平成 23 年度　特別研究報告書

MANETを用いた災害時避難経路把握

龍谷大学　理工学部　情報メディア学科

学籍番号　T080421　清水　敬太

指導教員　三好　力　教授

内容梗概

2011 年の大災害のように、都市部で大規模な自然災害が発生した場合、多くの帰宅困難者が発生する。帰宅困難者の多くは情報を収集する為にスマートフォンのような携帯情報端末に搭載されている GPS 機能を利用して移動するが、利用者同士の情報共有は行われない。各個人で勝手な避難行動を行うと2次災害を招く恐れがある。

ここで、無線アドホック通信を用いて、携帯情報端末を保持する歩行者(ノード)の歩行記録を共有することで同期が完了した時点で歩行可能道路を中心に周辺の道路情報がわかるというものを提案する。各ノードは GPS などで常に自身のおおよその位置を把握できるものとして、隣接端末が保持する道路情報を交換・同期するものである。

実験としては、2 つのノードがお互いの移動記録を交換することにより、正しく道路情報マップが作成できるかを調べた。加えて、各ノードが道路情報を取得した後、この情報を用いて移動した場合障害物などを回避出きるのかという実験を行った。

結果、今回のシミュレーションでは回避出きることが分かった。

目次

第１章 はじめに

1.1 本研究の背景

2011 年 3 月 11 日 14 時 46 分頃に発生した宮城県牡鹿半島の東南東沖 130km の海底を震源として発生した M9 の東日本大震災[1]は首都圏で震度 5 強が観測され、鉄道各社がまったく動かなくなった。このことにより、都内で約 300 万人の帰宅困難者が出た。帰宅困難者の中には会社や知人の家などで夜を明かした者も居るが、約 10 万人は学校やホールなどの公共施設に泊まった。大災害時の東京都のこれまでの基本方針は「自力で帰ることを支援する」というものであったが、この方針が帰宅困難者の増加の要因の一部であったと考えられる。反対に被災者を受け入れる場合、どの建物が避難・休憩所として利用できるか、位置はどこにあるのかを知らせる必要性がある。また、携帯電話の世帯普及率(図 1 参照)から見るに、私たちの日常生活に携帯電話は切って離せないものとなっている。これから先、携帯電話はスマートフォンに代わってくると考えられる。(図 2 参照)今回の震災において使うことが可能であった携帯情報端末に関する既存技術を以下に示す。

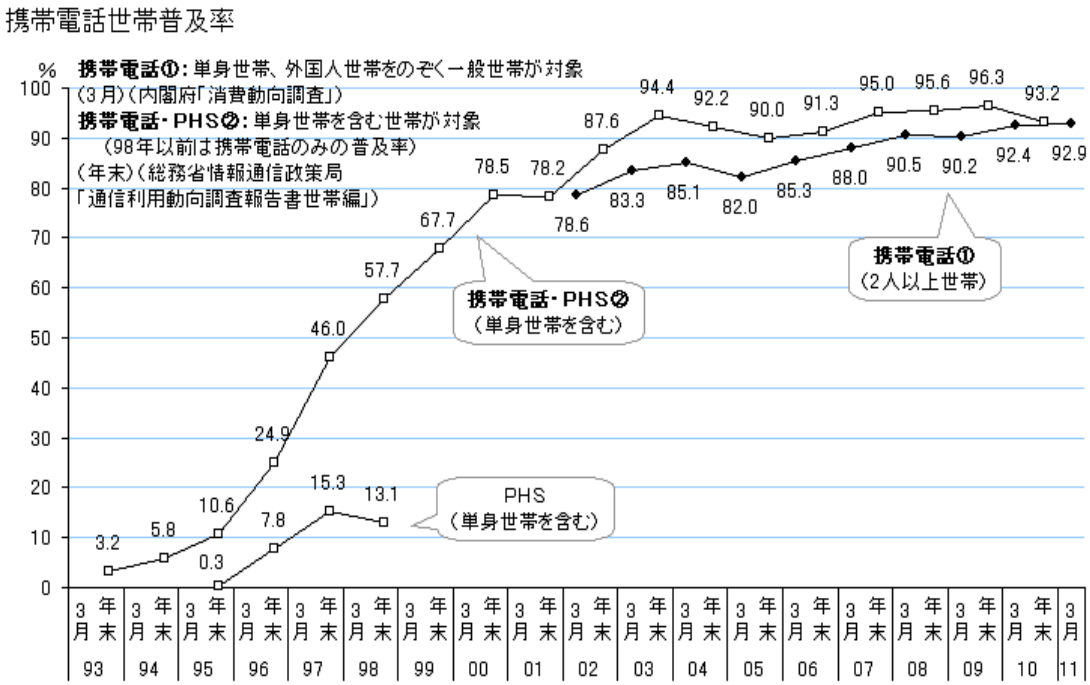

図１　携帯電話世間普及率[2]

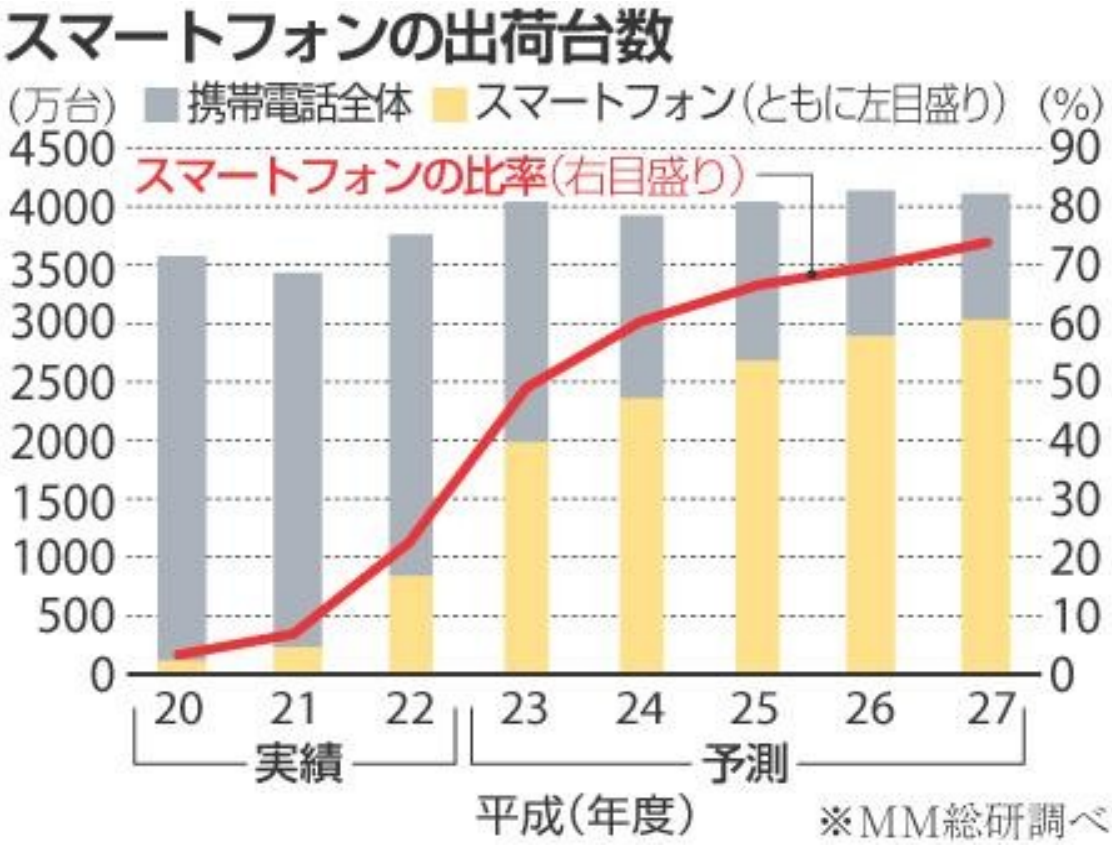

図 2　スマートフォンの出荷台数[3]

1.1.1　災害時ナビ

災害時ナビ[4]とは、au・docomo・softbank でサポートされている携帯電話で災害時に、自分の位置情報を把握して、避難場所までのルートを案内することが出来るアプリケーションである。しかし、使える端末が限られており、携帯電話のサービスエリア内かどうかで GPS が使えないことがある。さらに、自分の位置情報を把握できない場合は避難所マップなどの地図情報だけを見ることが出来るものである。

図 3　災害時ナビのサンプル画面

1.1.2　震災サポート

震災サポート[5]とは、NAVITIME の提供する機能で PC や携帯電話、iphone、android 端末で利用することが出来る。具体的に扱われる情報は鉄道運行情報、乗換案内、病院検索、電車混雑リポート、走行ログマップ(2011 年 10 月 24 日現在。鉄道運行情報、乗り換え案内以外の機能はプラットフォームによって参照できるかの違いがある)である。走行ログマップについては、車両が 1 日以内に通行したかと、1 週間以内に通行されたかどうかがわかる。今後、歩行者用の「EZ ナビウォーク」や「au one ナビウォーク」などに対応される予定である。

図 4　震災サポートのサンプル画面

1.1.3　HONDA インターナビ

HONDA インターナビ[6]とは、Honda が提供するサービスで、「全国の VICS 情報」と「フローティングカーデータ」をインターナビ情報センターで統合し、Honda 独自の「インターナビ交通情報」がつくりあげられる。ユーザはデータ通信によって渋滞予測情報を受け取り、ルート案内などに活用する。ここで、VICS 情報とフローティングカーデータについての簡単な説明をする。VICS 情報とは、(財)道路交通情報通信システムセンターが配信している、主要な幹線道路や高速道路を対象としてリアルタイムの交通情報(渋滞・事故・交通規制)で、カーナビは VICS センターで編集・処理されたデータを「FM 多重放送」「電波ビーコン」「光ビーコン」によって受け取りルート案内に活かすことができる。

フローティングカーデータとして扱うデータはどのクルマがどこを通ってどこへ行ったかというデータの集合である。インターナビ搭載車やスマートフォンなどで利用できるインターナビ会員の走行データから車線別情報まで扱うことができるので VICS 情報からでは案内できない距離的ではなく時間的最短ルートへ案内することが可能である。

図 5　フローティングカーデータでの道路情報把握

また、この 2011 年 3 月 11 日の東日本大震災が発生した時、インターナビ事業室の本拠である和光も震度 5 強の地震が襲った。Honda は、被災地における通行可能な道路の参考情報として、インターナビの会員の車両からの通行実績情報を GoogleEarth 上に最新版を公開した。Honda の東日本大震災でのインターナビによる取り組み「通行実績マップ」が「2011 年度グッドデザイン大賞」を 2011 年 11 月 9 日に受賞した。審査員の評価は以下のようなものであった。[引用 7]

会員の走行データを活用した最適ルート案内を2003年から Honda 独自の双方向通信型「インターナビ」として実施していた、その通行実績情報が東日本大震災の翌日から威力を発揮したことは記憶に新しい。被災地に向かう支援物資の車両、ボランティア団体、被災関係者などはこのオープン化された情報にどれほど助けられたことだろう。Google はこの通行実績情報データを「Google 自動車通行実績情報マップ」として3月14日から公開し、Yahoo も「Yahoo！地図」として4月21日から公開することにつながった。Honda は情報の正確さを求め、前日の24時間以内の情報のみを毎朝10時に更新するなど被災地に向かう人々へ提供し共有を続けた。その後、自動車各社もこれに追従し、情報を提供し始める。独自に進めていた双方向通信型カーナビによる 通行実績情報が大震災時に応用できると誰が予測しただろう。この実績をもとに世界の震災時に役立てられるように展開して頂きたい。長年の企業努力と震災後のサービス提供に対して、担当した審査委員一同は最も高い評価をしました。

1.1.4　モバイル空間統計

モバイル空間統計[8]とは、NTT と東京大学の共同研究で行われている携帯電話サービスを提供するために必要な各基地局のサービス範囲内の端末数などの運用データから、人口の地理的な分布(図 6 参照)を推計した統計情報のことである。現在行われている研究の一例としては、モバイル空間統計を使って、千葉県柏

市の季節・曜日・時間による属性別の人口変動を推計し、同市内の公共施設などの利用実態を推計した人口変動との関連性を分析し、都市空間の人口変動というデータを活用することで公共施設の利活用や機能転換などへの効果的な提案に結びつけようとするものである。

また、5 月 25 日、ワイヤレスジャパン基調講演で NTT ドコモ代表取締役社長山田隆持氏が行ったプレゼンテーションにおいて震災当日およびその後の帰宅困難者の統計情報を可視化できたということが実証され、2011 年内の NTT としての災害対策が紹介された。

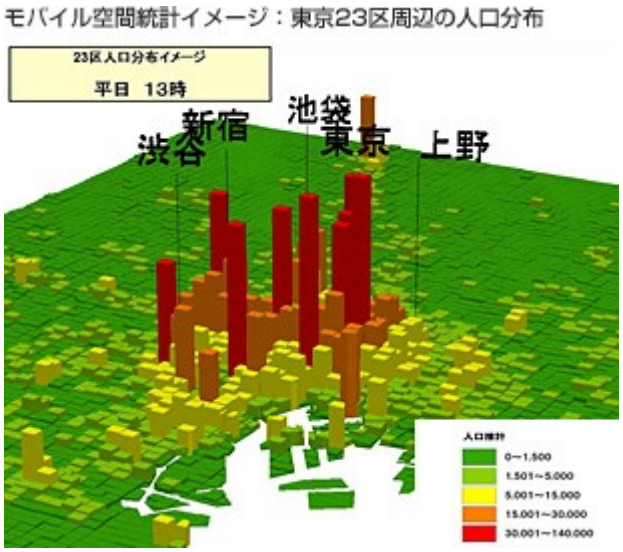

図 6　モバイル空間統計による人口分布の可視化

1.2 問題点

1.1 説で述べた既存技術にはいくつかの問題点がある。まず、携帯電話(フィーチャー・フォン、日本国内ではガラパゴスケータイと呼ばれるもの)で利用可能なサービスはインターネットに接続できる環境ですべての仕様を使える。しかし、大規模災害になると基地局の停電などにより電話やインターネットが使えなくなり、情報がリアルタイムに入ってこなくなる。したがって、被災者の災害についての情報取得が基地局の停電や故障などの状態に依存するという問題がある。

また、既存技術ではインターナビのように利用者のフローティングカーデータを一度メインサーバに集約し、編集、配信という手順を踏んで利用者に提供するためインターネットに問題なく繋がることができても処理を行うサーバ側に依存するという問題がある。加えて、現行の震災ナビやインターナビは自動車のナビシステムの応用であるため、歩行者には役に立たない。

また、災害の初期段階で被災地ごとの人々の間で避難所マップ(潜在的な危険場所と避難所の位置が記されたもの)や現在の危険場所・避難所の混雑状況等、現在必要とされている被災地の情報を必要なときに伝えることができるかという、サービスのレスポンスの面での問題がある。

以上のような問題が起きた場合、災害情報の無いまま各個人で勝手な避難行動を行い2次災害を招く恐れがある。

1.3 研究目的

本論文ではスマートフォンの GPS 機能を使い自身の歩んできた道を周辺の利用者とアドホック通信で共有することで、利用者周辺の道路情報把握でき、避難所等への現在歩行可能な道路情報がわかるようなシステムを提案する事を目的とする。

帰宅困難者の多くは情報を収集する為にスマートフォンのような携帯情報端末に搭載されている GPS 機能を利用して移動するが、既存サービスでは把握された GPS 位置情報による歩行履歴を利用しない。また、フローティングカーデータを利用するサービスでも利用者同士が直接通信しデータを共有するようなことを想定されていない。

ここで、無線アドホックネットワークは基地局を必要とせずに、端末同士で通信することが出来るので、GPS 情報による歩行情報をアドホック通信を用いてノード間共有することを考える。また、マルチホップ通信で全ノードと通信可能な状態にするという形式ではなく、次章の関連技術で述べる DTN の Store and forward という技術を用いてかつて受け取っていた他ノードからの情報も共に送ることで同期が完了した地点周辺の歩行可能な道を中心とした周辺の道路情報を知ることが出来るシステムを想定することで、より多くの有益な情報を各ノードに持たせることを考えた。

1.4 本論文の流れ

本論文は以下の章により構成されている。

第 1 章、研究内容の概要について述べている。

第 2 章、関連技術について述べている。

第 3 章、提案手法について述べている。

第 4 章、実験について述べている。

第 5 章、本論文に関するまとめと今後の課題について述べている。

第 2 章 関連技術

本章では提案手法に利用する関連技術の概要について述べる。

2.1 MANET の概要

MANET とはイーサネットや無線 LAN などのインフラ設備を必要とせず、無線で接続できる端末のみで端末相互の通信を実現する技術のことである。インフラ設備を必要としない為、端末さえ集まれば場所を選ばずに通信することが可能である。しかし、端末の移動性により、ネットワークトポロジーはその都度変化するもので、直接通信できない相手端末には図 7 の形態ようにマルチホップ通信を行うことで通信が可能である。また、周囲のどの方向の端末とも通信を可能とするため、アンテナは無指向性のアンテナを使用する。身近な利用例としては Nitendo のニンテドー DS や SONY の PSP がある。

また、Ad-hoc Network は自律分散型であり、移動中継型、ハイブリッド型、固定中継型の 3 種のネットワークの運用形態がある。(図 7 参照)

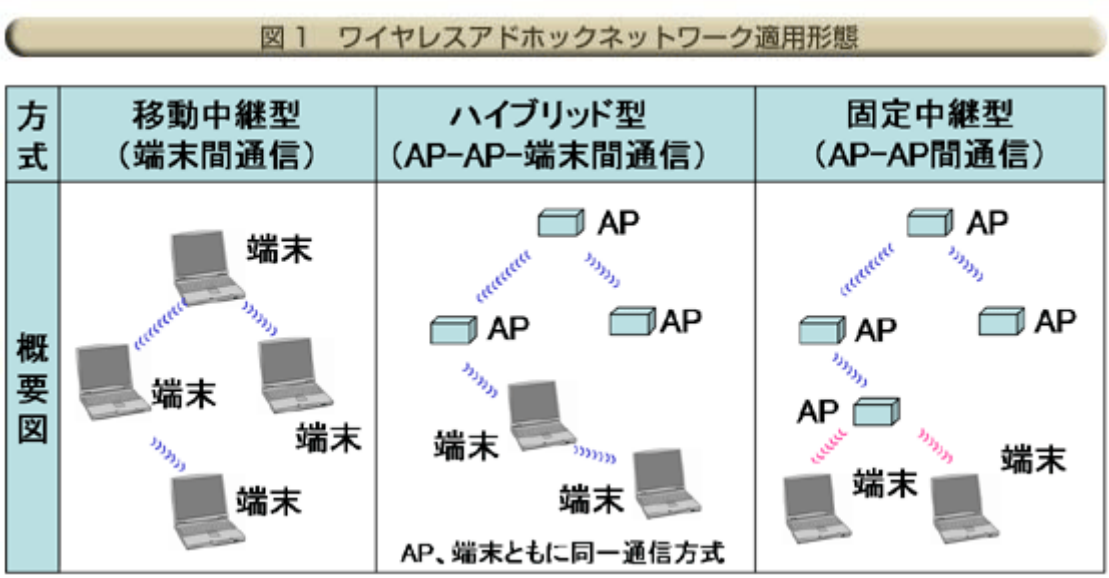

図 7　MANET の適用形態[引用 9]

NTT では少ない工事稼動でネットワーク構築が可能で、かつ安定なサービス提供が可能なアクセスポイント（AP）間をワイヤレスアドホックネットワークで構築する固定中継型あるいはハイブリッド型での運用を考えており、これらの高スループット化、VoIP・映像系アプリに対応するための QoS 制御、ハンドオーバ技術、またセキュリティ確保などの研究開発を行われている。

(1)高スループット化

単一 Ch でのワイヤレスアドホックネットワーク構築に無線 LAN（IEEE802.11）を使用する場合、隠れ端末問題、さらし端末問題によって

スルー プットが低下する。このため、複数 Ch 運用が有効で、自律的な Ch 割当、ルーティング、効率的なパケット転送方法が課題となる。

(2)QoS 制御

VoIP や映像情報送信などの利用が考えられ、ワイヤレスアドホックネットワークにおいても、優先制御・帯域制御技術などの QoS 制御が課題となる。

(3)ハンドオーバ

VoIP や映像情報送信などの利用シーンとして歩行程度の速度での移動を考慮した場合に、アクセスポイントをまたがって移動することも考えられることから、可能な限り品質劣化を抑えるためのハンドオーバ技術が必要となる。

(4)セキュリティ確保

ワイヤレスネットワークでは盗聴防止、端末のなりすまし防止、パケット改竄防止などのセキュリティ確保が必須であり、アドホックネットワークでは、さら に不正なアクセスポイントの設置が容易であること、また認証パケットなどを模擬した不正パケットの大量侵入により無線帯域が大幅に消費され通信が妨害され る可能性がある。このため、アクセスポイント自身の認証、不正パケットの検出・転送防止策も重要となる。

2.2 GPS

Global Positioning System[10]の略で、衛星を使った測位システムの一つである。測位システムの一般名称は、GNSS(Global Navigation Satelite Systems)で、「全地球測位システム」と訳される。

GPS はアメリカ合衆国が軍事用技術として実用化された技術で地上約 2 万 km のところを飛んでいる 24 機の GPS 衛星の中から上空にある数個の衛星から電波に乗せられた時刻情報を GPS 受信機で受信し計算することで、受信者が自身の地球上における位置(緯度、経度、高さ)を知るシステムである。近年は自動車（カーナビゲーションシステム）や携帯電話などに搭載され利用されている。携帯電話では精度によって様々だが、良いものならば30 M 以内の精度で求めることが出来る。

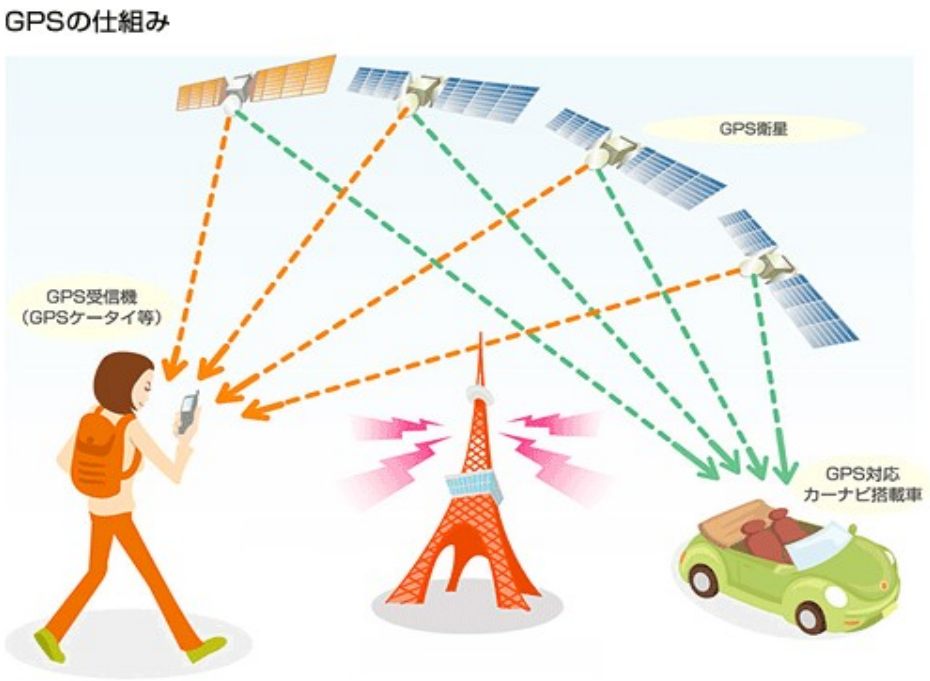

図 8　GPS の仕組み

2.3　DTN

delay/disruption-tolerant networking[11] の略で、Store and forward という交換方式を用いてデータ転送を行うもので、送信元から送信先までデータを送る際に中継地点でデータを保存しておき、通信可能になった時点でデータを転送する方式である。もともとは惑星間インターネットワークといわれる宇宙空間の通信手段の研究から生まれたものである。したがって、中断や切断が多発したり、大きな伝送遅延が生じたりする劣悪な通信環境でも、信頼性のあるデータ転送ができる。

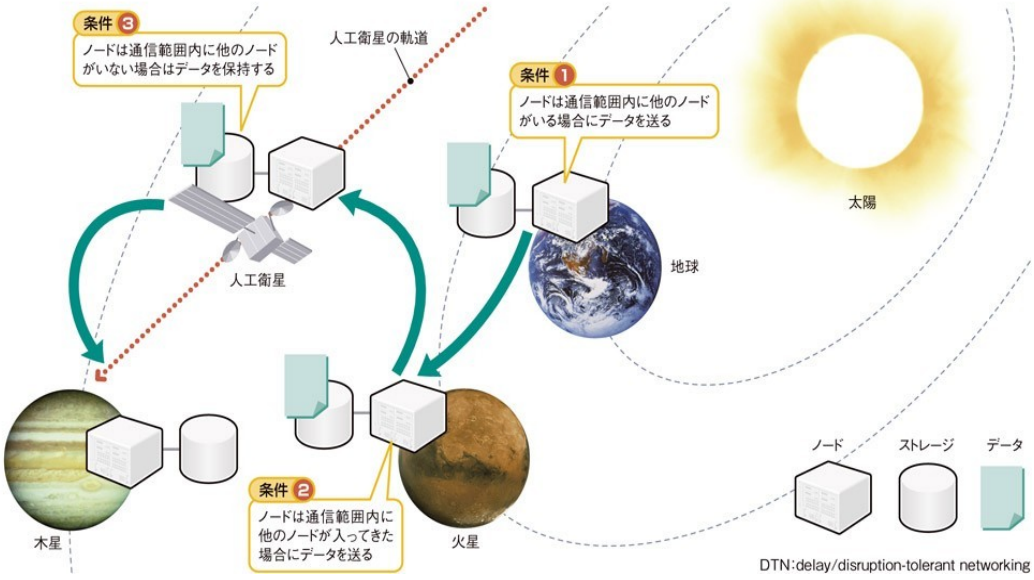

図 9　DTN によるデータ転送のしくみ

第 3 章 提案手法

本章では提案手法とこれに関するアルゴリズムについて述べる。

3.1 提案手法の概要

アドホック通信による通信で携帯情報端末が各々に情報を収集・処理できれば大規模災害発生直後で主に通信インフラの復旧までの周辺情報に乏しく知る術が無い状態のときであっても被災者の元へ情報を集めることが出来ると考える。今回は、必要な情報の中でも道路情報に着目する。

ここで提案する手法はフローティングカーデータのようなデータを歩行者向けに適応させ、通行履歴や地点ごとの歩行者の通行に関する問題などを記録しておくこと、および DTN のように通信で受け取ったデータを保持しておき通信範囲に携帯情報端末を保持する歩行者(以降、ノード)が入った場合に渡すというものである。具体的には、ノードが移動する事で歩行可能とわかった道路情報と別のノードが記録した道路情報を無線アドホック通信を用いて共有するものである。結果として同期が完了した地点周辺の歩行可能な道を中心とした周辺の道路情報を知ることが出来るシステムである。これは、現行の災害時ナビや震災サポートのような地図情報参照サービスには無い機能で、追加されると避難行動を行う際に役に立つ機能であると考えている。

3.2 提案手法のアルゴリズム

提案手法のシステムは大きく分けて 2 つのステップで構成されている。まず 1 つ目のステップでは各ノードは GPS 機能を使い自身のおおまかな位置情報を把握し、把握し始めた時点から単位時間ごとにベクトル方向でどちらに進んでいるかの移動記録を記憶しアドホック通信で近隣ノードと記録の共有をすることで、周辺の道路情報マップを作成するものである。

2 つ目のステップでは、1 つ目のステップで取得した道路情報マップを使い進行方向決定に利用する事で障害物や混雑を避けてより早く目的地に到着することができるものである。共有を行った後の各ノードの進行方向決定ついては各ノード自身がまだ訪れていない地点と、共有された道路情報マップで訪れられている地点では後者の方を優先的に進むようにする。

また、手に入れた道路情報を有効利用するために、通信範囲内のノードの集合の中からもっとも距離が近い順に交換するという通信を行う。

携帯情報端末は以下の機能を持つものと仮定する。

-IEEE802.11x 規格の無線 LAN 装置

-GPS

-経路情報を保存するのに十分な容量のストレージ

-地図データを納めたメディアとドライブ

提案手法を実現する為に必要なプログラムのアルゴリズムを次に示す。1 つ目の近隣ノードと現在保持している道路情報を共有する際に必要になるプログラムが下記の 1.である。2 つ目の道路情報マップを使って次の進行方向を決定することによって早く目的地につくプログラムが下記の 2.である。

1. プログラム 1
   1.1 開始
   1.2 GPS を用いて自身のおおまかな位置情報を把握し GPS 位置把握開始地点とする。
   1.3 目的地を設定する。
   1.4 通信半径内に通信可能なノードを検索する。
   1.5 1.4 でノードがあった場合、自身の GPS 位置把握開始地点と通信するまでに自身が動いた方向ベクトルを送信する。
   1.6 1.4 のノードが複数ある場合、ノードまでの距離が一番短いものと通信する。
   1.7 道路情報を受け取ると共有した道路情報用ストレージに道路情報を格納する。
   1.8 一定周期(以降、ターン)ごとに前位置からどの方向に進んだかの方向ベクトルを時間情報とともに通行履歴用ストレージに格納する。
   1.9 終了しない場合、1.4 に戻る。
   1.10 終了

2. プログラム 2
   2.1 開始
   2.2 GPS を用いて自身のおおまかな位置情報を把握し GPS 位置把握開始地点とする。
   2.3 目的地を設定する。
   2.4 通信半径内に通信可能なノードを検索する。
   2.5 2.4 でノードがあった場合、自身の GPS 位置把握開始地点と通信するまでに自身が動いた方向ベクトルを送信する。
   2.6 2.4 のノードが複数ある場合、ノードまでの距離が一番短いものと通信する。
   2.7 道路情報を受け取ると避難経路用ストレージに道路情報を格納する。
   2.8 共有することで取得した道路情報を元に次の方向を決定する。

2.9 ターンごとに前位置からどの方向に進んだかの方向ベクトルを時間情報とともに通行履歴用ストレージに格納する。

2.10　終了しない場合、2..4 に戻る。

2.11　終了

第 4 章 実験

4.1　実験概要

提案手法の有効性を確認するため、移動記録をとるノードを 2 つ配置してすれ違った時にお互いの移動記録を交換する事により、正しく道路情報が作成できるかを調べる実験 1 と、作成した道路情報を元に次に進む方向を変化させた場合、障害物を回避できるかの実験 2 を行う。

4.2　実験方法とシミュレータのアルゴリズム

実験環境としては以下の表のパラメータを用いる。

表 10：シミュレーションパラメータ

フィールドサイズ	100×100
ターン数	1000
位置情報確認	1 ターンごと
通信半径	10

想定した地図は 100×100 で 10000 のセルを設け、各ノードがどのセルに入っているのかを考える。加えて地図には縦、横ともに10セルおきに 1 セルの道が通っていて、道上にランダムに 6 つの障害物を設定しているものとする。

ノードについては、各ターンごとに常にいずれかの方向に移動した状態であるとし、10 ターンおきに通信範囲に入っているノードを探しているものとする。提案手法では各ノードは目的地を設定するが今回のシミュレーション実験では目的地をシミュレーション範囲外ということにして目的方向を決定することで、交差点での転回方向の重みにしている。つまり、目的の方角に対して曲がりやすいということである。

また、通信に関しては通信可能範囲内のノード全てに対し、距離を測ることによって最短距離に位置するノードと共有する。加えて、共有するデータを現在までに自身が進んできた歩行履歴とする。

4.2.1　実験 1

通信範囲に入ると各ノードは各々の道路情報の共有を行う。ノードの交換で得た道路情報が正しく自身の道路情報として作成できるかを調べる実験を行う。なお、3 章のプログラム1は実験 1 のアルゴリズムの 7.のステップで行う。

実験1のアルゴリズム

1. シミュレーションスタート
    1. 開始。
    2. モデルサイズ分のメモリを確保する。
    3. 地図をシミュレータに読み込む。
    4. ターン数を設定する。
    5. ノード関係の設定を行う。
        1. 各ノードの初期位置と目的方向を設定する。
    6. 通信範囲を決める。
    7. ターンの設定回数分繰り返す。
        1. ノード回数分繰り返す。
            1. ノードがフィールド設定の外にでていないかを確認する。
            2. 各ノードと通信できるかどうか確認する。
                1. 通信できな場合は 2.1 をスキップし、できる場合は処理を実行する。
                    1. 相手が複数いた場合の処理。
                        1. 一番近いノードと情報を交換する。
                    2. 通信相手と道路情報を交換しストレージに記憶する。
                2. 前回のターンの自分の GPS 位置を把握する。
                    1. 自分の目的方向を確認し優先度に応じた転回等の行動をする。
                    2. 結果を時間情報とともにストレージに記憶する。
    8. 終了

4.2.2 実験 2

アドホック通信で共有した道路情報をもとに地図を作成し、地図情報を自身の進行方向の決定に利用する事で障害物を回避出きるかを調べる実験を行う。なお、3 章のプログラム 2 は実験2のアルゴリズムの 7.のステップで行う。

実験 2 のアルゴリズム

2. シミュレーションスタート
    1. 開始。
    2. モデルサイズ分のメモリを確保する。
    3. 地図をシミュレータに読み込む。
    4. ターン数を設定する。
    5. ノード関係の設定を行う。
        1. 各ノードの初期位置と目的方向を設定する。
    6. 通信範囲を決める。
    7. ターンの設定回数分繰り返す。
        1. ノード回数分繰り返す。
            1. ノードがフィールド設定の外にでていないかを確認する。
            2. 各ノードと通信できるかどうか確認する。
                1. 通信できな場合は 2.1 をスキップし、できる場合は処理を実行する。
                    1. 相手が複数いた場合の処理。
                        1. 一番近いノードと情報を交換する。
                    2. 通信相手と道路情報を交換する。
                2. 前回のターンの自分の GPS 情報を把握する。
                    1. 道路情報を持っていない場合 1.へ、持っている場合 2.へ進む。
                        1. 自分の目的方向を確認し優先度に応じた転回等の行動をする。
                        2. 交換して持っている道路情報を考慮した優先度に応じた転回等の行動をする。
                    2. 結果を時間情報とともにストレージに記憶する。
    8. 終了

4.3 実験結果

4.3.1 実験 1

100×100 フィールドでシミュレーション行った時にアドホック通信によって道路情報を得ることができた様子の一例を図11から図 13 で示す。これらの図は自身の GPS 位置によって把握した歩行履歴を O で表し、通信によって得られた部分を L、通信結果と自身の歩行履歴と重複した部分を＋記号で示してある。また、実線で引いてある部分は実際に道はあるが通ったノードが無い部分を示している。

通信できていない状態の時のノード A とノード B がマークをつけた地点で持っている道路情報を図 11、12 に示す。10ターン後アドホック通信ができ、取得した B の道路情報を A の歩行履歴と合わせたものを図 13 に示す。

図 11 を見ると左下に道が有るがまだ A が歩行していないため地図の左下部分の道路情報がなく実線が多い事が分かる。図 12 を見ると左上方向への歩行履歴はあるが右上部分の道路情報が無い事が分かる。図 13 では左下部分の実線以外は道路情報が把握できているので、道路情報が正しく作成出きていることが分かる。

今回提案した手法は GPS 位置把握開始地点とその位置からターンごとにどういったベクトル方向に進んだかを記録するため、同じ道を歩いていても GPS 位置によっては道路情報がずれる場合がある。今回の実験では、地図をセルに分けて事前にセルごとに緯度、経度の範囲を決めてあるということで実行したため、セル上の動きで道路情報を表すことが出来る。

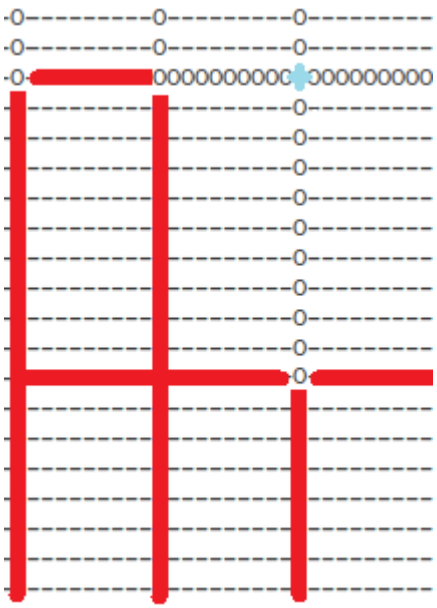

図 11 A の道路情報マップ

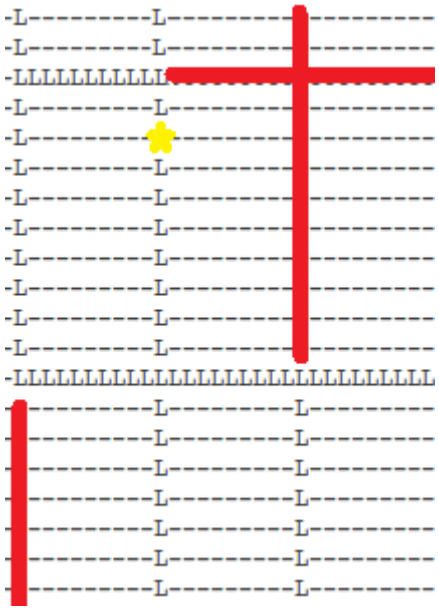

図 12 B の道路情報マップ

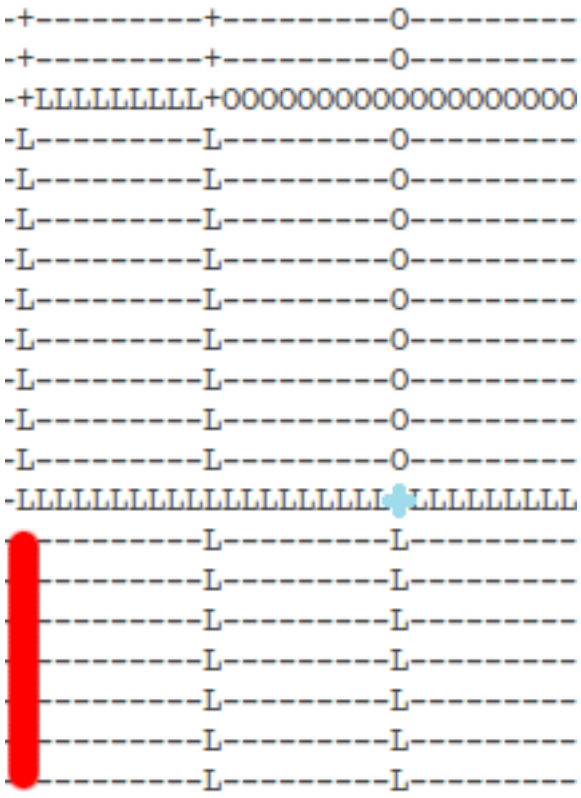

図 13 A の道路情報マップ

4.3.2　　実験 2

実験 1 で得られた道路情報を利用して自身の進行方向の決定に使っているかどうかを調べる実験を行った。実験結果の一例を図 14 に示す。これは、100×100 フィールドの中から変化のあった 40×70 範囲で抜粋した図である。この図は自身の GPS 位置把握開始地点を S、歩行履歴を O、障害物を B で表したもので、実線で引いてある部分は実際に道はあるが通ったノードが無い部分を示している。

一番上の丸印のある十字路の位置での処理で共有した道路情報を使った場合と使っていない場合でたどる経路が異なる。提案を用いていない場合は重み付けによる優先度の違いにより A の矢印で表した方向に進み、共有した道路情報を使う場合は同じく優先度の違いにより 1 の矢印で表した方向へ進む。進んだ方向の矢印は各時間毎で A と1、B と2、C と3、…といった対になって表している。今回のシミュレーション実験では道路情報を使わなかった場合に比べて障害物を 2 つ回避できたことが分かる。

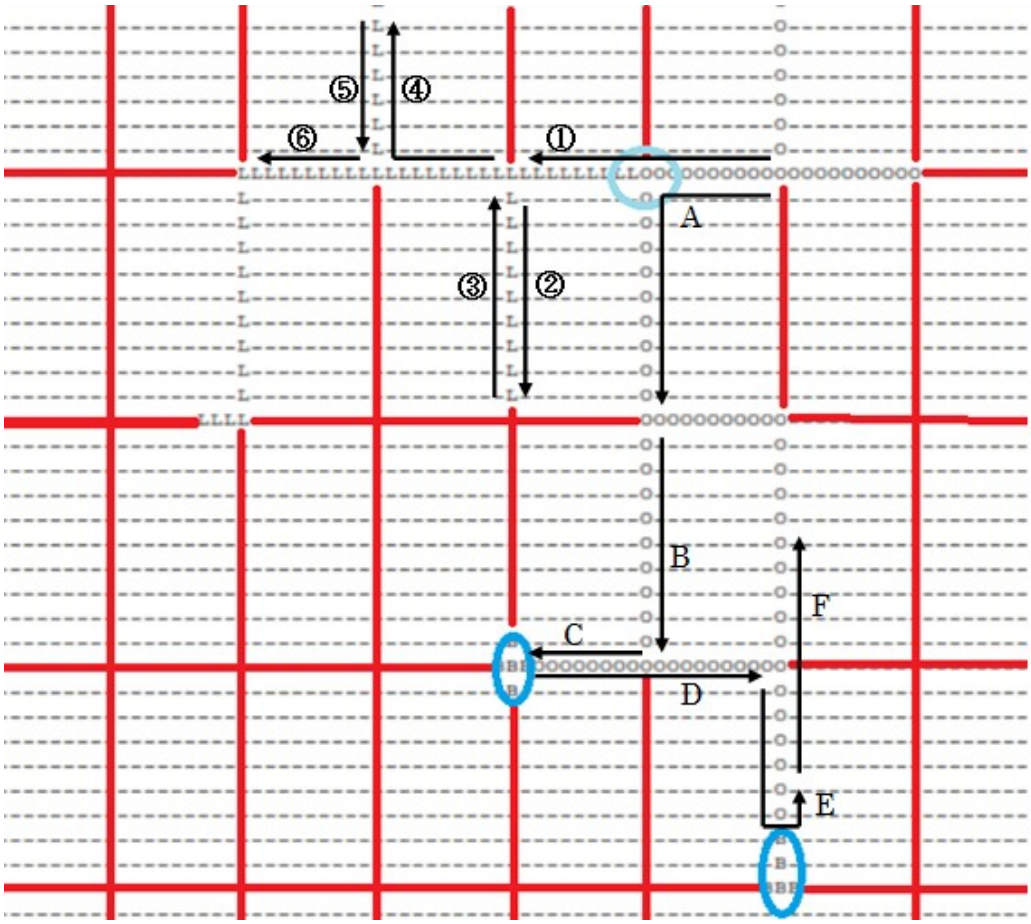

図 14　最終ターンにおける 2 つの通行履歴

今回の実験では一定距離(シミュレータにおいては道どうしの間隔に合わせて10セルで取っている)先まで道路情報を確認し障害物を示す道路情報が存在しない限りノードは重み付けによる優先度の違いによって進行方向を決定している ので障害物に向かってしまう可能性があるが、一定距離内に障害物を認識した場合その方向への行動を回避するため、目的地への到着は早くなる。

上記と同様の実験を 100 回行った結果を表 15 に示す。

表 15　実験 2 を 100 回試行した結果

	平均ステップ数	比較(%)
利用しない場合	750.5	100
利用する場合	482.2	64.3

表を見ると、提案手法によって平均 35.7％早く目的地へ到着できる事が分かる。

第 5 章 おわりに

本論文では、現行の地図情報参照サービスには無い機能であるスマートフォンの GPS 機能とアドホック通信で各ノードの歩行履歴を共有することで、利用者が周辺の道路情報を把握でき、避難所等への現在歩行可能な道路情報が分かるシステムを提案した。

提案手法はアドホック通信で携帯情報端末が各々に情報を収集・処理することで通信インフラの復旧までの間に被災者に情報を提供することができ、避難行動時の 2 次災害を回避するための手助けになることを想定している。提案手法が地図情報に着目したため、共有した道路情報を自身の道路情報と合わせて確認できるかということに加え、道路情報を用いて障害物を回避することが出きるかを確認した。

実験結果としては、実験 1 で道路情報マップをマス区切りを設けることによって異なるノードが同じ道を歩いた場合に同じ結果が出るように考慮し、ノードと共有した道路情報を正しく持っている道路情報と共有出来たことを確認した。実験 2 で実験1で作成した道路情報を用いて行動制御をかけた場合、障害物を回避するので平均 35.7％早く目的地へ到着でき提案手法の有効性が分かる。

今後は実際に GPS や携帯情報端末を用いて道路情報作成して地図の上で確認すると歩行できる道路の上に合致しているかを実験する。

謝辞

本研究を進めるにあたり、ご指導いただいた龍谷大学理工学部情報メディア学科の三好力教授には、貴重なお時間を割いてアドバイスをしていただき深く感謝いたします。また、研究室の皆様にも議論を通じて様々な知識や示唆を得ることができました。ここに、深く感謝いたします。最後に、公私に渡り支えてくれた友人に深く感謝いたします。

参考文献

[1]. http://www.jiji.com/jc/v4?id=earthquakewalk0001(2011/10/12)

[2]. http://www2.ttcn.ne.jp/honkawa/6350.html(2011/10/13)

[3]. http://sankei.jp.msn.com/economy/news/110821/biz11082101310000-n1.htm(2011/10/26)

[4]. http://www.au.kddi.com/saigaiji_navi/(2011/10/26)

[5]. http://corporate.navitime.co.jp/topics/pr/201103/28_1716.html(2011/10/26)

[6]. http://www.honda.co.jp/internavi/about/floating/(2011/10/26)

[7]. http://www.g-mark.org/archive/2011/award-grand.html(2011/12/27)

[8]. http://www.nttdocomo.co.jp/corporate/disclosure/mobile_spatial_statistics/(201110/31)

[9]. http://www.ntt.co.jp/mirai/organization/organization0305.html(2011/10/24)

[10]. http://www.tdk.co.jp/techmag/knowledge/200501u/index.htm(2011/10/27)

[11]. http://itpro.nikkeibp.co.jp/article/Keyword/20090203/324080/(2011/10/31)

付録

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define Mapx 100
#define Mapy 100
#define NODE 2
#define LATE 1000
#define RANGE 10
int Random(int min,int max);
int main(){
  int i=0,j=0,t,v,b=0,m,r,f=0,sh,map[Mapx+1]
[Mapy+1],dir[NODE],posx[NODE][LATE],posy[NODE]
[LATE],who[NODE]
[NODE],cx,cy,x,y,max,n=0,count[2],mode,sum[NODE],dsum=0,cwho[NO
DE];
  int sx,sy,ct,st;
  float so,s[NODE][NODE];
  FILE *fp;
  FILE *fp2;
  FILE *fp3;
  FILE *fp4;
  FILE *fp5;
  FILE *fpmax,*fpmin,*fpcmax,*fpcmin;
  fp = fopen("map.txt","r");
  fp2 = fopen("log.txt","w");
  fp3 = fopen("error.txt","a");
  fp4 = fopen("memo.txt","a");
  fp5 = fopen("log_move.txt","a");
  fpmax = fopen("max.txt","a");
  fpmin = fopen("min.txt","a");
  fpcmax = fopen("cmax.txt","a");
  fpcmin = fopen("cmin.txt","a");
  srand((unsigned)time(NULL));
  count[0]=0;
  count[1]=0;
  max = Mapx+1;
  while((m=fgetc(fp))!=EOF){
    map[i][j]=m;
    j++;
    if(j==max){
      j=0;
      i++;
    }
  }
  for(i=0;i<Mapx;i++){
    for(j=0;j<Mapy;j++){
      x=map[i][j];
      if(x==48)map[i][j]=0;//fprintf(fp,"%d",0);
      else if(x==51)map[i][j]=3;//fprintf(fp,"%d",3);
      else if(x==50)map[i][j]=2;//fprintf(fp,"%d",2);
      else if(x==55)map[i][j]=3;//map==7
      else if(x==56)map[i][j]=8;//fprintf(fp,"%d",8);
      else printf("地図→%d %d\n",i,j);//printf("%d",map[i][j]);
    }
  }
  for(i=0;i<NODE;i++){
    for(j=0;j<NODE;j++){
      s[i][j]=0.000000F;
      who[i][j]=0;
      dir[i]=0;
      posx[i][0]=0;
      posy[i][0]=0;
      sum[i]=0;
      cwho[i]=0;
    }
  }
  fprintf(fp2,"NODE.ID position(X,Y) direction\n");
  for(i=0;i<NODE;i++){//初期地点+進む方向性決定
    while(1){
    posx[i][0]=GetRandom(0,Mapx-1);
    posy[i][0]=GetRandom(0,Mapy-1);
    cx=posx[i][0];
    cy=posy[i][0];
    if(map[cx][cy]==3)break;
    }
    dir[i]=GetRandom(1,4);
   fprintf(fp2,"%d   (%d,%d)   %d \n",i,posx[i][0],posy[i][0],dir[i]);
    }
  mode=0;
  r=RANGE;//通信半径
  for(t=1;t<LATE;t++){///ターン数の管理(1~600)
    for(i=0;i<NODE;i++){///ノード番号の管理
      if(posx[i][t-1]>=0||posy[i][t-1]>=0||posx[i][t-1]<Mapx||posy[i][t-
1]<Mapy){
                if(t%10==0){//10 タイミングごとに通信
                  for(j=i+1;j<NODE;j++){
                    if((so=sqrt((posx[i][t-1]-posx[j][t-1])*(posx[i][t-1]-posx[j][t-
1])+(posy[i][t-1]-posy[j][t-1])*(posy[i][t-1]-posy[j][t-1])))<r){
                      s[i][j]=so;
                    }//通信範囲内に入った場合距離 so を保存
                    else s[i][j]=r+1;
                  }
                  if(i!=NODE-1){//ソート始まり
                    for(j=NODE-1;j>i;j--){
                      if(so>=s[i][j]){
                                so=s[i][j];
                      }
                    }//ソート終わり
                    for(j=NODE-1;j>i;j--){
                      if((s[i][j]==so)&&(so<r)){
                                if(mode==0){
                                  who[i][j]=t;
                                  who[j][i]=t;
                                  break;
                                }
                                else{
                                  who[i][j]++;
                                  who[j][i]++;
                                  break;
                                }
                      }
                    }
                  }//同期タイミング記入おわり
                }
                x=posx[i][t-1];
                y=posy[i][t-1];
                ////////////////////////////////////////////////////////////////////
                ///////////////          移動開始          ////////////////////
                ////////////////////////////////////////////////////////////////////
                if(map[x-1][y]==3&&map[x+1][y]==3&&map[x][y-
1]==3&&map[x][y+1]==3)sh=0;
                else if(map[x-1][y]==3&&map[x+1][y]==3&&map[x]
[y+1]==3)sh=1;
                else if(map[x][y-1]==3&&map[x-1][y]==3&&map[x+1]
[y]==3)sh=2;
                else if(map[x][y-1]==3&&map[x+1][y]==3&&map[x]
[y+1]==3)sh=3;
                else if(map[x][y-1]==3&&map[x-1][y]==3&&map[x]
[y+1]==3)sh=4;
                else if(map[x-1][y]==3&&map[x][y+1]==3)sh=5;
                else if(map[x][y-1]==3&&map[x+1][y]==3)sh=6;
                else if(map[x-1][y]==3&&map[x][y-1]==3)sh=7;
                else if(map[x][y+1]==3&&map[x+1][y]==3)sh=8;
                else if(map[x-1][y]==3&&map[x+1][y]==3)sh=9;
                else if(map[x][y-1]==3&&map[x][y+1]==3)sh=10;
                else if(map[x+1][y]==3)sh=11;
                else if(map[x][y-1]==3)sh=12;
                else if(map[x][y+1]==3)sh=13;
                else if(map[x-1][y]==3)sh=14;
                if(dir[i]==1){
                  sx=x,sy=y;
                  if(t!=1){
                    for(j=0;j<NODE;j++){
                      if(sx==x&&sy==y){
                                if(i!=j){
                                  if(who[i][j]!=0){
                                    ct=who[i][j];
                                    for(st=0;st<ct;st++){
                                      if(x==posx[j][st]&&y+1==posy[j][st]){
                                                sy++;
                                                break;
                                      }
                                    }
                                    if(sy==y){
                                      for(st=0;st<ct;st++){
                                                if(x-1==posx[j][st]&&y==posy[j]
[st]){
                                                  sx--;
                                                  break;
                                                }
                                      }
                                    }
                                  }
                                }
                      }else break;
                    }
                    f=0;
                    for(v=1;v<=t;v++){
```

```
                if(posx[i][v]==x&&posy[i][v]==y)f++;//訪れたことがあるか
のチェック
              }
              if(sh==0){
                if(f%4==0) y++;//fprintf(fp3,"%d ",1);//←
                else if(f%4==1) x--;
                else if(f%4==2) y--;
                else x++;
              }
              else if(sh==1){
                if(f%4==0||f%4==3) y++;
                else if(f%4==1) x--;
                else if(f%4==2) x++;
              }
              else if(sh==2){
                if(f%4==0||f%4==3)x--;
                else if(f%4==1)y--;
                else if(f%4==2)x++;
              }
              else if(sh==3){
                if(f%4==0||f%4==3)y++;
                else if(f%4==1)y--;
                else if(f%4==2)x++;
              }
              else if(sh==4){
                if(f%4==0||f%4==3)y++;
                else if(f%4==1)x--;
                else if(f%4==2)y--;
              }
              else if(sh==5){
                if(posy[i][t-2]-1==posy[i][t-1])x--;
                else if(posx[i][t-2]+1==posx[i][t-1])y++;
              }
              else if(sh==6){
                if(posy[i][t-2]+1==posy[i][t-1])x++;
                else if(posx[i][t-2]-1==posx[i][t-1])y--;
              }
              else if(sh==7){
                if(posy[i][t-2]+1==posy[i][t-1])x--;
                else if(posx[i][t-2]+1==posx[i][t-1])y--;
              }
              else if(sh==8){
                if(posy[i][t-2]-1==posy[i][t-1])x++;
                else if(posx[i][t-2]-1==posx[i][t-1])y++;
              }
              else if(sh==9){
                if(posx[i][t-2]+1==posx[i][t-1])x++;
                else x--;
              }
              else if(sh==10){
                if(posy[i][t-2]+1==posy[i][t-1])y++;
                else y--;
              }
              else if(sh==11) x++;
              else if(sh==12) y--;
              else if(sh==13) y++;
              else if(sh==14) x--;
              else fprintf(fp3,"型にハマらないデータ:%d time=%d",i,t);
            }
            //t=1 の時の処理
            else{
              if(sh==0||sh==1||sh==3||sh==4||sh==5||sh==8||sh==10||
sh==13)y++;
              else if(sh==2||sh==7||sh==9||sh==14)x--;
              else if(sh==11)x++;
              else if(sh==12)y--;
              else{
                if(rand()%2==1)y--;
                else x++;
              }
            }
            if(sx==posx[i][t-1]&&sy==posy[i][t-1]){
              posx[i][t]=x;
              posy[i][t]=y;
            }else{
              posx[i][t]=sx;
              posy[i][t]=sy;
            }
          }
          if(dir[i]==2){
            sx=x,sy=y;
            if(t!=1){
              for(j=0;j<NODE;j++){
                if(sx==x&&sy==y){
                  if(i!=j){
                    if(who[i][j]!=0){
                      ct=who[i][j];
                      for(st=0;st<ct;st++){
                        if(x==posx[j][st]&&y+1==posy[j][st]){
                          sy++;
                          break;
                        }
                      }
                      if(sy==y){
                        for(st=0;st<ct;st++){
                          if(x+1==posx[j][st]&&y==posy[j]
[st]){
                            sx--;
                            break;
                          }
                        }
                      }
                    }
                  }
                }else break;
              }
              f=0;
              for(v=1;v<=t;v++){
                if(posx[i][v]==x&&posy[i][v]==y)f++;//訪れたことがあるか
のチェック
              }
              if(sh==0){
                if(f%4==0) y++;//fprintf(fp3,"%d ",1);//←
                else if(f%4==1) x++;
                else if(f%4==2) y--;
                else x--;
              }
              else if(sh==1){
                if(f%4==0||f%4==3) y++;
                else if(f%4==1) x++;
                else if(f%4==2) x--;
              }
              else if(sh==2){
                if(f%4==0||f%4==3)x++;
                else if(f%4==1)y--;
                else if(f%4==2)x--;
              }
              else if(sh==3){
                if(f%4==0||f%4==3)y++;
                else if(f%4==1)x++;
                else if(f%4==2)y--;
              }
              else if(sh==4){
                if(f%4==0||f%4==3)y++;
                else if(f%4==1)y--;
                else if(f%4==2)x--;
              }
              else if(sh==5){
                if(posy[i][t-2]-1==posy[i][t-1])x--;
                else if(posx[i][t-2]+1==posx[i][t-1])y++;
              }
              else if(sh==6){
                if(posy[i][t-2]+1==posy[i][t-1])x++;
                else if(posx[i][t-2]-1==posx[i][t-1])y--;
              }
              else if(sh==7){
                if(posy[i][t-2]+1==posy[i][t-1])x--;
                else if(posx[i][t-2]+1==posx[i][t-1])y--;
              }
              else if(sh==8){
                if(posy[i][t-2]-1==posy[i][t-1])x++;
                else if(posx[i][t-2]-1==posx[i][t-1])y++;
              }
              else if(sh==9){
                if(posx[i][t-2]+1==posx[i][t-1])x++;
                else x--;
              }
              else if(sh==10){
                if(posy[i][t-2]+1==posy[i][t-1])y++;
                else y--;
              }
              else if(sh==11) x++;
              else if(sh==12) y--;
              else if(sh==13) y++;
              else if(sh==14) x--;
              else fprintf(fp3,"型にハマらないデータ:%d time=%d",i,t);
            }
            //t=1 の時の処理
            else{
              if(sh==0||sh==1||sh==3||sh==4||sh==5||sh==8||sh==10||
sh==13)y++;
              else if(sh==2||sh==6||sh==9||sh==11)x++;
              else if(sh==14)x--;
              else if(sh==12)y--;
              else{
                if(rand()%2==1)y--;
```

```
      else x++;
     }
    }
    if(sx==posx[i][t-1]&&sy==posy[i][t-1]){
     posx[i][t]=x;
     posy[i][t]=y;
    }else{
     posx[i][t]=sx;
     posy[i][t]=sy;
    }
   }
   if(dir[i]==3){
    sx=x,sy=y;
    if(t!=1){
     for(j=0;j<NODE;j++){
      if(sx==x&&sy==y){
                if(i!=j){
                 if(who[i][j]!=0){
                  ct=who[i][j];
                  for(st=0;st<ct;st++){
                   if(x==posx[j][st]&&y-1==posy[j][st]){
                              sy--;
                              break;
                    }
                   }
                  if(sy==y){
                   for(st=0;st<ct;st++){
                              if(x+1==posx[j][st]&&y==posy[j]
[st]){
                               sx--;
                               break;
                              }
                    }
                  }
                 }
                }
      }else break;
      }
     f=0;
     for(v=1;v<=t;v++){
      if(posx[i][v]==x&&posy[i][v]==y)f++;/
     }
     if(sh==0){
      if(f%4==0) y--;//fprintf(fp3,"%d ",1);//←
      else if(f%4==1) x++;
      else if(f%4==2) y++;
      else x--;
     }
     else if(sh==1){
      if(f%4==0||f%4==3) x++;
      else if(f%4==1) y++;
      else if(f%4==2) x--;
     }
     else if(sh==2){
      if(f%4==0||f%4==3)y--;
      else if(f%4==1)x++;
      else if(f%4==2)x--;
     }
     else if(sh==3){
      if(f%4==0||f%4==3)y--;
      else if(f%4==1)x++;
      else if(f%4==2)y++;
     }
     else if(sh==4){
      if(f%4==0||f%4==3)y--;
      else if(f%4==1)y++;
      else if(f%4==2)x--;
     }
     else if(sh==5){
      if(posy[i][t-2]-1==posy[i][t-1])x--;
      else if(posx[i][t-2]+1==posx[i][t-1])y++;
     }
     else if(sh==6){
      if(posy[i][t-2]+1==posy[i][t-1])x++;
      else if(posx[i][t-2]-1==posx[i][t-1])y--;
     }
     else if(sh==7){
      if(posy[i][t-2]+1==posy[i][t-1])x--;
      else if(posx[i][t-2]+1==posx[i][t-1])y--;
     }
     else if(sh==8){
      if(posy[i][t-2]-1==posy[i][t-1])x++;
      else if(posx[i][t-2]-1==posx[i][t-1])y++;
     }
     else if(sh==9){
      if(posx[i][t-2]+1==posx[i][t-1])x++;
      else x--;
     }
      else if(sh==10){
       if(posy[i][t-2]+1==posy[i][t-1])y++;
       else y--;
      }
      else if(sh==11) x++;
      else if(sh==12) y--;
      else if(sh==13) y++;
      else if(sh==14) x--;
      else fprintf(fp3,"型にハマらないデータ:%d time=%d",i,t);
     }
     //t=1 の時の処理
     else{
      if(sh==0||sh==2||sh==3||sh==4||sh==6||sh==7||sh==10||
sh==12)y--;
      else if(sh==1||sh==6||sh==8||sh==9||sh==11)x++;
      else if(sh==14)x--;
      else if(sh==13)y++;
      else{
       if(rand()%2==1)y--;
       else x++;
      }
     }
     if(sx==posx[i][t-1]&&sy==posy[i][t-1]){
      posx[i][t]=x;
      posy[i][t]=y;
     }else{
      posx[i][t]=sx;
      posy[i][t]=sy;
     }
    }
    if(dir[i]==4){
     sx=x,sy=y;
     if(t!=1){
      for(j=0;j<NODE;j++){
       if(sx==x&&sy==y){
                 if(i!=j){
                  if(who[i][j]!=0){
                   ct=who[i][j];
                   for(st=0;st<ct;st++){
                    if(x==posx[j][st]&&y-1==posy[j][st]){
                              sy++;
                              break;
                     }
                    }
                   if(sy==y){
                    for(st=0;st<ct;st++){
                              if(x-1==posx[j][st]&&y==posy[j]
[st]){
                               sx--;
                               break;
                              }
                     }
                    }
                   }
                  }
       }else break;
       }
      f=0;
      for(v=1;v<=t;v++){
       if(posx[i][v]==x&&posy[i][v]==y)f++;
      }
      if(sh==0){
       if(f%4==0) y=y-1;//fprintf(fp3,"%d ",1);//←
       else if(f%4==1) x=x-1;
       else if(f%4==2) y=y+1;
       else x=x+1;
      }
      else if(sh==1){
       if(f%4==0||f%4==3) x--;
       else if(f%4==1) y++;
       else if(f%4==2) x++;
      }
      else if(sh==2){
       if(f%4==0||f%4==3)y--;
       else if(f%4==1)x--;
       else if(f%4==2)x++;
      }
      else if(sh==3){
       if(f%4==0||f%4==3)y--;
       else if(f%4==1)y++;
       else if(f%4==2)x++;
      }
      else if(sh==4){
       if(f%4==0||f%4==3)y--;
       else if(f%4==1)y++;
       else if(f%4==2)x--;
      }
      else if(sh==5){
```

```
                if(posy[i][t-2]-1==posy[i][t-1])x--;
                else if(posx[i][t-2]+1==posx[i][t-1])y++;
               }
               else if(sh==6){
                if(posy[i][t-2]+1==posy[i][t-1])x++;
                else if(posx[i][t-2]-1==posx[i][t-1])y--;
               }
               else if(sh==7){
                if(posy[i][t-2]+1==posy[i][t-1])x--;
                else if(posx[i][t-2]+1==posx[i][t-1])y--;
               }
               else if(sh==8){
                if(posy[i][t-2]-1==posy[i][t-1])x++;
                else if(posx[i][t-2]-1==posx[i][t-1])y++;
               }
               else if(sh==9){
                if(posx[i][t-2]+1==posx[i][t-1])x++;
                else x--;
               }
               else if(sh==10){
                if(posy[i][t-2]+1==posy[i][t-1])y++;
                else y--;
               }
               else if(sh==11) x++;
               else if(sh==12) y--;
               else if(sh==13) y++;
               else if(sh==14) x--;
               else fprintf(fp3,"型にハマらないデータ:%d time=%d",i,t);
              }
              //t=1 の時の処理
              else{
               if(sh==0){
                if((rand()%2)==1)y=y-1;//←
                else x=x-1;//↑
               }else if(sh==1)x=x-1;
               else if(sh==2){
                if((rand()%2)==1)y=y-1;
                else x=x-1;
               }else if(sh==3||sh==4||sh==6||sh==7||sh==10||sh==12)y=y-1;
               else if(sh==5||sh==9||sh==14)x=x-1;
               else if(sh==13)y=y+1;
               else if(sh==11)x=x+1;
               else{
                if((rand()%2)==1)y=y+1;
                else x=x+1;
               }
              }
              if(sx==posx[i][t-1]&&sy==posy[i][t-1]){
               posx[i][t]=x;
               posy[i][t]=y;
              }else{
               posx[i][t]=x;
               posy[i][t]=y;
              }
             }
    }else{
             posx[i][t]=posx[i][t-1];
             posy[i][t]=posy[i][t-1];
    }
   }
  }
  for(i=0;i<NODE;i++){
   for(j=0;j<NODE;j++){
    if(i!=j){
             if(mode==0)printf("%d→ %d time %d ",i,j,who[j][i]);
             else printf("%d→ %d count %d ",i,j,who[j][i]);
             if(who[j][i]!=0)cwho[i]++;
    }
   }
   printf("\n");
  }
  t=0;
  n=0;
  //入力した番号のノードに対する情報
  for(v=0;v<NODE;v++){
   if(n!=v){
    t=who[n][v];
    if(t!=0){
             fprintf(fp4,"NODE_no: %d (x,y)=(%d,%d) t=%d\n",v,posx[v]
[0],posy[v][0],t);//受け取った元データの初期位置設定
             for(i=1;i<t;i++){
              if(mode==0){
               if(posx[v][i-1]==posx[v][i]&&posy[v][i-1]<posy[v]
[i])fprintf(fp4,"%d ",1);
               else if(posx[v][i-1]<posx[v][i]&&posy[v][i-1]==posy[v]
[i])fprintf(fp4,"%d ",2);
               else if(posx[v][i-1]==posx[v][i]&&posy[v][i-1]>posy[v]
[i])fprintf(fp4,"%d ",3);
               else if(posx[v][i-1]>posx[v][i]&&posy[v][i-1]==posy[v]
[i])fprintf(fp4,"%d ",0);
               else break;//fprintf(fp3,"型にハマらないデータ:検証部分\n");
              }
             }
             fprintf(fp4,"\n");
    }
   }
   else count[0]++;
  }
  //全ノードの情報
  for(v=0;v<NODE;v++){
   {
    fprintf(fp5,"NODE_no: %d (x,y)=(%d,%d) \n",v,posx[v][0],posy[v]
[0]);//受け取った元データの初期位置設定
    for(i=1;i<LATE;i++){
             if(posx[v][i-1]==posx[v][i]&&posy[v][i-1]<posy[v]
[i])fprintf(fp5,"%d ",1);
             else if(posx[v][i-1]<posx[v][i]&&posy[v][i-1]==posy[v]
[i])fprintf(fp5,"%d ",2);
             else if(posx[v][i-1]==posx[v][i]&&posy[v][i-1]>posy[v]
[i])fprintf(fp5,"%d ",3);
             else if(posx[v][i-1]>posx[v][i]&&posy[v][i-1]==posy[v]
[i])fprintf(fp5,"%d ",0);
             else break;//fprintf(fp3,"型にハマらないデータ:検証部分\n");
    }
    fprintf(fp5,"\n");
   }
  }
  fclose(fp);
  fclose(fp2);
  fclose(fp3);
  fclose(fp4);
  fclose(fp5);
  fclose(fpmax);
  fclose(fpmin);
  fclose(fpcmax);
  fclose(fpcmin);
}
int GetRandom(int min,int max){
 return min+(int)(rand()*(max-min+1.00)/(1.0+RAND_MAX));
}
```