迂回経路を用いたドロネーネットワークの提案と検討

著者	平山 雅夫, 大西 真晶, 上島 紳一
発行年	2008-04-07
その他のタイトル	Delaunay network using detour
URL	http://hdl.handle.net/10112/4691

迂回経路を用いたドロネーネットワークの提案と検討

平山　雅夫† 　大西　真晶†† 　上島　紳一†

†, †† 関西大学大学院総合情報学研究科
569-1095 大阪府高槻市霊仙寺町 2-1-1
E-mail: †{fb6m127,ueshima}@edu.kansai-u.ac.jp, ††ohnishim@nict.go.jp

あらまし　計算機器（以下ノード）の位置関係により定義されるドロネーネットワークは，平面上で局所性を持つため，(i) 対象領域の拡張に伴うネットワークの拡張性，(ii) ノードの位置に基づく地理的な経路選択，(iii) ノードの頻繁な参加・離脱に対する弾力性の保持などの面で有用である．そのため，アドホックネットワークを用いたアプリケーションにおいて応用性が広い．しかし，ノードの通信範囲に制限のある無線環境を考慮すると，ノードの位置に基づくドロネー図を直接的に構成できない．そこで本稿では，直接接続を持たないノード間に対して迂回経路による仮想リンクを定義し，無線環境でドロネー図を仮想的に構成するアルゴリズムを提案する．各ノードは，実行時の隣人までの接続通知に経路情報を繋ぎ合わせることで，迂回経路による仮想リンクを生成し，すべてのノード間で仮想的なドロネーネットワークを自律分散的に構成することができる．さらに，迂回経路は一般に 2 ノード間で複数存在するため，迂回経路の縮退アルゴリズムについても述べる．提案ネットワークの上で，欲張り法及びその拡張法を用いた経路選択法を提示し，本ネットワークのアプリケーションについても述べ，各ノードの維持負荷，迂回経路の効果などを定量的に検証する．

キーワード　MANET, アドホック, マルチホップ, 通信範囲

Delaunay network using detour

Masao HIRAYAMA†, Masaaki OHNISHI††, and Shinichi UESHIMA†

†, ††Graduate school of Informatics, Kansai University
2-1-1 Ryozenji, Takatsuki, Osaka, 569-1095 Japan
E-mail: †{fb6m127,ueshima}@edu.kansai-u.ac.jp, ††ohnishim@nict.go.jp

Abstract　A P2P Delaunay network, whose topology is defined by geometric adjacency of computational entities (nodes), has locality on a plane. It has the following advantages of (i) scalability in the spatial extension, (ii) geographical routing capability to specific points, (iii) resiliency against node's frequent join/leave. Hence, this network is expected to be widely applicable for ad-hoc network applications. However, in practical situations where node's communication range is limited, such as in wireless network environments, our autonomous generation algorithm of Delaunay network cannot be directly employed for construction of Delaunay network. To cope with this problem, the authors introduce a 'virtual link' by setting 'detour' among Delaunay neighbors that cannot have direct link due to limited communication range, and provide an autonomous and distributive generation algorithm for generating a Delaunay network virtually. In the algorithm, each node notifies its connection to neighbors using the proposed algorithm, in order to generate a detour link by merging routing tables. We also describe an algorithm of detour degeneracy for the case when multiple detours exist between two nodes. Then, we present a routing method using a geometric greedy routing and detour links, and evaluate our schemes in terms of network maintenance costs numerically.

Key words　MANET, ad-hoc, mutli-hop, communication range

1. はじめに

無線アドホックネットワーク (MANET) は，イベント会場などでの即時的なネットワークシステムや防災管理システムなどへの応用が期待されている．このネットワークにおいて，各ノードは，基地局やアクセスポイントを介さずに直接通信を行うため，自律分散的に経路を構成しなければならない．このため，各ノードの地理的な位置関係に着目したネットワークの自律分散的な手法の研究が注目を集めている [1], [2].

一方，ドロネー図構造のネットワークでは，(i) 地理的に近いノードとのみ接続し，(ii) ノードの隣人表が小さく，(iii) ネットワークの拡張性があり，(iiii) 任意のノード間を到達可能である，などの理由からノード位置を利用したネットワークに適用しやすいという特徴がある [3]〜[5]．しかし，このネットワークでは，各ノードが所持する隣人表 (以下リンク) はドロネー辺で繋がるリンクとして定義しているため，無線環境ではそのまま適用することができない．これは，ドロネー図において，ノード間で構成される辺（以下リンク）が，ノードの通信範囲以上の場合では従来手法によって直接的にドロネーリンクを構成できないためである．

そこで，本稿では，各ノードがドロネーリンクとして所持する隣人内で，ノードの通信範囲外に存在する隣人に対してはリンクを仮想的に構成し，その仮想リンク間を迂回経路によって通信可能とすることで，無線環境に適用する．

ここでは，通信範囲を考慮した無線アドホックネットワークのモデルとして Unit Disk Graph(以下，UDG) を用いて定式化を行う．この UDG モデル上において，各ノードはドロネー図生成のための従来アルゴリズムを拡張し，局所ドロネー化及びノード間へのリンク通知を繰り返し実行することで，提案ネットワークを自律分散的に構成する．特に，リンク通知は，自身の隣人表から特定の 2 ノードを選択し，その 2 ノード間が通信範囲より長い場合においては，2 ノード間の迂回経路の生成を行い，迂回経路とともにノードの情報を通知することによって，迂回経路による仮想リンクを構成することができる．

また，生成した迂回経路をそのまま利用すると，仮想リンク間の迂回経路に冗長な経路が存在する場合がある．そのため，迂回経路を短縮する縮退法についても述べる．さらに，生成ネットワークの利用法として，欲張り法およびその拡張法によって迂回経路を用いて仮想リンクを辿る欲張り法およびその拡張法による経路選択法についても述べる．

以降の章では，2 章では，提案手法の前提及びそのモデルとした UDG について述べたあと，それに基づく仮想リンク及び迂回経路の定義を行う．3 章では，迂回経路を用いた仮想リンク型ドロネーネットワークの生成アルゴリズムについて述べる．4 章では，欲張り法及びその拡張法を用いた本提案ネットワークの利用法について述べ，5 章で本提案ネットワークの生成コスト及び，経路選択の効果についての評価をホップ数の観点から行う．6 章では，関連文献について述べ，7 章では，まとめと今後の課題について述べる．

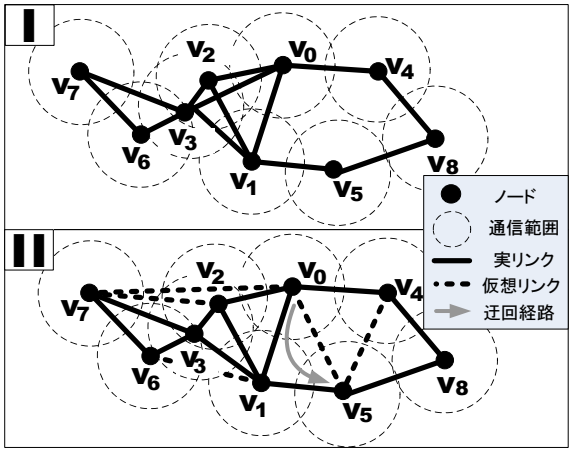

図 1　UDG 上で生成するドロネーネットワーク

2. 準　　備

まず，ノードの前提についてまとめ，定式化を行う．

2.1 UDG

ここでは，各ノードの前提についてまとめる．ノードは以下の前提に基づくものとする．

（1） 各ノードは，すべて同じ処理能力をもち，無指向性のアンテナを所持する計算主体である．

（2） 各ノードは，識別可能な固有の ID 及び，GPS などで自ノード位置 (X/Y 座標) 通信半径を所持するものとする．

（3） 各ノードは，通信範囲外のノードに対しての通信はマルチホップ通信を行い，直接もしくはマルチホップ通信によって，他ノードと (2) の情報及びについて交換可能である．

このような前提を表すモデルとして Unit Disk Graph(以下 $UDG(V)$) を用いる．$UDG(V)$ とは，全ノードの通信範囲の半径を r として，任意の 2 ノード $v_i, v_j \in V$ において，$\overline{|v_iv_j|} < r$ を満たす 2 ノードが接続する無向グラフである (ここで，V はユークリッド平面上のノード集合，$\overline{|v_iv_j|}$ は v_i,v_j 間の辺 $\overline{v_iv_j}$ の距離を表す)．$UDG(V)$ はこの定義に基づくため，平面グラフではなく，辺が交差する場合が存在する．また本稿で用いるネットワークは，与えられた任意の 2 ノード間の経路の存在を仮定するため，$UDG(V)$ が連結グラフであるとする．

図 1 の (I) に 9 個のノードから成る $UDG(V)$ を示す．図上では，通信範囲の直径を前述の r とするため，ノードを中心とした円が交差する場合にノード間が繋がる．そのため，図上では v_0 と v_1 間は繋がるため，通信経路が存在する．

Note: また，ここでは説明の簡単化のため，各ノードは同期的に計算処理を実行するが，非同期の場合にも対応できる．

2.2 仮想リンク

図 1 の (II) に $UDG(V)$ 上から，ノード間をドロネー辺で繋がる，提案ネットワークの例を図示する．図上では v_0 は，通信範囲内のノード $\{v_1,v_2,v_4\}$ とは実線で繋がり，通信範囲外のノード $\{v_5,v_6,v_7\}$ は破線によって繋がり，これらをノードが所持する隣人表とする．このように，ノードが通信範囲内のノードと繋がる辺を実リンク，直接通信可能でないノード繋がる辺

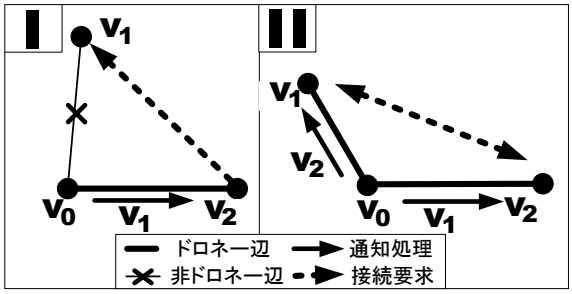

図 2　ドロネー図生成時の通知処理と接続要求

を 仮想リンクと定義する．また，これらの辺の両方をまとめて**リンク**と呼ぶ．

以上の関係をまとめると，任意のノード $v_i \in V$ が所持するノード集合を V_i と示し，$v_j \in V_i$ が，$r < \overline{|v_i v_j|}$ を満たす場合にがノード v_i は v_j と仮想リンクで繋がる．

図 1 の (II) において v_0 の所持する仮想リンクで繋がるノード v_5 と通信を行う場合には，実リンクを辿り，マルチホップ通信を行うことで v_5 と通信を行う．そのため，v_0 はノード v_5 と通信を行うために実リンクの経路が必要である．この実リンクの経路を**迂回経路**と定義する．

ノード $v_i \in V$ から仮想リンクで繋がるノード v_j までの迂回経路を $Path(v_i, v_j)$ によって表す．また，$Path(v_i, v_j)$ 上の具体的な経路を示す場合には，v_i から近いほうから順番に $(v_a, v_b, ...v_n)$ と表す．迂回経路は順番も考慮するため，$2 \leqq |Path(v_i, v_j)|$ の場合には，$Path(v_i, v_j) \neq Path(v_j, v_i)$ となる．

Note: ただし，仮想リンクに対する迂回経路は複数存在する．そのため，仮想リンク間の経路を常に最小のホップ数で到達できるとは限らない．図 1 の (II) では，v_0 の v_5 までの迂回経路 $Path(v_0, v_2)$ は，(v_1) 以外にも (v_4, v_8) などが考えられる．

3. 生　成　法

前章の準備に基づいた，ドロネー図の自律分散生成法について述べる．$UDG(V)$ 上のノード $v_i \in V$ が実行するドロネー図生成築アルゴリズムは所持する隣人数 $|V_i|$ によって異なり，
$(|V_i| = 1)$ の場合では，v_i は隣人表に 2 つ以上のノードを所持するまで他者からの通知を待つ．
$(|V_i| = 2)$ の場合では，v_i は三角化通知のみを行う．
$(|V_i| \geqq 3)$ の場合では，v_i は局所ドロネー化, 委譲, 三角化通知の 3 つの処理を順番に行う．

なお，初期状態において v_i は，$|v_i v_j| < r$ となる全ての v_j を V_i のノードとし，他ノードとの接続要求や通知を行うことによって V_i を更新する．これを全てのノードが繰り返し実行することで，全体としてドロネー図を構築する．ここで，上記で示した主要 3 つのアルゴリズムの内容について確認する．

局所ドロネー化とは，v_i を中心として時計順に並べ全ての隣人ノードを，外接円判定によって v_iと ドロネーリンクで繋がるノード（ドロネーノード）か，繋がらないノード（非ドロネーノード）かの 2 種類に分類する処理である．

委譲とは，v_i の局所ドロネー化によって非ドロネーノードを

Algorithm 1 ノード v_i による迂回経路生成

```
1: Given v_j, v_k ∈ V_i, Path(v_i, v_j) ← null
2: if r < d(v_j, v_k) then
3:     if v_i has no Path then
4:         Path(v_j, v_k) ← v_i
5:     else if v_i has Path(v_i, v_k) then
6:         Path(v_j, v_k) ← (v_i + Path(v_i, v_k))
7:     else if v_i has Path(v_i, v_j) then
8:         Path(v_j, v_k) ← (Reverse(Path(v_i, v_k)) + v_i)
9:     else
10:        Path(v_j, v_k) ← (Reverse(Path(v_i, v_k)) + v_i + Path(v_i, v_k))
11:    end if
12: end if
```

ドロネーノード内でその非ドロネーノードに最も近いノードへ委譲する通知処理である．また，三角化通知とは，ドロネーノード間が互いノード間の関係を認識させるために行う通知処理である．図 2(I) に委譲処理を示す．v_0 はドロネーノードである v_2 に非ドロネーノードである v_1 を通知する．ここで，v_0 から通知を受け取った v_2 は v_1 に対して通信経路の接続要求を行うが v_2 の通信範囲内にないため，直接接続できない．同様に，(II) の三角化通知においても，v_0 は v_1,v_2 の 2 つのドロネーノード間が相互に接続するため，両方に対して通知を行うが，通知を受けたノードが自身の通信範囲内に存在しない場合には接続できない．

そこで，ノード $v_i \in V$ がこれらの通知処理を行う場合には，迂回経路を付加して通知することでこれに対応する．これらの通知処理は，v_i が所持するノードから対象となる 2 つを取り出し，そのノード間の関係に基づき通知処理を行う．そのため，次節では，2 つのノード間に対する迂回経路の生成及びその縮退法について説明する．

Note: なお，ここではノードの隣人による迂回経路の生成は先に来た通知を優先する．つまり，v_i が v_j からの通知処理にによって $Path(v_i, v_k)$ の通知を受けた場合に，v_j 以外の他のノード v_l から v_k 及び $Path(v_i, v_k)$ の通知を受けると，v_l からの通知は破棄する．

3.1　迂回経路生成

本節では，各ノードによる迂回経路の生成アルゴリズムについてアルゴリズム 1 及び図 3 を用いて説明する．ここでは，ノード $v_i \in V$ が隣人表から 2 つのノード $v_j, v_k \in V_i$ を取り出した場合において，v_i が v_j に対して v_k の情報を通知する時における迂回経路 $Path(v_j, v_k)$ の生成について示す．また，v_j,v_k 間の迂回経路を $Path(v_j, v_k) = null$ としてアルゴリズムを始める．これらはアルゴリズム 1 の 2 行目に初期状態として与えている．

Step1. v_i は $r < \overline{|v_j v_k|}$ であるかをチェックする．これはアルゴリズム 1 の 2 行目に当たる．

Step2-1. v_i は v_k 及び v_j の両方と実リンクで繋がル場合，v_i 自身のみが v_j と v_k 間の迂回経路となるため，$Path(v_j, v_k)$ に v_i 自身のみを追加する．これはアルゴリズムの 3,4 行目に当たる．また，図 3(I) のように v_0 の所持するノード v_1, v_2 が実リンクで繋がっているとこの処理を実行する．

Step2-2. v_i の所持する v_k が他ノードから通知を受けたため，仮想リンクで繋がる場合には，迂回経路 $Path(v_i, v_k)$ を所持し

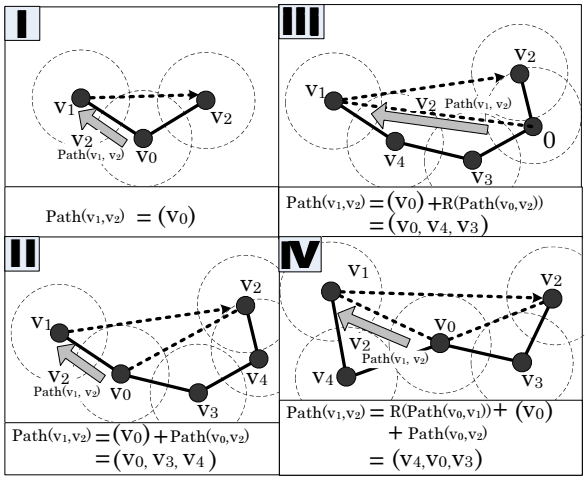

図 3　迂回経路の生成例

ている．そのため，v_i はまず $Path(v_j, v_k)$ に v_i を追加した後，$Path(v_i, v_k)$ を順番に追加する．これはアルゴリズム 1 において 5,6 行目に当たる．なお，アルゴリズム中 7 行目の”+”は Path の連結を示す．つまり，$Path(v_i, v_k) = (v_a, v_b)$ の場合，$Path(v_j, v_k)$ は (v_i, v_a, v_b) となる．図 3(II) において，v_0 は，$Path(v_1, v_2)$ に自身を格納した後，$Path(v_0, v_2)$ を格納する．

Step2-3. v_i の所持する v_j が仮想リンクで繋がる場合には迂回経路 $Path(v_i, v_j)$ を所持しているため，v_i は $Path(v_j, v_k)$ に反転させた $Path(v_i, v_j)$ を追加した後，v_i を追加する．アルゴリズム中 1 中 7,8 行目に当たり，$Reverse(Path(v_i, v_j))$ とは，経路の反転を表す．つまり，$Path(v_i, v_j) = (v_c, v_d)$ の場合には $Reverse(Path(v_i, v_j) = (v_d, v_c)$ となり，結果，$Path(v_j, v_k) = (v_d, v_c, v_i)$ となる．v_j から v_i への経路は，v_i から v_j への経路の反転であるため，このような処理を行うことで，v_j から v_k までの辿ることが可能な経路を構成できる．これは，図 3 中 (III) に当たる．

Step2-4. v_i の所持する v_j 及び v_k の両方が仮想リンクで繋がる場合には迂回経路 $Path(v_i, v_j)$ 及び $Path(v_i, v_k)$ を所持するため，v_i は $Path(v_j, v_k)$ に反転させた $Path(v_i, v_j)$ を追加した後，v_i を追加し，さらに $Path(v_i, v_k)$ を追加する．これは，アルゴリズム中 9,10 行目，図 3 における (IV) に当たる．

3.2　迂回経路の縮退

ノード v_i が v_j と v_k 間の迂回経路 $Path(v_j, v_k)$ の生成時，v_i が所持する $Path(v_i, v_j)$ 間と $Path(v_i, v_k)$ 間に共通するノードが存在する場合がある．この場合，$Path(v_j, v_k)$ を生成して v_j にそのまま通知を行うと，$Path(v_j, v_k)$ は v_j と v_k 間の通知処理時に共通するノードを 2 回経由する冗長な通信経路となるため，ネットワーク全体の通信量の増大につながる．ここでは，このような冗長な経路に対して 2 種類の縮退アルゴリズムについて述べる．

[縮退法 1]

ノード v_i がノード v_j から v_k 間の迂回経路 $Path(v_j, v_k)$ を v_i 以外の他ノードの通知によって得た情報によって構成すると，$Path(v_j, v_k)$ 内に v_j または v_k が含む場合が存在する．ここで v_i の通知によって v_j を含む $Path(v_j, v_k)$ を v_j が取得すると，

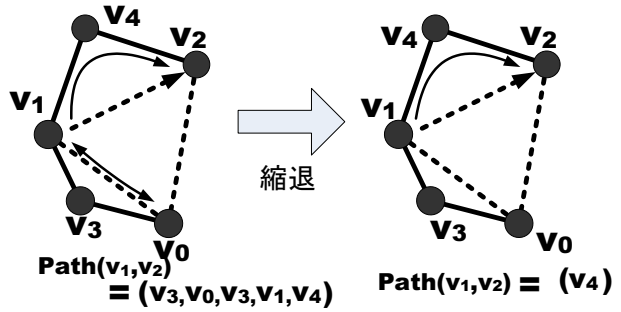

図 4　縮退法 1 による縮退例

Algorithm 2 ノード v_i による縮退法 1

```
1: Given Path(v_j, v_k), n = |Path(v_j, v_k)|, v_j, v_k ∈ V_i
2: for x = 1 to n do
3:     if v_x = v_j then
4:         for a = 1 to x do
5:             Delete(v_a)
6:         end for
7:     else if v_x = v_k then
8:         for b = x to n do
9:             Delete(v_b)
10:        end for
11:        x ← n
12:    end if
13: end for
```

v_j が v_k に通信を行う場合に，通信パケットは v_j から v_j までの経路を辿った後，v_j から v_k までの経路を辿る．また，同様に v_k が v_j からの通信パケットを受信する時，通信パケットは v_k までの経路を経由して v_k に到達する．

そのため，$Path(v_j, v_k)$ 内に v_j が含む場合は v_j までの経路及び v_k 以後の経路を削除できる．以下に，縮退アルゴリズムについてアルゴリズム 2 を用いて述べる．なお，迂回経路長 $|Path(v_i, v_j)|$ を n とし，$Path(v_j, v_k)$ は v_j から近いノードから順番に，迂回経路リストの番号を $(0 \leqq x \leqq n)$ とする．

Step1. まず v_i は，所持する $Path(v_i, v_j)$ のうちノード v_x(初期値は $x = 1$, つまり v_1) を取り出し,$v_x = v_i$ または $v_x = v_k$ かを検査する．これはアルゴリズム中 3,7 行目にあたる．

Step2-1. v_x が v_i でも v_k でもないと $x \leftarrow x+1$ として Step1 に戻る．これはアルゴリズム中 2 行目にあたる．なお，$n = x$ の場合にはアルゴリズムを終了する．

Step2-2. $v_x = v_i$ の場合には，v_1 から v_x 間の全てのノードを削除し，$x \leftarrow x+1$ として，Step1 に戻る．これはアルゴリズム中 3 から 6 行目に当たり，a の値を 1 から x まで変化させ，その間のノードをすべて削除する．なお，$x = n$ の場合にはアルゴリズムを終了する．

Step2-3. $v_x = v_k$ の場合には，v_x から v_n までのノードを全て削除し，アルゴリズムを終了する．これはアルゴリズム中アルゴリズム 7 から 11 行目に当たる．

これを図 4 を用いて説明する．図上では v_0 が生成する $Path(v_1, v_2)$ が $(v_3, v_0, v_3, v_1, v_4)$ であり，これを上記のアルゴリズムによって縮退する．まず，$Path(v_1, v_2)$ から v_x を順番に検査すると，4 番目に v_1 が存在する．そのため，Step2-2 によって v_1 を含めた v_1 以前のすべてのノードを削除し，$v_x = v_4$ として，v_4 の検査を終えるとアルゴリズムを終了する．この縮退法により $Path(v_1, v_2)$ は (v_4) のみとなる．

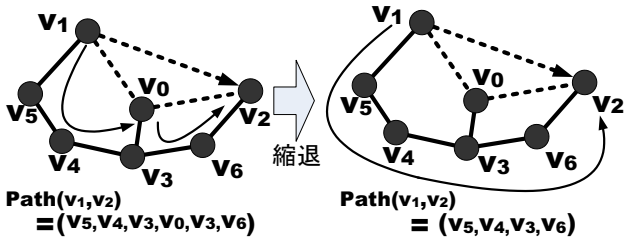

図 5　縮退法 2 による縮退例

Algorithm 3 ノード v_i による縮退法 2

```
1: Given Path(v_i, v_j), n = |Path(v_i, v_j)|
2: for x = 1; to n do
3:     for y = x + 1 to n do
4:         if v_x = v_y then
5:             while v_x = v_{y+1} do
6:                 Delete(v_y)
7:                 y ← y + 1
8:             end while
9:         end if
10:    end for
11: end for
```

[縮退法 2]

さらに，$Path(v_j, v_k)$ 内に v_j,v_k を持せず，迂回経路に共通するノードが存在する場合がある．そこでこのような場合における短縮アルゴリズムについてアルゴリズム 3 を用いて示す．また，ここでは前述と同様，$n = |Path(v_i, v_j)$ とし，v_j から近い順番として，v_x, v_y の x, y を迂回経路のリスト番号とする $(0 \leqq x, y \leqq n)$.

Step1. まず v_i は，所持する $Path(v_i, v_j)$ のうちノード v_x(初期値は $x = 1$, つまり v_1) を取り出す．次に，$y \leftarrow x + 1$ とした v_y を取り出し，$v_x = v_y$ かを検査する．これはアルゴリズム中 4 行目に当たる．

Step2-1. $v_x = v_y$ でない場合には，$y \leftarrow y + 1$ として Step1 に戻る．これはアルゴリズム中 3 行目に当たる．

Step2-2. $v_x = v_y$ の場合には，v_x から v_{y-1} 間の全てのノードを削除し，$x \leftarrow y$, $y \leftarrow y + 1$ として，Step1 に戻る．これはアルゴリズム中 4 から 9 行目であり，v_yから v_x 間の番号のノードを削除している．

Step3. $y = n$ となると，$x \leftarrow x + 1, y \leftarrow x + 1$ として Step1 に戻る．これを $x = n$ となるまで繰り返し，$x = n$ となる時点でアルゴリズムを終了する．

図 5 を用いて説明する．v_0 が生成する迂回経路 $Path(v_1, v_2)$ が $(v_5, v_4, v_3, v_0, v_3, v_6)$ である場合において上記のアルゴリズムによって縮退する．まず，v_x を v_5, v_4 として同じノードがないかを検査するここで，$v_x = v_3$ の検査中，$v_y = v_3$ となるため，v_x から v_{y-1} までのノードを迂回経路リストから削除する．ここでは，v_x から v_{y-1} の要素が v_3, v_0 であるため，これらを削除する．そして $v_x = v_3$, $v_y = v_6$ として検査を再開し，$v_x = v_6$ となった時点で縮退アルゴリズムを終了する．

v_i はこれらの迂回経路の生成及び縮退アルゴリズムを終了すると，三角化通知または委譲処理によって v_j に v_k と迂回経路 $Path(v_j, v_k)$ の二つの情報の通知を行う．これによって，$\overline{|v_j v_k|} > r$ の場合においても，v_j は v_k に迂回経路 $Path(v_j, v_k)$ を辿ることで v_k と通信を行うことができる．この迂回経路の

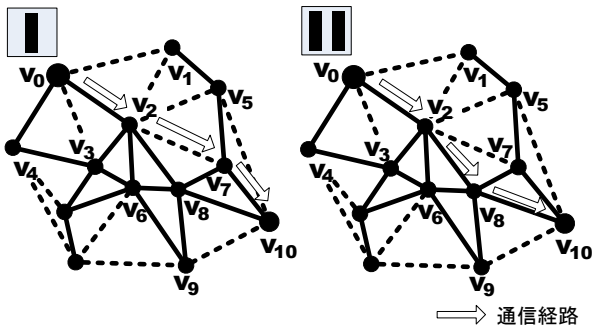

図 6　欲張り法及びその拡張法による経路選択

生成によって，実リンクで繋がらないノード間に対しては，迂回経路を経由させることで仮想リンクを構成できる．そのため，各ノードが従来アルゴリズムに加えてこれらのアルゴリズムを繰り返し実行することで，迂回経路を用いたドロネーネットワークを自律分散的に生成できる．

4. 利　用　法

4.1　欲張り法を用いた経路選択

本章では，迂回経路を用いたドロネーネットワークの利用法について説明する．ここでは，生成したネットワーク上において各ノードが欲張り法によって経路選択を行い，パケットを転送する手順について説明する．各ノードが生成するパケットはドロネーリンクで繋がるノードを辿ることによって，他の任意のノードに到達することができる．

ここで，アルゴリズムを実行するノードを v_i, 始点ノードを v_S，目的地ノードを v_G として，v_S から v_G までの欲張り法による経路選択を図 6[I] を用いて説明する．なお，目的ノードの位置座標は既知とし，初期値として $v_i \leftarrow v_S$ を与える．パケット内容は目的地点の位置座標を含み，迂回経路利用時には，迂回経路間のノードに対して迂回経路情報を付加する．

Note: また，v_S が v_G への応答を要求する場合には，v_s から v_G 間以外に v_G から v_S 間の経路が必要である．そこで，問合せ元ノードはパケット内容に経由ノードリストを付加したパケットを送信する．パケットを受信するノードは，ノード自身を経由ノードリストに格納して転送することで v_S から v_G までの経路が順番に格納される．これによって，問合せ先ノード v_G は，v_S から v_G 間で経由ノードリストに格納さた経路を逆順に辿ることで，v_S に応答することができる．

Step1.v_i は $\overline{|v_j vG|}$ が最小となる v_j を V_i の中から決定する．図 6[I では，始点ノード v_0 は，$V_0 = \{v_1, v_2, v_3, v_4\}$ から v_{10} までの距離をそれぞれ計算し，最も距離の近い v_2 を決定する．

Step2-1.v_j が v_i と実リンクで繋がる場合，直接 v_j にパケットを転送する．図中 [I] の v_0 は，v_2 を実リンクとして所持するため v_2 に直接パケットを送信する．

Step2-2.v_j が v_i と仮想リンクで繋がる場合には，v_i はパケットに $Path(v_i, v_j)$ の情報及び v_j の情報を付加する．$Path(v_i, v_j)$ 間のノードはこれに基づき，$Path(v_S, v_j)$ 間でパケットを v_j まで到達させる．図中 [I] の v_2 が目的ノードに最も距離の近い v_7 を Step1 によって決定するが，v_7 は仮想リンクであるため，直

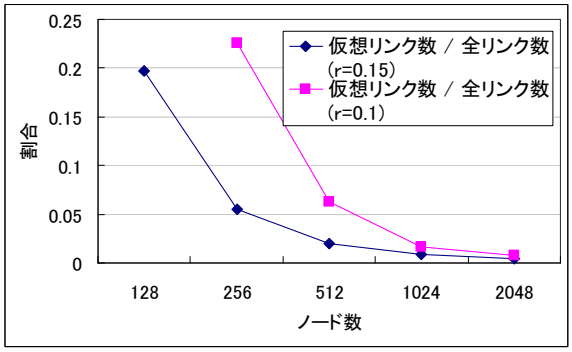

図 7　全リンク数に対する仮想リンク数の割合

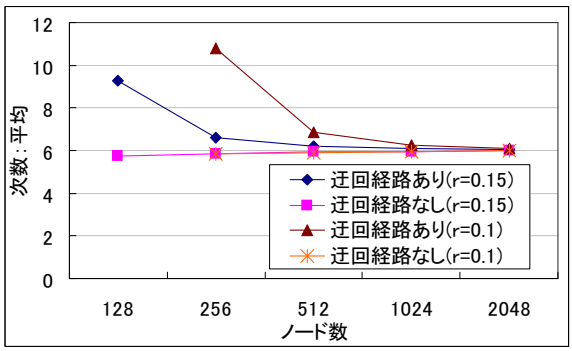

図 8　迂回経路を所持または所持しないノード数の次数平均

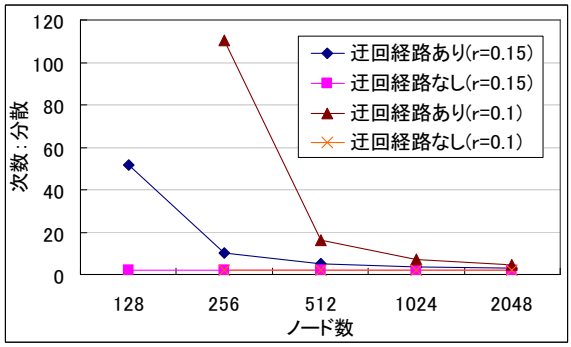

図 9　迂回経路を所持または所持しないノード数の次数分散

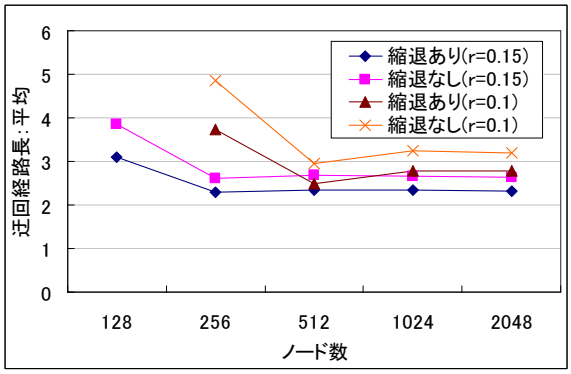

図 10　ノード数に対する迂回経路長の平均

接パケットを転送できない．そこで，v_2 は $Path(v_2, v_7) = (v_6)$ を順番に経由して v_7 まで到達する．

Step3. Step1 に戻り，$v_i \leftarrow v_j$ として $v_G = v_j$ となるまで Step1 及び Step2 を繰り返し，v_G までの経路選択を行う．図中 [I] の v_7 がパケットを v_{10} まで転送し，目的地ノードとなる v_{10} がこれを受け取ると欲張り法による経路選択を終了する．ここでは，v_0 から v_{10} 間にかかるホップ数は 4 となる．

4.2　欲張り法の拡張

前述した欲張り法の Step2-2 の場合において経路選択を実行するノード v_i が所持するノード $v_j \in V_i$ が仮想リンクで繋がる場合に，迂回経路 $Path(v_j, v_k)$ を辿ることで仮想リンク v_j まで到達すし，これを繰り返すことで目的地ノード v_G に到達する．しかし v_i が v_j までの途中である $Path(v_i, v_j)$ 間で目的地ノード v_G をリンクで繋がるノードが存在する場合には，そのノード間のリンクを辿ることでそのまま v_G に到達できる．そのため，前節の Step2-2 を拡張して以下のように変更する．なお，他の Step には変更がないため，ここでは Step2-2 のみについて述べる．

Step2-2.　v_j が v_i の仮想リンクの場合には迂回経路 $Path(v_i, v_j)$ のノードに対して順番にパケットを転送する．このとき，$Path(v_i, v_j)$ 内のノード v_k は自身のリンク V_k 内に v_G を所持しているかを調べる．V_k 内に v_G が存在する場合には，v_k は $Path(v_i, v_j)$ を用いず，v_k の所持するリンクを利用して v_G に到達し，経路選択を終了する．

図 6[II] に欲張り法の拡張による経路選択法を示す．図中 [II] の v_2 は，V_2 の中で最も目的地ノードに近い v_7 を決定する．ここで，v_7 は仮想リンクで繋がるため，v_2 は $Path(v_2, v_7)$ である v_8 にパケットを転送する．v_2 からパケットを取得した v_8 は自身の所持するリンク内に目的地ノードである v_{10} を所持するかを調べる．ここでは v_2 は v_{10} へのリンクが実リンクでであるため，v_2 は直接 v_10 へパケット転送する．欲張り法の拡張によって v_0 から v_{10} まで到達するホップ数は 3 となる．

5.　評　　価

本章では，提案手法によって生成したネットワークの特性及び，利用時の特性について評価する．なお，ここではノードを [0,1] × [0,1] の正方領域にランダムに分布させ，通信半径 r を対象となる正方領域の 0.1 および 0.15 と固定し，ノード数 N を変化させ，ネットワークの特徴について調べる．また，$N = 128, r = 0.1$ の場合において，正方領域内にノードをランダムに配置すると，ネットワークが連結しないため，ここでは省略する．なお，ノード配置による影響を考慮し，各シミュレーションを 100 回試行し，その平均を結果として示す．

5.1　生 成 評 価

ここでは，N を 128 から 2048 まで変化させ，ノードごとの次数とネットワーク全体に対する仮想リンク数，迂回経路長の平均と分散についてそれぞれ計測した．

図 7 に全ノード数 N に対して，全リンク数に対する仮想リンク数の割合について示す．仮想リンク数はノード数の増加に伴い小さくなっている．これは，ノード配置に対する対象領域とノードの通信範囲を変化させず，ランダムにノードを分布させた場合では，ノード数が増加すると対象領域内の密度が増加する．そのため結果として，通信範囲内に含まれるノード数も

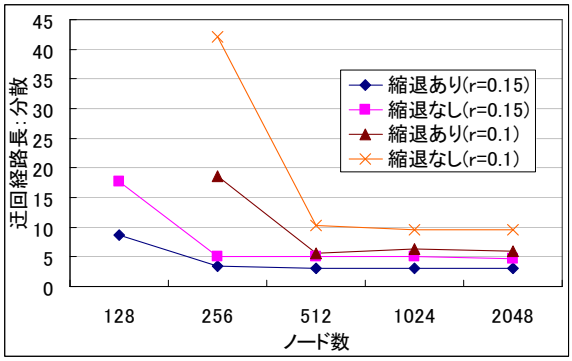

図 11　ノード数に対する迂回経路長の分散

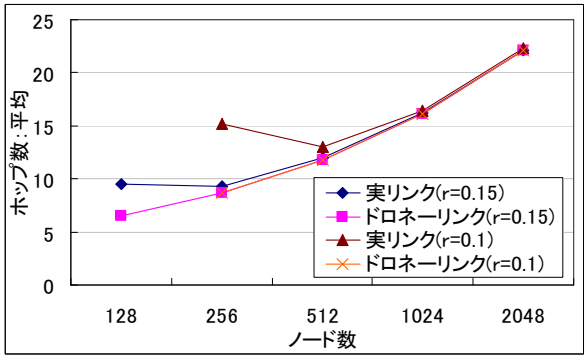

図 13　ノード数ごとのホップ数の平均

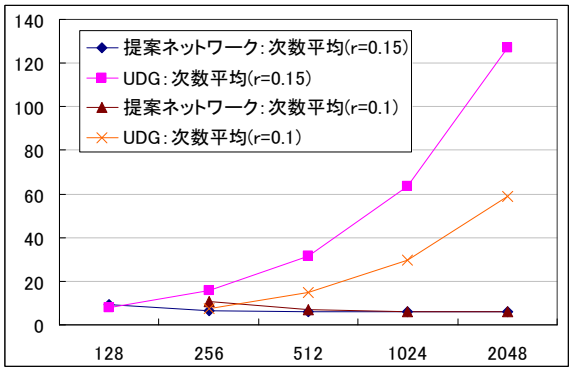

図 12　提案ネットワーク及び UDG でのノード数の次数平均

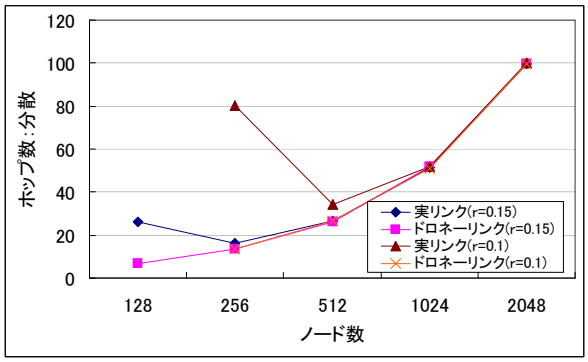

図 14　ノード数ごとのホップ数の分散

増加し，ノードの通信範囲より長いドロネーリンクが少なくなるためであると考えられる．

図 8,9 に UDG 上における提案ネットワーク構築後の，迂回経路を含めた場合と含めない場合におけるノード次数の平均及び分散について示す．なお，迂回経路を含めない場合には，ドロネーリンクで繋がるノードのみを次数とし，迂回経路を含める場合には，さらに，迂回経路長もノード次数として含める．

迂回経路を含めた場合におけるノードの次数は，ノード数が小さい場合には，分散及び平均が若干高くなっている．これは，ノード数が小さい場合には仮想リンク数が増えるため，その分，迂回経路長が長くなったためであると考えられる．一方，迂回経路を含めない場合においては，ノード数の指数的な増加に対しても次数は 6 前後と一定であり， 分散も小さい．これは，ドロネーネットワークの特徴を示している．

また図 8,7 の 2 つによってネットワーク内の迂回経路を所持するノード数が減少する一方，次数の平均はほぼ一定であるため，ノード数全体での次数の平均は小さくなる．

図 10,11 に縮退法を適用する場合と適用しない場合における，N に対する迂回経路長の平均及び分散を示す．通信半径が同じ場合では，縮退法を利用しないほうが縮退法を利用する場合と比べて迂回経路長の平均が大きくなっている．特に，$(N = 256, r = 0.1)$ では，平均約 1 ホップほど短縮されていることが確認できる．また平均及び分散は，$N = 512$ を超えると全てのグラフで一定であるが，$(N = 256, r = 0.1)$ による迂回経路における分散は，大きくなっている．これは，迂回経路の生成数が他の場合よりも多いため，迂回経路の縮退法による効果が顕著に表れたためであると考えられる．

図 12 に UDG の次数と提案ネットワーク (迂回経路を含めた) 場合のノードの次数について示す．UDG の次数はノード数全体の増加に伴い次数も増加している．一方，提案ネットワークでは，ノードの次数は一定である．ここで，$(N = 256, r = 0.1)$ の場合において UDG では 7.3, 提案ネットワークでは 10.8 程度であり，$(N = 128, r = 0.15)$ の場合においては UDG で 7.8, 提案ネットワークでは 9.2 程度であった．つまり，図 7,12 によって，全リンク数の約 18% を超える仮想リンク数が存在する場合では，UDG よりも提案ネットワークでは次数が大きくなっている．

5.2　利 用 評 価

本ネットワークの性能評価のため，N を変化させ，欲張り法及び拡張法を利用した場合におけるルーティングにかかるホップ数を計測した．

図 13,14 では，N を 128 から 2048 まで変化させ，ネットワークからランダムに 2 ノードを決定し，その 2 ノード間での欲張り法による経路選択を 100 回行ったときのホップ数の平均及び分散である．また，実リンク及びドロネーリンクで繋がる，それぞれの経由するノード数について計測した．$(N = 256, r = 0.1)$ において，実リンクの経由ノード数の平均が 15.1 ホップ，ドロネーリンクの経由ノード数の平均が 8.7 ホップと，約 6.4 ホップほどの差が生じた．又この条件では，同様に分散値も大きい．一方，ノード数 512 以上の場合では，実リンクおよびドロネーリンク間でのホップ数の差は小さいことが確認できる．

これは，ノード間のホップ数が，迂回経路の利用分だけドロ

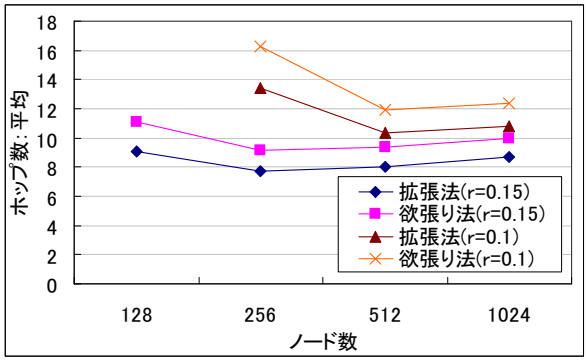

図 15　欲張り法と拡張法の比較

ネーリンクを辿る分のホップ数が増加するが，ノード数が増加することによって迂回経路数が減少し，迂回経路を利用する割合が減少したためである．

また，拡張法による効果を計測するため，N を 128 から 1024 まで変化させ，任意の 2 ノード間の全ての経路の中で，拡張法によって迂回経路内に目的地ノードが存在する経路を対象として計測を行った．図 15 に欲張り法と拡張法のホップ数の平均を示す．図中に示すように拡張法を利用することによって，すべてのグラフにおいて欲張り法よりも約 2 ホップ程度のホップ数の短縮が確認できる．

Note: 但し，迂回経路内に目的地ノードが存在するような短縮経路数は, 最も多い $(N = 256, r = 0.1)$ の場合で全経路数の 0.003 程度であるため，実利用上で大きい効果は得られないと考えられる．これは，今後の課題としたい．

6. 関 連 研 究

無線環境を考慮したネットワーク上での経路選択は，スケーラビリティを保持するため，フラッディングの利用を避ける必要がある．そこで，ID やノード位置に基づくネットワークの構造化や対象範囲の制限をした通信によって，ネットワーク全体の通信量を低減しつつ，任意のノード間に対して到達可能な経路選択法が研究されている．

Karp ら [1] は，ノードの位置関係によってガブリエルグラフを構築し，欲張り法及び Right-Hand-Rule を利用することで，フラッディングを利用せずに，平面グラフ上で任意のノード間での経路選択を保障している．一方，提案手法では，構成したドロネーリンクを辿ることによって任意の 2 ノード間で経路選択を行うことができる．Li 及び Wang ら [2], [6] は，通信範囲内のノードに焦点を当て，UDG 上から平面局所化ドロネー図と呼ばれる平面グラフを提案し，同構造を利用した欲張り法による経路選択について議論している．これに対して提案手法では，通信範囲外のノードには迂回経路によって到達可能なリンクを構成できるため，グラフが連結であれば，通信範囲に依存せずネットワークを構築することができる．

Liao [7] らの手法では，平面を格子状に分割し，各格子で代表ノードを割り当て，代表ノードのみが格子間で経路情報を交換し，格子内ノードのみに取得した経路情報をマルチキャストする．これによって，全格子内の任意のノード間で経路選択を可能にしている．しかし，この手法は各格子に割り当てられた代表ノードに負荷が集中する．

また，Caesar ら [8] らは，Chord 環を利用しており，各ノードがノード ID の近いノードを隣人表として所持し，ID 空間上で近いノード ID を順番に辿ることで ID のみによる経路選択法を実現している．また，ID 空間上の隣人がノードの通信範囲外にある場合には，マルチホップ通信による隣人までの経路を所持することで対応している．しかしながら，ノード数の増加に伴い ID 空間も大きくなるため，ノード数が大きい場合には隣人までのホップ数が増加する．一方，提案手法では，ノード数の増加に対して所持する隣人数は一定であり，拡張性が高い．

7. お わ り に

本稿では，迂回経路の生成法及び縮退法について述べ，無線環境におけるドロネー図の生成法について述べた．また，提案ネットワークの利用法として欲張り法およびその拡張法による経路選択を行った．さらに，数値シミュレーションによって各ノードの維持負荷および利用効果について評価を行い，任意の 2 ノード間の経路選択において到達可能であることを示し,MANET 環境での応用可能性を確認した．今後は，ノードの参加・離脱時におけるネットワークの再構築にかかる負荷の評価，実環境に応じたパラメータの設定による評価，及びネットワークシミュレータなどによる実パケット転送量の観点からの本ネットワークの性能評価などが考えられる．

文　　献

[1] B. Karp and H. T. Kung: “GPSR: greedy perimeter stateless routing for wireless networks”, MobiCom ’00: Proceedings of the 6th annual international conference on Mobile computing and networking, pp. 243–254 (2000).

[2] X.-Y. Li, G. Calinescu, P.-J. Wan and Y. Wang: “Localized delaunay triangulation with application in ad hoc wireless networks”, IEEE Transactions on Parallel and Distributed Systems, **14**, 10, pp. 1035–1047 (2003).

[3] 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, **47**, SIG4(TOD29), pp. 51 – 64 (2006).

[4] F. Araújo and L. Rodrigues: “Geopeer: A location-aware peer-to-peer system”, Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications (IEEE NCA04), Cambridge, MA, USA, pp. 39–46 (2004).

[5] P. Bose and P. Morin: “ISAAC: 10th international symposium on algorithms and computation”, ISAAC, pp. 113–122 (1999).

[6] Y. Wang and X.-Y. Li: “Efficient delaunay-based localized routing for wireless sensor networks: Research articles”, International Journal of Communication Systems, **20**, 7, pp. 767–789 (2007).

[7] W.-H. Liao, J.-P. Sheu and Y.-C. Tseng: “GRID: A fully location-aware routing protocol for mobile ad hoc networks”, Telecommunication Systems, **18**, 1-3, pp. 37–60 (2001).

[8] M. Caesar, M. Castro, E. B. Nightingale, G. O’Shea and A. I. T. Rowstron: “Virtual ring routing: network routing inspired by dhts”, SIGCOMM, pp. 351–362 (2006).