卒業論文　　2002年度(平成14年度)

異種無線メディア間における経路制御機構

指導教員

慶應義塾大学環境情報学部

徳田　英幸

村井　純

楠本　博之

中村　修

南　政樹

慶應義塾大学環境情報学部

高橋 ひとみ

卒業論文要旨　　2002年度 (平成14年度)

異種無線メディア間における経路制御機構

無線端末の急速な普及に伴い, 既存の通信インフラを利用せず, 動的なネットワークを構築するモバイルアドホックネットワーク (MANET) が注目を集めている．MANETを構築するルーティングプロトコルは多く提案されているが，多くのプロトコルはMANETを構築するための前提条件として，構築するノードの無線メディアはすべて同じ規格であり，またその無線メディアはIP (Internet Protocol) アドレスの使用が可能なことが挙げられている．しかし現在多くの無線メディア規格が提案されており，集まったノードが異なる無線メディアを所持している場合が考えられる．送信元と宛て先の無線メディアが異なる場合，現在は互いにノードの通信ができない．また，無線メディアの中にはIPをサポートしないものもあり，IPアドレス使用を前提としてMANETを構築することが不可能な場合もある．

MANETを構築するルーティングプロトコルの多くで使用される最短ホップを選択する経路制御では，異種無線メディア間のアドホックネットワークには適応が難しい．なぜなら無線メディアは各規格で転送性能が大きく異なり，経路選択の指標をホップ数とする経路制御では，規格上転送性能が小さいホップ，また転送性能が大きいホップも1ホップと判別してしまうため，ホップ数でない指標を用いた経路選択が必要である．そこで，指標にホップ数ではなく他の指標を用いる経路制御機構が必要である．

本研究はまず，異なる無線メディア，もしくはIPアドレスを使用できない無線メディアをノードが所持していた場合でも，MANETを構築できるルーティングプロトコルであるPTR(Protocol Transport Routing)の設計と実装，評価を行った．評価では実際にPTRを用い802.11bとBluetoothのネットワークを構築し，TCP転送性能の評価を行った．802.11bのみのネットワークではホップ数が増えると急激にTCPの転送性能が低下してしまう．一方Bluetoothを経路に1ホップでも含んだ場合，ホップ数が2，3程度増加した際でも，TCP転送性能の著しい低下は見られなかった．

そして異種無線メディア間で重要となる，経路選択の指標にホップ数を用いない経路制御機構である，ETR(Estimated-TCP-throughput maximization Routing Scheme)を提案した．ETRは定期的にプローブメッセージを送信し経路のRTTとパケットロス率を取得することで，送信元から宛て先へ最もTCP転送性能が高いと予想される経路を選択する経路制御機構である．このETRを用い，同一無線メディアにおけるアドホックネットワークでTCP転送性能の評価を行った．ETRを用いることでTCP転送性能が向上すると判明した．

そして最後にPTRにETRを組み合わせた，効率の良い異種無線メディアにおけるアドホックネットワークのルーティングについて提案を行う．

慶應義塾大学　環境情報学部
高橋 ひとみ

Abstract of Bachelor's Thesis
A Routing Protocol for Ad Hoc Heterogeneous Network

Advances in technology of wireless communication bring attentions that a network is called mobile ad hoc networks (MANET). Though many routing protocols have been proposed for MANET, assumptions for most of these protocols are that the nodes have same wireless media, and these wireless media use Internet Protocol (IP) address. Recently many wireless media have been standardized, thus the nodes constructing MANET do not always have the same wireless media. If the wireless media of a source and a destination node are different, the source node is not able to communication with the destination node. In addition, the node with the wireless media which uses no IP addresses can not construct of MANET.

A Routing scheme that a node selects a shortest-hop route is not suitable for MANET constructed with different wireless media. The reason is that such the routing scheme makes a node regard as same 1 hop whether throughput of wireless standards is high or not. Thus MANET constructed with different wireless media needs a routing scheme which does not use a hop-metric but other metrics.

This paper presents Protocol Transport Routing (PTR) and Estimated TCP throughput maximization Routing scheme (ETR). PTR builds MANET with different wireless media without IP addresses. We planed, implemented, and evaluated PTR. In evaluations of PTR, We built MANET with 802.11b and Bluetooth, and evaluated PTR by TCP throughput. In MANET constructed with only 802.11b, TCP throughput was degraded sharply when a number of hops increased. On the other hand, in MANET constructed with 802.11b and Bluetooth, TCP throughput was not degraded sharply when 2 or 3 hops increased.

ETR is a routing scheme which does not use a hop-metric. In ETR, a source node measures an end-to-end delay to its destination as well as a collection of loss ratio for each traversal link. The source node estimates the maximum TCP throughput based on the first approximation with these measured values, and selects a path dynamically. We evaluated ETR in MANET constructed with same wireless media. In our testbed network, ETR improves TCP throughput.

Finally we suggest an effective Ad Hoc network routing protocol for different wireless media.

Hitomi Takahashi

Faculty of Environmental Information
Keio University

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究動機

近年無線通信技術の発達が急速に進んでいる．より高性能になった規格が次々と提案・実現され，高速なデータ転送ができる携帯電話の規格，IMT-2000 [1] の FOMA [2] もサービスが開始された．携帯電話や無線 LAN の規格である 802.11a [3], 802.11b [4], Bluetooth [5] 等も日常生活で目にしない日はないほど爆発的に無線端末の普及が進んでいる．そして無線機器端末が集まった時，機器同士で瞬時にデータのやりとりを行いたいとういう要求が出てきた．そのため従来の基地局を介した通信ではなく，既存の設備を必要とせず無線端末が動的にネットワークを構築するモバイルアドホックネットワーク (MANET) が注目を集めている．MANET ではすべてのノードがルータの機能を持ち，送信元から宛て先まで直接通信できない場合でも中間ノードがデータを宛て先まで転送することで，送信元は宛て先とデータの通信ができる．

現在 MENET を構築するためのプロトコルは多く提案されている．しかし DSR (Dynamic Source Routing) [6] や，AODV (Ad Hoc On-Demand Distance-Vector Routing) [7]，FSR (Fisheye State Routing Protocol) [8] など，そのほとんどのプロトコルは前提環境として，ネットワークを構築するノードの所持する無線メディアは同じであり，IP アドレスの使用が可能なことが挙げられる．しかし無線の普及にともない無線メディアの規格は多く提案されており，集まった全てのノードが所持する無線メディアが同じ規格であるとは限らない．またノードが所持する無線メディアに対し IP がサポートされていない場合，MANET を構築することはできない．例えば IP アドレスを使用しない携帯電話では MANET を構築できないと考えられる．

また異種無線メディア同士のアドホックネットワークを組んだ場合，多くの無線メディアが存在することになり，それぞれ規格毎に転送性能が異なる．そのため複数経路が存在した場合，経路選択が転送性能に強く影響を及ぼすことが予想される．現在 MANET を構築するルーティングプロトコルの多くで最短ホップを選択する経路制御が使用されている．もちろん同一の無線メディア間のアドホックネットワークにおいても，無線は有線と異なり電波品質が通信性能に影響を与えるため，最短経路が転送性能において最善な経路とは限らない．しかし異種無線メディアが存在するアドホックネットワークでは，先に述べたように規格毎に大きく転送性能が変化する．そのため経路選択の指標をホップ数にした場合，規格上転送性能が小さいホップ，また転送性能が大きいホップも 1 ホップとして同等に判別してしまうため，経路選択にホップ数以外の指標を用いる経路制御機構が必要である．そのため異種無線メディアにおけるアドホックネットワーク上では，指標にホップ数ではなく他の指標を用いる経路制

御機構が必要である．このような経路制御機構を用いることで，異種無線メディア間のアドホックネットワークを効率良く構築出来ると考えられる．

そこで異種無線メディアを所持する無線端末同士が集まった場合でも，MANET を構築出来るルーティングプロトコルである，PTR (Protocol Transport Routing) を提案する．PTR では IP をサポートしない無線メディアが存在することを考え，IP アドレスを用いず独自の識別子を使用する．また無線メディアの規格が異なる場合，相互に通信ができない．複数の無線メディアを所持したノードにデータを転送し，無線メディアの切替えを行うことで他の無線メディアとの通信が可能となる．次に PTR を用い異種無線メディアの MANET を構築した際，複数経路が存在した場合最短経路を使用しない経路制御機構が必要である．そこで経路選択の指標としてホップ数以外の経路を用い，かつ通信品質を考慮し動的な転送性能の変化にも適応できる経路制御である ETR (Estimated-TCP-throughput maximization Routing Scheme) を提案する．そして最後に PTR 上で ETR を用い，異種無線メディア間における効率の良いアドホックネットワークルーティングプロトコルである，PTER (Protocol Transport Effective Routing) の提案を行う．

1.2 本研究の目的及び意義

現在，無線メディアが日常生活に溢れている．この現実世界において瞬時にデータの共有や通信を行うために MANET を構築する場合，既存のルーティングプロトコルは同じ無線メディア同士だけのネットワークを構築する．また IP アドレスをサポートしていない無線メディアは MANET の構築すらできない．しかし PTR は異種の無線メディアが存在する MANET を構築でき，ユーザは無線メディアを意識せずに通信が可能となる．また IP アドレスを使用しないノードでも MANET に参加でき，より多くのノードが MANET を構築できるようになる．

ホップ数を指標とし最短経路を選択する単純な経路制御機構が，既存の MANET におけるルーティングプロトコルで多く採用されている．しかし最短経路を選択する単純な経路制御機構の場合，通信品質の変動が大きい無線環境では，必ずしも転送性能において最善な経路とはならない．また異種無線メディア間で MANET を構築した際，無線メディアの規格では転送性能が大きく変わってしまう．そこで ETR を用いることで通信品質の変動にも適応し，転送性能が高いと予想される経路へデータの送信が可能となる．そして PTR に ETR を動作させることで，より効率的な異種無線メディアによる MANET が構築できる．

1.3 本論文の構成

本論文は全 6 章から構成される．2 章で研究背景である無線端末の普及を述べ，MANET について説明を行い，既存の MANET におけるルーティングプロトコルについて簡単に説明する．そして 3 章で異種無線メディア間で MANET を構築するルーティングで

あるPTRの説明を行い，4章で転送性能が高いと予想される経路へのデータ送信を可能にする経路制御機構であるETRの説明をする．そして5章においてPTRとETRを組み合わせた機構であるPTERの提案を行い，6章で本論文のまとめと今後の課題について述べる．

第2章　背景

本章では研究の背景について説明をする．

2.1　無線端末の普及

本節では研究の背景となる無線端末の普及について述べる．今日端末の小型化や性能の向上が進み，端末の移動性が向上した．端末が移動した際にもネットワークに接続したいという要求に伴い，無線通信技術の発達が急速に進んだ．図 2.1 で示すように，無線端末を日常生活で目にしない日はないほど爆発的に普及が進んでいる．無線は無線ノードの移動や設置場所の制約をなくし，迅速な LAN の構築が出来るため，これからも無線の普及は進んでいくと予想される．

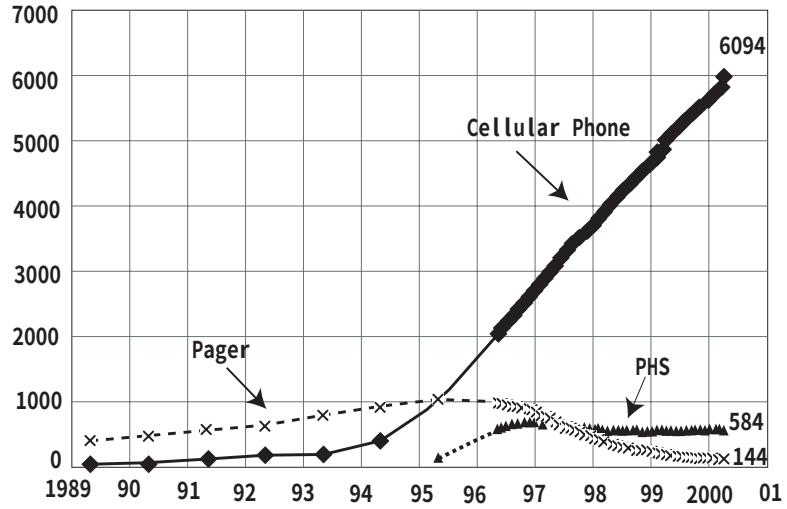

図 2.1: 無線端末の普及

また無線端末の普及に従い，高性能な無線規格も数多く提案されている．例えば高速なデータ転送ができる携帯電話の規格である，IMT-2000 [1] の FOMA [2] が挙げられる．また携帯電話だけでなく，多くの用途に適応する無線 LAN の規格も次々と提案・実現されている．例えばる 802.11a [3]，802.11b [4]，HiperLAN2 [9] などが提案，実現されている無線規格として挙げられる．そしてホームネットワークやケーブルの代わりとして使用される近距離無線 LAN の規格である IrDA [10] や HomeRF [11]，ま

た Bluetooth [5] も製品として目にするようになった.

2.2 アドホックネットワーク

前節で述べた様に無線端末の普及に従い，日常生活に無線機器が増えている．今日の無線ネットワーク体系は図 2.2 で示すように，無線端末から基地局までの 1 ホップのみが無線であり，基地局以降のネットワークは既存のインフラストラクチャの有線ネットワークを利用し，データ通信を行っている．

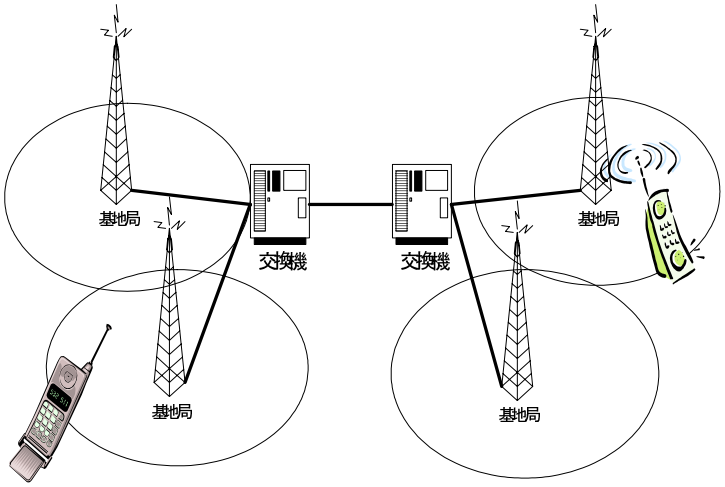

図 2.2: 従来の無線ネットワーク

しかし従来の無線ネットワーク体系の場合，既存のインフラストラクチャなしではデータ通信ができない．そのような問題を解決するために無線端末同士が既存のインフラストラクチャなしでも動的にかつ自律的にネットワークを構築できる，アドホックネットワークの研究が進められている．

アドホックネットワークは図 2.3 のように，無線端末だけで構築されるネットワークである．無線メディアには電波の届く伝送範囲があり，もし送信元から宛て先ノードまで直接電波の届かない距離にいた場合，中間ノードがルータとして機能し宛て先まで送信元ノードのデータを転送することで，送信元は宛て先ノードと通信が可能となる．またノードの移動によりデータ送信の経路が途切れた場合にも，経路の途切れを発見し動的に経路の再構築を行う．

このようなネットワークは既存のインフラストラクチャを必要としないため災害地や自動車ネットワーク，会議室でのデータ交換など，インスタントにネットワークを構築する場面に利用が期待されている．

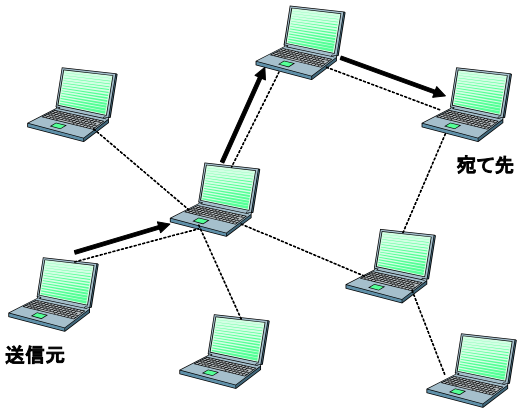

図 2.3: アドホックネットワーク

今日アドホックネットワーク構築するルーティングプロトコルや，それらのルーティングプロトコルを補助する機構は多く提案されているが，日常生活で使う場合は数多くの問題が考えられる．これらの問題を下に挙げる．

- アドレス割り当ての機構が無い

  MANET は自律分散なネットワークであり，有線ネットワークのように集中管理を行うサーバが存在せず，有線ネットワークに使用される DHCP の機構をそのまま適応できない．そのためユーザが独自にアドレスを割り当てなければならない．

- DNS の機構が存在しない

  先にも述べたように，MANET では集中管理を行うサーバを使用することが難しく，有線と同様な DNS の機構を用いることができない．そのため通信先のホストを指定する際，ホスト名でなく IP アドレスを直接指定しなければならない．

- 不安定な無線リンクである

  MANET を構築するネットワークは全て無線であり，リンクが非常に不安定である．またノードが移動するため有線ネットワークに比べてはるかにリンクの変更が多い．現在のネットワーク機構ではリンクの安定している有線ネットワークの環境を前提に設計された機構が多いため，MANET の環境には適さない．

- インターネットとの相互接続が出来ない

  MANET 内のノードと通信ができるが，通常の有線ネットワークへの接続ができない．

- セキュリティ対策が困難である

  MANET ではすべてのノードがルータの役割を果たしデータの転送を行えるため，セキュリティの対策が困難であると考えられる．データ転送を行う中間ノードでデータの盗み見や改竄が行われることが予想される．

先に述べた様な数多くの問題が挙げれる．しかしこれらを解決する機構は数多く提案されており [12, 13, 14, 15, 16]，近い将来に必要な技術になると考えられる．

本節で説明した MANET を構築するルーティングプロトコルは，全ての端末が同一な無線メディアを所持していることを前提としている．しかし前節で述べたように無線規格の多様化が進んでいるため，端末の所持している無線メディアが異なるという状況が考えられるようになった．そこで本論文では，無線規格の多様化を意識して設計された機構である PTR，ETR，PTER を提案する．

第3章　PTR

3.1　始めに

近年無線通信技術の発達が急速に進んでいる．より高性能になった規格が次々と提案・実現され，高速なデータ転送ができる携帯電話の規格，IMT-2000 [1] の FOMA [2] もサービスが開始された．携帯電話や無線 LAN である 802.11a [3], 802.11b [4]，Bluetooth [5] 等も日常生活で目にしない日はないほど爆発的に無線端末の普及が進んだ．そのため従来の基地局を介した通信体系ではなく，既存の設備を必要とせず無線端末が動的にかつ自律的にネットワークを構築するモバイルアドホックネットワーク (MANET) が注目を集めている．MANET ではすべてのノードがルータの機能を持ち，送信元から宛て先まで直接通信できない場合，中間ノードがデータを宛て先まで転送することで，送信元は宛て先とデータの通信ができる．

現在 MENET を構築するためのプロトコルは多く提案されている．しかし DSR (Dynamic Source Routing) [6] や，AODV (Ad Hoc On-Demand Distance-Vector Routing) [7]，FSR (Fisheye State Routing Protocol) [8] など，そのほとんどのプロトコルは，MANET に参加するノードはすべて同一の無線メディアを所持していることが前提である．またルーティングプロトコルが用いるノードの識別子は IP (Internet Protocol) アドレスであり，無線メディアは IP アドレスを使用可能なことが前提である．しかし多くの無線メディアの規格がある中，集まった無線ノードが同一でない無線メディアを所持している場合が考えられる．また，ノードが所持している無線メディアは IP アドレスをサポートしない規格である場合，既存のプロトコルでは MANET を構築できない．

異種無線メディアを所持する無線端末同士が集まった場合，IP アドレスを使用しない無線メディアが存在することを考え，IP アドレスを用いず独自の識別子を使用する必要がある．また無線メディアの規格が異なる場合，相互に通信ができない．そのため複数の無線メディアを所持したノードにデータを転送し，無線メディアの切替えを行うことで他の無線メディアとの通信が可能となるルーティングプロトコルが必要である．そこで，ノードの識別子として IP アドレスを用いずに MANET を構築する，異種無線メディア間におけるアドホックネットワークルーティングプロトコルである PTR (Protocol Transport Routing) を提案する．

3.2 本章の構成

本章の構成としてまず関連研究を紹介し，次に異種無線メディア間で MANET を構築できるルーティングプロトコルである PTR の設計と実装について述べる．最後に PTR を用い実際にネットワークを構築し評価を行い，その結論を述べる．

3.3 関連研究

本研究で提案する ETR と同様に多様なプロトコルに透過適な通信の実現を行う機構として，ETL (Extended Transport Layer) [17] が挙げられる．ETL においてアプリケーションは，既存のトランスポート層の代わりに ETL を使用する．ETL では ETL 層におけるホスト名，ポート名を割り当て，送信元ノードから宛て先ノードまで複数のノード間トランスポート通信により通信を行う．またトランスポート層におけるコネクションは，隣接ホスト毎にコネクションを確立していく．

ETL はトランスポート層に実装することで，PTR に比べ移植性やソフトウェアの軽量化が優れていると考えられる．しかし ETL は実装する層が PTR より上位層になるため PTR に比べオーバヘッドが高い．また ETL では隣接ノード毎にトランスポート層のコネクションを確立していくため，end-to-end での輻輳制御が不可能である．そのため宛て先ノードの帯域が極端に小さい場合，送信元ノードは宛て先ノードで輻輳が起きても分からずに，データの送信を続けてしまう．しかし PTR ではネットワーク層より下の層に存在し，トランスポート層のコネクションを end-to-end で確立できる．

3.4 PTR の設計

本節では異種無線メディアによる MANET を構築するルーティングプロトコルである，PTR の設計について述べる．

3.4.1 PTR の概要

PTR は異なる無線規格同士による MANET を構築できるルーティングプロトコルである．送信元が宛て先ノードまでの経路を動的に発見し，そしてノードの移動により経路が途切れた場合，経路の途切れを発見，経路の再構築を行う．また送信元が宛て先ノードまで直接通信できない距離にある場合，途中のノードが送信元から宛て先ノードまでデータの転送を行うことで，送信元は宛て先ノードとデータ送信が可能になる．

また無線規格が数多く提案される中，MANET を構築するノードがすべて同一の無線メディアを所持しているとは限らない．PTR はノードが異なる無線メディアを所持している場合でも MANET を構築可能なプロトコルである．例えば図 3.1 が示すように無線メディアが異なるノード A，ノード C でデータの送信を行うとする．ノード A は

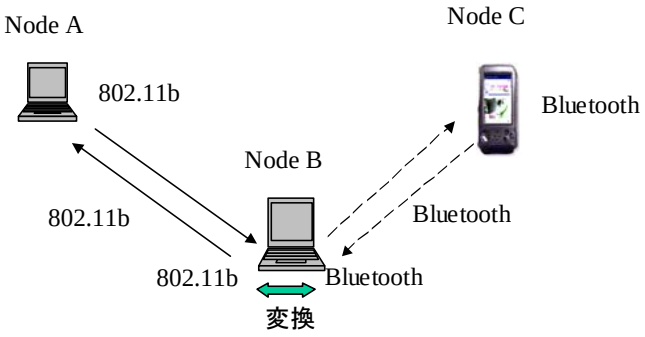

図 3.1: 無線メディアの切替え

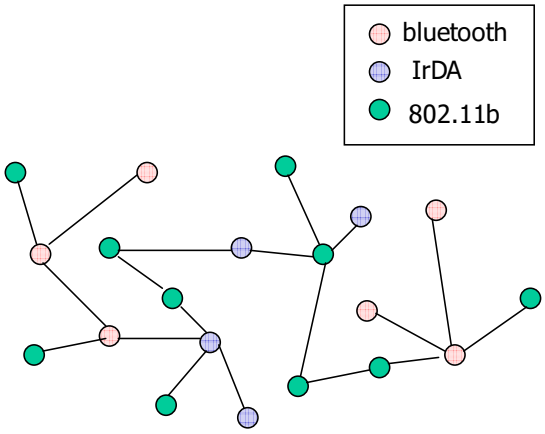

図 3.2: PTR を用いた MANET

802.11b，ノード C は Bluetooth の無線メディアのみを所持している．ノード A，ノード C は無線メディアが異なるため，もちろん直接通信はできない．しかし両方の無線メディアを持ったノード B にノード A，C はデータを送信し，ノード B は無線メディアを切替えデータをそれぞれの宛て先に対し転送することで，ユーザ空間の視点ではノード A とノード C は直接通信を行っているように見える．図 3.1 の動作を繰返してネットワークを構築することで，異種無線メディア同士の MANET が組めるようになる．また，ユーザの所持している無線メディアは必ずしも IP アドレスをサポートしているとは限らない．そのため PTR では IP アドレスに依るノードの識別を行わず，独自の ID を用い通信を行う．

PTR を用いることで図 3.2 の様な MANET を構築できる．ユーザは自分の所持して

いる無線メディアを意識することなしに透過的なネットワークを構築できる．

3.4.2 ノードのデータ送信手順

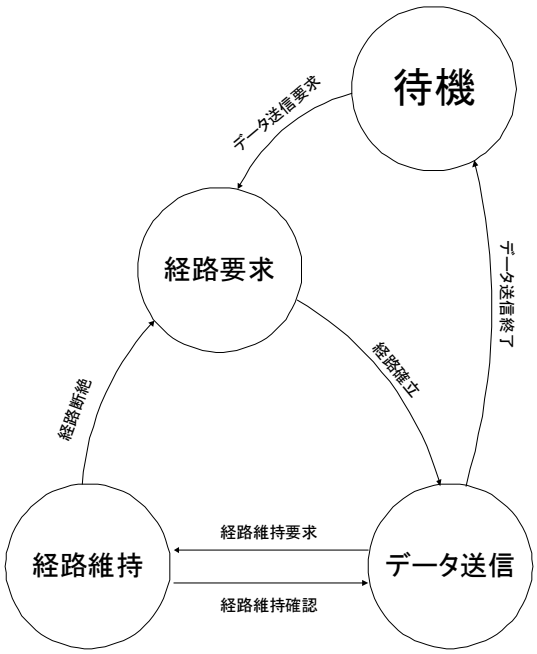

図 3.3: 初期状態遷移図

本小節ではPTRにおいて送信元ノードの具体的な送信手順を説明する．図 3.3 は送信元ノードの初期状態遷移を示している．まずノードはデータ送信の要求がない場合何も行わず待機する．上位層よりデータ送信の要求がPTRへ送られた場合，経路発見の状態になる．経路発見では送信元から宛て先までの経路の有無を調べ，経路が存在しない場合経路要求を送信する．

経路要求により送信元から宛て先までの経路が発見された場合，送信元はデータの送信を行う．データ送信が終了したならば，送信元は初めの待機状態に戻る．そしてデータ送信を行う際，送信元はノードの移動により経路が途切れていないかを判断する経路維持状態に移る．経路が途切れていない場合，送信元はデータの送信を続けるが，経路が途切れたと判明した場合，経路要求状態に戻り，新たな経路の発見を行う．

図 3.4 は各ノードにおけるPTRの動作を示している．それぞれ送信，転送，受信ノードに分けられ，送信元ノードにおいてPTRは宛て先までの経路確立や，データ送信の機能を提供する．転送ノードでは，PTRパケットを受信した際，ターゲットアドレスが自分のアドレスと一致するかを判断し，異なる場合はルーティング情報を元に次のノードへ転送する機能を提供する．最後に宛て先ノードでは，ターゲットアドレスに自分のアドレスが該当する場合，PTRヘッダを取り除き適切な上位層へデータを

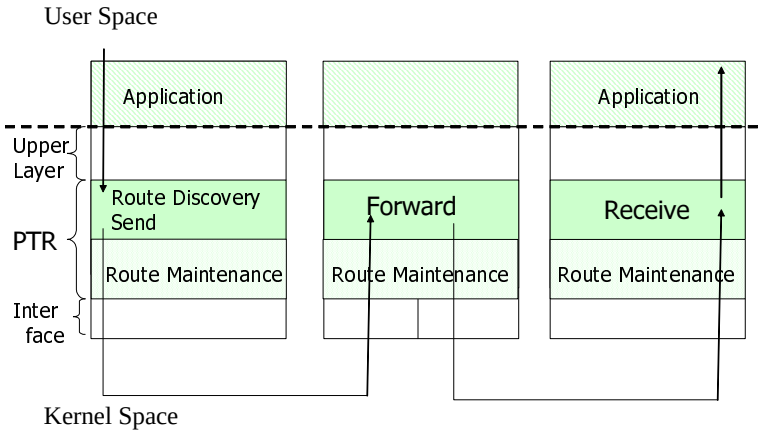

図 3.4: 各ノードにおける PTR の動作

渡す機能を持つ．またすべてのノードにおいて経路維持が存在する．これはデータの送受信を行う度に経路が途切れていないかを監視し，もし経路が途切れてしまった場合，経路の途切れを他のノードへ通知する機能となる．

3.4.3 各モジュール説明

PTR は経路発見，データ送信，データ転送，経路維持のモジュールに分割できる．本小節ではそれぞれのモジュールの機能について説明する．

経路発見モジュール

経路発見の概要　初めて送信元が宛て先ノードに対しデータ送信を行う場合，またデータ送信途中にノードの移動で経路が途切れてしまった場合，送信元から宛て先ノードの経路を発見しなければならない．

経路発見モジュールにおいて，まず送信元はターゲットアドレスとインタフェース情報を経路要求パケットにを加え，直接届く範囲のノードに送信を行う．経路要求パケットを受信したノードが宛て先ノードでない場合，経路要求パケットに自分のアドレスとインタフェース情報を付加して，直接届く範囲のノードにすべて送信する．またノードが経路要求パケットを送信する際，自分自身が持っているすべてのインタフェースに対し送信を行い，次々に経路発見要求パケットを送信していく．

もし自分自身のアドレスがターゲットアドレスと等しい場合，経路応答パケットを送信元に送信する．その際経路要求パケットにはそれまで転送を行ったノードのアドレ

スと受信したインタフェース情報が格納されている．それらのアドレスとインタフェース情報は経路表へキャッシュされ，ノードは経路応答パケットにそれらの情報をコピーする．経路応答パケットを送信する際，経路要求に含まれるアドレスとインタフェース情報からノードとインタフェースの選択が可能となり，送信元まで宛て先ノードは送信できる．またデータ送信を行っていた経路がノードの移動により途切れた際にも経路発見が行われる．

経路要求の手順　上位層からのパケットをPTRが受信し，宛て先アドレスを経路表に問い合わせる．経路表は表 3.1 の情報を扱う．

表 3.1: 経路表

項目	説明
Route ID	経路の識別子
Source Node ID	送信元ノードの識別子
Target Node ID	宛て先ノードの識別子
Hop Node ID	中間ノードの識別子
Interface	各ノードのインタフェース情報

もし経路表に宛て先ノードの識別子が存在する場合，経路が確立され判明されているため送信元はデータ送信が可能である．しかし経路表に問い合わせ，一致する宛て先ノードの識別子が無い場合，経路を確立しなければならないため経路要求を送信する．

経路要求パケットには表 3.2 の項目を格納する．*Packet Type* には，表 3.3 の項目のいずれかが格納される．経路要求の場合には *Packet Type* は *Route Request* となる．また *Source Node ID* は実際のデータ送信元ID，*Dstination Node ID* は実際のデータ送信先IDとなる．経路要求パケットを送信するノードでは，*Source Node ID* と *Req Source Node ID* に等しいIDが入ることになる．

送信元ノードは経路要求パケットに情報を格納後，所持するすべてのインタフェースから経路要求パケットを送信する．その際送信したインタフェース情報は経路要求パケットの *Interface* に格納される．

経路要求を受信したノードは以下の手順で動作する．

1. *Route ID*，*Req Source Node ID*，*Target Node ID* を調べる．もし同じタプルのパケットを最近受信していた場合は破棄し，転送しない．

2. 1 の条件に当てはまらなければ，*Target Node ID* を経路表に問い合わせる．もし経路表に同じ *Target Node ID* があれば，宛て先までの経路が判明しているので，宛て先までの経路を格納し送信元へ経路応答パケットを送信する．

表 3.2: 経路要求パケット

項目	説明
Packet Type	パケットタイプ
Header Len	ヘッダの長さ
Source Node ID	送信元ノードの識別子
Destination Node ID	宛て先ノードの識別子
Route ID	経路の識別子
Req Source Node ID	経路要求を送信したノードの識別子
Target Node ID	ターゲットノードの識別子
Hop Node ID	中間ノードの識別子
Interface	各ノードのインタフェース情報

表 3.3: パケットタイプ

項目	説明
Route Request	経路要求
Route Reply	経路応答
Data	通常データ
Ack	データ転送応答
Route Error	経路エラー

3. 1，2 の条件に当てはまらず，*Target Node ID* が自分の識別子と一致した場合，経路要求パケットに格納されている情報から経路応答パケットを作成し，送信元へ経路応答パケットを送信する．

4. 上記のどの条件にも当てはまらなければ，経路応答パケットの *Header Len* を変更し，自分の識別子を *Source ID*，*Hop ID* に，送信インタフェースを *Interface* に格納し，ノードが所持している全インタフェースから送信を行う．

経路要求パケットを受信したノードは一定時間 *Route ID*，*Source Node ID*，*Target Node ID* のタプルを保存する．短時間に再びこのタプルと同じ経路要求パケットを受信した場合，ノードはそのパケットを破棄することで，ネットワークにおける経路要求パケットのループを防ぐ．

経路応答の手順　宛て先ノードが経路要求を受信した場合，経路応答パケットを送信しなければならない．宛て先ノードは経路要求パケットの *Source Node ID*，*Hop ID*，

Interface の情報から経路応答パケットを作成する．また経路表にそれらの情報を格納する．

表 3.4: 経路応答パケット

項目	説明
Packet Type	パケットタイプ
Header Len	ヘッダの長さ
Source Node ID	送信元ノードの識別子
Destination Node ID	宛て先ノードの識別子
Route ID	経路の識別子
Req Source Node ID	経路要求を送信したノードの識別子
Target Node ID	ターゲットノードの識別子
Hop Node ID	中間ノードの識別子
Interface	各ノードのインタフェース情報

経路応答パケットでは表 3.4 に示す要素が格納され送信される．宛て先ノードは *Packet Type* を *Route Reply* に設定し，経路要求パケットを基にして必要な情報を格納する．

送信する際，経路要求パケットが送信された逆方向に経路応答パケットは送信される．例えば図 3.5 に示す様に，送信元である A が宛て先である G に対し，A，B，C，…，G の経路を通過し経路要求パケットが送信された場合は，逆の方向である G，F，E，…，A へ経路応答パケットを送信すれば良い．

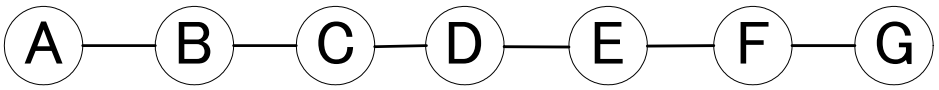

図 3.5: 経路応答パケットの送信

また経路要求パケットを受信した中間ノードは，宛て先まで経路応答を転送する際，自分の経路表へ経路応答の *Route ID*，*Source Node ID*，*Target Node ID*，*Hop ID*，*Interface* を格納する．次のノードへ転送する際は経路応答パケットに格納されている *Hop ID*，*Interface* の情報を元に転送を行う．

データ送信モジュール

データ送信の概要　送信元は宛て先の経路確立後，宛て先へデータを送信する．ユーザが指定した宛て先ノードの ID から経路表を検索し，宛て先までの経路を得る．こ

の時複数経路が存在する場合は最短ホップの経路をデータ送信の経路として選択する．経路表の情報から適切な次の転送先ノードの ID とインタフェースを使用しデータ送信を行う．

表 3.5: 通常データパケット

項目	説明
Packet Type	パケットタイプ
Packet ID	パケット識別子
Header Len	ヘッダの長さ
Data Len	データの長さ
Source Node ID	送信元ノードの識別子
Destination Node ID	宛て先ノードの識別子
Route ID	経路の識別子
Req Source Node ID	経路要求を送信したノードの識別子
Target Node ID	ターゲットノードの識別子

データ送信の手順　送信元は宛て先の経路確立後，*Target*，*Node ID* を鍵として経路表を検索する．経路表からは宛て先までの経路の有無が返る．宛て先までの経路が存在する場合，送信元ノードはデータに表 3.5 の要素を持つ PTR ヘッダをデータの先頭に付随させる．*Pacet Type* を *Data* とし，*Header Len*，*Data Len* を計算し次のノードに送信する．もし宛て先までの経路が複数存在する場合は，最短ホップの経路をデータ送信に使用する経路として選択する．経路表から最短経路の転送先ノードの ID，インタフェースを取得し，データ送信を行う．

データ転送モジュール

データ転送の概要　データ転送モジュールでは，ノードがデータパケットを受けとった際，そのパケットが自分の宛て先であるかを判断する．もし自分の宛て先でない場合は経路表を問い合わせ，次に転送するノードのアドレスとインタフェース情報を取得し，転送を行う．

データ転送の手順　データを受信したノードはまず，自分のノードの識別子が *Target Node ID* と一致するかを調べる．もし異なれば次のノードへ転送をしなければならない．そのため中間ノードは経路表より *Route ID*，*Req Source Node ID*，*Target Node ID* を鍵として検索する．もしこれらのタプルに適合する経路が存在するならば，経路表から *Hop ID*，*Interface* の情報を取得できる．そして *Source Node ID* を自分のノー

ドのID，*Destination Node ID* を転送先ノードのIDに書きかえ，経路表より取得した *Interface* より，適切なインタフェースに向けて転送を行う．

経路維持モジュール

経路維持の概要　経路維持モジュールではデータ送信時に，ノードの移動による経路の途切れを発見し，送信元が経路発見モジュールにより経路を新たに再検索を行うよう，送信元に知らせる機能を持つ．

表 3.6: データ転送応答パケット

項目	説明
Packet Type	パケットタイプ
Packet ID	パケット識別子
Header Len	ヘッダの長さ
Source Node ID	送信元ノードの識別子
Destination Node ID	宛て先ノードの識別子

表 3.7: 経路エラーパケット

項目	説明
Packet Type	パケットタイプ
Packet ID	パケット識別子
Header Len	ヘッダの長さ
Source Node ID	送信元ノードの識別子
Route ID	経路の識別子
Error Source Node ID	経路エラーパケットを送信したノードの識別子
Error Destination ID	パケットを送信できないノードの識別子

経路維持の手順　各ノードはデータの送信もしくは転送を行った場合，ある一定時間転送したパケットを保存する．転送先であるノードはデータを受信した際，表 3.6 の要素を持つ転送応答パケットを転送元ノードへ返す．転送応答パケットを受信した転送元ノードは転送が成功したと判断し，*Packet ID* の値が同じであるパケットを破棄する．

もしある一定時間内に転送応答パケットが転送先ノードより送信されなければ，データ転送が失敗したと判断し，転送元ノードは転送先ノードへ保存しておいたパケットの

再送を試みる．この再送が失敗した場合ノードの移動により経路が途切れたと判断し，経路エラーパケットを送信元もしくは宛て先に送信しなければならない．経路エラーパケットは表 3.7 の要素を格納する．経路エラーパケットでは *Packet Type* は *Route Error* となり，途切れた経路識別子を *Rout ID* に入れ，必要な情報を格納し送信する．経路エラーパケットを受信したノードは，自分所持している経路表の情報に関係ある経路ならばその経路を消去する．送信ノードが経路エラーパケットを受信した場合，新たな経路を再び確立するため経路要求パケットを送信する．

経路エラーパケットの送信先は経路の途切れていない逆の方向となる．これにより不必要なパケットの送信をせずに済む．たとえば図 3.6 の様に，A を送信元とし G を宛て先とする経路を仮定する． もし F が移動し，E，F 間のリンクが途切れた場合，E は F からの転送応答パケットを受信できない．そこで，E は A へ経路エラーパケットを送信する．

また C，D 間にデータがあり，B，C 間と E，F 間で同時に経路が途切れてしまった場合，送信元である A は次のデータを送信することで，B は C への経路の途切れを発見し A へ経路エラーパケットを送信する．A は B から C への経路が途切れたという経路エラーパケットを B から受信した後，再び G への経路要求を行い，新たな経路を発見する．

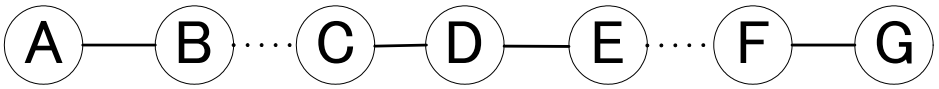

図 3.6: 経路維持

3.5 PTR の実装

本節では PTR の実装について述べる．今回，実装の時間がなかったため，ノードの識別子として PTR が独自に提供する識別子ではなく，IP アドレスを用いる．また使用する無線メディアとして 802.11b と Bluetooth を用いる．

3.5.1 実装環境

本研究では PTR を Redhat7.3 (Linux 2.4) に実装をし，無線デバイスとして 802.11b と Bluetooth を使用した．Bluetooth のデバイスは，3Com 3CREB96 [18] を用い，Bluetooth のプロトコルスタックとして，bluez-kernel-2.3 を用いた．今回ノードの識別子として IP アドレスを用いるために，Bluetooth 上で IP アドレスを使用した．また 802.11b の無線デバイスとして Melco WLI-PCM-L11 [19] を用いた．

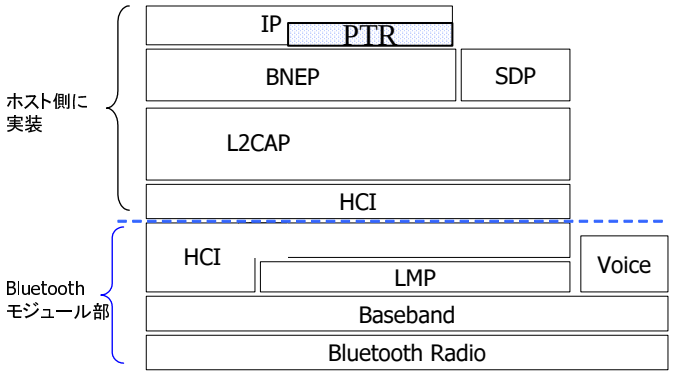

図 3.7: Bluetooth における PTR プロトコルスタック

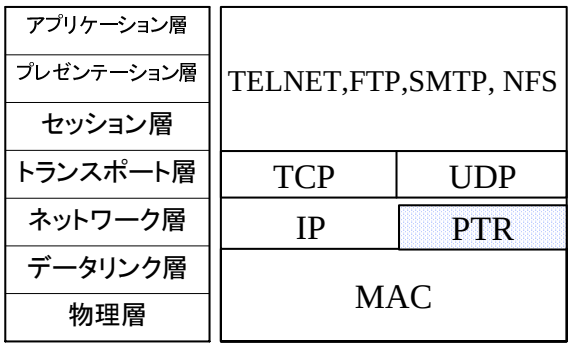

図 3.8: 802.11b における PTR プロトコルスタック

3.5.2 PTR のプロトコルスタック

本小節では PTR のプロトコルスタックについて述べる．今回はノードの識別子として IP アドレスを用いるため，ネットワーク層に PTR を実装し，また Bluetooth では，Bluetooth Networking Encapsulation Protocol (BNEP) を用いた．BNEP は IPv4, IPv6，IPX などのネットワークプロトコルを Bluetooth 上で使用可能にする機構である．BNEP を使用することでネットワーク層からは他の NIC と同様に Bluetooth を使用できる．図 3.7 では Bluetooth における PTR スタックを示している．PTR スタックは BlueZ のスタック上にあるネットワーク層に存在する．

図 3.8 では 802.11b における PTR スタックを示している．802.11b においても Blue-

tooth と同様，ネットワーク層に PTR のスタックが存在する．

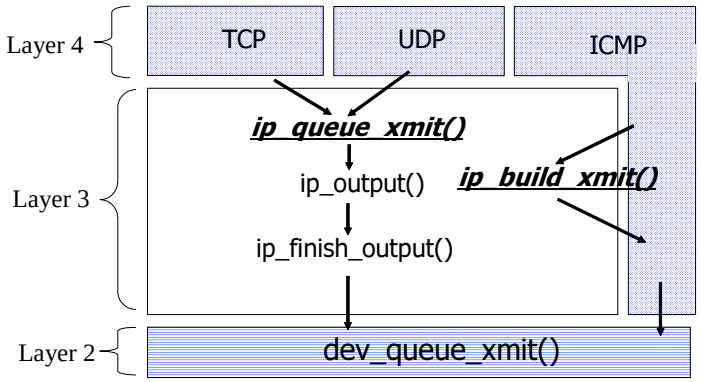

図 3.9: Linux におけるデータ送信時の関数

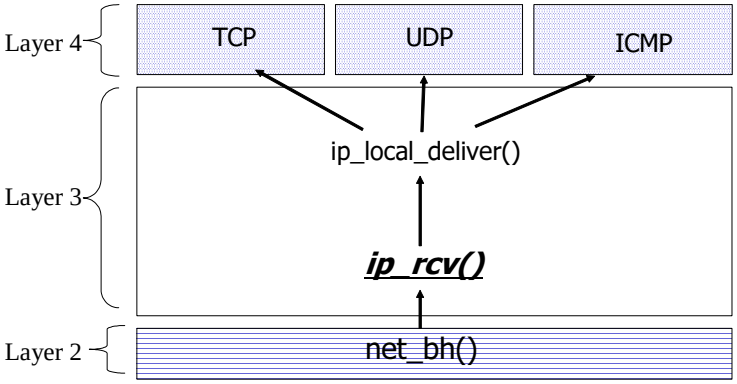

図 3.10: Linux におけるデータ受信時の関数

3.5.3 各機能における PTR の実装

本小節では各機能における，PTR の実装について述べる．

まず，PTR が Linux において具体的にどの関数に実装されるかを説明する．PTR はネットワーク層に存在するため，上位層から渡されたデータをネットワーク層の PTR

で受信し，アドレスや PTR ヘッダを付加しデータを送信しなければならない．今回用いたカーネルである Linux-2.4 のネットワーク層における大まかな関数呼び出しは図 3.9 の様になっている．

トランスポート層の TCP，UDP などのプロトコルは，ネットワーク層へデータを渡す際，必ず ip_queue_xmit() を呼び出す．しかし ICMP の場合，他のトランスポート層の様に ip_queue_xmit() 関数を呼び出さず，ICMP の関数内でインタフェース層のキューにデータを送信する．ICMP が使用する IP の関数は IP ヘッダを作成する ip_build_xmit() である．そこで PTR は ip_queue_xmit()，ip_build_xmit() の 2 箇所に PTR 関数を設置する．

また Linux ではデータ受信時の関数呼び出しは図 3.10 の様になっている．インタフェース層において net_bh() で受信したデータは IP にデータを渡す際，ip_rcv() を呼び出す．ネットワーク層に渡ったデータはその後 ip_local_deliver() からプロトコル番号に従い，上位層である TCP，UDP，ICMP などのトランスポート層に渡される．

PTR では受信したデータが PTR のパケットであるか否かを判断しなければならない．そこでデータを受信した際，必ず呼び出される関数である ip_rcv() に PTR の機能を追加した．

経路確立

ここでは経路が確立するまでの PTR の実装について述べる．送信元が上位層のデータを送信する際，PTR の経路表を調べる．もし宛て先までの経路が確立されてない場合，送信元は経路要求を送信し，送信元から宛て先までの経路を確立しなければならない．

経路要求　経路確立時の疑似コードを図 3.11，変数の説明を表 3.8 に示す．まず上位層から渡されたデータは PTR の関数である，Route_Exists() に渡される．ここで，PTR の経路表から宛て先である D の経路の有無を調べる．もしなければ宛て先 D までの経路を確立しなくてはならない．そのため送信元は Send_Route_Ruquest() を呼び出し経路要求パケットを送信する．Send_Route_Request() では Get_IF_Info() を行い，送信元が所持するインターフェース情報を取得する．経路要求パケットは所持している全インタフェースに関して送信されなければならないため，送信元は取得したインタフェース情報から Make_PTR_Req_Header() を用い，必要な個数の経路要求パケットを作成

表 3.8: 変数の説明

変数	説明
S	送信元ノードのアドレス
D	宛て先ノードのアドレス

```
・Source Node S:
Send data to Destination(D):

if(Route_Exists(D) == NULL){
    Send_Route_Request(D);
}else{
    Add_PTR_Header();
    Change_IPhdr();
}

We have no route to D, Send Route Request:

Send_Route_Request(D){
    Get_IF_Info();
    Make_PTR_Req_Header();
    Make_IPhdr();
    dev_queue_xmit();
}
```

図 3.11: 経路要求

する．経路要求パケットは図 3.12 で示すパケットヘッダとなる．経路要求パケットの格納する値は以下になる．

- pkt_type = Route_Request
- hdr_len = 32
- Req_src_addr = S
- Target_addr = D
- IF[0] = Bluetooth

PTR ヘッダのメンバである hdr_len は，今回 IP アドレスを使用し，かつ Hop_addr は格納されていないため 32Byte となる．Req_src_addr は経路要求を送信したノードである S のアドレスとなり，Target_addr は宛て先である D となる．そして IF には各ノードにおける出力インタフェースの識別子が格納される．またパケットヘッダのメンバとして存在する Route_ID は，それぞれの経路についている識別子である．中間ノードはこの Route_ID と，送信元，宛て先のアドレスで次のホップとインタフェース

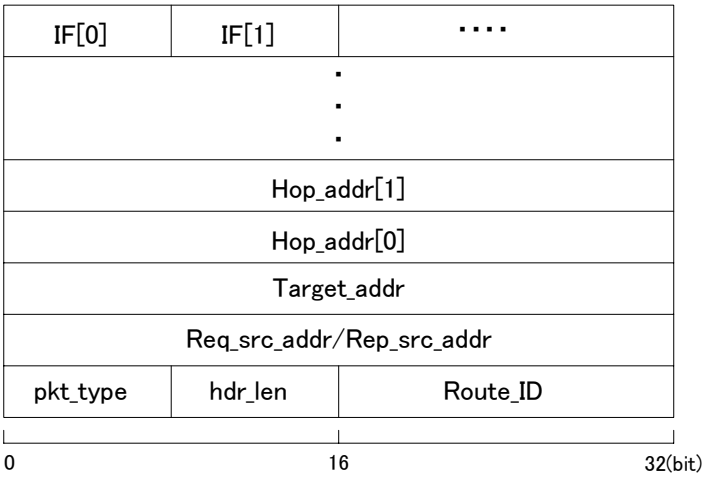

図 3.12: 経路要求，経路応答パケット

情報を経路表より取得する．Pkt_ID は後に説明するデータ転送応答で使用する，シーケンス番号である．以上の情報を PTR ヘッダに格納し，Make_IPhdr() を呼ぶ．

Make_IPhdr() は PTR コントロールパケットのために IP ヘッダを作成する関数である．IP ヘッダのそれぞれの値を以下に示す．

- `iph->daddr = 255.255.255.255`
- `iph->saddr = S`
- `iph->tot_len = 32+sizeof(iph)`
- `iph->proto = IPPROTO_PTR`

まず IP ヘッダにおける実際の送信先は D でなく，全ノードに経路要求を送信するためブロードキャストアドレスとなる．また PTR ヘッダをデータとして付随させるので，パケット全体の長さをしめす tot_len には PTR ヘッダと IP ヘッダ自体の大きさとなる．最後に IP ヘッダのプロトコル番号には PTR のプロトコル番号である IPPROTO_PTR を格納する．

経路応答　ノードが経路要求パケットを受信した場合，経路応答を破棄，他のノードに転送，もしくは送信元に経路応答を返さなければならない．

経路応答の疑似コードを図 3.13 に，変数の説明を表 3.9 に示す．経路要求を受信したノードは IP ヘッダのプロトコル番号を調べ，もしプロトコルが IPPROTO_PTR と定義されていた場合，PTR パケットだと判断し Rcv_PTR() を呼ぶ．Rcv_PTR() では，PTR ヘッダの pkt_type により処理が分かれる．もし pkt_type が Route_Request である場合，そのパケットは経路要求と判断し経路要求の処理関数が呼ばれる．

```
・Receive Route Request
if(iph->proto == IPPROTO_PTR)
    Rcv_PTR();

Rcv_PTR(){
    if(ptrhdr->pkt_type == Route_Request){
        if(Tuple_Same()){
            goto DROP;
        }else if(Route_Exists(D)){
            Add_Routing_Table();
            Send_Route_Reply();
            Save_Packet()
        }else if(ptrhdr->target_addr == MY_ADDR){
            Add_Routing_Table();
            Send_Route_Reply_from_Req();
            Save_Packet()
        }else{
            Forward_Route_Req();
            Save_Packet()
}
```

図 3.13: 経路応答

表 3.9: 変数の説明

変数	説明
iph	IP ヘッダのポインタ
ptrhdr	PTR ヘッダのポインタ
D	宛て先ノードのアドレス
MY_ADDR	自分のアドレス

Tuple_Same() は経路要求パケットに格納されている情報である Route_ID, Req_src_addr, Target_addr を調べる．もし同一のタプルを持った経路要求パケットをある期間内に受信している場合, DROP により受信した経路要求パケットは破棄される．次に Route_Exists() により，経路要求パケットの Target_addr が経路表に存在するか調べる．もし 0 以上の値が返ってきた場合, 宛て先までの経路が存在することになる．そこで Add_Routing_Table() により経路表を更新する．この経路表は設計で示した PTR の経路表とカーネルで実装されている経路表に対して変更が行われる．その後 Send_Route_Reply() を呼び出し，

経路要求パケットに格納されている Hop_addr と，自分の経路表から経路応答パケットを作成する．経路応答は図 3.12 で示すように，経路要求と同一のパケットフォーマットとなる．以下に具体的な経路応答パケットの内容を示す．

- `pkt_type = Route_Reply`
- `Rep_src_addr =` 経路応答を送信するノードのアドレス
- `Target_addr = S`

pkt_type は経路応答の Route_Reply となる．また hdr_len では経路要求を転送したノード数により可変となる．Route_ID は経路要求パケットと等しい値となり，Rep_src_addr は経路応答が Send_Route_Reply() から送信された場合，途中の中間ノードによって経路応答が送信されるため，中間ノードのアドレスが入る．また Target_addr は送信元のアドレスである S となる．そして IP ヘッダが付け加えられ，送信元へ送信される．

先の条件以外であり，また target_addr が経路要求を受信したノードのアドレスである場合も，経路応答パケットを送信元へ送信しなければならない．上記と同様に，Add_Routing_Table() を呼び出し経路表を更新する．そして Send_Route_Reply_from_Req() を呼び出し経路応答を送信元へ送信する．

上記の条件にどれも該当しない場合，Forward_Route_Req() を呼び出し，経路要求パケットの Hop_addr，IF に対し自分の情報を付け加え，hdr_len の値を更新する．その際，Target_addr，Req_src_addr，Route_ID の 3 つのタプルをある一定時間保存しておく．その後 IP ヘッダの tot_len を加算し，daddr は同様にブロードキャストに設定し次のノードに送信する．

経路要求に対して送信される経路応答が各ノードで受信された時の疑似コードを図 3.14，変数の説明を表 3.10 に示す．

プロトコル番号により PTR のパケットを判別後，PTR ヘッダの pkt_type を調べる．経路応答の場合 pkt_type は Route_Reply となる．経路応答を受信した全ノードは Add_Routing_Table() を呼び出し経路表を更新する．経路応答の Target_addr が自分のアドレスと等しい場合，自らが送信した経路要求に対する応答となるので，送信元がデータ送信を行う際，必要な経路が確立されたこととなる．そのため packet_routed となり通常データ送信に処理が戻る．

表 3.10: 変数の説明

変数	説明
iph	IP ヘッダのポインタ
ptrhdr	PTR ヘッダのポインタ
D	宛て先ノードのアドレス
MY_ADDR	自分のアドレス

```
・Receive Route Reply:

if(iph->proto == IPPROTO_PTR)
    Rcv_PTR();

Rcv_PTR(){
    if(ptrhdr->pkt_type == Route_Reply){
            Send_ACK();
            Add_Routing_Table();
            if(ptrhdr->target_addr == MY_ADDR){
                goto packet_routed:
            }else{
                Forward_Route_Rep();
                Save_Packet()
    }
}
```

図 3.14: 経路応答 (続き)

経路応答パケットを受信し Target_addr が自分のアドレスでない場合，経路応答を次のノードに転送しなければならない．そこで Forward_Route_Rep() を呼び出す．Forward_Route_Rep() では IP パケットを作成し，適切なインタフェースのキューにデータを渡す関数である．経路応答パケットにおける情報更新の必要はないが，経路応答パケットの Hop_addr の情報から取得した転送先ノードのアドレスを IP ヘッダの daddr へ格納し，適切なインタフェースで転送を行う．

データの送信

ここではデータ送信における PTR 機能の実装について述べる．データ送信機能は送信元ノードのデータ送信機能，転送ノードにおける転送機能，受信ノードにおける受信機能の 3 つに区分できる．

データ送信　データ送信では PTR ヘッダをつけ，IP ヘッダの情報を書き換える処理を行う．

図 3.15 にデータ送信機能の疑似コードを，表 3.11 に変数の説明を示す．上位層から渡されたデータは Route_Exists() により PTR の経路表に宛て先までの経路が確立されているかを調べる．もし経路がある場合，Add_PTR_Header() を呼び出し PTR ヘッダを IP ヘッダとデータの間に付け加る．通常のデータならば図 3.16 のヘッダがデータの前に付随する．通常データパケットヘッダにおけるメンバの値は以下となる．

```
・Source Node S:
Send data to Destination(D):

if(Route_Exists(D) == NULL){
    Send_Route_Request(D);
}else{
    Add_PTR_Header();
    Change_IPhdr();
}
```

図 3.15: データ送信

表 3.11: 変数の説明

変数	説明
D	宛て先ノードのアドレス

- pkt_type = Data
- hdr_len = 32
- Src_ID = S
- Target_ID = D

Route_ID はそれぞれの経路についている識別子である．hdr_len は今回 IP アドレスを使用するので，32Byte となる．中間ノードはこの Route_ID と，送信元，宛て先のアドレスで次のホップとインタフェース情報を経路表より取得する．Pkt_ID は後に説明するデータ転送応答で使用するシーケンス番号である，データ転送応答についての説明は後に述べる．以上の情報を PTR ヘッダに格納し，IPhdr_Change() を呼ぶ．

IPhdr_Change() において IP ヘッダの情報は以下となる．

- iph->daddr = NEXT_HOP
- iph->tot_len += sizeof(PTRhdr)
- ptrhdrh->trueproto = iph->proto
- iph->proto = IPPROTO_PTR

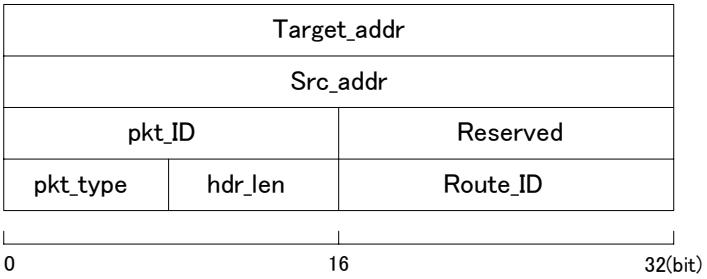

図 3.16: 通常データ，転送応答パケット

まず IP ヘッダにおける実際の送信先は D でなく，次の転送先における IP アドレスでなくてはならない．また PTR ヘッダをデータとして付随させるので，パケット全体の長さを示す tot_len に PTR ヘッダのパケットサイズ分を追加する．最後に IP ヘッダに格納されているプロトコル番号を PTR ヘッダに格納し，IP ヘッダのプロトコル番号には PTR のプロトコル番号である IPPROTO_PTR を格納する．

PTR ヘッダを付随させ，IP ヘッダを作成後，IP の通常の処理関数を呼び出す．送信するインタフェースは経路応答を受信した際に，PTR の経路表と通常の経路表に変更が加えてあるため，次の転送先ノードに対し適切なインタフェースで送信できる．

データ転送　データ転送では，通常データを受信し Target_addr が自分のものでない場合，次のノードに転送する処理を行う．図 3.17 にデータ転送の疑似コードを，表 3.12 に変数の説明を示す．

まず下位層から受信したデータのプロトコル番号を調べる．プロトコル番号が IPPROTO_PTR の場合 PTR の処理系にデータが渡される．PTR ヘッダの pkt_type が Data である場合，通常のデータとして処理される．まず PTR ヘッダの Target_addr が自分のアドレスであるかを調べる，もし自分のアドレスでない場合，次のノードに転送しなければならないため Forward_Data() を呼び出す．

Forward_Data() ではまず Search_NH() を呼び出し，PTR ヘッダの Target_addr，Src_addr，Route_ID をキーとし，PTR の経路表から次に転送すべきノードのアドレスを取得す

表 3.12: 変数の説明

変数	説明
iph	IP ヘッダのポインタ
ptrhdr	PTR ヘッダのポインタ
D	宛て先ノードのアドレス
MY_ADDR	ノードのアドレス

```
・Forward Node F:
Send data to Destination(D):

if(iph->proto == IPPROTO_PTR)
    Rcv_PTR();

Rcv_PTR(){
    if(ptrhdr->pkt_type == Data){
        Send_ACK();
        if(ptrhdr->target_addr == MY_ADDR){
            Rcv_IP_Header()
            Del_PTR_Header();
        }else{
            Forward_Data();
            Save_Packet()
}

Fowrwad_Data(){
    Search_NH();
    IP_Daddr_Change();
}
```

図 3.17: データ転送，受信

る．その取得したアドレスを`iph->daddr`の値に入れ，IP に処理を戻し転送を行う．

データ受信　データ受信ではノードが通常データを受信し，PTR ヘッダにおける Target_addr が自分のアドレスと一致した場合，上位層にデータを渡すため PTR ヘッダを取り除く処理を行う．図 3.17 にデータ転送，受信のコードを，表 3.12 に変数の説明を示す．

受信したプロトコル番号が IPPROTO_PTR であり，PTR ヘッダの pkt_type が Data である場合，通常データとして Rcv_PTR が呼ばれる．PTR ヘッダ内のメンバである Target_addr が自分のアドレスと一致した場合，自分宛てのパケットと判断し，Rcv_Data() が呼ばれる．

Rcv_IP_Header() では上位層へデータを渡すため，IP ヘッダの書き換えを行う．IP ヘッダは以下のように書きかわる．

- `iph->tot_len -= sizeof(PTRhdr)`
- `iph->proto = ptrhdr->trueproto`

まずパケット全体の長さを示す tot_len を PTR ヘッダを取り除くため，PTR ヘッダ分減らす．またデータを送信する際プロトコル番号を IPPROTO_PTR に書き換えているので，元のプロトコル番号に戻さなければならない．そこで PTR ヘッダの trueproto に格納してある元のプロトコル番号を IP ヘッダの proto に入れる．IP ヘッダの変更後，Del_PTR_Header() を呼び出し IP ヘッダと通常データの間に存在する PTR ヘッダを取り除き，IP の処理に戻す．

経路維持

ここでは経路維持における PTR 機能の実装について述べる．経路維持はノードの移動によりデータ送信に使用されている経路が途切れていないかを確認する転送応答機能と，経路が途切れた際それを他のノードに知らせる経路エラー機能に区分できる．

転送応答　転送応答ではノードがデータを受信した場合，受信したノードは転送元へ転送応答を送信しなければならない．転送先から転送応答が無い場合，転送元はパケットの再送をする．図 3.14，図 3.17 に疑似コードを示す．

プロトコル番号が IPPROOT_PTR であり，PTR における pkt_type が Route_Reply もしくは Data のパケットを受信した場合，Send_Ack() が呼ばれる．Send_Ack() では PTR の転送応答パケットを作成し，転送元ノードに送信する．その際 PTR ヘッダでは図 3.16 のパケットが作成される．具体的なメンバの値は以下になる．

- `pkt_type = ACK`
- `hdr_len = 32`
- `Route_ID = NULL`
- `Src_ID =` 転送先ノードアドレス
- `Target_ID =` 転送元ノードアドレス

pkt_type は転送応答となるため ACK となり，hdr_len は固定長の 32Byte となる．Route_ID は転送応答の場合，ノードが直接届く範囲にいると想定されるため NULL となる．pkt_ID は転送されたパケットの pkt_ID の値が格納され，転送応答を受信したノードは pkt_ID により，どのパケットに対する転送応答かを判別する．そして Src_addr は転送応答を送信するノードのアドレス，Target_addr は転送応答の送信先アドレスとなる．

ノードは転送応答の PTR ヘッダを作成後 IP ヘッダを作成し，PTR ヘッダの前に配置し送信を行う．IP ヘッダの具体的なメンバの値は以下となる．

- `iph->daddr = ptrhdr->Src_ID`
- `iph->saddr = ptrhdr->Target_ID`

- `iph->tot_len = sizeof(PTRhdr)+sizeof(iph)`
- `iph->proto = IPPROTO_PTR`

宛て先である daddr は転送元ノードのアドレスとなり，送信元を示す saddr は転送先ノードのアドレスとなる．全体のパケット長を示す tot_len は PTR ヘッダと IP ヘッダの和となる．またプロトコル番号は PTR のコントロールパケットなので IPPROTO_PTR となる．

もし転送先ノードから転送応答があった場合，Del_Packet() により転送応答の対象となったパケットを消去する．

経路エラー　転送元ノードはパケットの転送後，転送先ノードからの転送応答を受信できない場合，転送したパケットが受信されなかったと判断し再送を行う．もし再送を 2 回した後に，転送応答を受信できない場合，経路が途切れたと判断し経路エラーパケットを作成し送信する．

図 3.18 はパケットの再送と経路エラー送信の疑似コード，表 3.13 は変数の説明を示してる．Watch_Packet() は常に各ノードで呼び出される関数である．ノードは経路要求以外のパケットを送信，もしくは転送した場合，図 3.11，3.14，3.17，3.18 が示すように Save_Packet() を呼び出す．Save_Packet() は送信もしくは転送したパケットを一定時間保存する関数である．Watch_Packet() では保存されたパケットが RTT 以上であるかを常に監視する．もし RTT の時間以上保存されているパケットが存在し，かつそのパケットの再送が 2 回以下である場合，Resending_Packet() を呼びパケットの再送を試みる．また 2 回以上再送が行われたパケットである場合は，再送先の経路が途切れたと判断し Send_Error() を呼び，経路エラーパケットをデータ送信元へ送信しなければならない．

Send_Error() では図 3.19 のパケットを作成する．経路エラーパケットにおける具体的なメンバの値は以下となる．

- `pkt_type = Route_Error`
- `hdr_len = 48`

表 3.13: 変数の説明

変数	説明
pkt_ID	各パケットの識別子
RTT	経路の RTT
iphdr	IP ヘッダのポインタ
ptrhdr	PTR ヘッダのポインタ
MY_ADDR	ノードのアドレス

```
・A node always checks whether a packet need be sent or not:

while(1){
    Wacth_Packet(){
        if( Packet_time(pkt_ID) > RTT){
            if(Send_time(pkt_ID) < 3){
                 Resending_Packet();
            }else{
                 Send_Error();
    }
}

・If Ack is not received, the node must send Route Error Packet:

if(iph->proto == IPPROTO_PTR)
    Rcv_PTR();

Rcv_PTR(){
    if(ptrhdr->pkt_type == Ack){
        Del_Packet()
    }
    if(ptrhdr->pkt_type == Route_Error){
        if(ptrhdr->Target_addr != MY_ADDR){
            Forward_Data();
            Save_Packet();
        }
    }
    Del_Route();
}
```

図 3.18: 経路エラー

- Src_addr = 経路の送信元アドレス
- Target_addr = 経路の宛て先アドレス
- Error_Src_addr = 経路エラーパケットの送信元アドレス
- Error_Target_addr = 経路が途切れたノードの宛て先アドレス

pkt_type は経路エラーを示す Route_Error となる．ヘッダ長を示す hdr_len は固定長である 48byte となる．Src，Target_ID はエラーが起きた経路におけるデータの送信元と宛て先になり，また Error_Tareget_addr はデータ送信ができない転送先ノード,

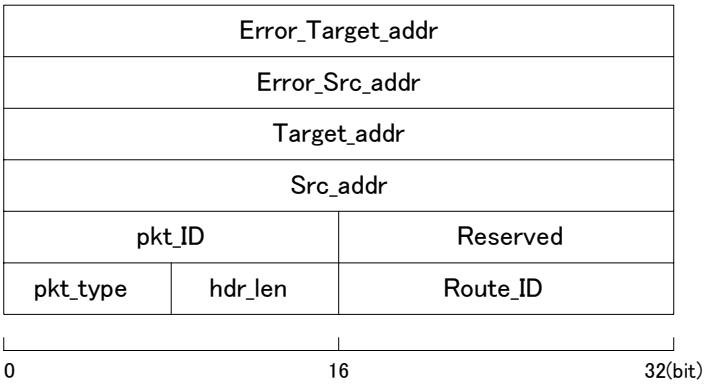

図 3.19: 経路エラーパケット

Error_Src_addr は経路エラーパケットを送信したノードのアドレスとなる．また，送信する際経路エラーパケットに IP ヘッダを付随させなければならない．IP ヘッダにおけるメンバの値は以下となる．

- `iph->daddr = 経路における次の転送先`
- `iph->saddr = ptrhdr->Error_Src_addr`
- `iph->tot_len = sizeof(ptrhdr)+sizeof(iph)`
- `iph->proto = IPPROTO_PTR`

まず経路エラーパケットの送信先である daddr は，経路表よりエラーの対象となった経路において，経路が途切れていない逆方向の転送先ノードアドレスとなる．そして saddr は経路エラーの送信元ノードアドレスとなる．tot_len，proto など値は上記に述べたものと同様となる．

各ノードは PTR ヘッダの pkt_type が Route_Error である経路エラーパケットを受信した場合，自分のアドレスがその経路における宛て先であるかを調べる．もし宛て先でない場合，その経路における終端ノードではないため，経路エラーパケットを次のノードに転送しなければならない，そのため Forward_Data() を呼び出し，通常データの様に次のノードへデータ転送を行う．経路エラーを受信したノードが経路の終端ノードならば，転送先がないためパケットを破棄する．

経路エラーを受信したノードは全て Del_Route() により PTR ヘッダ中の Error_Target_addr を含む経路はすべて破棄する．

3.6 評価

本節では RedHat7.3，Linux2.4 のカーネ上に PTR を実装したシステムを用い，実際に図 3.20 の様な 1 ホップから 4 ホップまでの実験ネットワークを構築し評価を行

った．MANET を構築するために無線メディアとして 802.11b，Bluetooth を用いた．Bluetooth を所持したノードは Bluetooth 上で IP アドレスを用いるため，予めコネクションを確立している．また今回の評価では，PTR の経路発見，経路維持の実装を行っていないため，PTR 層におけるコントロールパケットは送信していない．

今回ホップ数と TCP 転送性能，無線規格と TCP 転送性能の関係を評価項目とした．各評価では netperf [20] を利用して TCP データの送受信を行い，TCP 転送性能を求めた．

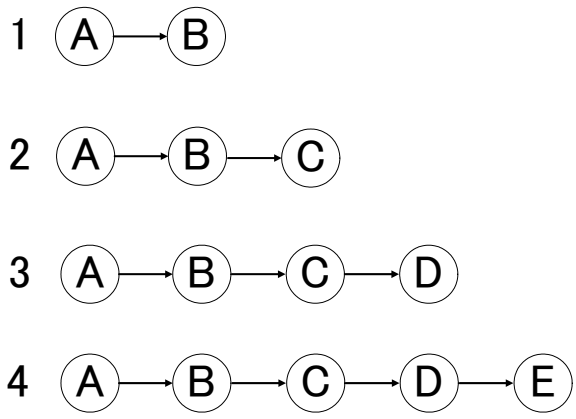

図 3.20: 実験ネットワーク

3.6.1 ホップ数と転送性能の関係

PTR を用いて構築した実験ネットワークに，以下の条件で TCP データの送信を行い評価を行った．

条件 a 無線メディアとして 802.11b のみを使用し，図 3.20 の様に 1 ホップから 4 ホップまでの実験ネットワークを構築した．各ホップの実験ネットワークに TCP データの送信を行う．

条件 b 無線メディアとして Bluetooth を 1 ホップのみ使用し，その他のホップは 802.11b を使用し，図 3.20 の様に 1 ホップから 4 ホップまでの実験ネットワークを構築した．各ホップの実験ネットワークに TCP データの送信を行う．

条件 a，b 共に TCP データの送受信を 10 回行い，TCP 転送性能を求めた．

条件 a の結果を図 3.21 に示す．横軸が転送性能を測定した際のネットワークにおけるホップ数，縦軸が TCP の転送性能 (kbps) となっている．10 回測定した TCP 転送性能の最大，最小，平均をプロットした．ホップ数が増えるにつれ TCP 転送性能が低下していることが分かる．特に 1 ホップから 2 ホップの TCP 転送性能の低下が激しく，平均で 526kbps もの差がある．

条件 b の結果を図 3.22 に示す．横軸，縦軸は条件 a と同様にホップ数と転送性能である．図 3.22 の A では始めの 1 ホップが Bluetooth となり，他のホップは 802.11b である．つまり実験ネットワークにおけるノード A，B 間のホップが Bluetooth となる．また図 3.22 の B では，最後の 1 ホップが Bluetooth となり，他のホップが 802.11b である．つまり実験ネットワークでは 4 の経路の場合，A，D 間は 802.11b であり，最後の 1 ホップである D，E 間は Bluetooth となる．それぞれの条件における TCP 転送性能の最大，最小，平均をプロットした．

経路における 802.11b のホップが増加するにつれて，わずかに TCP の転送性能が低下している．つまり経路に Bluetooth のホップが含まれている場合，802.11b のみのホップ数が増加しても，TCP 転送性能に大きな影響がないと分かる．

条件 a，b の実験より経路に Bluetooth のホップが存在した場合，802.11b のみの経路より，802.11b におけるホップ数の増加が TCP 転送性能に与える影響が少ないと判明した．

3.6.2 無線規格と転送性能の関係

PTR を用い，Bluetooth，802.11b のホップ数を以下の条件のように変化させ，TCP の転送性能を評価した．

条件 A 図 3.20 の 1 のような 1 ホップの経路を構築し，無線メディアとして 802.11b, Bluetooth を用い TCP データを送受信する．

条件 B 図 3.20 の 2 のような 2 ホップの経路を構築し，以下の状態で TCP データを送受信する．

- 2 ホップすべて 802.11b
- ノード A，B 間は 802.11b，ノード B，C 間は Bluetooth

条件 C 図 3.20 の 3 のような 3 ホップの経路を構築し，以下の状態で TCP データを送受信する．

- 3 ホップすべて 802.11b
- ノード A，C 間は 802.11b，ノード C，D 間は Bluetooth
- ノード A，B 間は Bluetooth，ノード B，C 間は 802.11b，ノード C，D 間は Bluetooth

条件 D 図 3.20 の 4 の様な 4 ホップの経路を構築し，以下の状態で TCP データを送受信する．

- 4 ホップすべて 802.11b

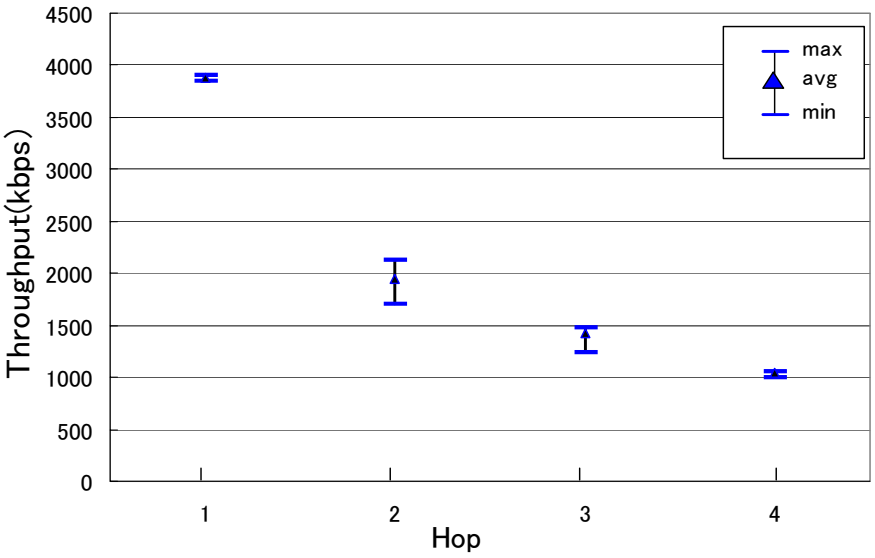

図 3.21: ホップ数と転送性能の関係

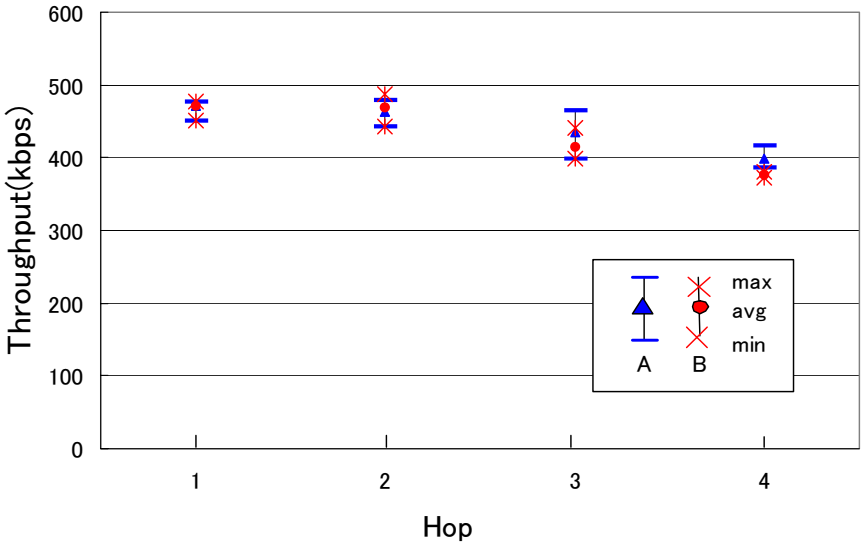

図 3.22: Bluetooth と 802.11b における転送性能の関係

- ノード A，D 間は 802.11b，ノード D，E 間は Bluetooth
- ノード A，B 間は Bluetooth，ノード B，D 間は 802.11b，ノード D，E 間は Bluetooth

全条件において TCP データの送受信を 10 回行い，結果として TCP 転送性能の最大，最小，平均値を示す.

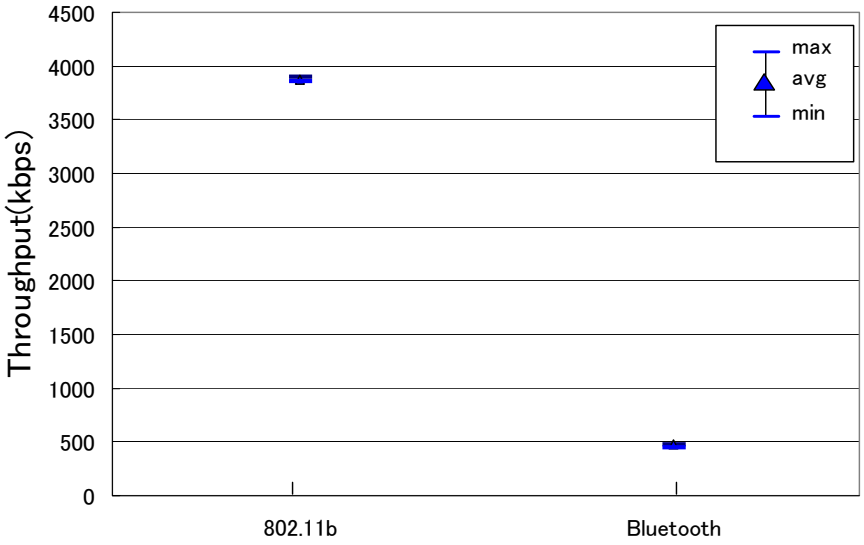

図 3.23: 1 ホップにおける TCP 転送性能

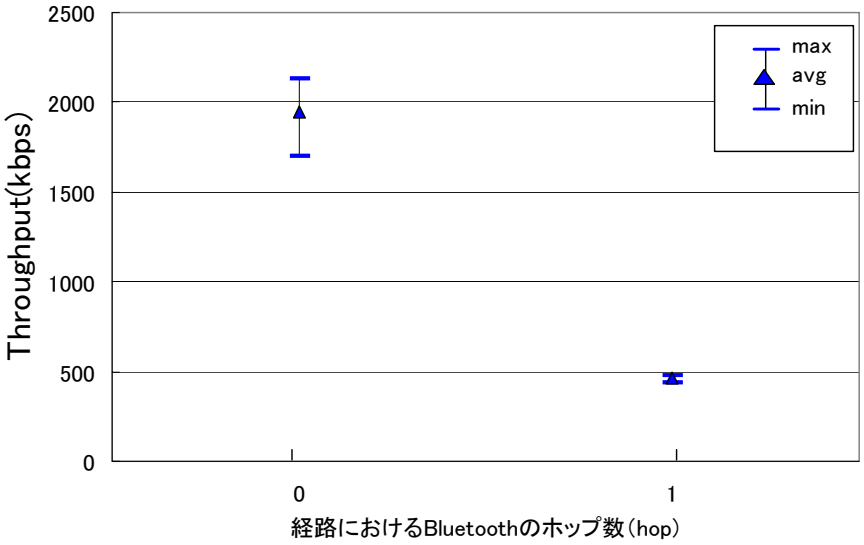

図 3.24: 2 ホップにおける TCP 転送性能

条件 A の結果を図 3.23 に示す．縦軸は TCP の転送性能 (kbps) である．Bluetooth に比べ 802.11b の転送性能は差が 3042kbps あり，約 8 倍も異なる．それぞれの規格における最大転送性能自体が，802.11b では 11Mbps，Bluetooth では 1Mbps と大きく異なるため，実測の転送性能においても Bluetooth と 802.11b に大きく転送性能の差が出たのだと考えられる．

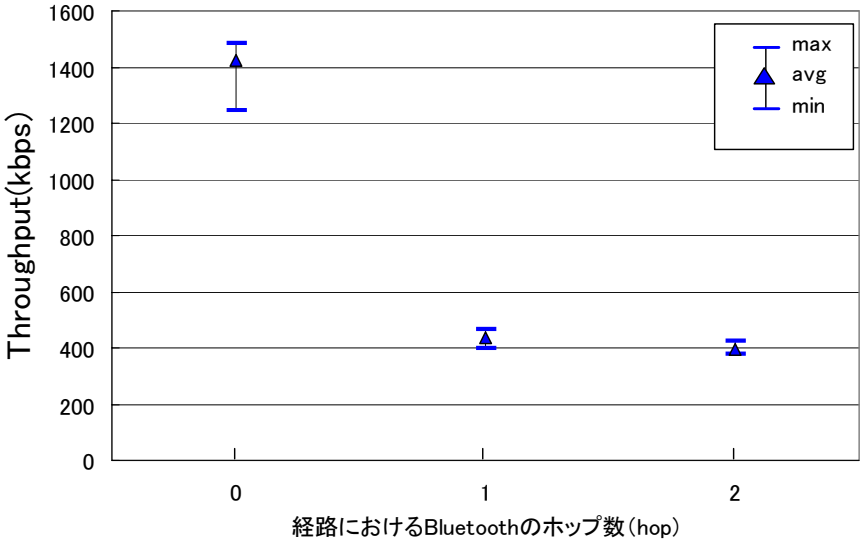

図 3.25: 3 ホップにおける TCP 転送性能

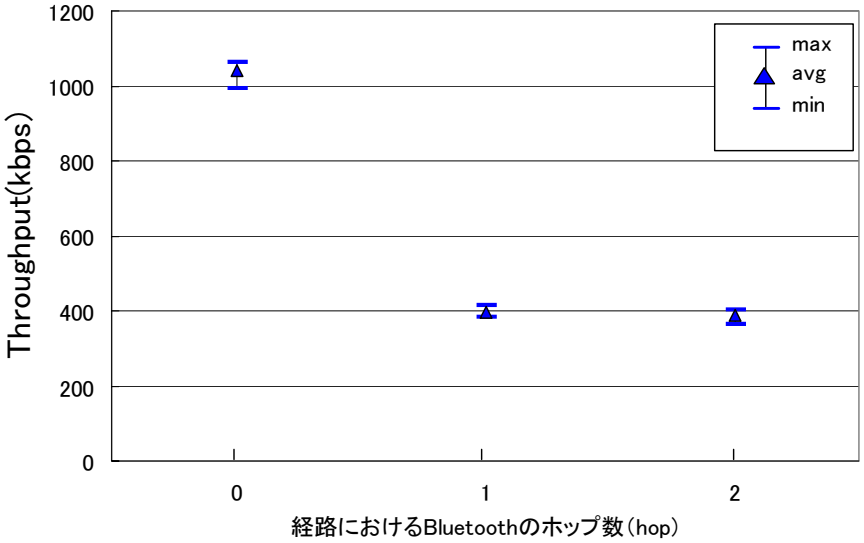

図 3.26: 4 ホップにおける TCP 転送性能

条件 B の結果を図 3.24 に示す．縦軸は TCP の転送性能 (kbps) であり，横軸は経路における Bluetooth のホップ数である．経路における Bluetooth のホップ数が 0 の場合，全てのホップが 802.11b であることを示す．条件 A の場合と同様に，802.11b のみの経路と Bluetooth を含む経路では大きく TCP の転送性能が異なる．802.11b のみの経路では Bluetooth を含む経路と比べ TCP 転送性能が約 4 倍良いことが分かる．

条件 A に比べ，条件 B では 802.11b の経路と Bluetooth を含む経路の転送性能の差が小さい．このことは Bluetooth を含む経路では 1 ホップ，2 ホップの経路における転送性能は差が平均 6kbps と小さいが，802.11b のみの経路では 1 ホップ，2 ホップの経路における転送性能の差が 526kbps と大きいためである．

次に条件 C の結果を図 3.25 に示す．縦軸と横軸は条件 B と同様である．802.11b のみ経路と，Bluetooth を 1 ホップ含む経路の転送性能は約 3 倍と大きく異なる．しかし，Bluetooth を 1 ホップ含む場合と 2 ホップ含む場合ではその差は 75kbps と小さいことが分かる．

最後に条件 D の結果を図 3.26 に示す．縦軸と横軸は条件 B と同様である．結果としては条件 C と同様な傾向が見られる．つまり，802.11b のみ経路と，Bluetooth を 1 ホップ含む経路の転送性能は約 3 倍と大きく異なる．しかし，Bluetooth を 1 ホップ含む場合と 2 ホップ含む場合ではその差は 9kbps と小さい．

これら 4 つの条件における結果から，802.11b のみの経路に比べ，1 ホップでも Bluetooth のホップを含んだ場合，転送性能は大きく低下してしまう．しかし 1 ホップでも Bluetooth のホップを含んだ場合，Bluetooth，及び 802.11b のホップが増加しても，TCP における転送性能の低下は小さいことが分かった．

3.7 問題点

本章では PTR を用い異種無線メディアにおける MANET を構築した際，考えられる問題を述べる．

現在多くの MANET を構築するルーティングプロトコルは，経路の指標としてホップ数を用い，送信元から宛て先まで複数経路が存在した場合最短経路を選択する．しかし無線は電波品質が転送性能に大きく影響を及ぼすため，最短経路が転送性能において最善な経路とならない場合が発生する．同一無線メディアの MANET において，経路により電波品質やパケットロス，遅延などが異なり，最短経路が転送性能において最善な経路とならない状況が発生すると考えられる．しかし PTR を用いて MANET を構築し，ノードが所持する無線メディアが異なる場合，ホップを指標にした経路選択は大きな問題となる．

無線メディアはそれぞれの規格毎に最大の転送性能が大きく異なる．表 3.14 に主な無線メディアの規格における転送性能を示している．無線メディアによって最大で 54Mbps から最低 1Mbps まで転送性能が大きく異なることが分かる．そのため，現在多くの MANET で使用されているホップ数を指標とした経路制御の場合，802.11a，802.11b，Bluetooth の経路もすべて 1 ホップという同一な経路でみなしてしまう．図 3.27 のように送信元から宛て先まで Bluetooth のホップを含む 3 ホップの経路，802.11a のみの 4 ホップの経路が存在する場合，指標をホップ数とした経路選択であると，3 ホップの経路を選択してしまう．しかし上の 3 ホップの最短経路より，下の 4 ホップの経路を使用した方が送信元から宛て先までの転送性能は高いと考えられる．なぜなら上の経路では最大転送性能が 1Mbps である Bluetooth のホップを含んでしまうため，Bluetooth

のホップがボトルネックとなってしまうからである．

例えば今回の評価では無線メディアとして Bluetooth と 802.11b を用いたが，規格上 Bluetooth の転送性能は 1Mbps，一方 802.11b では 11Mbps である．実測値の場合でも図 3.23 となり，転送性能の差が明らかに大きい．この 2 種類の無線メディアを含む MANET を考えた場合，前節の評価結果より Bluetooth を 1 ホップ含んだ 3 ホップの経路では転送性能が約 400kbps，802.11b のみの 4 ホップの経路では転送性能が約 1000kpbs となり，転送性能において最短経路が最善な経路でない．

そのため異種無線メディア間における MANET では，複数経路が存在した場合経路選択の指標としてホップ数以外の指標を用いる経路選択機構が必要であると考えられる．

表 3.14: 無線メディアと転送性能

規格	802.11a・HiperLAN	IrDA	802.11b	HomeRF	Bluetooth
転送性能	54Mbps	15Mbps	11Mbps	10Mbps	1Mbps

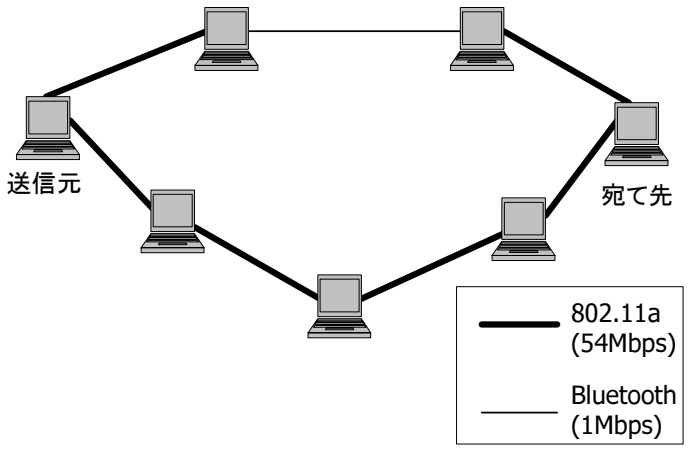

図 3.27: ホップを指標とした経路選択

3.8 まとめと今後の課題

ここでは本章のまとめと今後の課題について述べる．

3.8.1 本章のまとめ

本章では多種多様な無線メディア同士で構築できる MANET のルーティングプロトコルである PTR を提案した．PTR 層では独自の ID を用いてノードの識別を行い，無線メディアを複数所持するノードに転送することで，異なる無線メディアを所持するノード同士が透過的に通信できるプロトコルである．本章では PTR の設計，実装を述べ，実際に PTR で MANET を構築し評価実験を行った．

評価実験では実際に PTR を用い，802.11b と Bluetooth の実験ネットワークを構築し，TCP 転送性能の評価を行った．802.11b のみのネットワークではホップ数が増えると急激に TCP の転送性能が低下してしまう．一方 Bluetooth を経路に 1 ホップでも含んだ場合，ホップ数が 2，3 程度増加した際でも，TCP の転送性能に大きな低下は見られなかった．

最後に異種無線メディアで構成される MANET において，最短経路選択の問題点を指摘した．同一無線メディアの MANET においても，経路の遅延やロスによっては最短経路が必ずしも転送性能において最善な経路ではない．しかし異種無線メディアの MANET において，無線規格毎の転送性能が大きく異なることから，経路選択は大きく転送性能に影響を及ぼす事を指摘した．

3.8.2 今後の課題

今後の課題として以下に挙げる．

- 動的なアドレス割り当て

  PTR における独自の ID は手動で割り当てを行わなければならない．PTR におけるアドレスの割り当てを動的に行える機構が必要である．

- セキュリティ

  PTR では接近したノードは全て MANET に参加出来てしまい，セキュリティを全く考慮していない．ある集団において特定のメンバーのみで MANET の構築ができるセキュリティの機構が必要である．

第4章　ETR

4.1　はじめに

前章でも述べたように，MANET における経路制御の多くはホップ数を指標とし，複数経路が存在した場合は最短経路を選択する．また経路状態を調べるのは経路探索時のみである．しかしこのような単純な経路制御では，実世界における MANET 環境に適応しない．なぜなら，無線環境では同一のメディア同士においても異種の無線メディア同士においても有線環境と異なり，使用する経路の電波品質により通信性能が大幅に変動し，電波品質が悪ければ転送性能は落ちてしまう．そのため最短経路の電波品質が悪ければ，必ずしも最短経路が最善な経路とは限らない．また使用中の経路における電波品質の状態は端末の移動により，大きく変動する．そのため経路探索時に最善であった経路の電波品質が劣化し，他に良好な経路が存在するにも関わらず，既存の経路制御では電波品質の劣化した経路を使い続けてしまう．

また MANET を構築するノードの所持する無線メディアが異なる場合，規格における最大転送性能が大きく異なる．転送性能が 1Mbps である Bluetooth の経路，また転送性能が 11Mbps である 802.11b の経路もホップを指標にした場合，同等の 1 ホップとみなしてしまう．そのため最短経路に Bluetooth 端末が存在し，その他の経路としてホップ数が最短経路より多い 802.11b の経路があった場合，Bluetooth を含む最短経路を選択してしまう．このような異種無線メディア間で構築される MANET では，ホップ数を指標とした経路選択は適さないと考えられる．

そこで無線メディアで構築されるネットワークである MANET 上で，効率の良い通信を実現するには電波品質を考慮した指標を用い，かつ状況が変化した場合でも対応できる動的な経路制御が必要である．

本章では，MANET 上で効率の良い通信を実現するためにソースルーティングを行うプロトコルで動作する，遅延とパケットロス率を指標にした経路制御機構 ETR (Estimeted-TCP-throughput maximization Routing Scheme) を提案する．

4.2　本章の構成

本章の構成として，まず関連研究について述べ，次に既存の MANET を構築するルーティングプロトコルの問題を示す．その後提案する ETR の設計・実装について述べ，最後に ETR の評価を行う．

4.3 関連研究

現在，MANET 環境を構築する多くのルーティングプロトコルが提案されている．しかし，DSR [6]，AODV [7] 等，そのほとんどの経路制御は単純にホップ数を指標として最短経路を選択する．

また本研究で提案する ETR と同じように，ソースルーティングを行う MANET 上の経路制御として，C. R. Lin が提案する QoS ルーティング [21] がある．この経路制御では経路選択の指標として TDMA 方式のフリースロット数を用いる．送信元は宛て先までの経路を要求する際に，必要なフリースロット数と共に経路要求パケットを送信する．もし，要求したフリースロット数以上を確保できる経路が存在した場合にのみ，フリースロットを予約しデータ送信を行う機構である．しかし指標はフリースロット数のみに限定しているため，要求したスロット数以上の確保が可能であっても，電波品質が悪い場合は通信性能は低下することになる．またフリースロット数を発見し，確保するのはデータ送信の最初のみであり，この方法では頻繁に経路の電波品質が変化する MANET 環境において不十分である．なぜならデータ転送の初期段階で転送性能が良好であっても，ノードの移動により電波品質が悪化した場合，転送性能は落ちてしまうからである．しかし ETR は経路の指標としてパケットロス率と遅延を用い，かつ経路に定期的なプローブを行い電波品質が変化した場合でも柔軟に対応できる．

C-K. Toh が提案する ABR (Associativity-Based Long-lived Routing) [22] も ETR と同様に経路選択の指標としてホップ数以外の指標を用いる．このルーティングプロトコルは経路選択の指標として経路の存続している時間用いる．ABR では各ノードがビーコンを一定間隔で隣接ノードに送信する．隣接ノードは受信したビーコンを数えており，送信元が送信した経路要求パケットを転送する際に，ビーコンの回数を付加し転送する．送信元が経路を選択する際，ビーコンの回数を指標とし経路選択を行うことで，頻繁に経路の再構築をせずにデータ送信ができる．ABR はネットワーク中に止まっているノードが存在するか，もしくは全てのノードが同じ方向に移動している場合高い転送性能が期待できる．しかし全てのノードがランダムに移動する場合，高い転送性能は期待できない．ETR はノードの動きに依らず TCP 転送性能が高いと予想される経路でデータ送信ができる．

4.4 MANET における最短経路選択の問題点

本節では MANET において多くのプロトコルで用いられるホップ数を指標とした経路選択を使用した場合に発生しうる問題点について述べる．

4.4.1 同一無線メディアにおける問題点

前節で述べたように MANET を構築するプロトコルの多くは同じ宛て先に対し複数経路があった場合，ホップ数を指標にして最短経路を選択する．MANET を構築する

無線ノードが同一の無線メディアを所持している場合，全てのノードの最大転送性能はほぼ同一だと考えられる．そのため，ホップ数を指標とした経路選択はロスが起きないと仮定した場合，ホップ数の少ない経路つまり遅延が小さい経路を選択することで通信性能において最善な経路を選択でき，また実装も容易である．しかし通信性能は経路の遅延のみではなくパケットロス率によっても大きく変動する．MANET 環境では経路のパケットロス率・遅延が，電波状態やノードの位置により変動することが予想される．そのため最短経路が通信性能において最善な経路とは限らない．

最短経路が通信性能において必ずしも最善な経路ではないことを実証するため，MANET を構築するプロトコルである DSR (Dynamic Source Routing) [6] を用いて実験を行った．本節では，まず実験に用いた DSR の概要を述べ，次に実験について述べる．

DSR の概要

DSR は，CMU の Monarch プロジェクト [23] で提案された MANET を構築するプロトコルである．DSR では送信パケットのヘッダに送信先から宛て先までの経路を入れ，データの配送をするソースルーティングを行う．送信元の経路キャッシュに宛て先までの経路が存在しない場合は，経路要求パケットをブロードキャストにより全てのノードに対して送信する．目的の宛て先または，宛て先までの経路をキャッシュに保持しているノードは経路要求パケットを受信すると経路応答パケットを送信先へ送信する．発見された経路は全てキャッシュに保持され，明示的に経路が壊れたと分かるまでキャッシュから経路を削除しない．もし同じ宛て先まで複数経路が存在した場合は，ホップ数の短い経路が選択される．

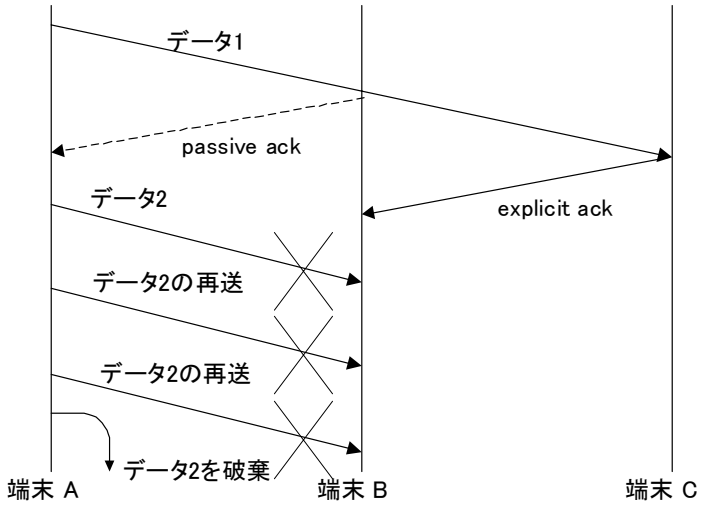

図 4.1: DSR のデータ送信手順

DSR 層は IP 層（ネットワーク層）の一部として存在している．MAC 層 (リンク層) で受信したパケットは IP 層に渡され，プロトコル番号により DSR 層に渡される．ま

た，DSR 層ではホップ毎のデータ転送処理も行っている．DSR 層では，ホップ毎のデータ転送の際，ある一定時間に DSR 層の ACK パケットによる返答がない，または自分が送信したパケットが次のホップへ転送されていない場合にパケットの再送を行い，2 回の再送に失敗するとそのパケットを破棄する．

図 4.1 は宛て先までの通信経路が決定した後のデータ送信手順を示したものである．端末 A は端末 B を経由し端末 C にデータを送信する．端末 A がデータ 1 を送信し，端末 B は端末 C へデータ 1 を転送する際，端末 A はプロミスキャスモードによりデータ 1 が端末 C へ転送されている事が分かる (passive ack)．これにより，端末 A は端末 B へデータ 1 の転送が成功したと判断する．次に，端末 C はデータ 1 を受信後，宛て先端末である端末 B へ DSR 層における ACK(explicit ack) を送信する．これより端末 B は端末 C へデータ 1 の転送が成功したと判断する．次に端末 A が端末 C へデータ 2 を送信する．しかし端末 B からの passive ack が受信できないため，再送を 2 回行った後パケットを破棄している．

パケットロス率と TCP 転送性能の関係

MANET において最短経路が通信性能において必ずしも最善な経路ではないことを実証するため，DSR を用いて以下のような実験を行った．

MANET 環境は，FreeBSD3.3R+PAO および MANET を構築するルーティングプロトコルの一つである DSR を用いて構築した．実験内容として，図 4.2 に示されるネットワークトポロジ上で，パケットロス率と TCP 転送性能の関係，TCP シーケンス番号の推移を調べた．TCP 転送性能の測定には netperf [20] を利用して 10 秒間の TCP データの送受信を 10 回行い，TCP 転送性能の平均値を求めた．中間ノードにおいてパケットをランダムに破棄させるコードを加え，DSR 層でパケットロスを発生させた．端末 S，A，D で構成される 2 ホップの経路 A，端末 S，B，C，D で構成される 3 ホップの経路 B を構築し，以下の 3 通りの条件で TCP 転送性能を測定した．各中間ノードの DSR 層におけるパケットロス率を $p(\%)$ とする．

条件 1 経路 A を用い，端末 A において $0 \leq p \leq 40$ のロスを発生させる．

条件 2 経路 B を用い，端末 B において $0 \leq p \leq 40$，端末 C において $p = 0$ のロスを発生させる．

条件 3 経路 B を用い，端末 B において $p = 0$，端末 C において $0 \leq p \leq 40$ のロスを発生させる．

この結果を図 4.3 に示す．ここで示されているパケットロス率は終端ノード間のパケットロス率ではなく，パケットロスを発生させたノードの DSR 層におけるパケットロス率である．条件 1，2，3 よりパケットロス率が高くなるにつれ TCP 転送性能が低下する事が分かる．また条件 1 と条件 2，3 を比較すると，パケットロス率によっては，経路 A の転送性能よりも経路 B の転送性能の方が良い状況が発生することが分かる．例

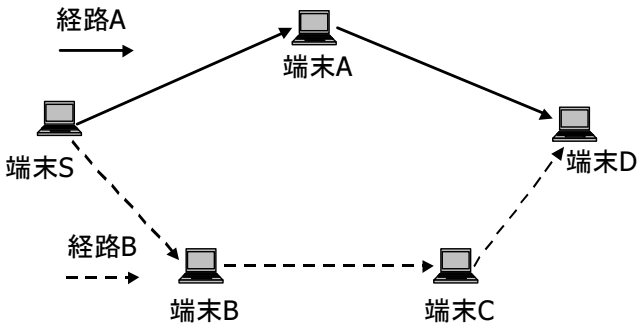

図 4.2: 実験ネットワークのトポロジ構成図

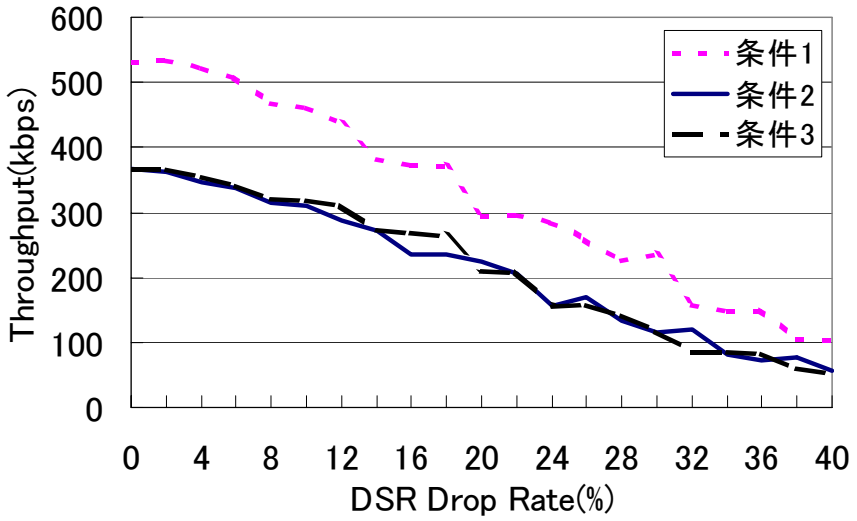

図 4.3: パケットロス率と TCP 転送性能

えば，端末 B，端末 C のパケットロス率が 0%，端末 A のパケットロス率が 20%以上だったならば，経路 A よりも経路 B を使用した方が TCP 転送性能は向上する．

次に経路 A と経路 B での TCP シーケンス番号の推移を示す．先の実験と同様に netperf を用い TCP データの送受信を 50 秒間行った．経路 A の端末 A において 40%の，経路 B の端末 B において 0%の DSR 層におけるパケットロスを発生させる．その結果を図 4.4 に示す．2 ホップの経路 A よりも 3 ホップの経路 B の方が傾きが大きい．このことからパケットロス率が 40%である 2 ホップの経路よりもパケットロス率が 0%である 3 ホップの経路の方が TCP 転送性能が良いことが分かる．これらの実験により，同一無線メディアにおける MANET では必ずしも最短経路が最善な経路ではないこと

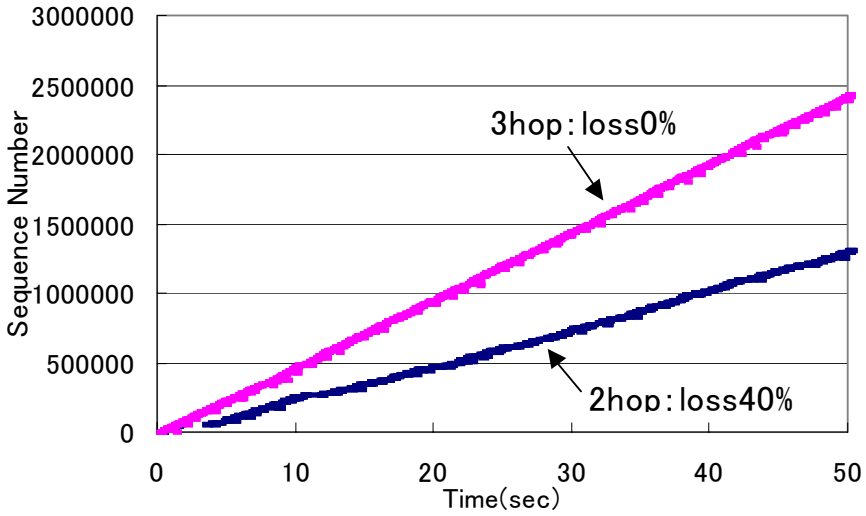

図 4.4: TCP シーケンス番号の推移

が実証された.

4.4.2 異種無線メディアにおける問題点

同一無線メディアにおける最短経路選択では，各ノードの最大転送性能がほぼ同一であるが，異種無線メディアにおける MANET では同一な電波品質の状況下においても，最大転送性能は大きく異なってしまう．表 3.14 で示したように，無線メディアによって規格上の最大転送性能は最大 54Mbps から最小で 1Mbps と大きな差がある．

経路選択の指標としてホップ数を用いた場合，複数の無線メディアが存在する MANET では，54Mbps の最大転送性能を持つ 802.11a，1Mbps の最大転送性能である Bluetooth の経路も同等の 1 ホップとして扱ってしまう．そのため，図 4.5 の様に，Bluetooth を所持したノードを含む 3 ホップの経路，802.11a のみの 5 ホップ経路が送信元から宛て先まで経路が存在した場合，3 ホップの経路を選択してしまう.

異種無線メディアにおけるホップ数と TCP 転送性能の関係

異種無線メディアが存在する MANET において経路選択の指標としてホップ数を用いた場合，最短経路が通信性能において最善ではないことを実証する．そのために前章で述べた PTR を用い，実験ネットワークを構築した．実験ネットワークでは 802.11b と Bluetooth を使用し，前小節と同様に netperf を使い 10 秒間の TCP データの送受信を 10 回行った．前章の評価から Bluetooth を経路に含んだ場合，1 ホップもしくは 2 ホップの経路でも転送性能に大きな差がないことから，Bluetooth を使用した実験ネッ

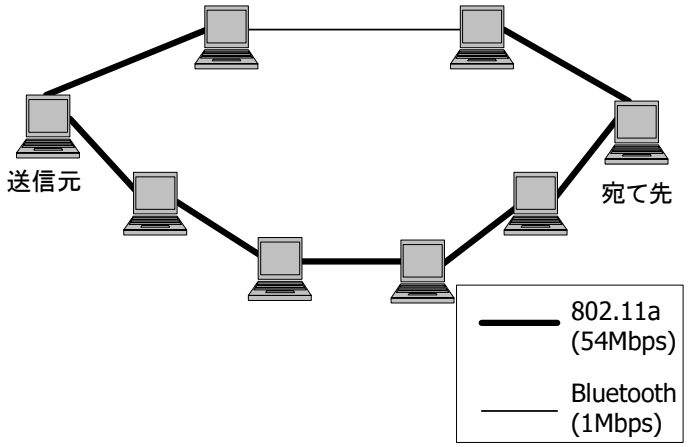

図 4.5: ネットワーク例

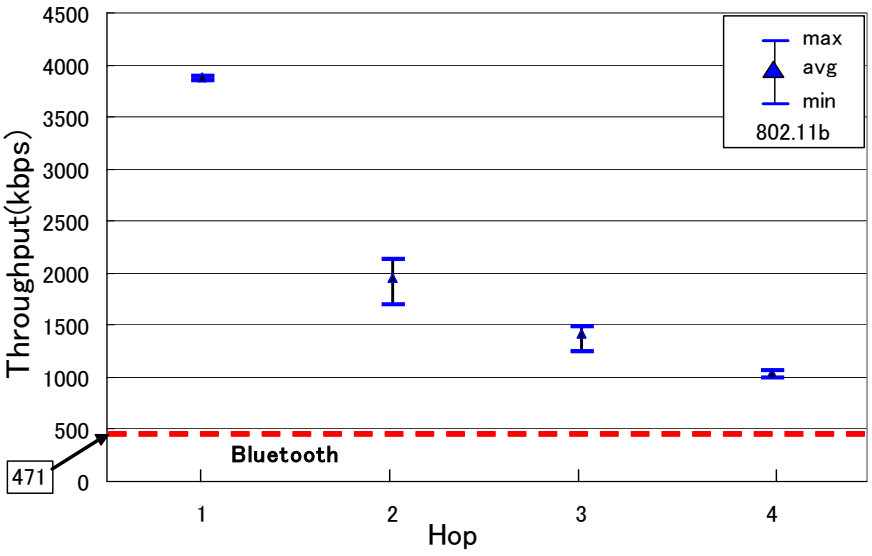

図 4.6: Bluetooth，802.11b の TCP 転送性能

トワークは Bluetooth のみの 1 ホップの経路を構築し評価を行った．また 802.11b では 802.11b のみを使用し，1 ホップから 4 ホップまでの経路を構築し，各経路で評価を行った．

結果を図 4.6 に示す．横軸はホップ数，縦軸は TCP の転送性能 (kbps) である．802.11b における転送性能の最大，平均，最小をプロットした．また Bluetooth では，Bluetooth の 1 ホップにおける TCP 転送性能の平均値を点線で示している．

802.11b，Bluetooth における規格上の最大転送性能はそれぞれ 11Mbps，1Mbps と

なっている．そのため，802.11b，Bluetooth の 1 ホップにおける転送性能を比較した場合，転送性能は 802.11b の方が Bluetooth に比べ格段に良い．また 802.11b ではホップが増える毎に転送性能が低下していることが分かる．しかし 802.11b と Bluetooth を比較した場合，802.11b のみの 4 ホップの経路でも転送性能は平均 1041kbps であり，1 ホップの Bluetooth における転送性能の約 2 倍となる．

この実験からもし送信元と宛て先が 802.11b 同士の無線メディアを所持していた場合，Bluetooth を含む最短経路を使用するより，最短経路より 4 ホップ多い場合でも 802.11b のみの経路を使用した方が，転送性能が高いと分かった．つまり異種無線メディア間においても，経路の指標をホップ数とした最短経路は必ずしも，転送性能において最善な経路とならないことが実証された．

4.5 ETR の設計

ETR は同じ宛て先に複数の経路が存在した場合，TCP 転送性能を予測し TCP 転送性能が良い経路に切り替える経路制御機構である．本節では ETR で用いる TCP 転送性能の指標，指標の算出方法，ETR のデータ送信手順について述べる．

4.5.1 TCP 転送性能の指標

同一の宛て先に対し複数の経路が存在した場合，最も通信効率が良い経路を選択するべきである．ETR では，各経路の TCP 通信性能を予測し，通信効率が良いと期待できる経路を選択する．現在，インターネットにおける IP トラヒックの 95%以上は TCP である [24]．このことから MANET においても同様な状況が起りうると想定し，本研究では TCP の通信性能に注目する．

前節の実験で，DSR 上でのパケットロス率と TCP 転送性能の関係，また無線メディアにおける TCP 転送性能の関係を示した．ETR は TCP 通信性能の指標として具体的には TCP 転送性能のモデルを用い，TCP の最大転送性能の理論値を計算し，TCP 転送性能が良い経路を選択する．そのため，ソースルーティングを行うプロトコルであれば ETR は実装が可能であり，特定のルーティングプロトコルに依存しない．一般的に知られている TCP 転送性能のモデルとして，M. Mathis が提唱したモデル [25] を基に，S. Floyd が変更を加えたモデル (1) がある [26]．ETR はこのモデルを用い，TCP 転送性能の理論値を求める．

$$T \geq \frac{1.5\sqrt{2/3} * B}{R * \sqrt{p}} \tag{4.1}$$

T : TCP の転送性能 (bit/sec)

B : TCP のセグメントサイズ (bit)

R : コネクションの最小遅延 (sec)

p : コネクションのパケットロス率

式 (1) を用い TCP のセグメントサイズ，コネクションの最小遅延，コネクションのパケットロス率から TCP 転送性能の最大値を算出できる．

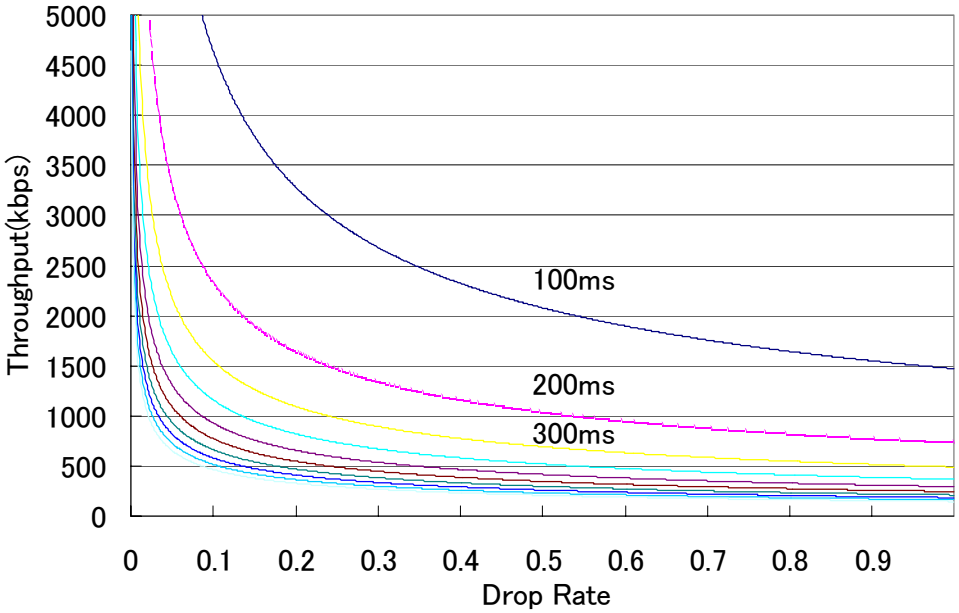

図 4.7: パケットロス率，遅延と TCP 転送性能の関係

図 4.7 は式 (1) を用い，セグメントサイズを 1500 バイト，遅延が 100ms 毎に 100ms から 1000ms である場合の TCP 転送性能の理論値を算出したものである．ただし，ここでのパケットロス率は終端ノード間の TCP におけるパケットロス率である．図 4.7 より，パケットロス率がある一定以上に達すると，パケットロス率の変動による TCP 転送性能の変動が少なくなることが分かる．そのため，遅延が短い経路を選択することで TCP 転送性能が良い経路を選択できる．

4.5.2 指標の算出方法

ETR では経路選択の指標である，式 (1) を用いた TCP 転送性能の理論値を算出するために，終端ノード間のパケットロス率と遅延が必要になる．また，指標の取得には送信元が宛て先までの全経路を把握しなければならないため，ETR では前提として送信元が宛て先までソースルーティングを行うものとする．

送信元が終端ノード間のパケットロス率と遅延を取得し，TCP 転送性能の理論値を算出するまでの手順の概要を図 4.8 に示す．図 4.8 では送信元ノード S から宛て先ノー

ドDへの経路が2つ存在し，SからDへTCPデータを送信している．以下に指標の算出手順を記述する．

1. 各ノードは現在までに転送したパケットからリンク毎のパケットロス率を計算する．送信元であるSは宛て先Dに対し，ロス率要求パケットを送信する．
2. 中間ノードはロス率要求パケットを受信すると，パケットを転送してきたノードとのリンクのロス率をロス率要求パケットへ入れ，Dに転送する．
3. Dがロス率要求パケットを受信したら，パケットロス率をロス率応答パケットに入れSに送信する．
4. ロス率応答パケットを受信した中間ノードもパケットロス率をロス率応答パケットに追加し，送信元まで転送する．
5. ロス率応答パケットを受信したSはノード毎のパケットロス率から終端ノード間のパケットロス率と，ロス率要求パケットを送信し応答されるまでの時間から遅延を算出する．
   経路の最小遅延とパケットロス率から式(1)を用い，SからDへのTCP転送性能の理論値を算出する．

2回目以降に送信元Sが同じ経路へ，ロス率要求パケットを送信する場合は，Sは経路の最小遅延をロス率要求パケットへ入れ宛て先Dに送信する．これより宛て先Dも，DからSへのTCP転送性能の理論値を算出できる．

各ノードにおけるパケットロス率の測定方法

パケットロス率の測定方法ついて述べる．各ノードは隣接ノードのアドレス毎にパケットロス率を管理し，転送したパケットからパケットロス率を測定する．各ノードはパケットの転送，またはある宛て先にデータ送信をしている際に，パケットを送信した隣接ノードに対して一定期間ロス率を測定する． またETRではロス率の測定の際，各ノードにおいて一定間隔毎にそれまで計測していたロス率・パケット数を破棄し新たに測定する．

ロス率を測定する際，無線環境ではその変動が大きいと考えられる．そのため測定したロス率をそのまま指標として用いた場合，ロス率の変動により経路切り替えが頻繁に起ってしまい，TCP転送性能が悪化する状況が起りうる．そこでETRでは以下の式を用いロス率の平滑化を行い，その値を指標として用いる．

$$sloss = \alpha \times sloss + (1 - \alpha) * loss \tag{4.2}$$

$sloss$: 平滑化されたロス率

$loss$: ロス率の実測値

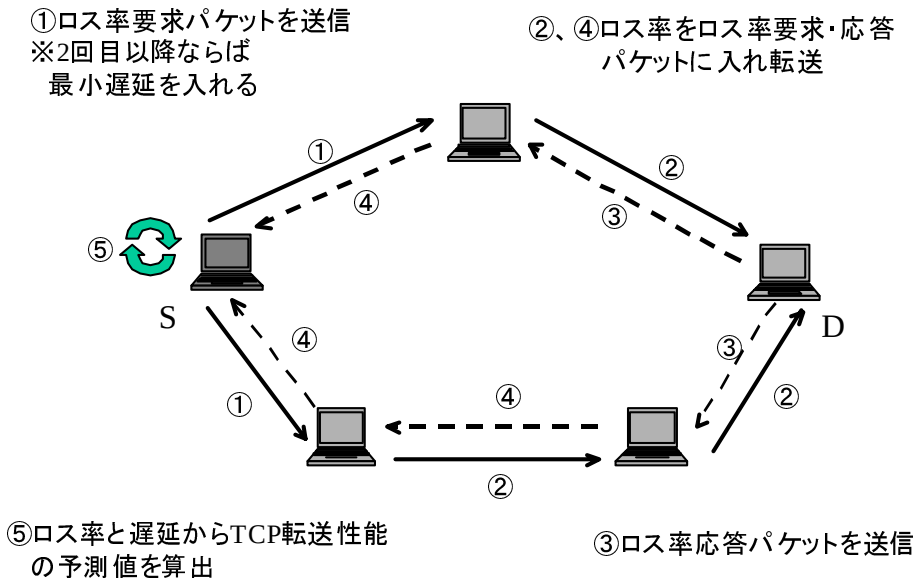

図 4.8: 指標の算出手順

この式ではそれまで計測していた値と新しい実測値から平滑化されたロス率を求める．例えば$\alpha = 0.8$の場合，新しく算出されるロス率の80%は以前に測定した値だが，20%は新しいロス率の実測値から導き出される．

遅延の測定方法

遅延の測定方法について述べる．送信元はロス率要求パケットを送信する際，そのパケットへタイムスタンプを行う．宛て先はロス率要求パケット中のタイムスタンプをロス率応答パケットに入れ，送信元に送る．送信元がロス率応答パケットを受信した時間と，ロス率応答パケット中のタイムスタンプの差を遅延とする．ただし，式に用いる遅延にはその経路での最小遅延とする．

4.5.3 ノードのデータ送信手順

送信元がTCPデータ送信を行なう手順について説明する．ノードがある宛て先に対して初めてTCPデータを送信する場合，遅延やパケットロス率が判明していないため，経路選択の指標であるTCP転送性能の予測値を算出できない．この場合は，送信元はまず最短経路を用いデータを送信する．ただし，データを送信する宛て先の経路

に対し，以前に算出したTCP転送性能の予測値が存在し，かつその値が最近算出されたものであれば，それを経路選択の指標として用いる．

MANETではノードの移動により遅延やパケットロス率が大きく変動する．ETRでは そのような状況変化に対応するため，送信元は同じ宛て先に対し複数経路が存在する場合，一定間隔でロス率要求パケットの送信を行う．そのため送信元はデータの送信を開始し，その宛て先に対し複数経路が存在した場合，一定時間毎にロス率要求パケットをそれぞれの経路に送信し指標を算出する．その後，送信元はロス率要求パケットの送信によって算出したTCP転送性能の予測値を指標として最善な経路を選択する．

4.5.4 各モジュール説明

ETRモジュールはMANETのルーティングプロトコル中に存在し，データ送信，転送，宛て先ノードにおいて，それぞれ以下のようにETRモジュールが呼び出される．

- 送信元ノード：ロス率要求送信・ロス率応答受信・経路選択モジュール
- 転送ノード：パケット転送モジュール
- 宛て先ノード：ロス率要求受信・ロス率応答送信モジュール

図 4.8 が示す指標の算出手順に従い，各手順に呼び出されるモジュールについて順を追い説明する．

1，ロス率要求パケットを送信

送信元ノードはロス率要求パケットを送信するため，ロス率要求送信モジュールを呼び出す．

ロス率要求送信モジュール　ロス率要求送信モジュールはノードがデータ送信を行っている間，定期的に呼び出される．このモジュールはロス率要求パケットを送信する機能がある．ロス率要求パケットは表 4.2 に示す情報が格納される．まずパケットがロス率要求，応答パケットのどちらであるかを示す *Packet Type* に，ロス率要求を示す *Loss Request* を格納する．*Packet Type* に格納する情報は表 4.1 が示すように，ロス率要求を表す *Loss Request* だけでなく，ロス率応答を表す *Loss Reply* もある．

その後，パケットの長さを *Packet Length* に入れ，どの経路に対して送信したかを識別するため *Route ID* に経路情報を加え，送信元が遅延を測定するためのタイムスタンプを *Time Stamp* に格納する．もしロス率要求パケットを送信する経路の最小遅延が判明している場合は *Minimum Delay* に遅延を入れ，宛て先ノードに対しロス率要求パケットを送信する．

表 4.1: パケットタイプ

項目	説明
Loss Request	ロス率要求
Loss Reply	ロス率応答

表 4.2: ロス率要求パケット

項目	説明
Packet Type	パケットタイプ
Packet Length	パケット長
Route ID	経路の識別子
Minimum Delay	最小遅延
Time Stamp	タイムスタンプ
Packet Loss	パケットロス率

2・4，ロス率をロス率要求，応答パケットにいれ転送

ETR パケット (ロス率要求，応答パケット) を受信した中間ノードは，パケットの転送処理を行うためにパケット転送モジュールを呼び出す.

パケット転送モジュール　パケット転送モジュールでは受信したロス率要求，応答パケットの *Packet Loss* に隣接ノードとのロス率を格納し送信を行うモジュールである．ただし格納するロス率はロス率要求，応答パケットを送信したノードとのロス率を格納する.

3，ロス率を応答パケットを送信

宛て先はロス率要求パケットを受信した際，ロス率応答パケットを送信するため，ロス率要求受信，ロス率応答送信モジュールを呼び出す.

ロス率要求受信モジュール　ロス率要求受信モジュールでは，ロス率要求パケットに存在する各ノードのロス率である *Packet Loss* より，経路全体のロス率を算出し，経路表に全体のロス率を格納する．また *Minimum Delay* に値が入っていた場合，その値は経路の最小遅延となるので，最小遅延も経路表へ格納する．その後，宛て先ノードは送信元ノードへロス率応答を送信するためにロス率応答送信モジュールを呼び出す.

ロス率応答送信モジュール　宛て先ノードはロス率要求パケットを受信した際，送信元へロス率応答を送信しなければならない．そのため宛て先ノードはロス率応答送信モジュールを呼び出す．

ロス率応答送信モジュールは表 4.3 に示す情報を格納したロス率応答パケットを送信する．まず宛て先ノードは，*Packet Type* にロス率応答パケットを示す *Loss Reply* をいれ，どの経路に対して送信したかを識別するために *Route ID* に経路情報を加え，パケット長を *Pacetk Length* へ，ロス率要求パケットに含まれていたタイムスタンプを *Time Stamp* へ格納し，ロス率応答パケットを送信元へ送信する．

表 4.3: ロス率応答パケット

項目	説明
Packet Type	パケットタイプ
Packet Length	パケット長
Route ID	経路の識別子
Time Stamp	タイムスタンプ
Pacekt Loss	パケットロス率

5，ロス率と遅延から TCP 転送性能の予測値を算出

送信元ノードはロス率応答を受信し，TCP 転送性能の予測値を算出するためにロス率応答受信モジュールと経路選択モジュールを呼び出す．

ロス率応答受信モジュール　ロス率応答受信モジュールでは，ロス率応答パケットに存在する各ノードのロス率である *Packet Loss* より，経路全体のロス率を算出しする．また受信した時間から *Time Stamp* の値を減算し，経路の RTT を算出する．そして，経路におけるロス率と遅延を経路表に格納する．

経路選択モジュール　経路選択モジュールはルーティングプロトコルの経路表より宛て先までの経路を取得し，もし複数経路がある場合は遅延とロス率より TCP 転送性能の論理近似値を算出する．そして TCP 転送性能が一番高いと予測される経路が選択される．もし経路のロス率や遅延がない場合は最短経路が返される．

4.6　ETR の実装

本節では，ETR の実装について述べる．今回実装が間に合わず前章の PTR を使用できないため，他の MANET のルーティングプロトコルを用いることにした．

4.6.1 実装環境

本研究ではETRをFreeBSD4.2Rで動作するDSR上に実装した．また無線メディアには802.11b規格のMelco WLI-PCM-L1 [19]を使用した．

4.6.2 各ノードにおけるETRの実装

ここでは送信元，中間，宛て先の各ノードにおけるETRの実装について述べる．

送信元ノード

送信元ノードでは，データ送信時にTCPの理論近似値を算出し経路選択を行う．また，データの送信を行っている間に定期的にロス率要求パケットの送信を行い，宛て先からのロス率応答パケットの処理を行う．

```
・Source Node:
D : Destination address
Send data to Destination(D):

if(Same_Destination_Route(D) > 1){
    Route_Use_Flag_UP(D);
    if(TCP_Estimation_State(D)== NEW){
        Route_to_Dst = Max_TCP_Route(D);
        Send_Data(Route_to_Dst);
    }
    if(TCP_Estimation_State(D) == (NULL || OLD)){
    Route_to_Dst = Shortest_Route(D);
    Send_Data(Route_to_Dst);
    }
}
Send_Data(Shortest_Route(D));
```

図 4.9: 経路選択

経路選択　図4.9は送信元がデータ送信時に経路を選択を処理する疑似コードを示している．先ず送信元データを送信する際，Same_Desitination_Route()により，その宛て先に対する経路が1つ以上つまり複数存在するか調べる．もし複数存在する場合には，Route_Use_Flag_UP()によりデータを送信する宛て先に対する経路のフラグを上

げ，フラグを UP した時間も経路表に入れる．その後，TCP_Estimation_State() によりその経路に対する TCP 転送性能の予測値が最近算出されたものか (NEW)，時間が経ったものか (OLD)，存在しないか (NULL) を調べる．返された TCP 転送性能の予測値の状態が NEW であれば，それを経路選択の指標として用い，TCP 転送性能が最も良い経路を使いデータ送信を行う．逆に，TCP 転送性能予測値が OLD または NULL であったならば，経路選択の指標としてその値を使用せずに，最短経路を選択してデータ送信を行う．

```
Route : Route ID

while(1){
    Route = Check_Use_Time();
    if(Route)
       Route_Use_Flag_DOWN(Route);
}

Each Interval time:
Route = Check_Use_Flag(Route);
if(Route){
   Send_Etr_Req(Route);
}
```

図 4.10: ロス率要求送信

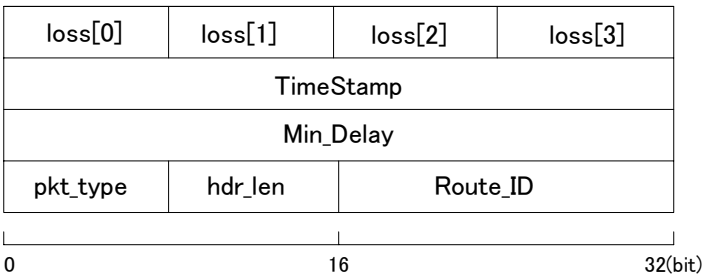

図 4.11: ロス率要求パケット

ロス率要求送信　図 4.10 にロス率要求を送信する疑似コードを示す．まず Cheak_Use_Time() でフラグが UP されてからの時間を調べる．ある一定時間が経っていた場合，その経路は最近，参照または使用されていないと判断し，その経路のフラグを下げる．

Each Interval time以下はある一定時間毎に呼び出される関数である．Cheak_Use_Flag() により，フラグがUPされている経路を経路表より探す．フラグがUPされている経路は最近参照されたまたは使用されたことを示し，もしUPされている経路があれば，Send_Probe() を呼びその経路に対しロス率要求パケットを送信する．

送信するロス率要求パケットは図 4.11 で示すパケットとなる．送信元ノードがロス率要求パケットに格納する値は以下となる．

- `ETR_Req->pkt_type = Loss_Req`
- `ETR_Req->hdr_len  =` ホップ数により可変
- `ETR_Req->Route_ID  =` 経路の識別子
- `ETR_Req->Min_Delay =` 最小遅延
- `ETR_Req->TimeStamp =` 送信時のタイムスタンプ

まずパケットの種類を表すpkt_typeはロス率要求を示すLoss_Reqとなる．hdr_lenは送信する経路のホップ数により，ロス率を格納するため必要なloss配列の大きさが変わるので可変となる．Route_IDはロス率要求の送信に使用した経路を識別するため，各経路に割り当てられている識別子となる．そして，Min_Delayには経路の最小遅延が存在した場合に限り遅延が格納され，送信時の時間をTimeStampの値に入れる．

ロス率応答受信　図 4.12 にロス率応答受信の疑似コードを示す．受信するロス率応答パケットは図 4.13 となる．Calculate_Loss_Rate() はロス率応答パケットのloss配列より，経路全体のロス率算出する．そしてCalculate_Delay() によりロス率応答を受信した時間からTimeStampの値を減算し遅延を計算する．最後に経路表からRoute_IDに対応する経路へ遅延とロス率を格納する．

```
Probe : Loss Request,Reply packets of ETR

Receive Probe Reply:

Calculate_Loss_Rate(Probe);
Calculate_Delay(Probe);
```

図 4.12: ロス率応答受信

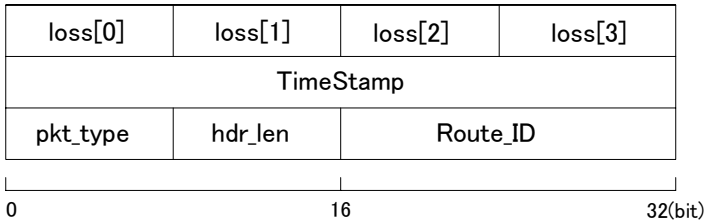

図 4.13: ロス率応答パケット

中間ノード

中間ノードでは受信したロス率要求，応答パケットへ隣接ノードのロス率を格納する処理を行う．

パケット転送モジュール　図 4.14 は中間ノードで受信したロス率要求・応答パケットを処理する疑似コードを示している．中間ノードではロス率要求・応答パケットの両方に対して同じ処理を行う．Insert_Packet_Loss() でロス率要求パケットまたはロス率応答パケットを送信したノードとのパケットロス率を loss 配列に加え，経路における次のノードへ転送する．

```
・Intermediate Node:
Receive Probe from S to D:

Insert_Packet_Loss(Probe);
Forward_Next_Hop(Probe);
```

図 4.14: パケット転送モジュール

宛て先ノード

宛て先ノードではロス率要求パケットを受信した際，送信元へロス率応答パケットを送信する処理を行う．

ロス率要求受信・ロス率応答送信　図 4.15 は宛て先がロス率要求パケットを受信した後の処理を疑似コードで示したものである．送信元はその経路に対し初めてロス率要求パケットを送信する場合，最小遅延を算出できないため，ロス率要求パケットに

おける最小遅延の値はNULLになる．宛て先ではロス率要求パケットを受信した際，Exist_Min_Delay() でロス率要求パケットにその経路の最小遅延が存在するかを調べ，最小遅延が存在すればRoute_IDに対応する経路に対する経路に遅延を格納する．また経路におけるロス率も同様にCalculate_Loss_Rate() によりロス率要求パケットに含まれるloss配列から，経路全体のロス率を算出し，対応する経路へロス率を格納する．

ロス率要求を受信した宛て先ノードは最小遅延の有無に関わらずロス率要求パケットを送信したノードへ，Return_Etr_Reply() を呼びロス率応答パケットを送信する．ロス率応答パケットのメンバは以下の様に設定される．

- `Etr_Rep->pkt_type = Loss_Rep`
- `ETtr_Rep->hdr_len  = (Etr_Req->hdr_len)-8`
- `Etr_Rep->Route_ID  = Etr_Req->Route_ID`
- `Etr_Rep->TimeStamp = Etr_Req->TimeStamp`

まずパケットの種類を表すpkt_typeはロス率応答を示すLoss_Reqとなる．hdr_len，Route_ID，TimeStampの値はロス率要求パケットの値がそのまま格納される．

```
・Destination Node:

Etr_Req : Loss Request packet of ETR
Etr_Rep : Loss Reply packet of ETR

Exist_Min_Delay(Etr_Req);
Calculate_Loss_Rate(Etr_Req);
Return_Etr_Reply(Etr_Rep);
```

図 4.15: ロス率要求受信・ロス率応答送信

4.7 ETRの評価

本節ではFreeBSD4.2RでDSRにETRを実装したシステムと既存のDSRで，TCP転送性能を比較し評価を行った．ロス率要求パケットの送信間隔を変化させ，経路切り替え時間とオーバヘッドを評価した．実験環境としては図4.2のような端末S，A，Dで構成する2ホップの経路A，端末S，B，C，Dで構成する3ホップの経路Bから成るネットワークを構築した

4.7.1 TCP 転送性能の評価

実験ネットワークに以下の 2 条件で TCP データを送信し，TCP 転送性能の評価を行った．

条件 A 経路 A，B を用い，TCP 転送を開始し 20 秒後に端末 A において 40%のパケットロスを発生させる．

条件 B 経路 A，B を用い，TCP 転送を開始し 10 秒後に端末 A においてそれぞれ 10，20，30，40，50%のロスを発生させる．

条件 A，B 共にロス率要求パケットを送信する間隔は 3 秒とし，ロス率要求パケットを送信するノードは端末 S のみとした．また ETR が TCP 転送性能の理論値を算出するために用いるパケットロス率の値は，DSR 層で送信した全てのパケットに対する，DSR 層で再送が 2 回失敗し破棄した終端ノード間におけるパケット数の割合とした．

条件 A の結果を図 4.16 に示す．図 4.16 は通常の DSR (DSR) と，DSR に ETR を加えたシステム (DSR+ETR) の端末 S における TCP シーケンス番号の推移を比較したものである．

ロスの発生後，DSR+ETR では式 (1) より経路 A の方が経路 B に比べ TCP 転送性能下がると判断し，ロス発生後約 8 秒で経路 A から経路 B に切り替えた．そのため，パケットロスが発生する経路 A を使い続ける DSR に比べ，DSR+ETR では経路状態が良好な経路 B へ切り替えることで TCP 転送性能が向上していることが分かる．

条件 B の結果を図 4.17 に示す．図 4.17 は DSR+ETR と DSR の TCP 転送性能を比較したものである．30 秒間の TCP 転送の送受信を行い，TCP 転送性能の最大値，平均値，最小値を示している．DSR ではパケットロスが発生した後の TCP 転送性能である．DSR+ETR ではパケットロスが発生し，経路が経路 A から経路 B へ切り替わった後の TCP 転送性能である．

ロス率が 10，20%では，DSR，DSR+ETR ともに経路 A を使い続ける．ロス率が 30，40，50%では，DSR はそのまま経路 A を使用するが，ETR+DSR ではパケットロスが発生後経路 B を使用するようになった．ロス率が 10%の時，DSR+ETR は DSR に比べ 16% TCP 転送性能が下がってしまう．しかし，ロス率が 20%以上になると，DSR+ETR は DSR に比べて TCP 転送性能が上がる．ロス率が高くなるにつれ DSR と DSR+ETR の TCP 転送性能の差は大きくなり，ロス率が 50%のときには，TCP 転送性能が 93%向上した．

4.7.2 経路切り替え時間とオーバヘッドの評価

ロス率要求パケットの間隔を変化させ，経路の切り替わる時間とロス率要求・応答パケットによるオーバヘッドの評価を行った．実験ネットワーク上に経路 A，B を用い TCP データ送信を行い，送信を始めてから 10 秒後に端末 A において DSR 層におけるロス率 40%のパケットロスを発生させた．前小節と同様にロス率要求パケットを

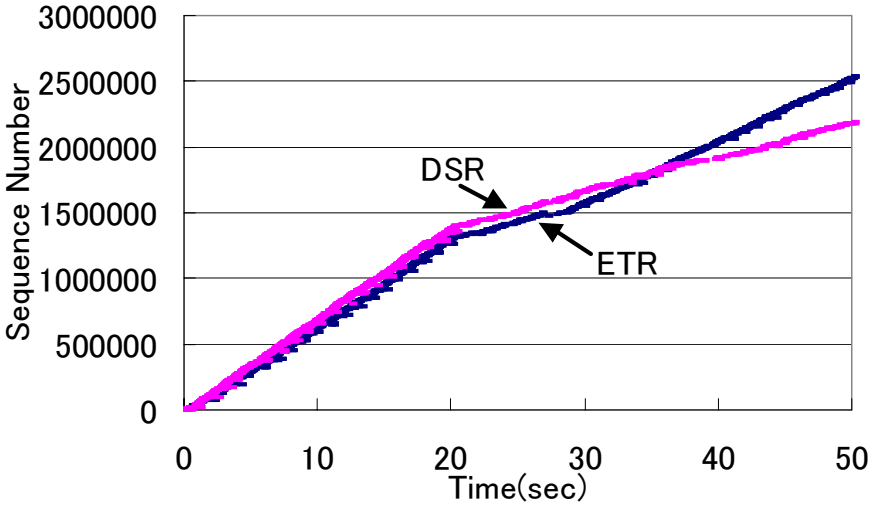

図 4.16: DSR+ETR，DSR の TCP シーケンス番号

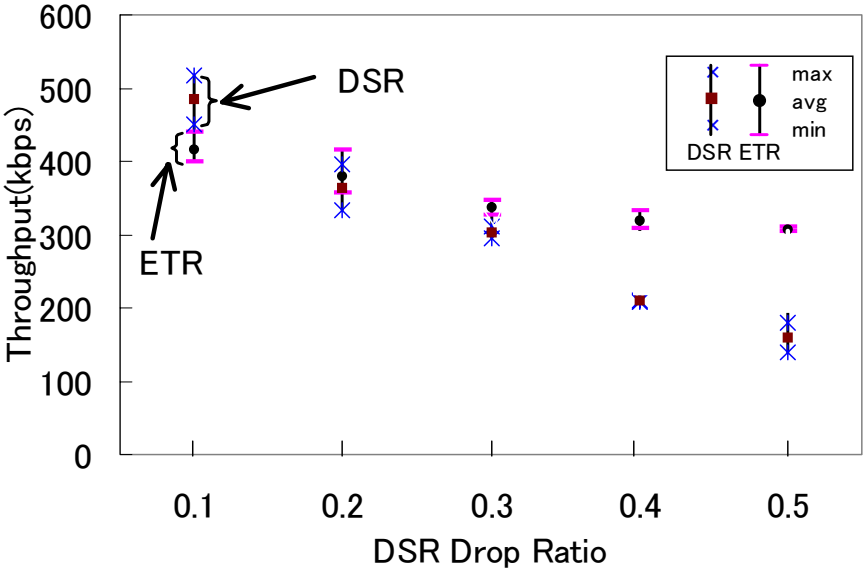

図 4.17: DSR+ETR，DSR の TCP 転送性能

送信するノードは端末 S とし，ロス率要求パケットを送信する間隔は 1，2，3，4，5 秒と変化させた．また同様に，ETR が TCP 転送性能の理論値を算出するために用いるパケットロス率の値は，DSR 層で送信した全てのパケットに対する DSR 層で再送が 2 回失敗し破棄した終端ノード間におけるパケット数の割合とした．

経路の切り替わる時間を図 4.18 に示す．この図は TCP データ送信開始後に 2 ホップの経路 A から 3 ホップの経路 B に切り替わった時間の最小値，平均値，最大値を示している．ロス率要求パケットの間隔が 1 から 4 秒まで約 11 秒前後で切り替わるが，

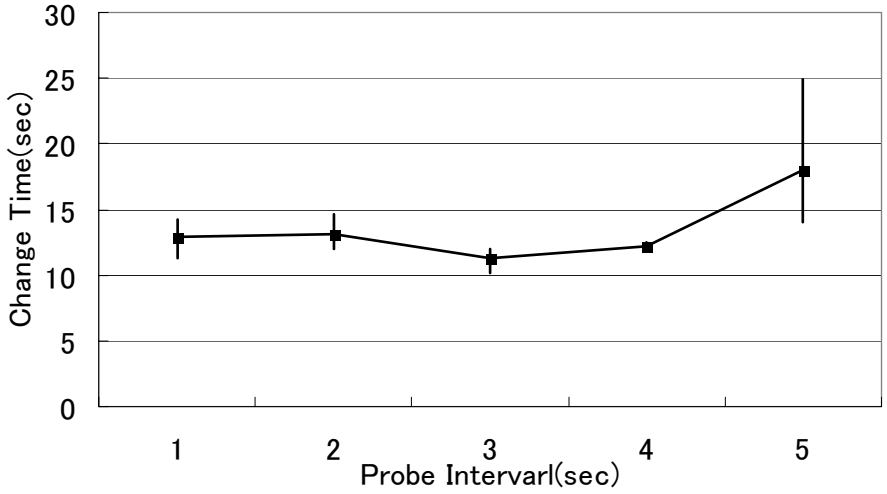

図 4.18: 経路切り替え時間

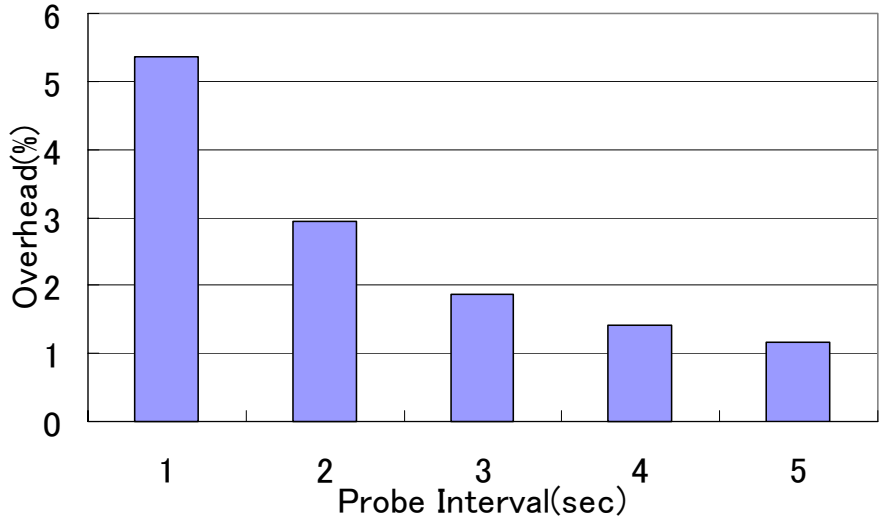

図 4.19: ロス率要求・応答パケットのオーバヘッド

5 秒の場合は切り替わる時間が大幅に遅れてしまうことが分かる．

次にロス率要求・応答パケットによるオーバヘッドの結果を図 4.19 に示す．この図は全体の送受信したパケット数に対して，ロス率要求・応答パケットの数とそのパケットへ返答する DSR 層での ACK パケット数を加えた割合を示したものである．パケット数は各ノードがパケットを送信した回数であり，複数ホップする場合は 1 回のホップ毎に 1 パケットを送信したと想定して計算した．ロス率要求パケットの間隔が長くなるごとに，ロス率要求パケットによるオーバヘッドが少なくなることが分かる．また，全体に占めるロス率要求・応答パケットのオーバヘッドは 2 から 5 秒の場合は 3%以下と少ないことが分かる．上記の 2 つの実験からロス率要求パケットの間隔は 2 ホップ

と 3 ホップの経路が存在する場合，3 から 4 秒の間が適当であると考えられる．

4.8 まとめと今後の課題

この章でのまとめと今後の課題について述べる．

4.8.1 本章のまとめ

本章では MANET 上でソースルーティングを行うプロトコルにおける，パケットロス率と遅延を指標とした経路制御機構である ETR の概要について述べ，指標の取得方法，算出方法について述べた．そして，Monarch プロジェクトで公開されている DSR 上で ETR を実装し，TCP 転送性能の評価を行った．ETR を用いる事で DSR 層におけるパケットロス率が 50%の場合 TCP 転送性能が 93%向上することが判明した．また ETR におけるプローブ間隔を変化させ，経路が切り替わる時間・オーバヘッドの関係を検証した．2 ホップの経路と 3 ホップの経路が存在する場合，プローブ間隔は 3〜4 秒が適切であると考えられる．

4.8.2 今後の課題

今後の課題として，以下のことが挙げられる．

- シミュレータの使用

  ETR のアルゴリズムを深く検証し評価を行うため，実機だけではなくシミュレータを用いノード数や移動パターンを変更させて ETR の性能を詳細に解析する．

- α の適切な値の検討

  ETR の設計で提案した，式 (2) における α の適切な値を検討する．

- ロス率要求パケットの送信間隔

  経路が 2 ホップの経路と 3 ホップの経路の場合にはロス率要求パケットの間隔は 3〜4 秒が適当であると考えられるが，ホップ数や経路が増えた場合の最適な間隔を解析し，動的にロス率要求パケットの送信間隔を変更するシステムを導入する．

- オーバヘッドの軽減

  ETR では経路のパケットロス率と遅延を測定する際，一定間隔でロス率要求パケットの送信を行うため，ネットワークへのオーバヘッドが高くなってしまう．より効率良く指標を取得する方法を提案し，プローブによるネットワークへのオーバヘッドを軽減させる．

- 負荷分散のメカニズム

  データを一つの経路だけで送信するのではなく，複数の経路に分散させてデータを送信するといった，ネットワーク全体の性能を考慮した負荷分散のメカニズムを導入する．

第5章　PTER

5.1　はじめに

4章で異種無線メディアにおけるアドホックネットワークでは経路選択の指標にホップ数を用いた場合問題が起こることを指摘した．そのためPTRで効率的なデータ送信を行う場合，ホップ数を指標としない経路選択機構の存在が不可欠である．そこで4章で述べた経路制御機構であるETRをPTR上で使用することで，異種無線メディア間において効率的なデータ送信が可能となる．

しかし4章で述べたTCP論理近似値を求め経路を選択する機構であるETRはソースルーティングを行うルーティングプロトコル上で実装が可能であるように設計されている．そのためロス率や遅延を測定する際，ETR独自のコントロールパケットを送信する．このETR独自のパケットをルーティングプロトコルのコントロールパケット，ヘッダ部分に包括してしまえば，ネットワークへのオーバヘッドが軽減できると考えられる．そこで，4章で述べたETRをPTRに統合したアドホックネットワークのルーティングプロトコルであるPTER (Protocol Transport Effective Routing) を本章で提案する．

本章の構成としてまずPTERの設計を述べ，最後に本章のまとめと今後の課題について述べる．

5.2　PTERの設計

異種無線メディアのMANETにおいて，TCPの理論近似値を用いた経路選択を行うルーティングプロトコルであるPTERの設計について述べる．

5.2.1　PTERの概要

PTERは3章で説明したPTRに，TCP転送性能の理論近似値を用いた経路選択機構であるETRを包括したルーティングプロトコルである．本小節ではPTERの概要について述べる，

3章で述べたようにPTRにおいてノードはまずデータ送信を行う際，送信元から宛て先までの経路を発見しなければならない．そのため経路要求を全ノードに対し送信する．そして宛て先が経路要求を受信した場合，送信元に対し経路応答パケットを送信する．その際，経路要求・応答のパケットヘッダに各リンクのロス率を格納する構造

体と，タイムスタンプを格納する場所を作成する．送信元は経路応答のロス率，タイムスタンプからTCP 転送性能の論理近似値が算出でき，データ送信の始めからTCP 転送性能が高いと予測される経路を用いデータ送信ができる．

経路の発見後，経路維持を行いながら通常データの送信を行う．その際，通常データに使用するパケットへッダへロス率，タイムスタンプを格納する．これよりデータ送信を行っている経路に対して，ETR 独自のコントロールパケットを送信する必要がなくなる．しかしデータ送信に使用していない同じ宛て先への経路に関しては，データを送信している期間ロス率要求パケットを送信しなければならない．もし現在使用している経路に比べ，TCP 転送性能の予測値が高い経路が存在した場合，送信元ノードは使用する経路を切替える．

5.2.2 PTR モジュールからの変更点

3 章で説明したPTR とPTER モジュールは基本的な機能は同様なものとなる．そこで，説明の重複を避けるため各モジュールの変更点を説明する．

経路発見モジュール

データ送信時に経路発見で使用される，経路発見モジュールの変更点について述べる．大きな変更点は以下に挙げる3 点となる．

1. 経路表
2. 経路要求パケット
3. 経路応答パケット

経路表　まず経路表の変更について述べる．PTER の経路表では各経路におけるパケットロス率，遅延の値が追加される．表 5.1 にPTER の経路表を示す．PTR からの変更点を太文字で示してある．PTER では経路選択にロス率と遅延が必要となる．そのため経路全体のパケットロス率である *Packet Loss* と，遅延である *Delay*，そしてパケットロス率と遅延から導き出されるTCP 転送性能の理論近似値である *TCP Throughput* を経路表に追加しなければならない．

経路要求パケット　次に経路要求パケットの変更点について述べる．PTER では経路要求パケットに送信時のタイムスタンプと各リンクのロス率を格納する．PTER の経路要求パケットは表 5.2 の情報となる．PTR からの変更点を太文字で示す．

送信元は経路要求パケットへ送信時のタイムスタンプを *Time Stamp* へ追加する．また経路要求を受信した各ノードは *Packet Loss* へ隣接するノードのパケットロス率を格納し転送する．

表 5.1: 経路表

項目	説明
Route ID	経路の識別子
Source Node ID	送信元ノードの識別子
Target Node ID	宛て先ノードの識別子
Hop Node ID	中間ノードの識別子
Interface	各ノードのインタフェース情報
Packet Loss	**パケットロス率**
Delay	**遅延**
TCP Throughput	**TCP 転送性能の論理近似値**

表 5.2: 経路要求パケット

項目	説明
Packet Type	パケットタイプ
Header Len	ヘッダの長さ
Source Node ID	送信元ノードの識別子
Destination Node ID	宛て先ノードの識別子
Route ID	経路の識別子
Req Source Node ID	経路要求を送信したノードの識別子
Target Node ID	ターゲットノードの識別子
Time Stamp	**タイムスタンプ**
Hop Node ID	中間ノードの識別子
Interface	各ノードのインタフェース情報
Packet Loss	**パケットロス率**

経路応答パケット　最後に経路応答パケットの変更点について述べる．PTER では経路応答パケットに経路要求パケットに格納されていたタイムスタンプ，各リンクのロス率を格納する場所が追加される．PTER の経路要求パケットは表 5.3 の情報となる．PTR からの変更点を太文字で示す．

宛て先は経路応答パケットへ経路要求パケットに格納されていたタイムスタンプを *Time Stamp* へ格納する，また経路応答パケットを受信した各ノードは *Packet Loss* へ隣接するノードのパケットロス率を格納し転送する．

表 5.3: 経路応答パケット

項目	説明
Packet Type	パケットタイプ
Header Len	ヘッダの長さ
Source Node ID	送信元ノードの識別子
Destination Node ID	宛て先ノードの識別子
Route ID	経路の識別子
Req Source Node ID	経路要求を送信したノードの識別子
Target Node ID	ターゲットノードの識別子
Time Stamp	**タイムスタンプ**
Hop Node ID	中間ノードの識別子
Interface	各ノードのインタフェース情報
Packet Loss	**パケットロス率**

データ送信モジュール

データ送信時に呼び出される，データ送信モジュールの変更点について述べる．大きな変更点はデータ送信時のパケットタイプが増える点となる

表 5.4: パケットタイプ

項目	説明
Route Request	経路要求
Route Reply	経路応答
Data	通常データ
Data+Loss Request	**通常データ+ロス率要求**
Data+Loss Reply	**通常データ+ロス率応答**
Ack	データ転送応答
Route Error	経路エラー

データ送信パケット　通常のデータを送信する際，PTER が付加するパケットヘッダは表 5.4 で示すように，*Data*，*Data+Loss Request*，*Data+Loss Reply* となる．

Data は 3 章で説明した PTR のパケットヘッダと同じである．PTER で追加される *Data+Loss Request*，*Data+Reply* は通常データパケットのヘッダに各リンクのロス率とタイムスタンプの値を格納する場所が追加されたヘッダとなる．

表 5.5: 通常データ+ロス率要求パケット

項目	説明
Packet Type	パケットタイプ
Packet ID	パケット識別子
Header Len	ヘッダの長さ
Data Len	データの長さ
Source Node ID	送信元ノードの識別子
Destination Node ID	宛て先ノードの識別子
Route ID	経路の識別子
Req Source Node ID	経路要求を送信したノードの識別子
Target Node ID	ターゲットノードの識別子
Time Stamp	**タイムスタンプ**
Delay	**経路の遅延**
Packet Loss	**パケットロス率**

送信元はデータ送信を行う際，通常データのパケットヘッダをデータに付加させ宛て先まで送信する．しかし同じ宛て先の経路が経路表に複数存在した場合，ある一定間隔毎に表 5.5 の情報を格納したパケットヘッダでデータを送信する．このパケットヘッダを付加して送信する場合，*Packet Type* は *Data+Loss Request* となる．送信元はデータ送信時の時間を *Time Stamp* に入れ．もしデータ送信を行う経路の遅延が判明している場合は *Delay* に遅延を格納し送信する．またデータ送信に使用していない経路においては，表 5.5 に示すパケットヘッダを用い送信する．

パケットヘッダの *Data+Loss Request* を受信した宛て先は，表 5.6 に示す，通常データのパケットヘッダにロス率応答の情報を含んだパケットで返答する．通常データ+ロス率応答パケットを宛て先が送信する際，*Packet Type* には *Data+Loss Reply* となり，通常データ+ロス率要求パケットに格納されていたタイムスタンプを *TimeStampe* にいれ通常データ+ロス率要求が使用した経路と同じ経路を用い送信する．

もし中間ノードがパケットヘッダの *Packet Type* に *Data+Loss Request* もしくは *Data+Loss Reply* の値になっているパケットを受信した場合，*Packet Loss* に隣接ノードのロス率を格納し，転送を行う．

5.3 まとめと今後の課題

本節では本章のまとめと今後の課題について述べる．

表 5.6: 通常データ+ロス率応答パケット

項目	説明
Packet Type	パケットタイプ
Packet ID	パケット識別子
Header Len	ヘッダの長さ
Data Len	データの長さ
Source Node ID	送信元ノードの識別子
Destination Node ID	宛て先ノードの識別子
Route ID	経路の識別子
Req Source Node ID	経路要求を送信したノードの識別子
Target Node ID	ターゲットノードの識別子
Time Stamp	**タイムスタンプ**
Packet Loss	**パケットロス率**

5.3.1 本章のまとめ

本章では 3 章で PTR に 4 章で述べた TCP 転送性能の論理近似値を指標とした経路選択機構である ETR の機構を付加したルーティングプロトコルである PTER を提案した．また PTER の設計において ETR からの変更点を述べた．

5.3.2 今後の課題

今後の課題としてまず PTER 実装を行い，評価を行う．また実機の実装だけではなくシミュレータを用い，アルゴリズムの検証を行うことが挙げられる．

第6章　結論

6.1　本論文のまとめ

本論文では本研究の背景について述べ，MANET の説明を述べた．まず，多種多様な無線メディア同士で構築できる MANET のルーティングプロトコルである PTR を提案した．PTR 層では独自の ID を用いてノードの識別を行い，無線メディアを複数所持するノードにデータを転送することで，異なる無線メディアを所持するノード同士が透過的に通信できるプロトコルである．本章では PTR の設計，実装を述べ，実際に PTR で MANET を構築し評価実験を行った．

評価実験では実際に PTR を用い，802.11b と Bluetooth の実験ネットワークを構築し，TCP 転送性能の評価を行った．802.11b のみのネットワークではホップ数が増えると急激に TCP の転送性能が低下してしまう．一方 Bluetooth を経路に 1 ホップでも含んだ場合，ホップ数が 2，3 程度増加した際でも，TCP 転送性能に著しい性能の低下は見られなかった．

そして，PTR を用い可能となる異種無線メディアで構成される MANET における最短経路選択の問題点を指摘した．同一無線メディアの MANET においても，経路の遅延やロスによっては最短経路が必ずしも転送性能において最善な経路ではない．しかし異種無線メディアの MANET において，無線規格毎の転送性能が大きく異なることから，経路選択は大きく転送性能に影響を及ぼすことを指摘した．

次に PTR で指摘した問題を解決する機構である ETR について述べた．ETR は MANET 上でソースルーティングを行うプロトコルにおける，パケットロス率と遅延を指標とした経路制御機構である．まず ETR の概要について述べ，指標の取得方法，算出方法について述べた．そして，Monarch プロジェクトで公開されている DSR 上で ETR を実装し，TCP 転送性能の評価を行った．ETR を用いる事で DSR 層におけるパケットロス率が 50%の場合 TCP 転送性能が 93%向上することが判明した．また ETR におけるプローブ間隔を変化させ，経路が切り替わる時間・オーバヘッドの関係を検証した．2 ホップの経路と 3 ホップの経路が存在する場合，プローブ間隔は 3〜4 秒が適切であると判明した．

最後に異種無線メディアによって MANET を構築するルーティングプロトコルである PTR に経路選択機構である ETR を付加したルーティングプロトコルである PTER を提案した．また，PTR からの設計の変更点について述べた．

6.1.1 今後の課題

本小節では今後の課題について述べる．

PTR における課題

- 動的なアドレス割り当て

  PTR における独自の ID は手動で割当を行わなければならない．PTR におけるアドレスの割り当てを動的に行える機構が必要である．

- セキュリティ

  PTR では接近したノードは全て MANET に参加出来てしまい，セキュリティを全く考慮していない．ある集団において特定のメンバーのみで MANET を構築が出来るとい，セキュリティの機構が必要である．

ETR における課題

- シミュレータの使用

  ETR のアルゴリズムを深く検証し評価を行うため，実機だけではなくシミュレータを用いノード数や移動パターンを変更させて ETR の性能を詳細に解析する．

- α の適切な値の検討

  ETR の設計で提案した，式 (2) における α の適切な値を検討する．

- ロス率要求パケットの送信間隔

  経路が 2 ホップの経路と 3 ホップの経路の場合にはロス率要求パケットの間隔は 3〜4 秒が適当であると考えられるが，ホップ数や経路が増えた場合の最適な間隔を解析し，動的にロス率要求パケットの送信間隔を変更するシステムを導入する．

- オーバヘッドの軽減

  ETR では経路のパケットロス率と遅延を測定する際，一定間隔でロス率要求パケットの送信を行うため，ネットワークへのオーバヘッドが高くなってしまう．より効率良く指標を取得する方法を提案し，プローブによるネットワークへのオーバヘッドを軽減させる．

- 負荷分散のメカニズム

  データを一つの経路だけで送信するのではなく，複数の経路に分散させてデータを送信するといった，ネットワーク全体の性能を考慮した負荷分散のメカニズムを導入する．

PTER における課題

- 実装今後の課題としてまず実装を行い，評価を行う．また実機の実装だけではなくシミュレータを用い，アルゴリズムの検証を行う事が挙げられる．

謝辞

本研究を進めるにあたり，御指導を頂きました，慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝致します.

また，貴重な御助言を頂きました東京電機大学工学部情報メディア学科助教授戸辺義人博士に感謝の意を表します.

間 博人氏，斉藤 匡人氏をはじめとする慶應義塾大学徳田研究室 ECN 研究グループの諸氏また徳田・村井・楠本・中村・南研究室の方々には，研究会の活動を通じて多くの御意見，御助言を頂きましたこと拝謝します.

最後に研究の日々を共に過ごした，青木 基衣氏，峰松 美佳氏，滝沢 允氏，その他多くの友人に深謝し，謝辞と致します.

2003 年 1 月 22 日

高橋 ひとみ

参照論文

- 高橋ひとみ, 斉藤匡人, 間博人, 徳田英幸.
“モバイルアドホックネットワークにおける TCP 転送性能を考慮した経路制御機構, ” 情報処理学会 コンピュータシステムシンポジウム, pp. 9-16,
Nov. 2001.

- 高橋ひとみ, 斉藤匡人, 間博人, 徳田英幸.
“モバイルアドホックネットワークにおけるパケットロス率と遅延を考慮した経路制御機構, ” 情報処理学会 第 63 回 全国大会, Vol. 3, pp. 297-298,
Sep. 2001.

参考文献

[1] IMT-2000. International Telecommunication Union (ITU) IMT-2000. http://www.itu.int/imt.

[2] FOMA NTT DoCoMo. http://www.nttdocomo.co.jp/mc-user/foma.

[3] IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High Speed Physical Layer in the 5 GHz band*, 2000.

[4] IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High Speed Physical Layer in the 5 GHz band*, 2000.

[5] Bluetooth., http://www.bluetooth.com.

[6] Johnson. D, Maltz. D, Hu. Y, and Jetcheva. J. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. IETF Internet-Draft [Work in Progress], (2002).

[7] Perkins. C, Royer. E, and Das. S. Ad Hoc On Demand Distance Vector (AODV) Routing. IETF Internet-Draft [Work in Progress], (2002).

[8] Gerla. M, Hong. H, and Pei. G. Fisheye State Routing Protocol (FSR) for Ad Hoc Networks. IETF Internet-Draft [Work in Progress], (2002).

[9] HiperLAN2. http://www.hiperlan2.com.

[10] IrDA. http://www.irda.org.

[11] HomeRF. http://www.homerf.org.

[12] Vixie. P (ed), Thompson. S, Rekhter. Y, and Bound. J. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136, (1997).

[13] Mesut Gunes, Jorg Reibel. An IP Address Configuration Algorithn for Zero-conf. Mobile Multi-hop Ad-Hoc Networks. International Workshop on Broadband Wireless Ad-Hoc Networks and Services, (2002).

[14] Balakrishnan. H, Padmanabhan. V, Seshan. S, and Katz. R. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. IEEE/ACM Transactions on Networking, (1997).

[15] Wakikawa. R, Malinen. T, Perkins. C, Nilsson. A, and Tuominen. A. Global connectivity for IPv6 Mobile Ad Hoc Networks. IETF Internet-Draft [Work in Progress], (2002).

[16] Zapata. M. Secure Ad hoc On-Demand Distance Vector (SAODV) Routing. IETF Internet-Draft [Work in Progress], (2001).

[17] 徳田，中島，森川，斉藤，西尾，戸辺，中澤，大越. .HOME プロジェクトにおける情報家電コンピューティング. 第 62 回全国大会予稿集,(2001).

[18] 3Com. http://www.3com.com.

[19] Melco. http://www.melcoinc.co.jp.

[20] Netperf. http://www.netperf.org.

[21] Lin. C. An On-demand QoS Routing Protocol for Mobile Ad Hoc Networks. IEEE INFOCOM (2001).

[22] Toh. K. Associativity-Based Routing for Ad-Hoc Mobile Networks. Journal on Wireless Personal Communications (1997).

[23] The MONARCH Project at Carnegie Mellon University. http://www.monarch.cs.cmu.edu.

[24] Thompson. K, Miller. G, and Wilder. R. Wide-Area Internet Traffic Patterns and Characteristics. IEEE/ACM Transactions on Networking,(1997).

[25] Ott. T, Kemperman. J, and Mathis. M. The Stationary Behavior of Ideal TCP Congestion Avoidance. Unpublished Manuscript,(1996).

[26] Floyd. S, Fall. K. Promoting the Use of End-to-End Congestion Control in the InternetWide-Area. IEEE/ACM Transactions on Networking, (1999).