P2P ドロネーネットワークにおける遠隔接続経路の自律分散生成法

大西真晶[†] 坪井新治[†] 平山雅夫[†]
江口隆之[†] 上島紳一[†]

本稿では，P2P ドロネーネットワークにおける遠隔接続経路群（LRC）の自律分散生成法および遠隔地点への経路選択と範囲問合せへの適用について述べる．ドロネー図の構造を持つオーバレイネットワークである P2P ドロネーネットワークでは，各ノードは近傍ノードのみと接続する．この特徴は地理的な経路選択や範囲問合せに有効である．しかし，ノード数の増加に対してネットワークの直径が増大するため，遠隔ノード間の通信や広範囲の問合せに要するクエリのホップ数が増大し，通信遅延が生じるという課題が残されていた．そこで，空間全体に LRC を構成することで 2 ノード間のホップ数を低減する．我々は，平面上の各ノードが，ホップ数に基づいた水平/鉛直方向の経路を協調的に生成することで，平面全体への LRC を完成させるボトムアップな生成法を与える．この水平/鉛直の LRC を用いた経路選択法，効率的な範囲問合せ法を提案する．さらに，ノードの参加・離脱時に，LRC を部分的に修正し構造を維持するアルゴリズムについて述べる．そして，数値シミュレーションにより，LRC の生成と維持にかかる負荷や LRC を用いた経路選択と範囲問合せの効果について検証した．また，応用システム例として災害時の被災者支援システムを提案し，実利用を想定したシミュレーションにより，LRC を組み込んだ P2P ドロネーネットワークおよび関連システムの GeoPeer に対して 2 種類の構造を設定し，3 者を比較評価した．

Building Long Range Contact over P2P Delaunay Network Distributively

Masaaki Ohnishi,[†] Shinji Tsuboi,[†] Masao Hirayama,[†]
Takayuki Eguchi[†] and Shinichi Ueshima[†]

In this paper, the authors propose an autonomous distributive generation algorithm of LRC (Long Range Contact) over P2P Delaunay network, and discuss on a routing method to distant location and on range query. In a P2P Delaunay network, which builds-in a Delaunay diagram as a topology using node location, a node has connections only with its neighboring nodes. This structural property has advantages for a geometric routing and range query mechanism among nodes. However, in case a P2P Delaunay network consists of a large number of nodes, its diameter increases which causes a serious communication delay in the wide area of range query. To overcome the problem, the authors show a method for nodes to construct LRC collaboratively in a bottom-up fashion both horizontally and vertically over a P2P Delaunay network to the entire target space, and provide algorithms for geometric routing as well as efficient range queries by use of horizontal/vertical LRC's. Moreover, algorithms for partial structural maintenance due to node join/leave are given. The authors examine the loads for generating/maintaining LRC, and the efficiency for routing and range query using LRC through numerical simulations. As a practical application, we set a scenario of a communication network for victims in a disaster area and rescue crues, and compare the efficiencies between the proposed system and two types of network topologies using GeoPeer numerically.

1. はじめに

近年，P2P ネットワークを用いた空間情報システムの研究が注目を集めている[1)~4)]．これらのシステムでは，情報が発生した位置などをキーとするデータの複数ノード上での分散管理について述べられており，ノードの負荷均衡を保ちつつ，ノードの脆弱性を考慮した P2P ネットワーク構成やデータ管理を行っている[5)]．

我々も，空間内の情報を分散的に管理する目的で**P2P ドロネーネットワーク**を提案している[6),7)]．このネットワークは，ノードの位置をもとにノード間をドロネー図状に接続し，IP ネットワークのようなベー

† 関西大学大学院総合情報学研究科
Graduate School of Informatics, Kansai University

スネットワークのオーバレイとして構成する．そして，(i) ノードの低次数，(ii) 対象平面の被覆性，などのドロネー図の幾何学的な特徴を利用することで，地理的な経路選択や範囲問合せを効率的に行うことができる特徴を持つ．さらにノードの位置を利用しているため，対象空間を拡張することもでき，P2P システムの持つスケーラビリティを活かした大規模の空間情報システムの実現も可能にしている．しかし，一方で，各ノードが近傍のノードのみと接続されるため，ネットワーク内のノード数が増加すると，直径が大きくなるという課題が残されていた．つまり，遠隔地のノードとの通信に，クエリが経由するノード数（ホップ数）が増加するため，通信の遅延が発生する．

そこで，本稿では，P2P ドロネーネットワークにおける任意の 2 ノード間のホップ数を低減させる短縮経路として，遠隔接続経路（Long Range Contact 以下，LRC）とその P2P 方式による自律分散生成法を提案する．提案手法では，各ノードが自律的にボロノイ領域を生成し，ボロノイ領域の最大幅を基準に他ノードとの通信量を制限しながら，通信領域内で選択したノードと通信を行うことで，対象平面全体に対して，ノードが協調的に LRC を生成することができるという特徴を持つ．

我々が提案する LRC は以下の特徴を持つ．

- 平面上の任意の位置に分布した 2 ノード間を $O(\log N)$ のホップ数で到達できる（N は総ノード数）．
- 平面上の指定した範囲への問合せを $O(\log N)$ のホップ数で実行できる．
- LRC を備えたノードの次数は，N の増加に対し，N の対数に比例する．
- 対象平面の拡張に対しても LRC を容易に構成できる．
- ノードの新規参加/離脱時にも LRC の構造を維持することができる．

本稿では，LRC を用いた経路選択アルゴリズムを与えて，遠隔地への通信や指定した範囲への問合せに対する有効性を検証する．

また，提案手法を組み込んだ P2P ドロネーネットワークの応用として，災害発生時の被災者支援システムの構築を考える．被災者が小型通信端末を保持し，提案ネットワークを構成することで，(i) 被災者の位置の把握，(ii) 被害状況の把握，(iii) 位置に則した被災者への情報提供を高速に行うことができる．この利用想定に基づいたシミュレーションにより，提案ネットワークと従来手法である GeoPeer[2] を比較し，評価する．

ノード位置を明示的に用いて空間情報を管理する P2P モデルとして GeoPeer[2] がある．Araújo ら[2),8)] は，基盤ネットワークにはドロネー図を用い，さらに，必要に応じて指定した 2 ノード間に短縮経路を構成して 2 つのドロネーネットワークを接続している．短縮経路は指定した 2 ノード間の欲張り法によるホップ数を基準に構成している．

これに対して提案手法では，任意の方向にある遠隔地への通信や指定範囲への問合せを目的として，ネットワーク内のノードすべてが自律分散的にホップ数に基づいて LRC を構成することで，平面上におけるすべての地点間のホップ数を短縮する経路の構成を実現している点が異なる．このため，構成の手間は大きくなるが空間全体に LRC を構成する利点が得られる．また，LRC の作成は，Araújo らの手法では，1 つのノードが 1 ホップずつ経路を進みながら構築するのに対して，提案手法では，他のノードが作った LRC 経路を利用して，複数のノードが協調して各ノードの LRC を構成する点が異なっている．前者は lazy な構成法，後者は eager な構成法とみることができる．

また，Naor ら[9)] は，与えられたノード集合へ連続的データを配置する目的で，距離 2 分化（distance halving）リンクを提案し，連続領域内の任意の 2 地点間のホップ数を $O(\log N)$ で実現する手法を提案している．特に，平面上でノードの分布が一様でない場合に対して，ボロノイ領域とドロネー図を利用しホップ数を低減する手法を示唆しているが，具体的な構成アルゴリズムを与えていない．これに対し本提案手法では，P2P 方式を用いて具体的にネットワークのボトムアップな構成法を提案し，シミュレーションによる評価を与えている．

提案手法は，管理対象の領域を拡張する場合や同領域を管理する複数のネットワークを融合する場合にも適用できる手法であり，地理情報システムや空間データベースの基盤として広く適用可能である．

以降では，2 章で P2P ドロネーネットワークの特徴と LRC 構成要素について，3 章では LRC の自律分散生成アルゴリズムについて述べる．4 章では LRC の利用として範囲問合せと遠隔地点への経路選択とノードの参加/離脱に対する構造の維持について述べる．5 章では LRC の生成負荷と効果についてシミュレーションで評価する．6 章では，現実的な応用面から提案手法を評価する．また，7 章では提案手法と関連研究について比較する．

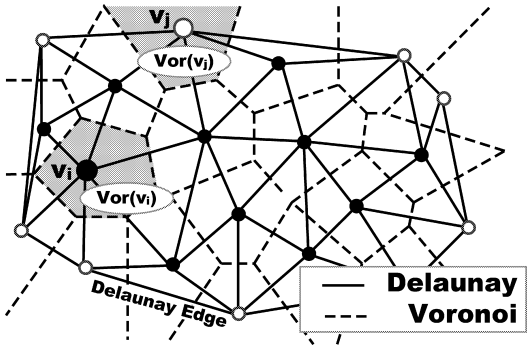

図 1 P2P ドロネーネットワーク
Fig. 1 P2P Delaunay network.

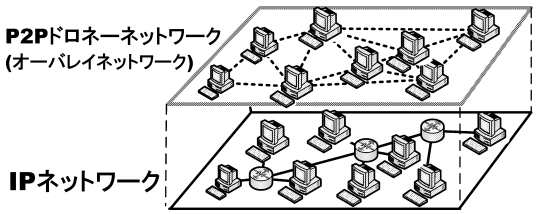

図 2 P2P ドロネーネットワークを IP ネットワークのオーバレイに構成
Fig. 2 P2P Delaunay network on IP network.

2. P2P ドロネーネットワークと LRC の構成要件

2.1 基本的な特徴

図 1 に P2P ドロネーネットワークを示す．ノードを計算主体，ドロネー辺を通信経路とする．ノード v_i のボロノイ領域 $Vor(v_i)$ は，ノードの中で最も近いノードが v_i である平面上の点の集合である．本稿では，IP ネットワークのオーバレイに P2P ドロネーネットワークを構築することを想定しているので，各ホップに対して実際のデータ転送は IP ネットワークの転送に置き換えられる（**図 2**）．

P2P ドロネーネットワークでは，各ノード v_i は $Vor(v_i)$ 内のデータを管理する．すなわち，位置に即したデータは，地理的に最も近いノードに格納される．P2P ドロネーネットワークは以下の特徴を持つ．

局所性：ノードは地理的に近いノードと接続され，局所的な通信がネットワーク全体に影響を与えない．そのため，ネットワークのスケーラビリティに優れ，大量のノードからなるネットワークを構成できる．また，対象平面の拡張にも，ネットワークを伸張することにより対応することができる．

低次数：ボロノイ図の持つ幾何学的特徴により，6 前後となる（オイラーの定理[10] より）ため，隣人表（経路表）が小さく構成できる．ノード次数はネットワークサイズに依存しない．

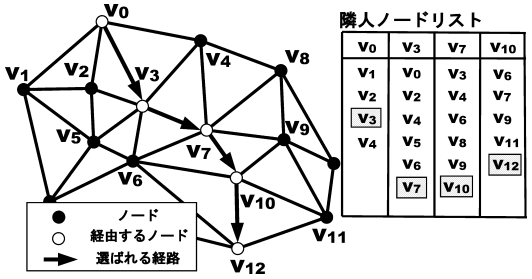

図 3 欲張り法による経路選択
Fig. 3 Routing by Greedy forwarding.

被覆性：各ノードのボロノイ領域の全体が対象平面を被覆するため，すべての位置に即したデータを矛盾なく，適切なノードに格納できる．また，ノードの新規参加/離脱に対しても，そのつど，自律分散アルゴリズムを局所的に実行することにより，構造を更新することができ，ネットワーク全体に影響を与えない．

欲張り法：通信において各ノードで幾何的経路選択が可能である．つまり，目的地ノードに対して，各ノードでホップの方向が一意に定まり，位置座標を指定したクエリの転送が可能である．各ノードは，ドロネー辺により接続しているノードの中から目的地点までの距離が最も近いノードを選択し，クエリを転送する．

図 3 に v_0 を出発ノード，v_{12} を目的ノードとした欲張り法による経路選択の例を示す．v_0 はドロネー接続するノードから v_{12} までの距離が最も近い v_3 を選び，クエリを転送する．以下同様に，v_3，v_7，v_{10} においても隣接ノードから v_{12} までのユークリッド距離の近い v_7，v_{10}，v_{12} をそれぞれ選択してクエリを転送し，目的地点まで到達する．

到達可能性：ノードのボロノイ領域の全体が対象平面を被覆しており，また，ドロネー辺に沿って通信経路を定めているため，任意のノードから別の任意のノードに対して P2P ドロネーネットワークを通って到達可能である．さらに，任意に指定した地点への問合せを，マルチホップ方式で到達させることができる．

範囲問合せ：クエリが互いに隣接するボロノイ領域を結ぶドロネー辺上を移動するため，範囲問合せを行いやすい．**図 4** に v_0 が黒枠で示された範囲に対して問合せを行う例を示す．まず，v_0 は欲張り法によって v_3 にクエリを転送する．v_3 は問合せ範囲とボロノイ領域が交差するため，クエリを転送してきた v_0 以外の隣接ノードのすべてにクエリを転送する．問合せ範囲にボロノイ領域を含む v_4，v_5，v_6 では v_3 同様にフラッディングを行う．このようにボロノイ領域が問合

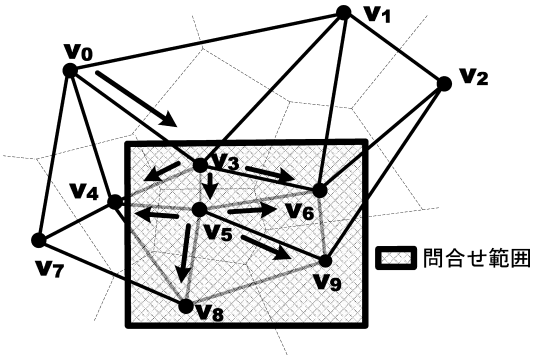

図 4 P2P ドロネーネットワークでの範囲問合せ
Fig. 4 Range queries on P2P Delaunay network.

せ範囲に含まれるすべてのノードは，クエリのフラッディングを行う．一方，v_1 は問合せ範囲から外れるため，範囲問合せのクエリを破棄する．また図中 v_4，v_6 のように v_3 と v_5 から重複してクエリを受信した場合，2 度目以降に受信したクエリを破棄して，それ以上転送しない．

Note：提案手法では，このように対象平面をボロノイ領域の全体とみて，ドロネー辺により各ボロノイ領域を接続する構造とみている．これにより，上記のネットワークの性質が得られノード位置に基づいて管理領域が明確に定義されている．GHT などのデータ中心型格納法では，物理ノードの存在する実平面と DHT が作る論理平面を一体化してとらえ，物理ノードにデータを格納する方法を用いている[4),5)]．同手法では，対象平面をタイル型に矩形分割することで，各ノードの担当範囲を決定しており，隣人ノード間でホップすることで，P2P 方式によるデータの分散管理を実現している．これに対して，本提案手法では，各データにとって距離的に最も近いノードにそのデータを格納し，ボロノイ領域の隣接関係を表すドロネー辺で通信を行っている．

2.2 LRC の構成要件

2.1 節で述べたように，P2P ドロネーネットワークは，ノードの位置的な隣接関係によりネットワークを接続するため，拡張性に優れる反面，ノード数の増加に対して，ホップ数に関してネットワークの直径が増大する．実際，平面上に一様に分布した N ノードに対して，$O(N^{1/2})$ の直径となる．このため，遠隔ノード間の通信や広範囲を対象とした範囲問合せにおいて，通信の遅延が起きる．これでは上位レイヤで構築される応用システムに対してネットワーク基盤として不十分である．そのため，P2P ドロネーネットワークの上に短縮経路を構成し，ホップ数を低減させることで，通信遅延を低下させることを考える．

LRC 構成のための要件は，以下のとおりである．

- **（要件 1）** LRC を用いて任意の 2 点間を結ぶホップ数が $O(\log N)$ 程度である．これは，始点ノードから終点ノードへのホップ残数がホップごとに半減する収束割合を意味し，既存の Chord やグラフの短縮路の生成にも用いられる実質的な割合である．
- **（要件 2）** P2P ネットワークの特性であるノードの新規参加/離脱への対応，ならびに対象平面の拡張性の保持のため，LRC を自律分散的に構成できること．サーバなどを配置すれば，ネットワークのスケーラビリティが失われるため，LRC 構成においても自律分散生成が望ましい．
- **（要件 3）** 平面の任意に指定した矩形部分への範囲問合せがスケーラブルに実現できること．ボロノイ領域の隣接性に基づく，ドロネーネットワークを用いた範囲問合せがスケーラブルに行えることを意味する．
- **（要件 4）** 平面上で任意のノード分布に対して適用できる構成法であること．平面上のノードの分布状況は，応用を考慮すると一般にノード分布は一様とは限らないためである．
- **（要件 5）** ノード次数が低く抑えられること．各ノードの持つ隣人表（経路表）のサイズが低く保たれるのが望ましい．

次章では，これらの要件を満たす LRC の具体的な生成アルゴリズムを与える．

3. LRC 生成アルゴリズム

本章では，上記要件に基づき，ノードが自律分散的に遠隔接続経路として LRC を構成し，すべてのノードが協調して P2P ドロネーネットワーク全体に LRC を生成するアルゴリズムを与える．

3.1 基本的な考え方

3.1.1 ノードの前提

ノードは以下の前提に基づく[6)]．

- ノードは自律的に動作する計算主体である．
- 通信プロトコルに IP（Internet Protocol）を用い，他ノードと協調して IP オーバレイネットワークを形成する．
- ノード間通信はマルチホップ方式とする．
- ノードの経路表に，ドロネー隣接ノードと LRC ノードの情報を保持する．

また，各ノードが経路表に保持する隣人ノード（LRC ノード）情報を，次の情報を含む構造体とする．

- ノード ID
- IP アドレス

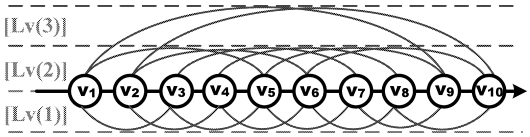

図 5 1 次元ネットワークにおける LRC
Fig. 5 1-dimensional network.

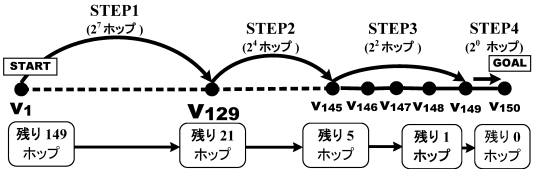

図 6 LRC による経路決定例
Fig. 6 Example of routing on LRC.

- 位置座標
- Lv（このノードがどの Lv の LRC ノードかを示す．これについては次節で述べる）．
- ボロノイ領域情報

3.1.2 1 次元ネットワークにおける LRC

まず，隣接ノード間が 1 次元に連結する P2P ネットワークにおける LRC の構成法について考える．**図 5** に，LRC 生成後のネットワークを示す．各ノードは 2^i（$i \geq 0$）先のノードとの遠隔の経路を構成している．すべてのノードがこのような経路を構成することで，任意の 2 ノード間を $O(\log N)$ のホップ数で到達する経路を構成することができる．たとえば，v_1 は隣接ノード v_2 以外に，v_3，v_5，v_9 に対して遠隔の経路を構成している．v_1 は隣接ノードの経路のみを用いると v_5 まで 4 ホップかかるが，遠隔経路を用いることで 1 ホップで到達できる．

各ノードが LRC で接続する遠隔ノードを **LRC ノード**と呼ぶ．また，特に 2^i ホップ先の LRC ノードを **Lv(i) ノード**，2^i ホップ先の LRC ノードに接続する経路を **Lv(i) 経路**と呼び，これらの遠隔基準の単位を単に **Lv** と呼ぶ．

• LRC の生成

この遠隔接続経路では，Lv(i+1) ノードは Lv(i) ノードの Lv(i) ノードになっているため，各ノードは，Lv(i) ノードに問い合わせることで Lv(i+1) ノードのアドレスを取得する．たとえば，図 5 で v_1 は，v_1 の Lv(1) ノードである v_3 に問合せを行うことで，の Lv(2) ノード v_5 のアドレスを取得する．これを繰り返すことで，各ノードは LRC を段階的に生成する．

• LRC を用いた経路選択

n を目的ノードまでのホップ数，v_0 を出発ノードとしたとき，v_0 は，$i \leq \lfloor \log n \rfloor$（$\lfloor \cdot \rfloor$ は床関数）を満たす i を決定する．v_0 は Lv(i) ノードにクエリを転送し，受信した Lv(i) ノードは同様にクエリを転送する．これを繰り返し，クエリを目的ノードに到達させる．

図 6 で，v_1 を出発ノード，v_{150} を目的ノードとして LRC を用いた経路選択例を示す．まず，v_1 は Lv(7) 経路を用いて v_{129} にクエリを転送する．同様に，v_{129} は Lv(4) 経路を用いて v_{145} に，v_{145} は Lv(2) 経路を用いて v_{149} に，v_{149} は隣接ノード v_{150} にクエリを伝播する．その結果，LRC を用いることで，v_1 から v_{150} まで 4 ホップで到達できる．

Note：この遠隔接続経路の構造および経路選択法は，構造型 P2P ネットワークの Chord [11] の手法と同じである．提案手法や Chord では，ノード間のホップ数を再帰的に 2 分する形に遠隔経路を構成し，2 ノード間のホップ数を短縮している．ただし，Chord がノードの ID 空間を 2 分するように経路を構成するのに対して，提案手法では 1 次元に並んだノードの順序に関して経路を構成する．これは提案手法が，Chord と異なり，ノードが空間に一様に分布することを仮定しないためである．

3.1.3 平面における LRC

次に，平面上にノードがランダムに分布し，ノードがドロネー辺によって P2P ドロネーネットワークとして接続されている場合を考える．この場合，平面上のノードが一直線に並ぶことが期待できず，1 次元の場合の方法をそのまま拡張できない．各ドロネー辺に LRC を構成することで，全方位への LRC を構成することも考えられる．しかし，各ノードは直線上に並んでいないため，各ノードが互いの経路を利用して協調的に LRC を生成することが難しい．そこで平面上では，水平/鉛直方向にノードのボロノイ領域の幅を考慮して LRC を生成することを考える．この水平方向と鉛直方向の LRC を利用して目的地への経路選択を行う．

•帯の幅

ボロノイ領域が平面全体を分割することに着目し，各ボロノイ領域の最大幅を用いて帯を設定する．各ノードは自身のボロノイ領域の最大幅を持つ**帯**を利用し，その帯とボロノイ領域が交差するノードを LRC ノードとする経路を構成する．つまり，水平方向の場合，各ノードは，自身のボロノイ領域の鉛直方向の最大幅を計算し，その幅を持つ帯とボロノイ領域が交差するノードを LRC の構成対象ノードとする．

図 7 に，v_0 が水平方向の帯を構成する様子を示す．v_1 から v_9 のボロノイ領域は v_0 の帯と交差するため，

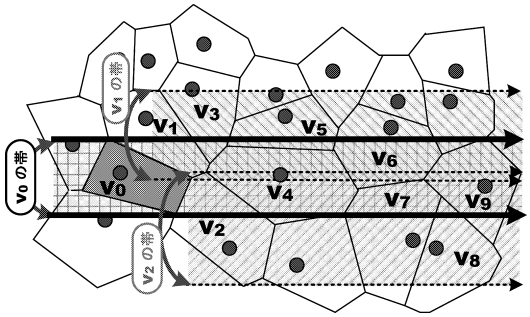

図 7 v_0 の水平帯とボロノイ領域が交差するノード群
Fig. 7 The nodes whose Voronoi regions intersect with the horizontal belt of v_0.

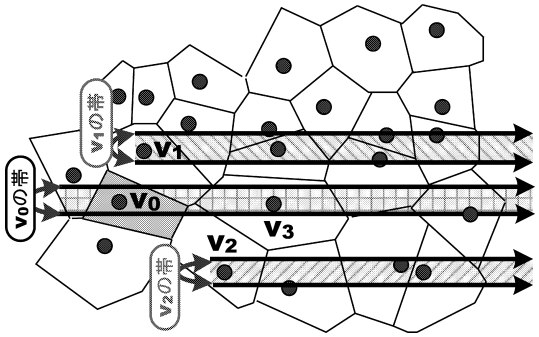

図 8 ボロノイ領域の最大幅より狭い幅の場合
Fig. 8 Narrower than the maximum width of Voronoi region.

これらのノード群が v_0 の LRC の対象のノードとする．同様に，各ノードは，鉛直方向にも自身のボロノイ領域の水平方向の最大幅を計算し，その幅を持つ帯とボロノイ領域が交差するノード群を，LRC の構成対象ノードとする．

提案手法において，水平/鉛直の 2 方向に幅を決定するのは，以下の理由による．

•平面で互いに独立な 2 つの一定方向を定めておけば，P2P の対称的な通信方式をすべてのノードで等しく行うことで，LRC を協調的に生成しやすい．

•各ノードの水平帯幅をボロノイ領域の鉛直方向の最大幅は P2P 通信に必要十分な大きさである．すなわち，あるノード v_0 の LRC ノードを v_0 の帯とボロノイ領域が交差するノードと定めておけば，v_0 が他ノード v_1 からノード情報の問合せを受けた場合にも，v_0 は自身の LRC ノードの中から v_1 の帯とボロノイ領域が交差するノードを返すことができる．そのため，v_0 は自身の帯幅とボロノイ領域が交差するノードと通信し，LRC を構成しておけばよい．

実際，帯がそれより狭い場合に水平/鉛直方向に LRC を構成できない．**図 8** に帯幅が狭い場合を示す．ここで v_0 の帯とボロノイ領域が交差するノード v_3 は，v_0 の LRC ノードである．しかし，v_0 の LRC ノードである v_1，v_2 の帯が v_3 のボロノイ領域とは交差していないため，v_3 は v_1，v_2 の LRC ノードではない．そのため，v_0 は v_3 と情報交換することができず，v_3 への LRC を構成できない．

また，幅が大きい場合は，v_0 は自身の管轄でない部分の情報が必要となる．したがって，この幅は必要十分な大きさである．

•各ノードの水平帯群は，平面を被覆する（**図 9**）．このため，各ノードの水平帯を利用することで水平方向に対して任意の地点の管理ノードへ経路選択ができる．また，帯は鉛直方向の帯についてもそのままあてはまる．つまり，これら水平/鉛直 2 方向の帯群が空間全体を被覆することで，LRC を用いて平面上の任意の地点へ到達できる経路を確保することができる．

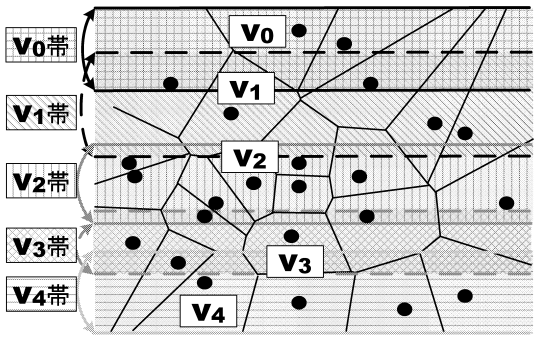

図 9 各ノードの帯による平面全体への被覆
Fig. 9 The belt of nodes which cover the entire plane.

提案手法のように，水平/鉛直方向の帯を設定することで，(i) ノードによる通信量の制限と，(ii) ノードの対称かつ，協調的な動きを可能にしながらも，(iii) 平面の被覆性が失われず，さらに，平面上の 2 ノードを結ぶ複数の経路が確保できる．

3.2 LRC の自律分散生成法

各ノードが自律的に実行するアルゴリズムは，次の部分処理からなる．

(1) **ボロノイ領域の自律生成**：ドロネー隣接ノードの位置情報を用いてボロノイ領域を生成する．

(2) **LRC 経路の協調的生成**：Lv(i) ノードから Lv(i+1) 経路を協調的に構成する．

(3) **Lv(i) ノード決定**：(2) 内において，水平/鉛直帯を用いて LRC ノードを決定する．

また，以降で用いる {**東，西，南，北**} の方向を，それぞれ次のように定義する．平面上のあるノード v_1，v_2 の XY 座標がそれぞれ (x_1, y_1)，(x_2, y_2) であるとき，

- **東**：$x_1 < x_2$ ならば，v_2 は v_1 の東方向のノード，
- **西**：$x_1 > x_2$ ならば，v_2 は v_1 の西方向のノード，
- **南**：$y_1 > y_2$ ならば，v_2 は v_1 の南方向のノード，
- **北**：$y_1 < y_2$ ならば，v_2 は v_1 の北方向のノード，

である．以降の図では，左下に原点があるものとするとするものとする．つまり，図の右方向を東，左方向を西，上方向を北，下方向を南とする．

Note：(i) 本稿では，説明の簡単のためノードは同期的に動作することを仮定するが，非同期に実行させることも可能である．(ii) 図 1 の v_j のような非有界ボロノイ領域を管理領域とするノードは，ドロネー図の周縁部（凸包）上に存在する（図 1 中白丸ノード）．それらのノードでは，ノードの外側に対して LRC が必要ないため，外側へ向けては構成しないものとする[12]．

3.2.1 ボロノイ領域の自律生成

ドロネー図とボロノイ図の双対性を利用し，平面全体をノードを母点としたボロノイ領域に分割する．提案手法では，すべてのノードがそれぞれのボロノイ領域を生成することによって，分散的に平面全体をボロノイ領域分割する．

[ボロノイ領域生成アルゴリズム]

図 10 のノード v_0 を例に，ボロノイ領域の自律生成アルゴリズムを説明する．

STEP1：v_0 は自身のドロネー隣人情報を真北を始点として時計回りに並べ替える．v_0 の隣人リストはノード $\{v_1, v_2, v_3, v_4, v_5\}$ となる．

STEP2：v_0 は v_1 と v_2 を取り出し，$\triangle v_0v_1v_2$ の外心を計算する．この外心をボロノイ点 w_1 とする．

STEP3：同様に，$\triangle v_0v_2v_3$ より w_2，$\triangle v_0v_3v_4$ より w_3，$\triangle v_0v_4v_5$ より w_4，さらに $\triangle v_0v_5v_1$ からボロノイ点 w_5 を取得する．これにより，すべての隣り合う三角形の外心の計算を完了する．

STEP4：v_0 は外心 $w_1, .., w_5$ を時計回りに接続する．そして，ボロノイ領域分割処理を終了する．

3.2.2 LRC 経路の協調的な生成

各ノードが東西南北の 4 方向に，帯とホップ数に基づいて，LRC を生成するアルゴリズムについて述べる（**図 11**，**図 12**）．3.1.2 項に示した要領で，各ノードは自身の Lv(i) ノード群から Lv(i+1) ノード群の情報を取得する．各ノードが自律分散的にこれを繰り返し，LRC を協調的に生成する．

[LRC 経路生成アルゴリズム]

図 12 に，ノードが協調的に LRC 経路を生成するアルゴリズムを示した．ここでは，アルゴリズムで使用される変数，関数について述べ，さらに，図 11 における v_0 を例にアルゴリズムの手順を述べる．

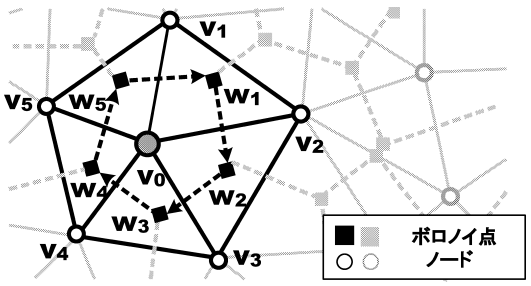

図 10 ボロノイ領域の分散生成：v_0 がドロネー隣人ノードの位置から自身のボロノイ領域を生成

Fig. 10 Autonomous generation of a Voronoi region: v_0 constructs its Voronoi region with Delaunay neighbor node.

まず，以下が図 12 において使用される変数，関数である．

direction：東西南北の方角の 1 つを，$X+$，$X-$，$Y+$，$Y-$ のいずれかとして格納する．

voronoiBelt：垂直/平行帯の端となる平行線を示す 2 つの x 座標値か y 座標値の組．

voronoiVertices：ノードのボロノイ領域である多角形の頂点座標のリスト．

delaunayNeighbors：ノードがあらかじめ構築している P2P ドロネーネットワークにおける隣接ノード情報のリスト．

connectionLv[x]：ノードが持つ LRC の Lv ごとの接続先ノード情報のリストのリストであり，x を指定することによって，任意の Lv 数のリストにアクセスできる．

makeVoronoiBelt(voronoiVertices, direction)：voronoiVertices をもとに，指定された方向軸に平行な voronoiBelt を計算して返す．

selectLRCNodes(nodes, direction)：ノード情報のリストの中から，direction の方向への LRC となるようなノードの組を選び出し，そのノード情報のリストを返す．詳細は，後述の **[Lv(i) ノード決定アルゴリズム]** に示す．

connectionLv[x].getConnectionLv(y, voronoiBelt,direction)：connectionLv[x] の中に格納されたノード情報の指すすべてのノードについて，connectionLv[y] の中のノード情報の中でそのボロノイ領域と，voronoiBelt の帯領域の direction 方向の部分が重なるすべてのノード情報のリストを取得してマージし，1 つのリストを生成して返す．

変数，関数については以上である．

次に，図 11 におけるノード v_0 が東方向に LRC を構成する場合を例として図 12 について解説する．v_0

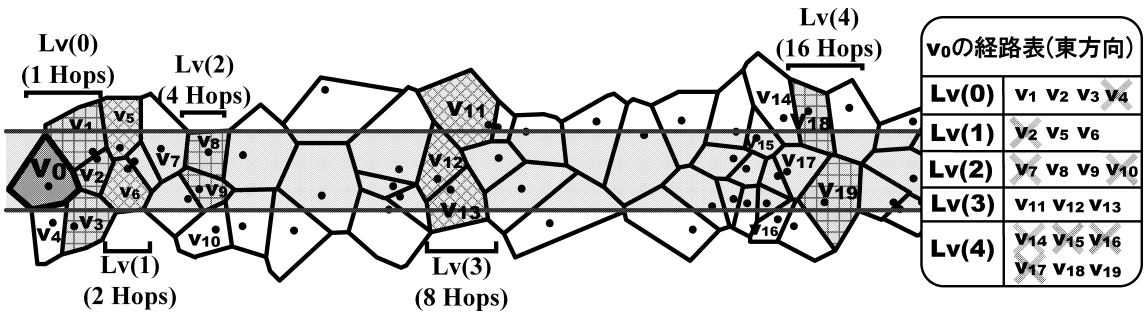

図 11 P2P ドロネーネットワーク上の遠隔接続経路：v_0 による東方向の遠隔接続経路
Fig. 11 LRC on P2P Delaunay Network: links of v_0 toward east.

```
Node.generateLRC( direction )
 1: voronoiBelt = makeVoronoiBelt(voronoiVertices, direction)
 2: connectionLv[0] = selectLRCNodes(delaunayNeighbors, direction)
 3: i = 1
 4: DO
 5:     IF ( connectionLv[i] = connectionLv[i-1]
                  .getConnectionLv(i-1,voronoiBelt, direction )
          == "Fail")
 6:         THEN EXIT DO
 7:     connectionLv[i] = selectLRCNodes(connectionLv[i], direction )
 8:     i = i + 1
 9: LOOP
```

図 12 遠隔接続経路の構成処理
Fig. 12 Algorithm for collaborative generation of LRC.

```
Node.selectLRCNodes(prospectiveNodes, direction)
 1: border = voronoiBelt.getUpper()
 2: DO
 3:     IF voronoiBelt.getBottom() > border
 4:         EXIT DO
 5:     crossedNodes = prospectiveNodes.getCrossNodes(border)
 6:     crossedFarthestNode =
            crossedNodes.getFarthestNode()
 7:     ADD crossedFarthestNode TO longRangeNodesList
 8:     border = crossedFarthestNode.voronoiVertices.getBottom()
 9: LOOP
10: RETURN longRangeNodesList
```

図 13 Lv(i) ノード決定アルゴリズム
Fig. 13 Algorithm for selection of Lv(i) nodes.

は，これと同様の手順で東方向以外にも同様に LRC を構成する．なお，“n 行目” という表記は図 12 内の行番号を意味する．

STEP1：v_0 自身のボロノイ領域をもとに東西方向の帯を作成する（1 行目）．Lv(0) ノードのリストを，v_0 の隣人ノードの中で東方向にあるノードで，帯とボロノイ領域が交差する $\{v_1, v_2, v_3\}$ とする（2 行目）．

STEP2：Lv(0) であるノード $\{v_1, v_2, v_3\}$ の Lv(0) 内のノードの中から，v_0 の東方向の帯とボロノイ管理領域が重なるノードすべてを取得し，v_0 の Lv(1) とする．これにより，v_0 は Lv(1) 候補ノード $\{v_2, v_5, v_6\}$ を得る（5 行目）．その後，Lv(1) の中のノードから，Lv(i) ノード決定法により，東方向の Lv(i) ノードを決定し，改めて Lv(1) とする（7 行目）（この Lv(i) ノード決定法の詳細については後述する）．

STEP3：同様に i を増加させながら，**STEP2** の処理を繰り返し，Lv(2), Lv(3), . . . を決定する．そして，最終的に Lv(5) を構成しようとするとき，v_0 の Lv(4) 内のノードは，どのノードも東方向の Lv(4) 内にノードがない．この場合に，「getLv() == ”Fail”」の条件式に該当し，v_0 の LRC 構成処理を停止する．

3.2.3 Lv(i) ノード決定

各ノードが取得した Lv(i) 候補ノードの中から，実際に経路を構成する Lv(i) ノードを選択するアルゴリズムを示す．各ノードは Lv($i+1$) 経路を構成するとき，自身の Lv(i) ノードから，Lv($i+1$) の候補となるノードを取得している．しかし，候補ノードすべてと経路を構成してはノードの接続数が大きくなりすぎる．そこで，取得した候補ノードから実際に接続するノードを絞り込み経路を構成する．実際に接続する LRC ノード集合は，

(1) ノード自身からの距離がより遠く，
(2) それらのボロノイ領域で帯を縦断/横断できる，

の条件を満たすように決定する．(1) の条件は，具体的にはノード v が東方向の LRC を構成する場合，$|v_x - v'_x|$ の値がより大きくなる v' を選択するということを意味する．

[Lv(i) ノード決定アルゴリズム]

LRC 経路生成アルゴリズムにおいて用いられる，selectLRCNodes のアルゴリズムを**図 13** に示す．ここでは，アルゴリズムで使用される変数，関数についての説明を述べ，さらに，**図 14** においてノード v_0 が Lv(1) の候補ノードとして $\{v_2, v_5, v_6\}$ の情報を取得した場合を例に説明する．

まず，以下が図 13 において使用される変数，関数である．

prospectiveNodes：ある Lv の LRC 接続を生成する際に用いるためのノードのリストである．このリストから，適切な組合せノードを選択して生成する．

direction：東西南北の方角の 1 つを，$X+$，$X-$，

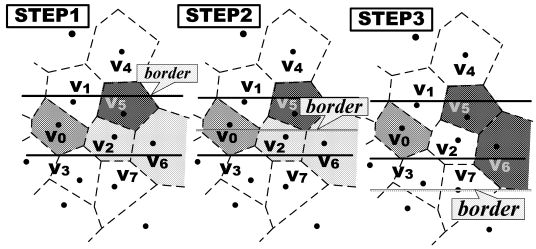

図 14 東方向経路における Lv(1) ノード決定の例
Fig. 14 Example for selection of Lv(1) nodes to the east.

$Y+$，$Y-$ のいずれかとして格納する．

border：ある Lv の LRC 接続先として確定したノードのボロノイ領域の帯による，LRC 接続作成主体のノードの帯の被覆領域の境界線の x 座標，もしくは y 座標の値．

crossedNodes：ある border と交わるボロノイ領域を持つノードのノード情報リスト．

crossedFarthestNode：crossedNodes の中で，最も LRC 接続作成主体のノードから遠いノードのノード情報を格納する．

longRangeNodesList：LRC 接続先として確定したノードのノード情報のリスト．

voronoiBelt：LRC を作成する主体であるノードにおける垂直/平行帯の端となる平行線を示す 2 つの x 座標値か y 座標値の組．

voronoiBelt.getUpper()：voronoiBelt の 2 つの座標値の組のうち，大きい方の値を返す．

voronoiBelt.getBottom()：voronoiBelt の 2 つの座標値の組のうち，小さい方の値を返す．

prospectiveNodes.getCrossNodes(border)：prospectiveNodes のノード情報の中で，border とボロノイ領域が交わるノード情報を prospectiveNodes から削除しつつ，その削除したノード情報のリストを生成したうえで，そのリストを返す．

crossedNodes.getFarthestNode()：crossedNodes のノード情報の中で LRC を作成する主体であるノードの位置座標から，最も遠いノードのノード情報を返す．

crossedFarthestNode.voronoiVertices .getBottom())：実行された時点での voronoiBelt と同じ方向の帯を crossedFarthestNode について求め，その帯の端となる平行線を示す 2 つの x 座標値，もしくは，2 つの y 座標値の組の中で小さい方を返す．

変数，関数については以上である．

次に，以下に図 13 を図 14 における v_0 が Lv(1) の候補ノードとして $\{v_2, v_5, v_6\}$ の情報を取得した場合を例として解説する．なお，“STEP” は図 14 に対応し，“n 行目” は図 13 の行数と対応する

STEP1：まず，v_0 は *Vor(v_0)* の北端を通る東西への水平線を求め，border とする（1 行目）．次に prospectiveNodes の中から，border とボロノイ領域が交差するノードをすべて求め，crossedNodes とする（5 行目）．さらに，crossedNodes の中から，最も自身から遠いノードを求め，crossedFarthestNode を取り出す（6 行目）．その後，crossedFarthestNode を最終的な LRC 接続先のリストである LongRangeNodeList に追加し，prospectiveNodes の中から削除する（7 行目）．以上の手順を v_0 の Lv(1) の候補に適用した場合，$\{v_5\}$ が LongRangeNodeList に追加される．また，*Vor(v_5)* の南端を通る東西への水平線を求め，border とする（8 行目）．

STEP2：5–8 行目の処理を繰り返す．v_0 の例では，残りの候補ノード $\{v_2, v_6\}$ の中から，*Vor(v_5)* の南端を通る基準線である border とボロノイ領域が交差するノードをすべて求め，crossedNodes とする（$\{v_2, v_6\}$）．次に，crossedFarthestNode として，v_6 を選び，LongRangeNodeList に追加する．そして，*Vor(v_6)* の南端を通る東西への基準線を border とする（8 行目）．

STEP3：5–8 行目の処理を繰り返すに従って，border は次第により南に下りてゆき，いずれ v_0 の帯の外側の位置となる．このとき，3–4 行目の LOOP の終了条件が満たされるため，10 行目の LongRangeNodeList 内の結果を返す処理が行われる．v_0 の例では，LongRangeNodesList に $\{v_5, v_6\}$ の 2 ノードが入っており，それが結果として返される．

3.2.4 LRC の生成例

図 11 を用いて，v_0 が東方向に LRC を生成する過程を示す．

まず，v_0 を含むすべてのノードが自律的にボロノイ領域を生成する．v_0 は *Vor(v_0)* のボロノイ点の中から，y 座標の最大/最小値を用いて水平帯を生成する．ドロネー隣人による v_1，v_2，v_3，v_4 を Lv(0) 候補ノードとして，Lv(0) ノード決定を行う．ここでは v_4 が除かれ v_1，v_2，v_3 のみが Lv(0) ノードとなる．

次に，v_0 は Lv(0) ノードすべてに Lv(1) 候補ノードを要求する．要求を受けた Lv(0) ノード $\{v_1, v_2, v_3\}$ は，それぞれの Lv(0) ノードから v_0 の水平帯にボロノイ領域が交差するノードを結果として v_0 に返答する．そして，v_0 は Lv(0) ノード群から Lv(1) の候補ノード $\{v_2, v_5, v_6\}$ を取得する．Lv(0) と同様に候補ノードから Lv(1) ノード $\{v_5, v_6\}$ を選択し，Lv(1) 経路を構成する Lv(1) ノード群を決定する．これら，

Lv(i) ノード情報要求，ノード選択，経路構成を繰り返し，同様に Lv(2) 経路，Lv(3) 経路，Lv(4) 経路と遠隔接続経路を構成する．v_0 は Lv(4) ノード $\{v_{18}, v_{19}\}$ においても情報を要求するが，v_{18}，v_{19} がそれぞれ Lv(4) 経路を持たないため，ノード情報を返さない．これを受けて v_0 は Lv(5) 以降の経路を構成できないと判断し，LRC 生成処理を終了する．

4. LRC の利用

本章では LRC を利用した平面上の任意の地点への経路選択と範囲問合せについて述べる．

4.1 LRC を用いた経路選択

鉛直，水平の 2 方向に生成した LRC を利用し経路選択を行う．この経路選択法における規則を次のように定める．

(1) クエリを水平方向，鉛直方向に転送する．
(2) クエリの転送は，送信元ノードを通る x 軸に平行な直線と折り返しノードを通る軸に平行な直線が交差する点を方向転換地点として定め，その地点を含むボロノイ領域を管理するノードにクエリの送信方向を変更する．
(3) LRC を構成しない（ドロネー図の凸包上の）ノードがクエリを受信すると，LRC を生成しているノードへいったん転送する．

本稿ではアルゴリズムの説明の簡単のため，方向転換は基本的に 2 段階に定めている．ただし，ノード分布や問合せ地点の設定によっては，3 段階以上に方向転換して対応する．これについては後で述べる．このように，ノードの協調によって LRC が全平面に生成されるため，2 地点を結ぶ複数の経路が存在している．

[クエリの構造]

LRC を用いた経路選択において，ノードが生成するクエリは以下に示す 4 つの情報を持つ．クエリが経由するノードはこれらの情報を書き換えて経路選択を行う．

- **目的地**：平面上に存在する目的地の座標値．
- **方向**：クエリを転送する方向．{ 東，西，南，北 } をその値とする．
- **方向転換の地点**：鉛直方向から水平方向（または水平方向から鉛直方向）へと方向転換する地点の座標値．目的地点 (d_x, d_y)，出発ノードの位置 (s_x, s_y) のとき，$|d_x - s_x| > |d_y - s_y|$ の場合には (d_x, s_y) とし，$|d_x - s_x| < |d_y - s_y|$ の場合には (s_x, d_y) とする．
- **帯幅**：出発ノードもしくは方向転換地点を管理するノードによるボロノイ領域の X/Y 座標の最大/最小値．

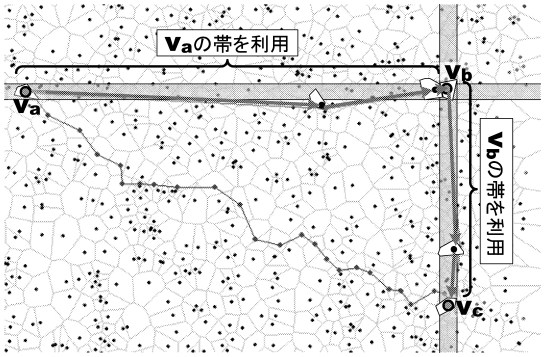

図 15 v_a から v_c の経路選択法の比較
Fig. 15 Routes v_a to v_c by LRC (heavy line) and the greedy-algorithm (thin line).

[経路選択アルゴリズム]

以下のアルゴリズムにより，クエリは出発ノードから目的地点を管理するノードまで到達する．

STEP1：出発ノードによりクエリの生成を行う．クエリは上記に示した目的地，方向転換地点，方向，帯幅の 4 つの情報を含む．

STEP2：ノード v は以下の処理のうちの 1 つを実行し，クエリを転送する．

⇒[CASE 1]：*Vor(v)* が目的地点を含む場合．v は経路選択を終了する．

⇒[CASE 2]：v がドロネーネットワークの凸包上に存在するため，LRC 経路を生成しない場合．v は LRC 経路を構成している隣人ノードにクエリを転送する．クエリを受けた隣人ノードは自身を出発ノードとして帯幅，方向転換の地点，向きの 3 つの情報を書き換える．これは LRC 経路を積極的に利用して経路選択する．この場合に，例外的にクエリの転送方向が 3 段階以上に転換する．

⇒[CASE 3]：ボロノイ領域が方向転換地点を含む場合．方向転換が行われるため，v はクエリが情報として持つ帯幅と方向を変更し，経路探索を行う．

⇒[CASE 4]：上記以外の場合．v は出発ノードもしくは方向転換地点を管理するノードの帯上にボロノイ領域が存在する，目的地点に最も近く目的地点を越えない Lv(i) ノードにクエリを転送する．クエリを通過する各ノードがこれらの処理を行うことで，クエリを転送する．

[経路選択の例]

図 15 で，北西ノード v_a が目的地点 v_c にクエリを転送する例を示す．図中太線は LRC による経路選択を，細線は欲張り法による経路選択を示す．

- 出発ノード v_a はまずクエリを生成し，クエリの持つ 4 つの情報に目的地点を v_c の座標，方向に

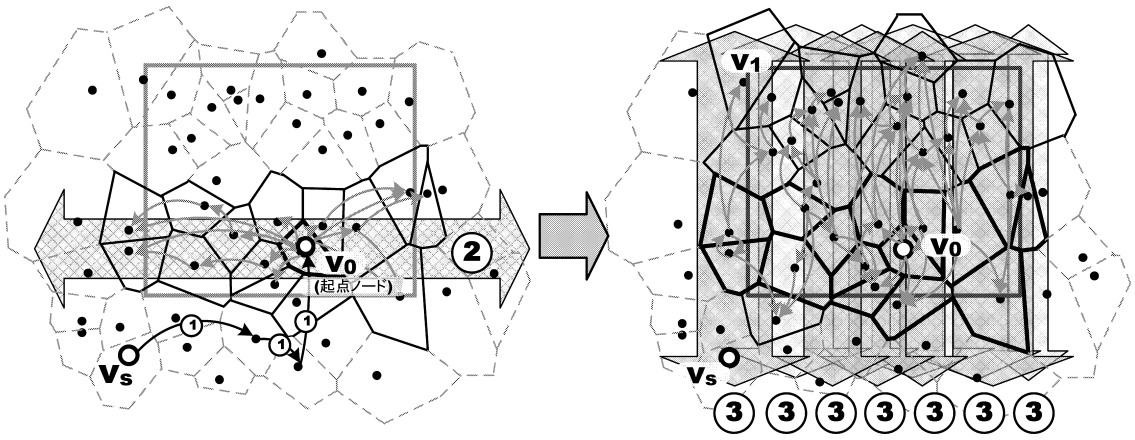

図 16 LRC を用いた再帰的な転送部分
Fig. 16 Recursive query transfer with LRC.

“東”，方向転換の地点に v_c の x 座標および v_a の y 座標，帯幅に $Vor(v_a)$ の y 座標の最大/最小値を設定する．出発ノード v_a からクエリを v_a の水平帯内にボロノイ領域が含まれるノードに転送し，方向転換地点をそのボロノイ領域に含む v_b まで転送する．

- 次に，v_b はクエリの持つ情報を，方向に “南”，帯に $Vor(v_b)$ の x 座標の最大/最小値を設定して v_c までクエリを転送する．
- v_c はクエリの目的地点をボロノイ領域に含むため，経路選択を終了する．

図 15 で比較すると，目的地まで欲張り法を用いた場合では 24 ホップ要するが，LRC を用いた場合では 5 ホップであり，ホップ数が低減していることが確認できる．

4.2 LRC を用いた範囲問合せ

本節では，LRC を利用する検索範囲を指定した問合せについて述べる．範囲問合せでは，問合せ範囲とボロノイ領域（管理領域）が交差するノードすべてに対してクエリを転送する．これらのノードを**問合せ対象ノード**と呼ぶ．範囲問合せ処理に LRC を用いることで，広大な範囲の大量の問合せ対象ノードに対しても高速にクエリを転送することができる．そのため，問合せ範囲をスケーラブルに設定可能である．なお，本稿では範囲問合せの対象範囲は矩形領域とする．

[範囲問合せ法の概要]

図 16 は範囲問合せが行われる様子を示す．ノードから出る矢印は，そのノードがクエリを転送することを意味する（ただし，図では簡単のため，すべての矢印を描いていない）．これを用いて説明する．

まず，問合せを行うノード v_s は，前節の LRC 経路選択法により，1 つのクエリを問合せ対象ノードに到達させる．このノードは任意でよいため，ここでは，問合せ範囲の中心を目的地に設定しクエリを転送するものとする．このクエリ転送処理が図 16 中 ① の矢印である．このクエリを受信した問合せ対象ノードが起点となり，他の問合せ対象ノードへのクエリ転送を開始する．この起点となるノードを**起点ノード**と呼ぶ．図 16 では v_0 が起点ノードとなる．

起点ノードから他の問合せ対象ノードへのクエリ転送は，次のように行う．

- 起点ノードは，クエリ内の帯情報を自身の水平帯に設定する．その水平帯とボロノイ領域が交差するすべての問合せ対象ノードに対して，クエリを水平方向（東方向と西方向）に LRC を用いて転送する．これらの水平方向からのクエリを受信したノードも同様の方向に，設定された帯に従ってクエリを再転送する．このクエリ転送処理を図 16 中 ② として示した．
- また，水平方向からクエリを受信したノードは，自身の鉛直帯を設定したクエリを生成する．そして，その帯とボロノイ領域が交差するすべての問合せ対象ノードに，鉛直方向（北方向と南方向）にこのクエリを転送する．鉛直方向からクエリを受信したノードも同様の方向に，設定された帯に従ってクエリを再転送する．このクエリ転送処理は図 16 中 ③ である．

以上のように，①–③ の 3 段階の処理を行うことで，すべての問合せ対象ノードにクエリを転送することができる．

また，図 16 から，②，③ において，重複してクエリを受信するノードが存在することが分かる．重複し

```
Node.doRangeQueries( query )
 1: if( Node.isOriginNodeOf( query ) )
 2:     query.setVoronoiBelt( Node.getVoronoiBelt( HORIZONTAL ) )
 3:     q1 = query.clone(), q2 = query.clone()
 4:     q1.setDirection( WEST ), q2.setDirection( EAST )
        /*クエリを転送する. 但し, getQueryTargetLRCNodes( q1.Range, q1.Direction )
          の戻り値が存在しない場合は転送しない. 以下も同様. */
 5:     SEND q1 TO Node.getQueryTargetLRCNodes( q1.Range, q1.Direction ),
        SEND q2 TO Node.getQueryTargetLRCNodes( q2.Range, q2.Direction )
 6:     query.setVoronoiBelt( Node.getVoronoiBelt( VERTICAL ) )
 7:     q1 = query.clone(), q2 = query.clone()
 8:     q1.setDirection( NORTH ), q2.setDirection( SOUTH )
 9:     SEND q1 TO Node.getQueryTargetLRCNodes( q1.Range, q1.Direction ),
        SEND q2 TO Node.getQueryTargetLRCNodes( q2.Range, q2.Direction )
10: else if( Node.hasBeenReceived( query.ID ) )
11:     DISCARD query
12: else if( query.Direction == WEST or EAST )
13:     SEND query
            TO Node.getQueryTargetLRCNodes( query.Range, query.Direction )
14:     query.setVoronoiBelt( Node.getVoronoiBelt( VERTICAL ) )
15:     q1 = query.clone(), q2 = query.clone()
16:     q1.setDirection( NORTH ), q2.setDirection( SOUTH )
17:     SEND q1 TO Node.getQueryTargetLRCNodes( q1.Range, q1.Direction ),
        SEND q2 TO Node.getQueryTargetLRCNodes( q2.Range, q2.Direction )
18: else if( query.Direction == NORTH or SOUTH )
19:     SEND query
            TO Node.getQueryTargetLRCNodes( query.Range, query.Direction )
```

図 17 LRC を用いた範囲問合せアルゴリズム：各ノードのクエリ処理手順
Fig. 17 Algorithm for Range query with LRC: Query processing for each nodes.

てクエリを受信したノードは 2 度目の受信以降はそのクエリを破棄し，それ以上転送しない．

Note：ノードが重複してクエリを受信するのには，2 つの理由がある．1 つは，②，③ の帯に従ったクエリ転送では，各ノードは転送方向と同方向にある LRC ノードすべてに対してクエリを転送するためである．もう 1 つは，③ の鉛直帯が重なり合うためである．帯が多く重なる領域と，ボロノイ領域が交差するノードは多くのクエリを受信することになる．

[クエリの構造]

クエリは，起点ノードまで転送するための前節で述べた経路選択情報に加えて，次の情報を含む．

1. ID：このクエリの固有の識別子を表す．

2. 問合せ範囲：問合せ範囲の全体を表す．**図 17** では，Range が問合せ範囲を表しており，問合せ範囲の矩形領域の x 座標の最大値，最小値，y 座標の最大値，最小値を格納する．

3. 帯幅：このクエリによって問い合わせる 2. の部分領域を表す．

4. 転送方向：このクエリを転送する方向（東西南北のいずれか）を表す．図 17 では，Direction が転送方向を表しており，$X+$，$X-$，$Y+$，$Y-$ のいずれかを格納する．

[範囲問合せアルゴリズム]

図 17 に，範囲問合せ処理において，クエリを受信したノードが実行するアルゴリズムを示す．ここでは，アルゴリズムで使用される変数，関数について述べ，さらに，アルゴリズムの各部分について解説する．

まず，以下が図 17 において使用される変数，関数である．

query：前述の［クエリの構造］に基づくクエリ．なお，図 17 において，query の set が頭につくメソッドは，メソッド名に該当する情報を引数から設定する．また，clone メソッドは新規にメモリを確保したうえで query を複製する．

Node.getVoronoiBelt()：引数の指定に基づいて，doRangeQueries 実行中のノードの水平，垂直いずれかの帯情報を返す．

Node.getQueryTargetLRCNodes(query.Range, query.Direction)：ノードの接続先となっているノード情報のリストの中から，query.Range の示す領域に含まれ，かつ，query.Direction の示す方向のノードの帯領域に含まれるすべてのノード情報のリストを返す．

Node.hasBeenReceived(query.ID)：query.ID をチェックし，過去に同じ ID の query をチェックしたことがあれば，true を，なければ，false を返す．また，false を返す場合，返す前に，過去にチェックした ID の履歴のリストに，この ID を追加する．

変数，関数については以上である．

次に，アルゴリズムの各部分について解説する．問合せを行うノードは，前述のクエリを生成し，これを問合せ範囲の中心点に向けて LRC により転送する．これにより，最初にクエリを受信した問合せ対象ノードが起点ノードとなる．起点ノードはクエリを図 17 の 1–9 行目の手順で処理する．

起点ノードの処理：クエリを受け取ったノードは，起点ノードであるかどうかを，クエリの内容に基づき判定し，起点ノードであった場合，2 行目からの処理を行う（1 行目）．まず起点ノードは水平方向にクエリを転送する（2–5 行目）．

(1) クエリに自身の水平帯を設定し（2 行目），
(2) このクエリを 2 つにコピーし，それぞれの転送方向を西，東に設定する（3–4 行目）．
(3) 西方向を設定した q1 は，**起点ノードの西方向の LRC ノードで，かつ問合せ対象ノードであるすべてのノード**に q1 を転送する．同様に東方向を設定した q2 を東方向に転送する（5 行目）．

その後，鉛直方向にクエリを転送する（6–9 行目）．クエリに自身の鉛直帯を設定し，水平方向と同様の手順で南北に転送する．水平方向が鉛直方向に，東西方向が南北方向に変更される以外は 2–5 行目と同様の処

理である．

なお，(3) において，転送先となるノードが存在しない場合，これ以上クエリを転送しない．たとえば，図 16 では，ノード v_1 が北方向にクエリを転送しようとしている．しかし，v_1 の北方向 LRC ノード内に問合せ対象ノードは存在しないため，これ以上クエリを転送しない．これは以降の転送処理においても同様である．

次に，起点ノード以外の問合せ対象ノードがクエリを受信した場合の処理について述べる．

クエリの破棄：過去に受信したクエリと同一の ID を持つクエリを受信したノードは，受信したクエリを破棄する（10–11 行目）．

水平方向から受信したクエリの処理：水平方向に向けて転送されてきたクエリを受信したノードは，12–17 行目の手順でクエリを処理する．

(1) まず，受信したクエリの帯幅と転送方向に従って転送する（13 行目）．
(2) 次に，クエリの帯幅を自身の鉛直帯で再設定し，北方向と南方向に転送する（14–17 行目）．

鉛直方向から受信したクエリの処理：鉛直方向に向けて転送されてきたクエリを受信したノードは，18，19 行目に示す手順で，クエリに設定された帯幅，転送方向に従って転送する．

以上により起点ノードから東西南北方向に，起点ノードからの東西方向のクエリを受信したノードは，南北方向に，南北方向のクエリを受信したノードは，さらに南北方向に転送し，クエリの示す範囲内を管理領域として持つすべてのノードにクエリが行き渡る．

転送の終了：クエリの転送は，起点ノードから，問合せ範囲の外側方向へと転送されるため，いつか問合せ範囲の外側に到達し転送が止まる．

[範囲問合せの例]

図 18 は範囲問合せを実行した例である．これを用いて，クエリを受信したときのノードの振舞いを説明する．v_0 が起点ノードとして，クエリを受信したとする．

起点ノード v_0 はクエリの帯幅に自身の水平帯を設定し，東西の LRC ノードでかつ問合せ対象ノードであるノードに，LRC を用いてクエリを転送する．さらに，v_0 はクエリ帯幅に自身の鉛直帯を再設定し，南北方向にも転送する．

v_0 から水平方向（東方向）に向けたクエリを受信した v_1 は，まず v_0 の水平帯と交差する東方向の LRC ノードでかつ問合せ対象ノードであるノードにクエリを転送する．さらに，クエリの帯幅を v_1 の鉛直帯で再設定し，鉛直方向（南北方向）にもクエリを転送する．

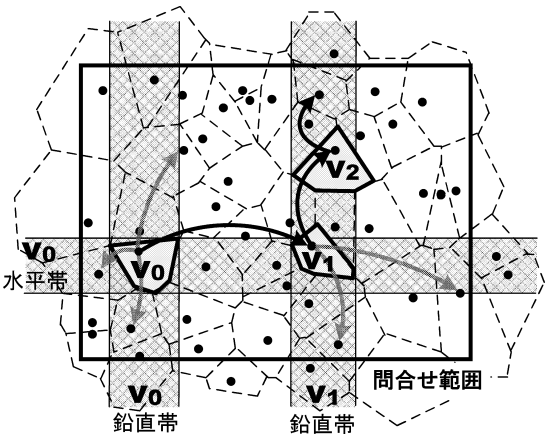

図 18 LRC を利用した範囲問合せの例
Fig. 18 Example for range query utilizing LRC.

鉛直方向（北方向）に向けたクエリを受信した v_2 は，北方向 LRC ノードで，かつ問合せ対象ノードであるノードにクエリを転送する．

クエリを受信したノードすべてがこの要領で動作し，範囲問合せを行う．

[問合せ結果の集約]

問合せ範囲内の各ノードへクエリが到達した後，各ノードから問合せ結果をクエリ発信元に送信する必要がある．このとき，クエリ発信元のノードに各ノードがじかに問合せ結果の送信を行うと発信元のノードは大量の問合せ結果を受信し，負荷が大きくなる．そこで，クエリ発信元の負荷を他ノードに分散させる問合せ結果の集約方法を提案する．

基本的な考え方を以下に述べる．まず，範囲問合せの際のクエリ転送時の経由ノードのネットワーク構造が，クエリ発信元ノードをルートとし，転送元ノードを上位ノード，転送先ノードを下位ノードする木構造となっていることに注目する．そこで，木構造の分岐点となる各ノードは，そのノードより下位に属するノードからの問合せ結果をとりまとめてから，より上位に相当するノードへ転送する動作を行うものとする．これを繰り返すことにより，各ノードからの問合せ結果は結合されて集約されながら，クエリ発信元ノードに到達することとなる．

具体的には各ノードは，問合せ結果の集約のために，追加で以下の動作を行う．

- 他のノードからクエリを受信した場合，そのクエリの受信が 2 度目でなければ，転送元ノード ID を上位ノードとして記録し転送を行う．転送を行う際には，転送先ノードの ID をすべて下位ノー

ドとして記憶する．

- 他のノードから転送されたクエリが 2 度目以降のものであった場合，その旨を即座に転送元のノードに通知する．この通知を終端通知と呼ぶ．
- 記録した下位ノードすべてからの終端通知，問合せ結果を受信した場合，各ノードは，下位ノードからの問合せ結果と自ノードにおける問合せ結果を結合，集約したデータを作成し，自身の上位ノードに自ノードおよび，その下位のノードの問合せ結果として送信する．

以上の動作をクエリを受け取った各ノードが行えば，クエリ転送の経由ノードによる木構造のネットワークにおいて最も末端に位置するノードは，必ず自身がクエリを転送した他のノードすべてから終端通知を受け取る．ゆえに末端のノードから徐々に上位へのノードに対して問合せ結果が送信され，最終的にクエリの発信元ノードまで検索結果が集約されたデータが到達する．各ノードが以上の手法を同期的に行った場合，クエリ発信元ノードからのクエリを問合せ範囲内のすべてのノードに転送する際とは逆方向に末端ノードからの通信が発生する．このため，問合せ結果の集約の際に各ノードが行う通信回数は，クエリを問合せ範囲に送信する際の通信回数と等しい．

4.3 LRC の構造維持

ノードの参加/離脱によってネットワーク構造が組み換わることで，ノード間のホップ数の関係が変化する．このとき，ホップ数に依存した構造を持つ LRC では有効な経路選択ができなくなる可能性がある．そこで，各ノードが自律的に LRC ノードの再決定を行い，新たなホップ数に対応して LRC の構造を維持する機構が必要である．本節では，LRC の構造維持アルゴリズムについて述べる．ここでは簡略化のため，1 次元空間上のネットワークについて説明する．平面上では LRC 生成アルゴリズムと同様に，帯の構造を組み込むことでそのまま対応できる．

[LRC 構造維持処理の概要]

まず，LRC の構造維持処理を例を用いて示す．

図 19 に，v_5 がネットワークから離脱した場合の，東方向への LRC の構造を維持する例を示す．v_5 の離脱によりノード v_1，v_2，v_3，v_4 の東方向へのホップ数は 1 ホップずつ変更される．このため，v_5 より西のノードは新しいホップ数による LRC の構造に対応して LRC ノードを再決定する．まず v_5 を Lv(0) ノードとしていた v_4 は，v_5 が離脱したため Lv(0) ノードを v_6 に再決定する．v_4 は，東方向の Lv(0) ノードの変更を自身を Lv(0) ノードとしている v_3 に通知する．

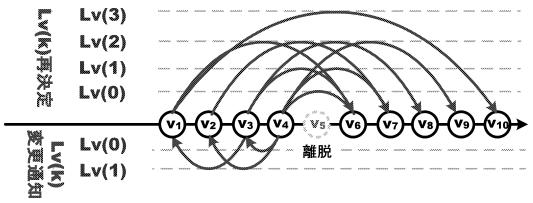

図 19 ノード v_5 離脱時の構造維持処理の例
Fig. 19 Example for maintenance of network when node v_5 leaves.

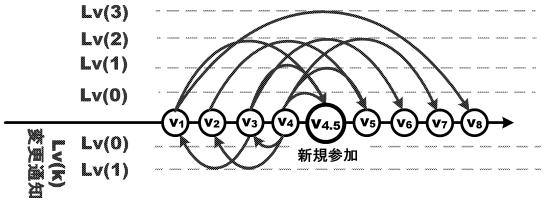

図 20 ノード $v_{4.5}$ 参加時の構造維持処理の例
Fig. 20 Example for maintenance of network when node $v_{4.5}$ joins.

また v_4 は，Lv(0) ノードを変更したため Lv(1) ノードより高い Lv の LRC ノードに対しても東方向への LRC の再決定をし，西方向へは再決定された Lv と同じ LRC の変更の通知を行う．同様に v_3 は v_4 より Lv(0) ノードの変更通知を受け，東方向の Lv(1) ノードを 2 ホップ先の v_6 に再決定し，v_3 自身を Lv(1) ノードとしている v_1 に変更を通知し，さらに高い Lv でも再決定と変更通知を行う．

参加の場合も同様である．**図 20** では，$v_{4.5}$ がネットワークに新たに参加している．この場合，v_4 は Lv(0) ノードを v_5 から $v_{4.5}$ に再決定し，それを v_3 に通知する．ノード v_3 は通知を受けて，Lv(1) ノードを $v_{4.5}$ に再決定し，それを v_1 に通知する．v_1 は通知を受け，Lv(2) ノードを $v_{4.5}$ に再決定する．

このように，各ノードが，(i) 通知を受けると LRC ノードを再決定し，(ii) LRC ノードを再決定するとさらに高い Lv の LRC ノードの再決定を行い，(iii) LRC ノードを再決定すればノードに通知する，という手順を繰り返すことで全体の LRC 構造を維持する．

[LRC 構造維持アルゴリズムの部分処理]

構造維持は各ノードが 2 つの処理を実行することによって行われる．なお，これ以降の Lv の値 k は 0 以上の整数とする．

Lv(k) 変更通知：Lv(k) ノードを変更したことを，構造変化が生じた反対方向の Lv(k) ノードに通知する．

Lv(k) 再決定：構造変化が生じた方向の Lv(k) ノードと協調して Lv(k+1) を再決定する．

また，ノード v_a の Lv(k+1) 再決定は次の場合に

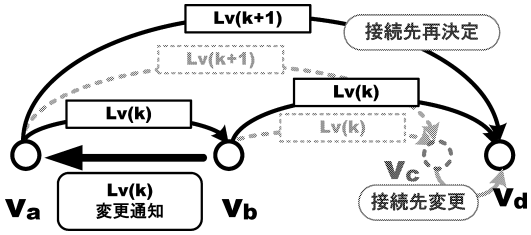

図 21 v_a による Lv(k+1) ノード再決定処理：v_a の Lv(k) ノード v_b から，Lv(k) ノード変更通知を受けた場合
Fig. 21 Reconfiguration for Lv(k+1) node of v_a: in case v_a receives notification to change Lv(k) node from v_b.

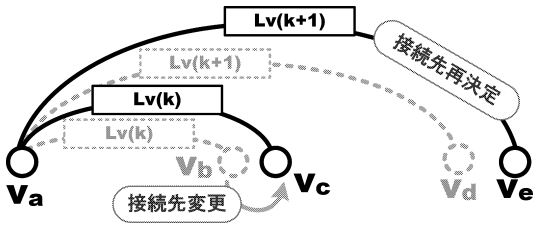

図 22 v_a による Lv(k+1) ノード再決定処理：v_a の Lv(k) ノード v_b の Lv(k) 経路が変更された場合
Fig. 22 Reconfiguration for Lv(k+1) node of v_a: in case v_a reconfigure the Lv(k) node of v_b.

行われる．

(1) v_a の Lv(k) ノード v_b から Lv(k) 変更通知を受ける場合．

(2) v_a の Lv(k) 経路が変更された場合．

(1) について**図 21** を用いて示す．v_b の Lv(k) ノードが v_c から v_d に変更される．v_b を Lv(k) ノードとする v_a でも東方向のホップ数に変更があるため，v_b は v_a に Lv(k) 変更通知を送り，v_a は Lv(k+1) ノードの再決定を行う．

また，(2) を**図 22** を用いて示す．v_a は Lv(k) ノードを v_b から v_c に再決定したため，Lv(k) ノードに問合せを行い決定した Lv(k+1) ノードにも変更が起こる．そのため，v_c に問い合わせることで Lv(k +1) ノードを v_d から v_e に再決定する．

[構造維持アルゴリズム]

LRC 構造を維持するアルゴリズムの流れについて述べ，この手順を**図 23** に示す．

- P2P ドロネーネットワークに参加したノード，もしくは，P2P ドロネーネットワークから離脱しようとするノードは通知元ノードとなり，自身に対して接続するすべてのノードに対して，それぞれの接続 Lv(k) に応じた Lv(k) 変更通知を行う．
- Lv(k) 変更通知を受信したノードは Lv(k) 再決定処理を行う．

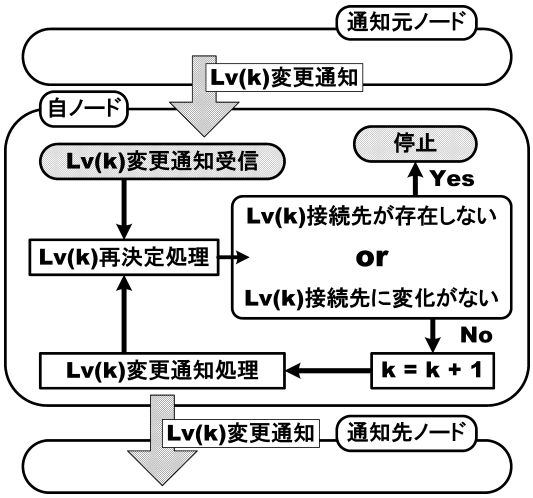

図 23 LRC 構造維持の手順
Fig. 23 Phase of LRC maintenance.

- 再決定したノード群が再決定前と同じもしくは再決定対象のノードが存在しなかった場合，処理を終了する．
- それ以外の場合，k=k+1 とし，Lv(k) 変更通知処理と Lv(k) 再決定処理の双方を行う．

各ノードがこの処理を自律分散的に行うことで，LRC の構造を維持できる．

[維持アルゴリズムの停止性]

変更通知処理と再決定処理による LRC 構造の維持処理は必ず停止する．変更通知処理と 2 回目以降の再決定処理は，必ず k の値を 1 増加させた後に行われる．このため，各ノードが持つ k の値は処理を重ねるごとに増加し続け，より高いホップ数先のノードとの接続についての維持処理となっていく．このため，いつか，P2P ドロネーネットワークの外縁部までのホップ数を超えるホップ数先の LRC の Lv(k) に到達し，そこで LRC 構造の維持処理は停止する．

5. 評　　価

本章では，P2P ドロネーネットワークにおける LRC の生成および維持にかかる負荷と LRC を利用した経路選択および範囲問合せの効果について評価する．ここでは正方領域 $[0,1]\times[0,1]$ にノードを一様分布させることを想定する．また，N はノードの総数を示す．

5.1 LRC 生成にかかる負荷

LRC 生成時のノードの計算負荷およびノード間通信によるネットワークの負荷を，各ノードが Lv(i+1) ノードを問い合わせる回数と，逆に問合せを受ける回数を計測することで評価する．

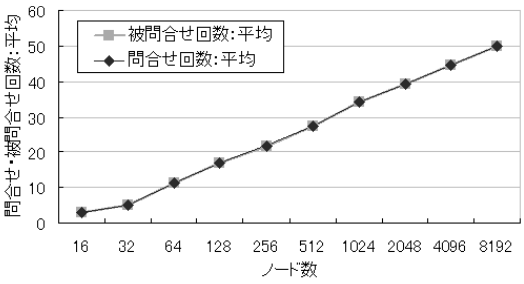

図 24 LRC 生成時のノード数に対する問合せ，被問合せ回数の平均

Fig. 24 Average frequencies of query submission and reception to N.

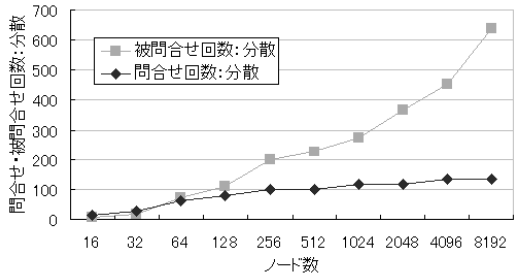

図 25 LRC 生成時のノード数に対する問合せ，被問合せ回数の分散

Fig. 25 Variance frequencies of query submission and reception to N.

まず，総ノード数を指数的に変化させた場合の問合せ回数の平均値と分散値を，**図 24**，**図 25** にそれぞれ示す．図 24 では，問合せと被問合せの平均は等しい．これは問合せを行う先のノードが発生すれば，必ず問合せを受けるノードが存在するためである．また，ノード数の増加に対して，いずれの平均も対数的に増加した．ノード数の増加に対して大幅に通信回数が増加しないことが分かった．

一方，図 25 では，問合せ回数の分散値はノード数の増加に対して小さい値で一定であるが，被問合せ回数の分散は大きくなっている．この理由を確認するため，P2P ドロネーネットワークにおける各ノードの帯幅について調べる．**図 26** は，各ノードの帯幅の平均と分散である．ノード数の増加に対して帯幅の平均値は減少するが，分散はほぼ一定である．つまり，各ノードが平面に一様に分布する場合でも，各ノードの帯幅にはばらつきがあり，他のノードより大きな帯幅を持つノードが存在していることが分かる．

ここで，帯幅が平均より大きなノードが各 Lv の LRC を構成する場合を考えてみると，そのノードの帯幅を帯幅の平均値で割った程度の個数のノードに，Lv(i) 候補ノード情報を問い合わせればよいことが分

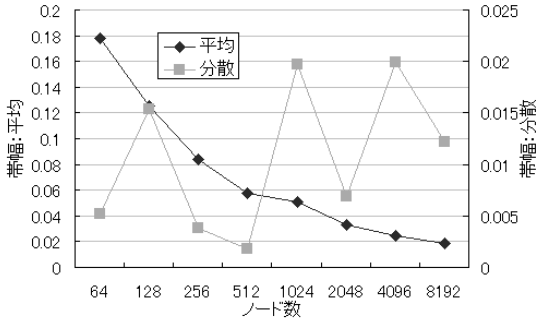

図 26 ノードごとの帯幅の平均と分散

Fig. 26 Average and variance of Voronoi-belt width to N.

かる．一方，帯幅の大きなノードが，他ノードから Lv(i) 候補ノード情報を問い合わせられる回数は，そのノードが問い合わせる回数よりも多くなる．

実際，LRC 構成手順を考えれば，帯幅が大きいノードは，次の特徴を持つ．すなわち，(i) 他ノードから生成アルゴリズム内の Lv(i) ノード決定処理において，LRC ノードに選ばれやすい．また，(ii) LRC ノードに選ばれやすいということは，生成アルゴリズム内の LRC の協調的な生成処理において，Lv(i) 候補ノードとして返されやすくなる．そして，(iii) Lv(i) 候補ノードとして返されやすいということは，さらに LRC ノードとして選ばれやすくなる．これら (i)–(iii) を考え合わせると，帯幅の大きなノードは，多くのノードから LRC ノードとして選ばれ，多くの問合せを受けることになる．また，帯幅が小さいノードの場合は，その逆である．以上のことから，問合せ回数と被問合せ回数の分散に差が生じているものと考えられる．

5.2 LRC の保持にかかる負荷

LRC を保持することによる各ノードの負荷として，LRC 完成後のノードの次数について評価する．

図 27，**図 28** にノードが LRC を保持する場合と保持しない場合について，それぞれ次数の平均と最大値および分散を示した．LRC を持たない従来の P2P ドロネーネットワークでは，次数の平均は 6 以下であり，最大値もノード数の増加に対して一定である．同様に分散も低く，ドロネー図の幾何学的特徴を示している．

一方，LRC を保持する場合では，ノード数の指数的な増加に対して次数の平均と最大値は増加しており，N=8192 の場合で平均が 42，最大値が 75 となっている．これはノード数が増加してネットワークの直径が大きくなるに従い，各ノードがより高い Lv の LRC を構成することになるためである．

また，次数の分散は各ノードの帯の幅に依存する．

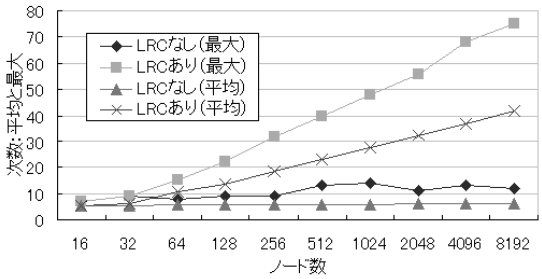

図 27 LRC 生成後のノード数に対する次数の平均と最大値
Fig. 27 Average and maximum of node degree to N.

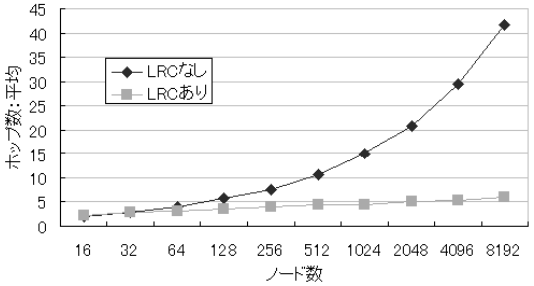

図 29 LRC 生成後のノード数に対するホップ数平均
Fig. 29 Average hop counts to N.

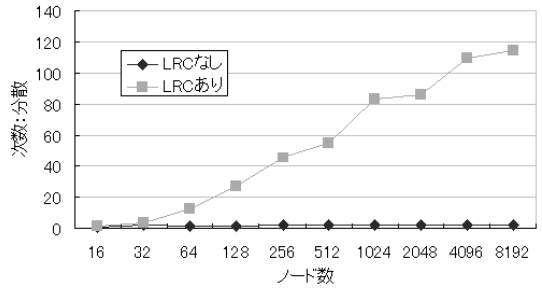

図 28 LRC 生成後のノード数に対する次数分散
Fig. 28 Variance of node degree to N.

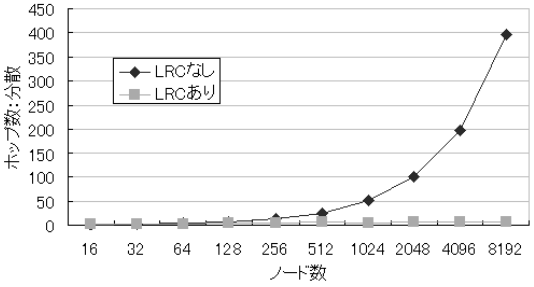

図 30 LRC 生成後のノード数に対するホップ数分散
Fig. 30 Variance of hop counts to N.

各ノードは LRC ノードを，LRC ノードのボロノイ領域が帯を縦断/横断するように決定する（3.2.3 項）．そのため，帯幅が広いノードほど，帯を縦断/横断するために多くの LRC ノードを必要とする．図 26 について前節で述べたように，$[0,1]\times[0,1]$ 内において，ノード数の増加に対して帯幅の平均は減少しているが，分散は減少していない．そのため LRC を保持する場合，ノード数が増加するに従い次数の分散も増加する．

5.3 経路選択の効果

LRC の性能評価として任意の 2 ノード間のホップ数を計測する．**図 29**，**図 30** にそれぞれノード数を変化させ，LRC を利用する場合と利用しない場合のホップ数の平均と分散を示す．ホップ数の平均，分散は各ノード数のネットワークにおいて，すべての 2 ノードの組合せを取り出して計測した．

LRC を持たないネットワークではホップ数の平均と分散は大きく，平均ホップ数はネットワーク直径の増大に比例して増加しているものと考えられる．また，分散もネットワーク直径の増大にともない増加している．これは 2 ノード間の経路上のノード数の差が大きくなるためであると考えられる．一方，LRC を生成した場合にはノード数の変化に対して平均/分散ともに低く一定である．このことから，LRC により任意の 2 ノード間を $O(\log N)$ のホップ数で到達可能となっていることが確認できる．

5.4 範囲問合せの評価

問合せ範囲の大きさを空間全体の 1/16 と 1/4 の面積として，ネットワークのノード数を変化させ，LRC を用いた範囲問合せの効果を計測する．ここでは LRC を利用せず欲張り法とフラッディングを用いて問い合わせた場合と，4.2 節で述べた手法により問い合わせた場合について比較する．

図 31 に LRC を利用した場合と利用しない場合での範囲問合せに要するクエリの最大ホップ数の平均値を示す．このホップ数とは，範囲問合せの起点ノードからのクエリ転送に要したホップ数である．これにより，範囲問合せ処理の実行に要する時間を評価する．LRC を利用しない場合では，ドロネー辺で接続する隣人ノードのみにクエリを転送するため，ホップ数は問合せ範囲の大きさとノード数の増加に対して増加している．これは，各ノード間の接続が近傍間のみで行われる従来の P2P ドロネーネットワークで，ホップ数がネットワークの直径や問合せ範囲のノード数に依存するためである．そのため，空間が拡張した場合の範囲問合せにおいては有効に機能しない場合がある．

一方，LRC を利用する手法では，問合せ範囲の増加に対して要する最大ホップ数にほぼ変化がなく，ノード数の増加に対しては $O(\log N)$ の割合で増加してい

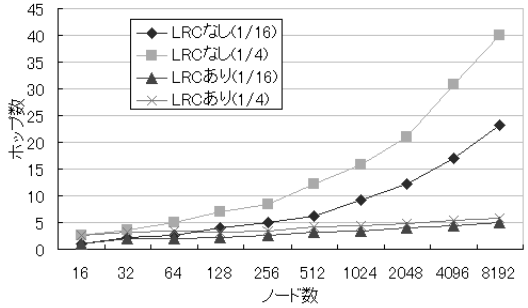

図 31 ノード数に対する 1 回の範囲問合せに要する最大ホップ数の平均
Fig. 31 Average hop counts for range query to N.

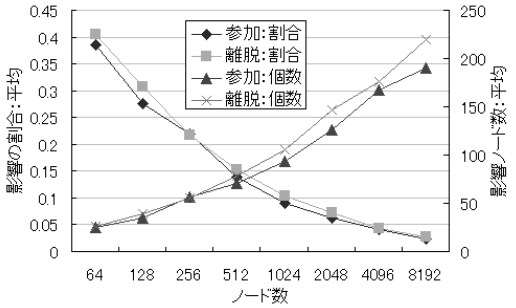

図 33 ノード参加/離脱時の影響ノード数平均
Fig. 33 Average of the number of nodes influenced by join/leave of a node.

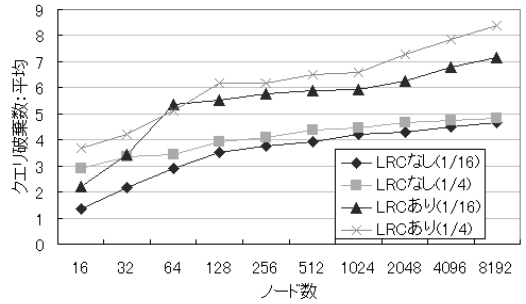

図 32 1 ノードあたりの範囲問合せクエリ破棄数の平均
Fig. 32 Average counts of discarded queries for range query to N.

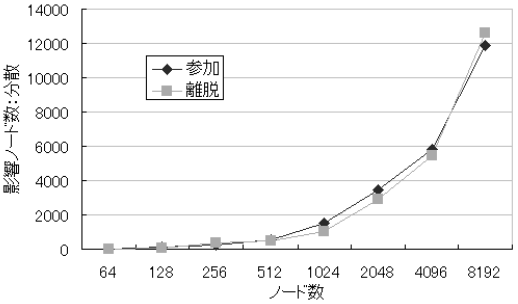

図 34 ノード参加/離脱時の影響ノード数分散
Fig. 34 Variance of the number of nodes influenced by join/leave of a node.

る．LRC を利用することでノード数が増加した場合においても，平面上の任意の大きさの領域に対して範囲問合せを効果的に行うことができることが確認できた．

また，**図 32** に範囲問合せにおける問合せ対象ノード 1 つあたりのクエリの破棄数を計測した．図 32 により，LRC を利用する手法では，利用しない手法と比較して，ノードが破棄するクエリが 2 倍程度に増加することが確認された．これは LRC により次数が増加していることで，各ノードが 1 度にクエリを転送する先が増加するためであると考えられる．

5.5 ノードの参加/離脱時の LRC 維持処理によるネットワーク負荷

ノードの参加と離脱の際の LRC の構造維持に要する負荷について計測した．ノード参加は，LRC 完成後に対象平面上の任意の地点にノードを発生させる．離脱はネットワーク内のノードよりランダムに 1 ノードを選び離脱させる．その参加/離脱の際に，LRC 維持アルゴリズムの通知処理または問合せのいずれかの処理を行ったノードを影響ノードと呼ぶ．ネットワークのノード数を変化させ，参加と離脱について，それぞれ 100 回試行し影響ノード数の平均と分散を計測した（**図 33**，**図 34**）．

影響ノード数の平均はノード数 N に比例して増加している．また，参加/離脱ノードの周辺ノードのボロノイ領域の大きさによって通知先の数に差が生じるため，分散は大きくなっている．これは，ノード分布を正規化しているためノード数の増加につれて各ノードの帯幅が小さくなり，LRC の再構成を要する範囲の割合が小さくなっているためであると考えられる．

6. 災害時の被害状況把握を想定した比較シミュレーション

本章では，実利用を想定したシミュレーションにより，異なる利用傾向から生成された 2 種類の GeoPeer 型ネットワーク[2] と提案ネットワークを比較する．

6.1 災害発生時における被災者支援システムと要件

災害発生時の被害状況を把握するための被災者支援システムを考える．

このシステムでは災害とは，広域地震から地下街での火災などまで，様々な規模のものを扱う．被災者はインターネットに接続でき GPS などにより位置情報を取得可能である小型通信端末を保持し，それらからオーバレイネットワークを構成可能であるものとする．被災者への支援にあたっては，(i) 被災者の位置と状況

把握，(ii) 被害状況の把握，(iii) 位置に則した誘導経路などの指示のための被災者への情報提供が重要である．災害発生時に被災者の持つ端末が提案ネットワークを構成し，支援にあたる者は，範囲問合せにより (i) を，位置を指定して被災者に問い合わせることで (ii) を行う．また，地域ごとに被災者に避難情報を配信することで (iii) を行うことができる．中央集中型のシステムではホストコンピュータが被害を受け停止すれば，システム全体が利用できなくなる．一方，分散型である提案システムでは，被害を免れた通信基地局が存在すれば，その周囲の被災者の情報の把握が可能である．また，P2P 方式では，大規模災害の際の膨大な数の被災者に対応するためのスケーラビリティも確保できる．

このシステムでは，次のような要件が求められる．

要件 1 被災地周辺の任意領域に対して，被災者情報の問合せを行う必要がある．

要件 2 平常時に構成したネットワークでの通信とは異なる種類の通信（被災者情報の問合せによるものである）が突然に増加するため，それに対応できる必要がある．

この 2 点を組み込んだ広域地震のような大規模な災害を想定したシミュレーションにより，提案ネットワークと GeoPeer を比較する．

6.2 シミュレーション設定

本シミュレーションにおいて比較する各ネットワークと，比較する項目について述べる．

6.2.1 提案ネットワークと GeoPeer 型ネットワーク

提案ネットワークでは，位置に基づくノード集合全体への LRC を構成して利用状況に依存せず空間全体に対する短縮経路を構成しているため，利用傾向に適したネットワーク構造は構築されない．しかし，それまでの傾向と違う通信が突然発生しても対応できる構造を保ち続けているものと考えられる．これに対して，従来手法である GeoPeer では，通信発生時に同時に LRC を構成することにより，通信が頻発する部分への経路を短縮する．そのため，GeoPeer では利用の傾向が一定である限り，それに適したネットワーク構造を保ち続ける特徴を持つ．そこで本シミュレーションでは，GeoPeer を異なる 2 種類の通信傾向モデルから生成し，それぞれを比較対象とする．また，以降では，この GeoPeer の LRC を **GeoPeer 型 LRC** と呼ぶ．

6.2.2 GeoPeer 型ネットワークの生成

まず，比較において用いる GeoPeer 型ネットワークの生成法について述べる．

[ノードの配置とドロネーネットワークの生成]

$[0,1] \times [0,1]$ の正方領域にノードを一様に配置し，ドロネー図状の接続関係のネットワークを生成する．

[GeoPeer 型 LRC の生成方法]

利用法から想定される通信傾向に基づき，通信元ノードと通信先ノードを決定し，その間を欲張り法に基づいてクエリを転送する．クエリ内には，ノード間で転送されるたびに，何ホップ目かを記録する．また，GeoPeer 型 LRC の接続先の候補となりうるノードのネットワークアドレスにも記録しておき，2^x（x は自然数）の整数倍のホップ数にあたるノードがクエリを受信した際に，クエリ内の情報から適切なノードを選び直接通信を行い，GeoPeer 型 LRC を生成する．

[LRC 生成本数]

LRC を備えた 3 つのネットワーク（提案ネットワークを含む）の性能比較は，ネットワーク全体の LRC の合計本数を同一にして行う．具体的には提案ネットワークで LRC 付きのネットワークを構築し，ネットワーク全体での LRC 本数を計測する．GeoPeer 型ネットワークは，LRC の合計本数が提案ネットワークと同数以上に生成する．

[GeoPeer 生成時の通信傾向モデル]

GeoPeer は通信傾向に応じて，ネットワークの構造が決定される．そこで今回は，LRC 接続が分散傾向にある GeoPeer と集中傾向にある GeoPeer の 2 種類を生成して用いる．具体的には以下のように生成する．

(1) 通信が集中する利用傾向において生成される GeoPeer は，都市とその周辺部などを想定して作成する．正方領域を 3×3 の 9 つの正方領域の並びにより区切ったうえで，その中央の正方形の領域を通信の集中する都市部と位置づける．各ノードは 9 割の確率で都市部のノードから，1 割の確率で全ノードから，ランダムに 1 ノードを選び出し，そこまでの経路上に GeoPeer 型 LRC を作成する．

(2) 通信が分散する利用傾向において生成される GeoPeer は，各ノードが他ノードの中からランダムに 1 ノードを選び出し，そこまでの経路上に LRC を生成する．

(1) は地震による被害を受けた地域内に，過去に通信が集中する領域（たとえば都市である）が存在していた場合を想定している．この場合，GeoPeer 型 LRC がその領域に集中する．そして，(1) と比較を行うために (2) の条件からも別にネットワークを生成する．以降，(1) を**集中型 GeoPeer**，(2) を**分散**

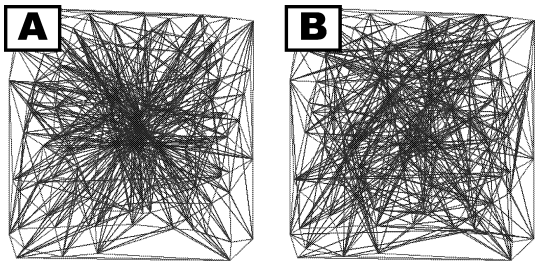

図 35 [A] 集中型 GeoPeer と [B] 分散型 GeoPeer
Fig. 35 [A] Centralized GeoPeer and [B] Decentralized GeoPeer.

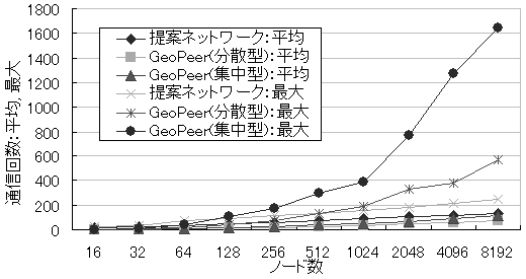

図 36 ネットワーク生成に要する通信回数の平均と最大値
Fig. 36 Average and Maximum numbers of data-transfer frequencies for each network construction.

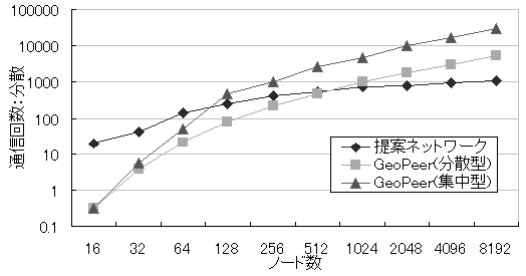

図 37 ネットワーク生成に要する通信回数の分散
Fig. 37 Variance numbers of data-transfer frequencies for each network construction.

型 GeoPeer と呼ぶ（**図 35**）.

6.2.3 各ネットワークの比較項目

シミュレーションでは，分散型 GeoPeer，集中型 GeoPeer，提案ネットワークと，LRC 無しの P2P ドロネーネットワークの 4 つで比較する．各ネットワークについて，(比較項目 1）LRC 生成時の各ノードへの通信負荷，(比較項目 2）LRC 生成直後の各ノードの接続数，(比較項目 3）LRC 生成後に任意の範囲に問合せを行った際の，指定範囲とボロノイ領域が交差するノード（4.2 節において，問合せ対象ノードとした）へのクエリ到達速度について比較する．各項目の比較方法について以下に説明する．

•(比較項目 1）LRC 生成時の通信負荷

LRC 生成時の各ノードへの通信負荷は，LRC 生成過程における各ノードの他ノードへのクエリ転送回数の平均，分散，最大値を計測し比較する．

•(比較項目 2）LRC 生成直後の各ノードの接続数

LRC の生成により各ノードの接続数は増加する．このとき，多くの LRC の接続先とされるノードは，ネットワークの各所からの通信が集中することになる場合があり，計算能力の低い小型通信端末を用いるという利用想定からみて望ましくない．各ネットワークの LRC 生成直後の各ノードの接続数の平均，分散，最大値を計測し比較する．

•(比較項目 3）範囲問合せ処理

範囲問合せ処理について比較評価し，前述の要件 1，2 に対応可能であるか確かめる．本シミュレーションでは，範囲を指定した通信による災害時の状況把握および地域ごとの被災者への避難情報の配信を想定しており，任意の地点から任意の位置，大きさの矩形範囲に問合せを行うことを想定している．そこで，すべてのノードから平面上の全領域を問合せを試行し，それに要するホップ数を計測し，その平均，分散，最大値を計測する．また，重複して転送されたクエリを各ノードが破棄した回数についても計測し，それらをもとに比較する．

6.3 シミュレーションによる比較評価

本節では各ネットワークの性能を，利用想定に基づき比較評価する．

6.3.1 ネットワーク生成時の通信負荷の比較

各 LRC 生成時に要する通信回数を計測した（**図 36**，**図 37**）.

図 36 に LRC 生成時における全ノード数に対する各ノード間の通信回数の平均と最大値を示した．平均については，どのネットワークもノード数の増加に対してほとんど増加していない．一方，最大値をみると，集中型 GeoPeer の通信回数が他のネットワークと比較して，ノード数の増加に対して大きく増加している．これは，まず通信が集中する地域が存在するというシナリオが影響している．さらに GeoPeer ノードは，他ノードとの通信が必要となった場合，LRC を用いて通信しながら，同時にその通信対象ノードとの LRC を構成するための制御用のパケットをドロネーネットワーク上を欲張り法により転送していく．そのため LRC がすでにある程度生成されている地域であっても，パケットが多く通過することになり，通信集中地域の通信回数が増大しているものと考えられる．

図 37 に LRC 生成時における全ノード数に対する各ノード間の通信回数の分散を示した．分散を示す縦

軸は対数軸としている．ノード数が少ないネットワークでは，提案ネットワークの方が分散値は大きいが，ノード数が増加するに従って GeoPeer の分散値が大きくなる．集中型 GeoPeer は通信が集中している地域に存在するノードをパケットが通過することで，その地域のノードの通信回数のみが増大していることによると考えられる．また，分散型 GeoPeer においても高くなっている理由としては，ドロネーネットワーク上の欲張り法による経路選択においては，通信元と通信先がランダムでも，平面の中央を経由するパケットが比較的多くなることが原因であると考えられる．

6.3.2 LRC 生成直後の各ノードの接続数の比較

各ネットワークの完成後のノード次数を計測した（**図 38**，**図 39**）．

図 38 に各ネットワークの LRC 生成後のネットワークにおける全ノード数に対する各ノードの次数の平均と最大値を示した．なお，前述のとおり，LRC の本数は 3 つの LRC 付きのネットワークとも同一にしている．まず，それら LRC 付きのネットワークでの次数の平均値はほぼ等しくなっており，また，ノード数の増加に対してもほとんど増加していない．しかし，最大値は GeoPeer の方が大きくなっており，ノード数の増加に対して大きく増加している．特に集中型 GeoPeer ではその傾向が強く，通信が集中する地域のノードの接続数が大幅に増加する瞬間があることを示している．

また，図 39 には，LRC 生成後のネットワークにおける全ノード数に対する各ノードの次数の分散を示した．これは集中型 GeoPeer が最も大きくなっており，通信の集中する領域とそれ以外の領域との LRC 接続数に差が生じていることが原因であると考えられる．また分散型 GeoPeer においても分散の値は大きくなっている．これは分散的な利用傾向から生成した分散型 GeoPeer においても，平面領域全体の中で中央部分が，GeoPeer 型 LRC 構成の基準とする欲張り法の経路選択で制御パケットが多く通過することで，中央部分と周辺部分の間で接続数に差が生じているためであると考えられる．一方，提案ネットワークでは分散値は低くなっている．これは提案ネットワークが通信傾向にかかわらず平面領域全体に対して LRC を構築するためである．

6.3.3 範囲問合せの比較

各ノードから平面上の領域全体に範囲問合せを実行し，各ネットワークで要するホップ数と重複受信したクエリの破棄数を比較した（**図 40**，**図 41**，**図 42**）．範囲問合せ処理において要するホップ数とは，5.4 節

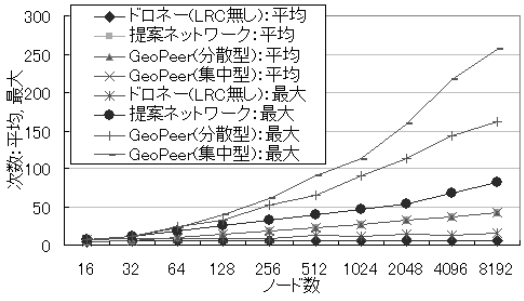

図 38 ネットワーク生成後の次数の平均，最大値
Fig. 38 Average and Maximum number of degrees for each node.

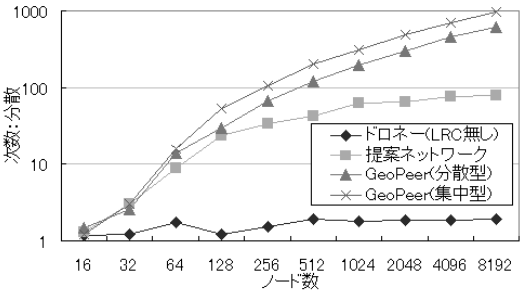

図 39 ネットワーク生成後の次数の分散
Fig. 39 Average and Maximum number of Degrees for each node.

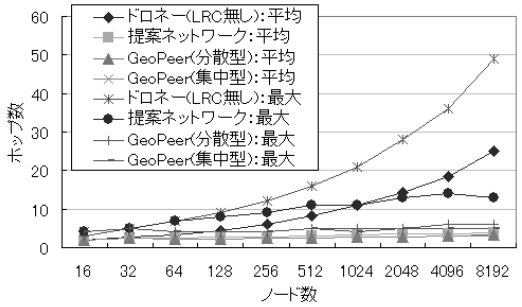

図 40 範囲問合せ処理におけるホップ数の平均と最大
Fig. 40 Average and Maximum hop counts for range query.

での設定と同様に，最も多くホップしたクエリのホップ数である．これにより，範囲問合せに要する時間を評価する．

なお，提案ネットワークでは 4.2 節で述べた手法により範囲問合せを行い，GeoPeer 型ネットワークでは GeoPeer 型 LRC のフラッディングによって範囲問合せを行う．

図 40 に範囲問合せ処理に要するホップ数の平均値と最大値を示した．平均値は，2 種類の GeoPeer と提案ネットワークでほぼ等しいことが分かった．最大値

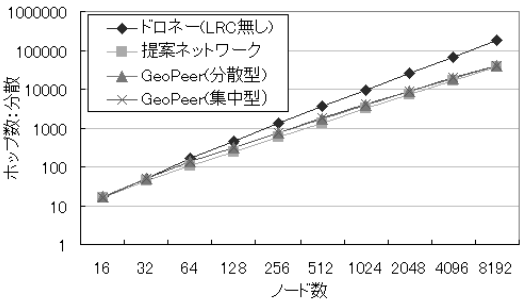

図 41 範囲問合せ処理におけるホップ数の分散
Fig. 41 Variance of hop counts for range query.

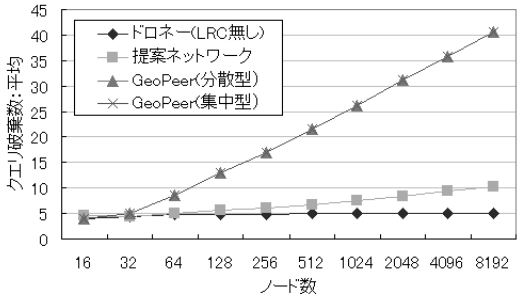

図 42 範囲問合せ処理における問合せ対象ノードあたりの平均クエリ破棄数
Fig. 42 Average number of discarded queries for each node in query-range.

は 2 種類の GeoPeer でほぼ等しい．一方，提案ネットワークでは，それらの 2 倍ほどのホップ数を要していることが分かった．この差は GeoPeer がフラッディングを用いているのに対して，提案ネットワークでは帯に基づいて X 軸方向と Y 軸方向に，転送方向を考慮して転送を行うためであると考えられる．

また，図 41 に範囲問合せ処理に要するホップ数の分散を示した．LRC なしのネットワーク以外の各ネットワークにおけるホップ数の分散はほぼ等しい．これにより，LRC を保持するネットワークでは，範囲問合せを実行するノードの位置に依存せず等しいホップ数でクエリが到達することを確認できた．

図 42 に，範囲問合せ実行時の重複受信したクエリを破棄した回数の平均値を示した．このクエリ破棄数は，5.4 節と同様の方法で計測した．破棄したクエリ（重複して受信したクエリ）の数を計測することで，ノードの通信負荷，処理負荷を比較する．提案ネットワークの平均クエリ破棄数と比較して，GeoPeer の平均クエリ破棄数が大きいことが分かった．提案ネットワークでは各ノードの帯に基づいた LRC の構造を用いてクエリの転送先を計画的に決定できるのに対して，提案ネットワークのような構造を持たない GeoPeer 型 LRC ではそれが難しい．そのため，GeoPeer では範囲問合せを行う場合に，各ノードは問合せ対象ノードである LRC ノードすべてにクエリを転送する．その結果，多くのクエリが転送されることとなり，クエリの破棄数が大きくなっている．

6.4 各ネットワークの特徴

本比較の結果，以下のような知見が得られた．

GeoPeer では GeoPeer 型 LRC 生成用パケットが，LRC 生成済みの地域を多く通過するため，LRC 生成に要する通信回数が増大した．特に平面の中央部分をパケットが多く通過することから，分散の値が大きくなった．また，この通信回数の偏りは，GeoPeer 型 LRC 生成後の各ノードの接続数にも影響を与えており，中央部分の接続数が大きくなる結果をもたらしている．範囲問合せは高速に実行可能であるが，ネットワーク全体の構造化が行われていないことから，クエリ転送にフラッディングを使わざるをえず，通信負荷，処理負荷が大きいことが確認できた．

提案手法では，LRC を空間全体に均一に生成する．各ノードの LRC 生成のための通信が特定の領域に集中しないことから，分散を低く抑えられた．また，GeoPeer と異なり，各ノードが他ノードの生成済み LRC をもとに新たな経路を構成するという協調的な LRC 生成法を用いているため，全体として通信回数を少なく抑えられた．またネットワーク全体に分散した構造の LRC を生成するという特徴から，各ノードの LRC 接続数も分散している．範囲問合せにおいても，帯とホップ数に基づいた構造を利用して問合せ範囲に計画的にクエリを転送することができ，GeoPeer に比べてクエリ破棄数を減少させることができた．しかし，範囲内のすべてのノードへ完全にクエリが到達するのに要する時間は，一般的なフラッディングを使用した GeoPeer と比較すると多くかかることが確認できた．

7. 関 連 研 究

P2P 方式を用いて平面上のノードの位置を陽に取り扱う場合や，対象が 2 次元平面上の地点や領域である地理データを扱う場合などでは，効率的な通信や検索のためにノード間およびデータ間の相互の関連性を持たせる工夫が必要である．

[平面上のノード集合に対する短縮経路]

Araújo ら[2),8)] は位置情報を用いたアプリケーションへの適用を目指し，ドロネー図を利用した 2 つの LRC の構成法を示している．Hop Level の方式では通信を行う 2 ノードが既知である場合を想定し，2 ノード間

の経路を欲張り法によって決定している．欲張り法により決定した2ノード間の経路上に存在するノードがホップ数に基づきLRCを構成することで，経路上のノード間を$O(logN)$のホップ数で到達できる．しかし，Hop LevelによるLRCでは平面上の任意の領域に対して問合せを行うことを想定していない．各ノードは必要に応じて指定したノードに対してLRCを構成している．また，Hop Levelに基づくLRCを空間全体に構成する場合，欲張り法によって構成する経路が多数存在し，各ノードは多数の経路上に存在する可能性が高くなり，次数が増加する．eCAN-Likeの方式では空間全体を固定された部分領域に分割し，部分領域内のノード間が通信を行うことでLRCを構成している．また，各ノードの次数は格子の分割数に依存する．

これに対して提案手法では，各ノードが自身のボロノイ領域による帯を用いて，LRCノードの数を自律的に制限している．このため，欲張り法によってあらかじめノード間の経路が既知である必要はなく，各ノード間が協調し，LRCを生成する．そのため，各ノードはこのLRCを利用することのみで任意の2ノード間のホップ数を$O(logN)$に短縮している．さらに，提案手法ではノードが存在する空間全体に対してLRCを生成する．スケーラブルな空間に対しても平面上の任意の領域に対する範囲問合せを行うことができる．また，各ノードが協調して分散的にLRCを構成しているため，ネットワーク構造としてのスケーラビリティだけでなく，構成法としてのスケーラビリティも保持している．

Naorら[9]は，連続領域を動的に分割してDHT上のノードにリンクを構成する手法を提案している．ノードへの配置データを離散データから連続データへ拡張するため，1次元並びに2次元ユークリッド領域上の各点からoutリンクを生成する距離2分関数を定義し，well-connectedなLRC型のデータ間グラフ（expanderと呼ぶ）の構成法を与えている．特に平面では，データ点集合で一意に定まるボロノイ/ドロネー図を利用してoutリンクを生成し，ノードへ配置する数学的方法を示唆している．Naorらが固定したノードにデータを配置するためのLRCを構成するのに対して，本提案手法はノードの位置を基にしたネットワークトポロジをオーバレイに構成している．また，Naorらのリンク生成のための関数をそのまま我々の手法に適用することも数学的には可能であるが，膨大なLRCリンク数によりノードの経路表が巨大になり実用的でない．また，2次元平面に対してボロノイ/ドロネー図を用いたリンク構造を示唆しているが，実際，ランダムに配置された2次元データに対して実システムとして構築できるかは不明である．本提案手法は，ノード集合の位置により一意に定まるボロノイ領域とドロネー図を用いることでノードの位置する連続領域をボロノイ領域の集合と見なし，ボロノイ領域間のリンクとして遠隔接続経路をノード間に構成している．これにより，リンク数を少なく保つことができる．さらに，各ノードのボロノイ領域の幅に対して水平/鉛直走査を行うためLRCリンクの数が圧倒的に小さく，結果的に経路表のサイズを小さく抑えている．

金子ら[3]は物理的なノード位置を考慮した空間情報管理のためのP2PネットワークとしてLL-Netを提案している．LL-Netでは空間を格子状の部分領域に再帰的に分割し，それらにノードを配置する．さらに，部分領域間を接続する経路を階層的に構成することで遠隔の通信にも対応する．しかし，格子区間の分割は静的に行われるため，特定の領域に負荷が集中したときに対応できない場合があることを示唆している．対して提案手法では，部分領域の階層化を行わず，同一層でLRCを構築している．これは，提案ネットワークを用いて，温度センサ機能を持つノード，高速計算や広い通信帯域を持つノードなどの，異なる役割を持つノードをそれぞれグループ化して部分ネットワークを構成し，部分ネットワーク間の関係を動的に柔軟に決定できる仕組みを実現したいためである．その仕組みに基づいて，高い処理能力や広い通信帯域を持つノードからなる部分ネットワークを構成し，他の部分ネットワーク全体のハブとした階層構造を導入することも考えている．このため，本稿では各部分ネットワーク内で効率良く通信可能にするために，LRCを構成することを提案している．

Xuら[13]は，オーバレイの経路選択効率を上げる補助的データ構造として，d次元空間（d-トーラス）を部分領域に分割し，複数の異なる長さの経路長（ホップ数）を持つexpresswayの自律分散構成法を提案している．expresswayは物理的なノード位置に依存せず，オーバレイとして論理的に自己組織化構成することができるが，部分領域の代表ノードが通信ノードとなっているためホットスポットができやすく，ノードが新規に追加されたとき接続を再構成するのに負荷がかかる．

[グラフと短縮経路]

P2Pネットワークのノードの隣人関係をグラフと見なし，次数，経路長，直径，バンド幅などの観点から議論する研究があるが[5),14)~16)]ノードの位置を陽に

は扱っていないため，平面上のノード集合を対象としたネットワークに対してはそのまま適用しにくい．

また，Harvey ら[17]，Aspnes ら[18] は P2P 環境におけるノードの頻繁な参加/離脱やネットワークの脆弱性に対応するため，ネットワークのトポロジに確率的グラフ構造を拡張した構造を提案している．これらもノードの位置情報を用いておらず，平面上のノード集合に対してそのまま適用することはできない．また，最上位レベルのノードに負荷の高いホットスポットができる可能性も残されている．

[空間充填曲線]

Shu ら[19]，Xu ら[13] はヒルベルト曲線，z 曲線などの空間充填曲線により対象空間の部分領域に対して 1 次元 ID 付けを行い，この ID に基づいて Skipnet によりオーバレイネットワークを構成している．これにより $O(\log N)$ の部分領域内のデータアクセスを可能にしている．しかし，ID の近接関係が必ずしも地理的な近接関係に一致せず，平面上における範囲問合せなどに利用する場合ではその経路選択が複雑になる．また，空間充填曲線を用いるため，空間全体の大きさを固定とする必要があり，対象空間の動的な拡張には対応しにくい．これらに対して，本提案手法ではノード位置とノード集合が構成するボロノイ図を明示的に用いてオーバレイネットワークを構成している．提案手法ではネットワークの局所性に対して，ノード間の遠隔接続経路を 2 次元平面上で協調的に構成することで空間の拡張に対しても範囲問合せを効果的に実行できる．各ノードがドロネー図と LRC の自律分散生成アルゴリズムを併用すれば，ノードの移動や複数の P2P ドロネーネットワークの統合への対応なども期待できる．

8. おわりに

本稿では，P2P ドロネーネットワークにおける LRC の分散生成法とその利用法として範囲問合せおよび遠隔地点への経路選択について述べた．さらにシミュレーションにより LRC の生成/維持にかかる負荷と LRC を利用した効果について評価した．提案手法では，対象空間をノード位置に基づいたボロノイ領域の集合として，その水平/鉛直の幅（帯）に基づいてネットワーク内の各ノードが LRC をボトムアップに生成した．さらに，空間全体へ構成された LRC を利用して任意の 2 地点間のホップ数を低減させることで，スケーラブルな空間に対しても範囲問合せができることを確認し，その効果について示した．この提案 LRC は GIS などの位置情報をキーとしてデータを保存し扱うアプリケーションを P2P ドロネーネットワーク上に構築した際に任意のサイズの範囲問合せにおいて使用できるほか，異種の様々な粒度のデータに応じた近傍を定義して相互の影響を計算するような物理シミュレータを P2P ドロネーネットワーク上に構築する場合にも利用可能である．今後の課題としては，LRC 生成後においてノード次数の分散を減少させるようにアルゴリズムを拡張することなどがあげられる．

謝辞 貴重なご指摘を賜った査読者に衷心より感謝の意を表す．本研究の一部は，平成 15–19 年度文部科学省私立大学学術研究高度化事業オープン・リサーチ・センター整備事業（知識ネットワーク社会創造のための人的・情報環境の構築に関する研究）ならびに学術フロンティア推進事業（合意形成のための認知的・数理的情報処理システムの構築）の支援を受けた．

参考文献

1) Chawathe, Y., Ramabhadran, S., Ratnasamy, S., LaMarca, A., Shenker, S. and Hellerstein, J.: A case study in building layered DHT applications, *ACM SIGCOMM*, pp.97–108 (2005).
2) Araújo, F. and Rodrigues, L.: GeoPeer: A Location-Aware Peer-to-Peer System, *Proc. 3rd IEEE International Symposium on Network Computing and Applications* (*NCA'04*), pp.39–46 (2004).
3) 金子 雄，春本 要，福村真哉，下條真司，西尾章治郎：ユビキタス環境における端末の位置情報に基づく P2P ネットワーク，情報処理学会論文誌：データベース，Vol.46, No.SIG18 (TOD28), pp.1–15 (2005).
4) Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R. and Shenker, S.: GHT: A Geographic Hash Table for Data-Centric Storage, *Proc. 1st ACM international workshop on Wireless sensor networks and applications* (*WSNA'02*), pp.78–87 (2002).
5) Hellerstein, J.M.: Toward network data independence, *ACM SIGMOD Record*, Vol.32, No.3, pp.34–40 (2003).
6) 大西真晶，源元佑太，江口隆之，加藤宏章，西出 亮，上島紳一：ノード位置を用いた P2P モデルのためのドロネー図の自律分散生成アルゴリズム，情報処理学会論文誌：データベース，Vol.47, No.SIG4 (TOD29), pp.51–64 (2006).
7) Ohnishi, M., Nishide, R. and Ueshima, S.: Incremental Construction of Delaunay Overlaid Network for Virtual Collaborative Space, *3rd Proc. Conference on Creating, Connecting and Collaborating through Computing* (*C5*), pp.77–84, IEEE CS Press (2005).

8) Araújo, F. and Rodrigues, L.: Long Range Contacts in Overlay Networks, *Euro-Par*, pp.1153–1162 (2005).
9) Naor, M. and Wieder, U.: Novel architectures for P2P applications: the continuous-discrete approach., *ACM Symposium on Parallel Algorithms and Architectures* (*SPAA*), pp.50–59 (2003).
10) Okabe, A., Boots, B., Sugihara, K. and Chiu, S.N.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd Ed., Wiley Series in Probability and Statistics (1992, 2000).
11) Stoica, I., Morris, R., Karger, D., Kaashoek, M. and Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *Proc. ACM SIGCOMM '01 Conference*, pp.149–160 (2001).
12) de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O.: *Computational Geometry: Algorithms and Applications*, 2nd ed., Springer (2000).
13) Xu, Z., Mahalingam, M. and Karlsson, M.: Turning Heterogeneity into an Advantage in Overlay Routing, *IEEE INFOCOM* (2003).
14) Loguinov, D., Casas, J. and Wang, X.: Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience, *IEEE/ACM Trans. Netw.*, Vol.13, No.5, pp.1107–1120 (2005).
15) Kaashoek, M.F. and Karger, D.R.: Koorde: A Simple Degree-Optimal Distributed Hash Table, *Proc. 2nd International Workshop on Peer-to-Peer Systems* (*IPTPS03*), Kaashoek, M.F. and Stoica, I. (Eds.), Lecture Notes in Computer Science, Vol.2735, pp.98–107, Springer (2003).
16) 岡下 綾, 有次正義, 柴田幸夫：P2P ネットワークにおける一般化 Kautz ダイグラフに基づく分散ハッシュ表を用いた検索アルゴリズム，日本データベース学会 Letters，Vol.3, No.2, pp.101–104 (2004).
17) Harvey, N.J., Jones, M.B., Saroiu, S., Theimer, M. and Wolman, A.: SkipNet: A scalable overlay network with practical locality properties, *4th USENIX Symposium on Internet Technologies and Systems* (*USITS '03*) (2003).
18) Aspnes, J. and Shah, G.: Skip Graphs, *Proc. 14th annual ACM-SIAM symposium on Discrete algorithms* (*SODA*), pp.384–393 (2003).
19) Shu, Y., Ooi, B.C., Tan, K.-L. and Zhou, A.: Supporting Multi-dimensional Range Queries in Peer-to-Peer Systems, *IEEE International Conference on Distributed Computing Systems*, pp.155–164 (2005).

(平成 18 年 12 月 20 日受付)
(平成 19 年 4 月 17 日採録)

(担当編集委員 春本 要)

大西 真晶（学生会員）
昭和 53 年生．平成 14 年関西大学総合情報学部卒業．平成 16 年同大学院総合情報学研究科博士前期課程修了．現在，博士後期課程在学中．仮想空間の高度利用，P2P ネットワークの研究に従事．電子情報通信学会，IEEE 各学生会員．

坪井 新治
昭和 58 年生．平成 18 年関西大学総合情報学部卒業．同年同大学院博士前期課程へ進学．P2P システムに関する研究に従事．

平山 雅夫
昭和 58 年生．平成 18 年関西大学総合情報学部卒業．同年同大学院博士前期課程へ進学．P2P システムに関する研究に従事．

江口 隆之
昭和 57 年生．平成 17 年関西大学総合情報学部卒業．平成 19 年同大学院総合情報学研究科博士前期課程修了．同年日立ソフトウェアエンジニアリング（株）入社．P2P システムに関する研究に従事．

上島 紳一（正会員）
昭和 30 年生．昭和 53 年京都大学工学部数理工学科卒業．昭和 58 年同大学院工学研究科博士課程単位取得退学（京都大学工学博士）．現在，関西大学総合情報学部教授．マルチメディア情報システム，柔軟な情報ベースに関する研究に従事．電子情報通信学会，ACM，IEEE 等の会員．