| Chord | $m$ | $2+m$ |
| HIERAS | $l \times m$ | $l \times (2+m)$ |
| proposal | $m$ | $3n(l-1)+(2+m)$ |

る遅延時間が大きくなる．また，検索中期以降は下位の階層で検索が行われる．同様に 3.1，3.2 節より，下位の階層は物理的に近いネットワークとなるので，検索の中期以降のホップでは 1 ホップあたりの通信にかかる遅延時間が小さくなる．

また， Chord の検索は，検索の初期や中期以降に関わらず遅延時間の偏りは無いため，検索の全過程において 1 ホップあたりは同じ程度の遅延時間になる．しかし，LCLV では，検索の初期数ホップのみが大きい遅延時間になり，それ以降は小さい遅延時間になるためトータルでの遅延時間は小さくなる．

### 3.4　LCLV における保持すべき情報

LCLV において各ノードが保持する必要のある情報は， Chord と同様にフィンガーテーブル，successor，predecessor である．しかし， LCLV においては，上位の層で successor，predecessor を最大で $n$ 個保持する必要がある．これは，階層化による経路の集中を防ぐことと，ノードが故障したときの冗長化が目的となる．また，同様の理由により各階層で代表となるノードを $n$ 個覚える必要がある．また，一番下位の階層での情報は Chord と同じ情報を保持する． LCLV，Chord，HIERAS のコストの比較を表 1 に示す．ここで，$m$ は ID のビット数，$l$ は階層数，$n$ は LCLV における保存するノードの数である．

LCLV におけるテーブル量は，階層数に依存しないため Chord と同じ大きさである．しかし， HIERAS では各階層おいて Chord と同じ大きさのテーブルが必要であるため，階層数倍のテーブルが必要となる．

維持コストについて比較すると，LCLV は $m > 3n$ ならば HIERAS よりも維持コストが小さい．現実的なネットワークの規模を考慮した場合，$m = 160$ 程度であり，仮に $n$ を比較的大きな 50 としても維持コストは HIERAS よりも小さくなる．また，実際のピアツーピアネットワークでは $n = 5$ 程度で充分であると考えられる．

## 4　評価実験

LCLV の効果を評価するために， Chord，HIERAS，LCLV のそれぞれをネットワークシミュレータである NS-2[12] 上に実装し評価実験を行った．実験では，各ノードがランダムな $key$ を検索し，検索にかかった遅延時間の平均，アプリケーション層でのホップ数の平均，物理層でのホップ数の平均を測定する．

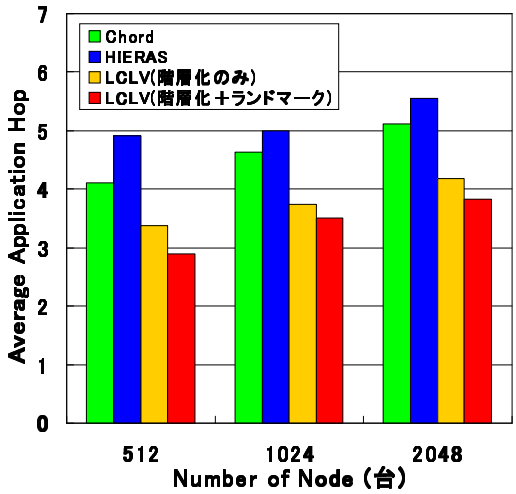

Fig. 5　Average Application Hop(TS-model 1)

また， LCLV においては，ランドマークベクトルによる ID の割り当ての効果を確認するため，階層化のみを行った場合，階層化とランドマークベクトルによる ID の両方を行った場合の二種類の実験を行った．

ID の大きさを 24 ビットとし，ノード数を 512 台から 2048 台まで変化させて実験を行った． LCLV における種々のパラメータは，ランドマークの数を 15 台，階層数を 3 とし，第一階層の ID は 4 ビット，第二階層の ID を 4 ビット，第三階層の ID を 16 ビットに設定した．また，実験に使用したネットワークは GT-ITM(TS モデル)[11] を使用した．TS モデルの設定において， HIERAS の論文 [5] では Transit-Transit(TT) 間の遅延時間を 100ms，Transit-Stub(TS) 間を 20ms，Stub-Stub(SS) 間を 5ms としていることを考慮し，本研究では，以下のような 3 つの設定で実験を行った．

1. TS モデル 1 (設定の変更無し)
2. TS モデル 2 (HIERAS と同じ設定
   TT 間 100ms，TS 間 20ms，SS 間 5ms)
3. TS モデル 3 (遅延時間の比率を変更
   TT 間 100ms，TS 間 80ms，SS 間 60ms)

3. については，HIERAS での設定の比率を変更し，より均質なネットワークに近づけた場合の HIERAS と LCLV の性能を比較する目的で用いた．

### 4.1　TS モデル 1 での結果

図 5 にアプリケーション層でのホップ数の比較のグラフ，図 6 に物理層でのホップ数の比較のグラフ，

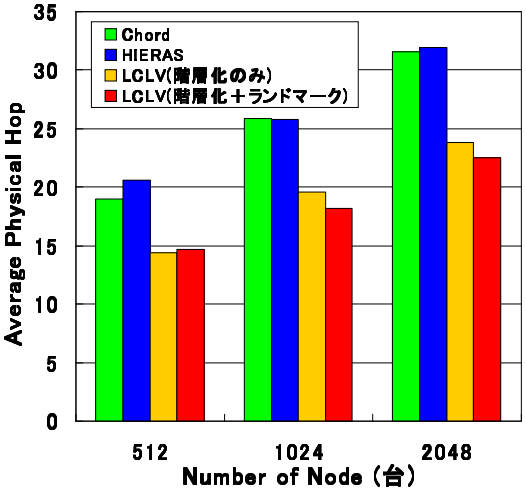


Fig. 6　Average Physical Hop(TS-model 1)

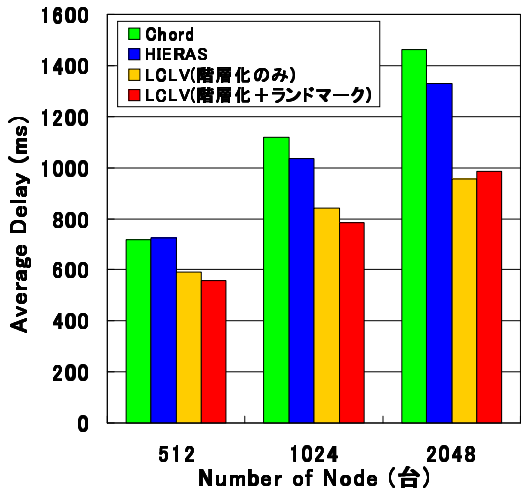


Fig. 7　Average Delay(TS-model 1)

図 7 に遅延時間を比較したときのグラフをそれぞれ示す．図 5 と図 6 より， HIERAS は Chord よりもアプリケーション層でのホップ，物理層でのホップ共にホップ数が増加していることが確認できる．また，3.2 節で説明したように， LCLV にはホップ数減少の効果があるため， Chord，HIERAS よりも検索にかかるホップ数が減少していることも同様に確認できる．

図 7 より， HIERAS は Chord よりも検索にかかる遅延時間がわずかながら減少している．これは，HIERAS の階層化の方法では， TS モデル 1 のような遅延時間に規則性の無いネットワークを適切に階層化出来なかったからと考えられる．

また，LCLV とその他の手法を比較すると，LCLV の方が他の手法よりも遅延時間が大きく減少している．これは，階層化のみの場合においても遅延時間が減少していることから，図 5 や図 6 においてホップ数が減少した分だけ遅延時間が減少したからである．

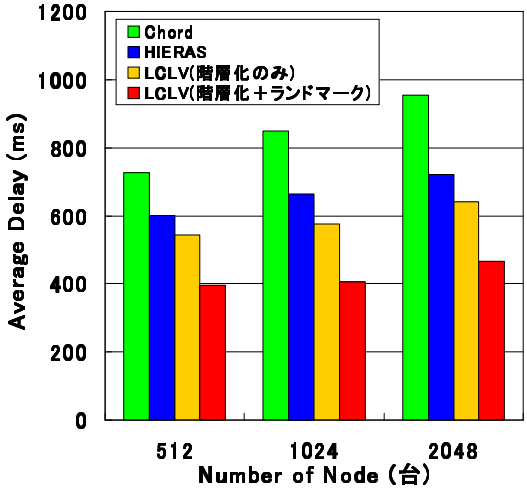


Fig. 8　Average Delay(TS-model 2)

LCLV 同士を比較してみた場合，階層化のみの場合，階層化とランドマークベクトルの両方を用いた場合の両方において同程度の遅延時間になっている．このときのランドマークベクトルを用いた場合の検索の過程を追跡すると，検索の初期，中期以降のいずれにも関わらず同程度の遅延時間になっていた．つまり，物理的に近いノード同士であるにも関わらず，分散ハッシュ上の ID では遠くなってしまう場合が多数存在していた．これは，図 2 において数字が 2 と 7 の位置は座標的には近いが分散ハッシュ上の ID 的には遠くなってしまっていることが原因であると考える．

### 4.2　TS モデル 2 での結果

図 8 に， HIERAS の論文と同じ設定である TS モデル 2 で実験を行ったときの遅延時間のグラフを示す．また，ホップ数に関してのグラフは 4.1 節でのグラフと同様の傾向であった．

図 8 に示すとおり， HIERAS は Chord よりも大きく遅延時間が減少している．このことは，先の 4.1 節の結果と違い，TS モデル 2 のネットワークは遅延時間に規則性があるため，ネットワークの階層化が適切に出来たためである．

また， LCLV と他の手法を比較すると， LCLV の方が他の手法よりも遅延時間が大きく減少している．これは，4.1 節と同様の理由である．

LCLV 同士を比較してみると，階層化のみに比べ，ランドマークベクトルも使用している方は，遅延時間が大きく減少している．これは，3.3 節で説明したように，ランドマークベクトルによる ID の生成により，物理的に近いノード同士が分散ハッシュ上でも近い ID を得ることが出来たことによる，遅延時間減少の効果である．

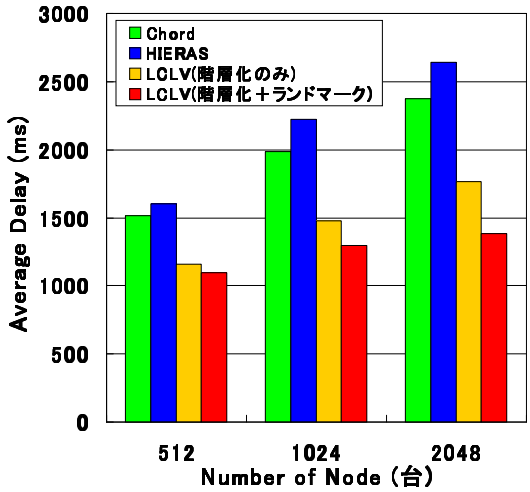

Fig. 9　Average Delay(TS-model 3)

### 4.3　TS モデル 3 での結果

図 9 に，遅延時間の比率を HIERAS の設定から変更した TS モデル 3 で実験を行ったときの遅延時間のグラフを示す．

図 9 より， HIERAS は Chord よりも遅延時間が大きくなった．このことは，TS モデル 3 のように，遅延時間に規則性はあるものの差が少ないネットワークでは， HIERAS によるネットワークの階層化を適切に行うことができず，逆に遅延時間を増加させる結果となったためである．

LCLV と他の手法を比較すると，図 7，図 8 と同様に他の手法より遅延時間が減少している．また，LCLV 同士を比較すると，HIERAS のように遅延時間が増加してしまうことは無く，ランドマークベクトルを使用した場合の方が遅延時間が減少している．これは，ランドマークベクトルによる ID の生成が，TS モデル 3 のように，遅延時間の差が少ないネットワークにおいても適切に働いたためである．

## 5　まとめ

分散ハッシュを使用したオーバレイネットワークは，物理ネットワークでの遅延時間などを考慮したものになっていないため，検索ホップ中に無駄が含まれてしまう可能性があった．本研究では，ランドマークベクトルを使用し物理ネットワークにおける近さを分散ハッシュでの ID に反映させた．また，その ID を使用して階層化を行うことで，検索にかかるホップ数，遅延時間共に減少させることができた．また別の手法による階層化よりも，より効率良く検索が可能であることを実験により確認した．

## 参考文献

[1] I. Stoica, R. Morris, D. Karger, M.Kaashoek, and H. Balakrishnan, ”Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”. Proc. ACM SIGCOMM, pp.149-160, Aug. 2001.

[2] A. Rowstron and P. Drushel, ”Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems”. Proc. 18th IFIP/ACM Int’l Conf. Distributed System Platforms (Middleware), pp.329-350, Nov. 2001.

[3] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, ”Tapestry: An Infrastructure for Fault-Tolerance Wide-Area Location and Routing”. Technical Report UCB/CSD-01-1141, Computer Science Division, Univ. of California, Berkeley, Apr. 2001.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, ”A scalable content addressable network”. Technical Report, TR-00-010, U.C.Berkeley, CA, 2000.

[5] Z. Xu, R. Min, and Y. Hu,1 ”HIERAS: A DHT Based Hierarchical P2P Routing Algorithm”. Proc. of hte 2003 International Conference on Parallel Processing, pp.187-194, Aug. 2003.

[6] Y. Zhu, and Y. Hu, ”Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems”. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL 16, No. 4, APRIL 2005.

[7] S. Ratnasamy, M. Handley, R.M. Karp, and S. Shenker, ”Topologically-Aware Overlay Construction and Server Selection”. Proc. IEEE INFOCOM, vol. 3, pp. 1190-1199, June 2002.

[8] Z. Xu, C. Tang, and Z. Zhang, ”Building Topology-Aware Overlays Using Global Soft-State”. Proc. 23rd Int ’l Conf. Distributed Computing Systems (ICDCS), pp. 500-508, May 2003.

[9] Z. Xu, M. Mahalingam, and M. Karlsson, ”Turning Heterogeneity into an Advantage in Overlay Routing”. Proc. IEEE INFOCOM, vol. 2, pp. 1499-1509, Apr. 2003.

[10] T. S. Eugene, Ng, H. Zhang, ”Towards global network positioning”. ACM SIGCOMM Internet Measurement Workshop 2001.

[11] E.W. Zegura, K. L. Calvert, and S. Bhattacharjee, ”How to model an internetwork”. Proceedings of the IEEE Conference on Computer Communication, San Francisco, CA, pp. 594.602, Mar. 1996.

[12] http://www.isi.edu/nsnam/ns/