修 士 論 文

マルチパスルーティングを適用したアドホックネットワークの耐故障性の評価に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

橋本将彦

2011 年 3 月

修 士 論 文

マルチパスルーティングを適用したアドホックネットワークの耐故障性の評価に関する研究

指導教員　知念賢一 特任准教授

審査委員主査　知念賢一 特任准教授
審査委員　篠田陽一 教授
審査委員　丹康雄 教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

0910047 橋本将彦

提出年月 2011 年 2 月

概 要

アドホックネットワークとは、基地局のようなアクセスポイントの介在なしに相互に接続する形態をとるネットワークのことであり、通信には無線通信が利用されることが多い。既存の無線ネットワークでは、端末同士が通信を行う場合でも基地局を経由する必要があり、基地局が存在しない環境ではネットワークを構築することができない。これに対して無線アドホックネットワークでは、端末同士が互いに無線通信を行い、それぞれの端末がルーティングを行う。また、移動性を持った無線端末で構築されるアドホックネットワークを、モバイルアドホックネットワークと呼ぶ。このモバイルアドホックネットワークは、個々の端末が自由に移動するので、トポロジが頻繁に変化する。そのため、通信相手までの経路選択、すなわちルーティングが重要な問題となる。

モバイルアドホックネットワークのルーティングプロトコルとして、様々なプロトコルが提案されている。代表的なプロトコルとして、リアクティブ型プロトコル、プロアクティブ型プロトコル、そしてハイブリット型プロトコルがある。リアクティブ型は、通信要求があったときのみ経路を計算するため、普段は余分な経路制御パケットを送信せず、電力効率が良い。しかし、経路が確定するまで時間がかかるため、通信が開始されるまでに遅延がある。続いて、プロアクティブ型は、常に最新の経路を保持しておき、通信要求があったときにすぐに通信を開始できるようにしている。そのために、経路制御パケットを常時送信しているので、電力効率は悪い。最後のハイブリット型は、リアクティブ型とプロアクティブ型を状況によって使い分けるプロトコルである。ただし、2つのプロトコルを組み合わせるために複雑な制御が必要になっており、効率的なルーティングを行うことが難しい。

このように、既存のモバイルアドホックネットワークのルーティングプロトコルは、電力効率や経路の確定にかかる遅延を考慮したプロトコルは存在するが、モバイルアドホックネットワークにおける重要な要素の一つである耐故障性に優れたプロトコルは数少ない。既存の研究では、通信の耐故障性を高めるためのルーティングプロトコルとして、マルチパスを利用するプロトコルが提案されている。しかし、このマルチパスを利用したルーティングの研究は、シミュレーションによるものが多く、実装されたものはほとんど存在しない。なぜなら、既存のカーネルではマルチパスの使用を想定していないため、マルチパスルーティングが正常に動作する環境が存在しないからである。そのため実際に動作しているソフトウェアが存在せず、実環境での実験が困難であるため、ほとんどの研究がシミュレーション実験にとどまっている。しかし、シミュレーションでは、実時間に即した実験ができず、耐故障性のような突発的な問題に対する評価や、ネットワークのスケーラビリティの評価を十分にできない。

そこで、本研究では、まず耐故障性の評価を行うために必要なマルチパスルーティングのランニングコードが動くカーネルを作成した。マルチパスルーティングを動かすためには、まず、カーネル内のルーティングテーブルでマルチパスを扱えるようにする必要があ

る。既存のカーネルのルーティングテーブルは、ラディックスツリーの構造をしており、各リーフノードに経路情報が格納されている。この各経路情報にマルチパスを格納するための配列を追加して、必要な時に、インデックス番号によって指定したマルチパスへアクセスができるようにした。さらに、ラディックスツリーを新しくパトリシアトライで作り替え、マルチパスの追加及び削除をできるようにした。パトリシアトライは、ラディックスツリーと同様の基数木の一種であるが、ラディックスツリーよりも効率的にツリーを構築できる。

以上を実装したカーネルを用いることによって、マルチパスルーティングのランニングコードを動かせるようになり、シミュレーションでは評価が難しかったモバイルアドホックネットワークの耐故障性の評価を実環境で行えるようになった。

目 次

第1章　はじめに

1.1　研究背景

災害発生地のような通常のインフラが使用できない環境では、無線で接続できる端末のみで構成可能なアドホックネットワークが活用できる。このような状況では、正確な災害情報や避難経路といった情報を確実に送受信する必要があるため、途中で通信が切断されたり、大きな遅延が発生するようなことは避けなければならない。

アドホックネットワークは端末が移動しているため通信が切断されやすい。また、帯域の狭い無線を利用しているため大きな遅延や輻輳が発生しやすいという問題がある。このように、ノードの移動、バッテリーの限界、ノードの故障など様々なアドホックネットワーク特有の制約によって、トポロジが変化するため安定したネットワークを構築しにくい。そのためアドホックネットワークでは、このような変化に対応できるような経路制御が重要である。特に、経路上のリンクが何らかの理由で切断されてしまった場合に素早く経路を回復できるような、耐故障性の高い経路制御が望ましい。

1.2　研究目的

マルチパスルーティングでは、通信の切断時に備えて代替経路を探索する。この代替経路を経路の切断時に利用することによって、アドホックネットワークの耐故障性の向上が期待できる。しかし、現状のカーネルではマルチパスの使用を想定していないため、マルチパスルーティングが正常に動作する環境が存在しない。そのため実環境での実験が困難であるため、ほとんどの研究がシミュレーション実験にとどまっている。しかし、シミュレーションでは、実時間に即した実験ができず、耐故障性のような突発的な問題に対する評価や、ネットワークのスケーラビリティの評価を十分にできない。

そこで、本研究では、まず耐故障性の評価を行うために必要なマルチパスルーティングのランニングコードが動くカーネルを作成して、そのカーネルを使って、マルチパルルーティングの耐故障性の評価実験を行う。

1.3　論文構成

本章では、論文の章構成について説明する。第 2 章では、アドホックネットワークについて、その定義と既存のルーティングプロトコルについて説明する。第 3 章では、アドホックルーティングの性能指標について説明する。第 4 章では、既存のマルチパスルーティングについて説明する。第 5 章では、本研究における提案と、提案手法に対する課題について説明する。第 6 章では、マルチパスルーティングを実現するためのカーネルの設計を行う。第 7 章では、実装の説明と、その動作確認を行う。第 8 章では、実験内容の説明と、その結果について考察する。第 9 章では、本研究の成果についてまとめる。

第2章　アドホックネットワーク

本章では、アドホックネットワークの定義と、アドホックネットワークのルーティングプロトコルについて説明する。

2.1　アドホックネットワークの定義

従来の無線通信には、基地局が存在し、基地局を介してノード間の通信を行う。このような通信形態をインフラストラクチャーモードと呼ぶ。それに対して、アドホックネットワークとは、基地局やルータなどが介在せず、ノードが相互に接続する形態をとるネットワークのことを指す。そのため、アドホックネットワークは地理的条件に左右されず、自由に移動できる。このアドホックネットワークは、無線通信によって構成されることが多い。

図 2.1: アドホックネットワーク

それに対して、アドホックネットワークは、基地局が存在せず、ノード同士が直接通信を行う。例えば、図 2.1 のようなネットワークである。このように、各中継ノードで経路制御を行っており、この中継ノードは、無線端末機器であることが多く、移動性を持ってる。このようなアドホックネットワーク上では、ノードの移動、バッテリーの限界、ノードの故障など様々な理由によって、ノード間のリンクが切断される。そのため、アドホックネットワークはトポロジが急激に変化するといった特徴を持つ。このようなネットワー

クで安定した通信を行うためには、アドホックネットワークの特性を考慮した経路制御の方法が重要になってくる。

2.2 アドホックルーティングプロトコル

アドホックネットワークは、ノードの移動があるためトポロジが急激に変化する。そのため、ノード間の経路を管理するルーティングプロトコルの重要性は高い。そのルーティングプロトコルは、通常のネットワークで使用されているプロトコルとは、想定されている環境が違うため、アドホックネットワーク独自の特徴に対応したルーティングプロトコルが必要である。

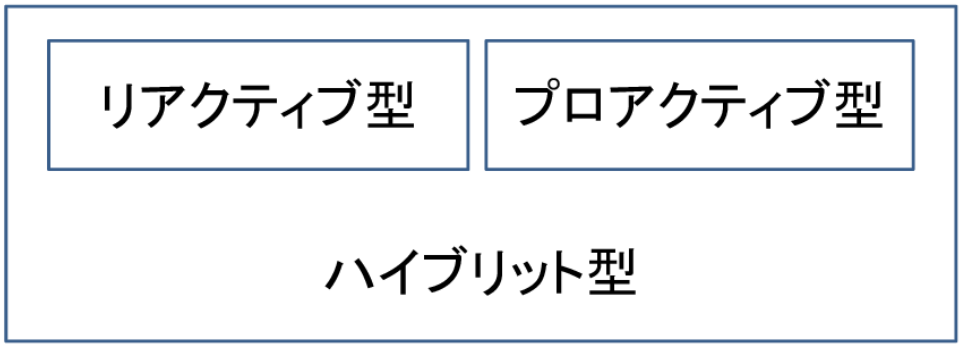

図 2.2: アドホックルーティング

アドホックルーティングプロトコルは現在、プロアクティブ型、リアクティブ型、ハイブリット型の 3 つに分類されている。

2.2.1 プロアクティブ型

プロアクティブ型のルーティングプロトコルは送信要求が発生する前に、互いのノードが通信しあい、あらかじめ各端末への経路をルーティングテーブルへ記述する。そうすることによって、送信要求が発生した場合に即時にデータを送信することができる。新たな端末がネットワーク内に入ってきたり他の端末とのネットワークが切断したときなどにルーティングテーブルが更新され、常に最新の経路情報を知ることができる。このプロトコルのメリットは、あらかじめ経路を計算しておくため通信の要求が起きるとすぐに通信が開始できることである。デメリットは、常に経路情報を更新し続けるため、通信要求がないときもパケットを送信している。そのため、電力消費が激しい。代表的なプロトコルとして、OLSR [2] や、TBRPF [14] 等がある。

プロアクティブ型で重要となるのは、どのぐらいの周期で経路表の更新を行なうのか、また、どのぐらいの範囲までをカバーするように経路表を作るのかを決めることである。常に最新の情報得るには、トラフィックが増大してしまうし、広範囲のノードまでカバーするには、電力を多大に消費してしまうからである。

OLSR(Optimized Link State Routing)

代表的なプロアクティブ型ルーティングプロトコルとして、OLSR がある。OLSR はリンクステート型ルーティングプロトコルとも呼ばれる。OLSR では、各ルータがトポロジデータベースを持っており、このデータベースをネットワーク全体にフラッディングを行い、すべてのノードで同じトポロジデータベースを保持する。このとき、より効率的にフラッディングを行えるように MPR(MultiPoint Relay) 集合というノードの集まりを作成している。

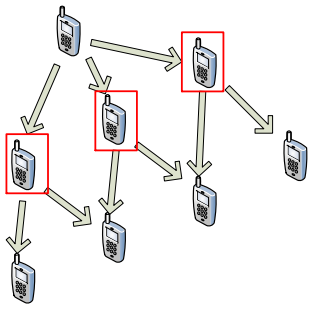

図 2.3: 通常のフラッディング

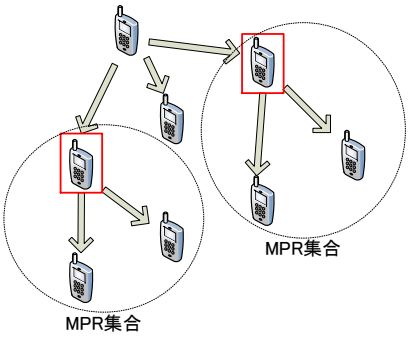

図 2.4: MPR 集合を用いたフラッディング

図 2.3 は、無線ネットワーク上で一般的なフラッディングを行った場合の、パケットの移動を矢印で表した物である。このとき、パケットを受信したノードはすべて、パケットの再送信を行っているが、同じパケットを重複して受け取るノードが存在する。そこで、OLSR では、図 2.4 のようにパケットの再送信の回数が最小限になるようなノードの集まりを作成して、フラッディングを行う。そのようなノードの集まりのことを MPR 集合と呼び、各ノードがこの MPR 集合を隣接ノードから選ぶことによって、効率的なフラッディングが行なえるようになる。ネットワーク全体のトポロジー情報はその効率的なフラッディング基盤を利用して交換され、各ノードで経路表が計算される。

2.2.2 リアクティブ型

リアクティブ型のルーティングプロトコルは、送信元ノードが通信を要求した時にのみ経路を作成する。いったん経路が発見されると、宛先への通信が終了するか、経路が使用不可能になるまでは、何らかの手段によってその経路を保持する。メリットとしては、通信要求時にのみ経路を計算するため、無駄なパケット送信を行わない。しかし、通信要求があってから経路が確定するまでに少し遅延が生じる。代表的なプロトコルとして、DSR [5] [19] や、AODV [17] [18] [19] 等がある。

アドホックネットワークは、モバイルノードを使用することが多いため、ノードのバッテリーが限られている。そのため、頻繁に電波を発信していると、すぐにバッテリーがなくなってしまう。また、ノードは移動しているため、数分前に作った経路は意味がないことが多い。そのため、通信の直前に経路表を作成するリアクティブ型プロトコルが考えられた。リアクティブ型は、通信を行わないときは、電波の発信が行われないため、ノードのバッテリーを節約できる。しかし、データの通信では、要求があってから、経路を確定する必要があるため、データが送信されるまでの時間が長くなってしまう。

AODV(Ad-hoc On-demand Distance Vecter Routing)

代表的なリアクティブ型ルーティングプロトコルとして AODV がある。AODV はオンデマンド型ルーティングプロトコルとも呼ばれる。AODV では、シーケンス番号をルーティングに利用している。各ノードは固有のシーケンス番号を管理しており、経路表の各送信先に対するエントリにこの番号とその有効性を含んでいる。それぞれのノードが持つシーケンス番号は経路が変更されるたびに増加させるため、古い経路と新しい経路を区別することが可能になり、ループが発生しないようになっている。

2.2.3 ハイブリット型

ハイブリッド型のルーティングプロトコルは、プロアクティブ型のルーティングプロトコルとリアクティブ型のルーティングプロトコルの両方を用いた複合的なルーティングプロトコルである。プロアクティブ型とリアクティブ型の利点と欠点を持っており、条件に応じてプロアクティブ型とリアクティブ型を使い分けるルーティングプロトコルが多い。代表的なものに Zone Routing Protocol:ZRP [3] がある。これは、近くにいるノードにはプロアクティブ型、遠くにいるノードにはリアクティブ型で経路制御を行うルーティングプロトコルである。しかし、どこまでプロアクティブ型を利用するかを決定するための適切な手段が存在しないため、性能がとても悪い。このように、ハイブリット型のルーティングプロトコルは、経路制御がとても複雑なため、経路が不安定になりやすく、現在のところ実用的ではないといえる。

第3章　アドホックルーティングの性能指標

本章では、アドホックルーティングプロトコルの性能を評価するための指標について説明する。

3.1　アドホックルーティングの評価指標

アドホックネットワークにおけるルーティングプロトコルを評価する際に考慮すべき項目について説明する。

3.1.1　ネットワークの規模

ノードの数やアドホックネットワークの構成範囲がアドホックルーティングの性能に与える影響を評価するための評価指標である。アドホックネットワークは、インフラに依存せずどのような場所でも構築可能なため、ネットワークの規模が可変である。そのため、ネットワークの管理者は、構築するアドホックネットワークの規模によって、どのルーティングプロトコルが高性能かを考慮する必要がある。

規模に対する適性を評価することによって、災害時のように広範囲に対して構築したい場合や、イベント会場のような非常に多くの数が狭い範囲に密集している場合など、状況によって最適なルーティングプロトコルの選択が可能になる。

3.1.2　ノードの移動度

アドホックネットワークの構築には無線端末を利用されることが多く、ノードが移動性を持っていることが多い。例えば、携帯電話やPDA、ノートパソコン、携帯ゲーム機などによってアドホックネットワークを構築可能である。ノードが移動すると、ネットワークのトポロジが変化するため、そのトポロジの変化に追従して経路を更新する必要がある。このノードの移動性は、他のネットワークにはないアドホックネットワークに独特の特徴である。そのため、ノードの移動性に対する耐性は、アドホックネットワークのルーティングプロトコルには絶対に必要な能力である。

3.1.3 パケット到達率

送信したパケットのうち、宛先に届いたパケットの割合のこと。変化を続けるトポロジに対して、どれだけ経路を維持することができているかを示す。アドホックネットワークはトポロジの変化が急激なため、その変化に対する追従性が重要である。この性能が低いルーティングプロトコルでは、信頼性の高いアドホックネットワークを構築できない。

3.1.4 遅延

アドホックネットワークの通信遅延には2種類ある。経路が確定するまでの遅延と通信そのものにかかった遅延である。経路が確定するまでの遅延とは、経路の収束速度を表す。この遅延が小さいほどトポロジの変化に対する追従性が高いため、アドホックネットワークの信頼性を評価するための重要な指標であるといえる。

また、アドホックネットワークは無線通信で構築されていることが多い。無線通信は、その性質上周囲の環境の影響を受けやすく通信経路が不安定になりやすい。経路上のノードに性能の低いノードが混ざっていた場合、そのノードがボトルネックとなり通信にかかる遅延が大きくなってしまう。そのため、アドホックネットワークでは、最小ホップ数の経路が常に最適であるとは限らない。そこで、実際に通信にかかる遅延を測ることによって、確定した経路の通信効率を評価することができる。

3.2 アドホックルーティングの評価手法

アドホックルーティングプロトコルの評価を行う上で、最も確実な方法は実機で動作させることである。しかし、実際に評価に必要なだけの実機を用意するのは手間もお金もかかるため現実的ではない。そのため、一般的には、シミュレーションか、エミュレーションを使って評価する。

3.2.1 シミュレーション

シミュレーションとは、模型や数学モデルを用いて現実に似た状況を試行する模擬実験のことである。代表的なネットワークシミュレータとして、NS や Click などがある。これらは、使用実績が多く、サポートされているルーティングプロトコルも多い。また、ソースコードも公開されているため、自作することによって新しいルーティングプロトコルの評価も行える。

しかし、モデルを利用した模擬実験の結果は、定式から導き出されたものであり、結果が前提に組み込まれている。そのため、一般性のあるデータをとることができない。使用するルーティングプロトコルも、シミュレータ独自のコードで書かれている必要があり、実際のランニングコードは動かしていない。また、実験の規模が大きくなるに従って、必

要な計算量が増加するため、大規模な実験を行うことが難しい。例えば、1 千台や 1 万台の規模のネットワークをシミュレーションした場合、数分間のシミュレーションに何日もかかってしまう。このように、シミュレーションは、現実的な規模のネットワークでの実験を行うのに向いていないと言える。

ns

ns(Network Simulater) [13] とは、最も利用されているネットワークシミュレータであり、現在 ns-2、ns-3 までバージョンアップされている。ソースコードは C++と Python で書かれており、Linux と UNIX、そして Windows の Cygwin 上で動作する。

Click modular router

Click modular router [6] [7] [8] [21] とは、多数のモジュール群で構成されるソフトウェアルーターのことである。

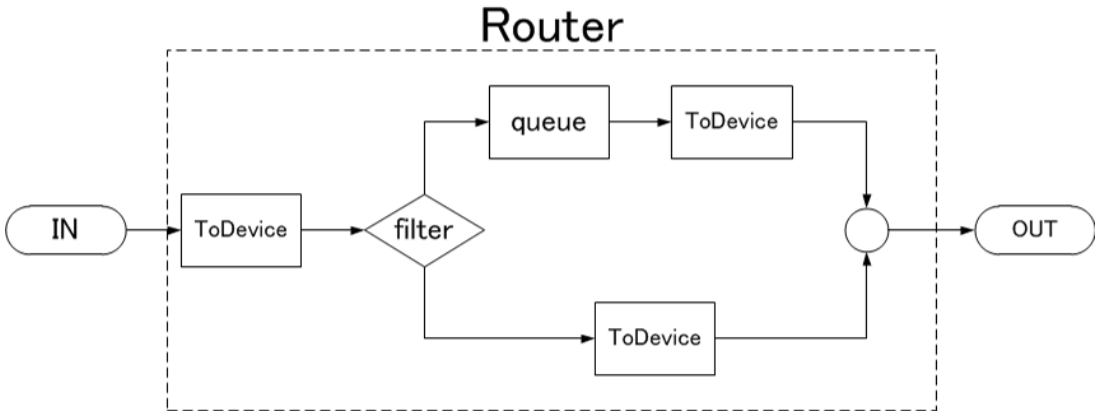

図 3.1: Click ルータ

Click は、エレメントと呼ばれる単純なルータの機能の持ったモジュールを組み合わせることによって、ルーティングシステム全体を柔軟にシミュレーションすることができる。

3.2.2 エミュレーション

エミュレーションとは、特定のハードウェアが行う処理に似せた処理を別のハードウェアやソフトウェアで実行することである。ネットワークエミュレータは、ネットワークの特性の変化を示したシナリオファイルを入力することで、そのネットワークの特性、パケットロスや遅延などをエミュレートする機能を持っている。

代表的なネットワークエミュレータとして、FreeBSD の dummynet や、Linux の Netem などがある。この模倣ネットワーク上でルーティングの実験を行うことにより、有線環境上で、無線通信で構成されたネットワークや、ノードが移動するネットワークにおけるルーティングの性能を評価できる。この場合、実際にルーティングプロトコルのランニングコードを動かしているため、シミュレーションよりも現実に近い実験が可能になる。

dummynet

dummynet [22] とは、帯域、遅延時間、パケットロス率などの制御を行うことができるツールであり、FreeBSD に標準で組み込まれている。パケットのフィルタリングは IP Firewall(ipfw) のパイプルールにより行われる。

QOMET

QOMET(Quality Observation and Mobility Experiment Tool) [1] とは、無線 LAN エミュレータであり、IEEE 802.11 や IEEE 802.15.4 などの通信規格がエミュレート可能である。

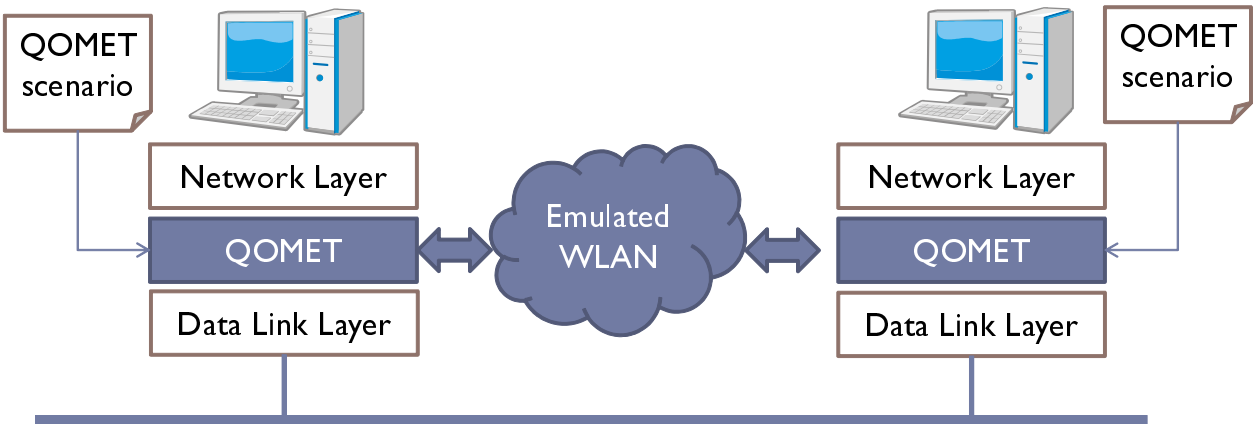

図 3.2: QOMET

図3.2のように、ネットワーク層とデータリンク層の中間で、入力されたシナリオに従って遅延やパケットロス率を制御する。このとき、遅延やパケットロス率は dummynet によって制御されている。

3.2.3 大規模実験環境

大規模実験環境とは、インターネットと分離された疑似インターネット環境のことである。支援ソフトウェアによって実験トポロジの構築が柔軟に変更可能であるため、実ノードによる実環境との同一性と、シミュレータの実験のしやすさという長所をあわせ持つ実証環境を構築できる。

StarBED

StarBED [12] は、北陸リサーチセンターによって提供されているネットワーク実験設備であり、約 1000 台の PC とスイッチ、支援ソフトウェアから構成されている。StarBED において実験トポロジは、VLAN を用いて設定可能である。実験トポロジの設定や、実ノードへの設定は管理用ネットワークを通して行われる。管理用ネットワークは、管理用トラフィックの実験への影響を防ぐため実験用ネットワークと分離されており、各実験用ノードは管理用と実験用の 2 つのネットワークに接続されている。実験を支援するためのファイルサーバや DHCP サーバ等や、実験ネットワーク構築のための支援ソフトウェアである SpringOS が存在する。

第4章　マルチパスルーティング

本章では、既存のマルチパスルーティングについて説明する。また、本研究でアドホックに対応させて評価するマルチパスルーティングアルゴリズムについても説明する。

4.1　アドホックネットワークにおけるマルチパス

アドホックネットワークにおいて提案されているマルチパスルーティングプロトコルについて説明する。

4.1.1　オンデマンド型マルチパス

SMR(Split Multipath Routing)

SMR [9] とは、リアクティブ型プロトコルである DSR [5] [19] をマルチパスに拡張したもの。ディスジョイントな経路を用意する。ディスジョイントな経路とは、SPT から決定した経路をプライマリ経路として、そのプライマリ経路で使用したリンクを使用しないように決定された代替経路のことである。プライマリの経路に障害が起きた場合、この代替経路に切り替えることで、すぐに通信を再開することができるように考案されたプロトコルである。

代替経路を、プライマリ経路と同じリンクを使用しないようにすることで、プライマリ経路の途中のリンクが切れても、代替経路は通信可能である。問題は、経路計算に時間がかかるため、通信を開始するまでの遅延が増加することと、経路が重ならないようにしているので経路のホップ数が増加することである。

こういった課題を解決するために、様々な拡張プロトコルが研究されている。しかし、より効率的にディスジョイントな経路を求めるためにホップ数以外の、電波強度のような新しいメトリックを追加したりするため、複雑な制御が必要になっている。そのため、制御パケットの数が増加しやすく、電力消費が大きくなるというデメリットがある。

4.1.2 リンクステート型マルチパス

MP-OLSR(Multipath OLSR)

MP-OLSR [23] とは、プロアクティブ型プロトコルである OLSR をマルチパスに拡張したもの。このとき、ある宛先に対する経路として、コストが等しい最適経路が複数ある場合、この複数経路をイコールコストマルチパスと呼ぶ。その宛先へのパケットの転送を複数のネクストホップへ分散することによって、トラフィックを分散してもよいことになっている。また、2 本の経路がある場合には、コストの低い方をメイン経路、コストの高い方をバックアップ経路として利用する場合もある。

4.2 Drouting

Drouting [16] とは、有線でのマルチパスルーティングプロトコルである。最大経路を計算できる、タグ転送、高い障害回避率といった特徴がある。複数の経路をタグによって切り替えることが可能で、経路に障害がおきた場合でもすぐに別の経路を利用できる。

4.2.1 MARA (Maximum Alternative Routing Algorithm)

MARA [15] は、Drouting で利用されている経路を計算するためアルゴリズム。既存のダイクストラと同等の計算量で、すべてのリンクを利用した最大数の経路を計算できる。

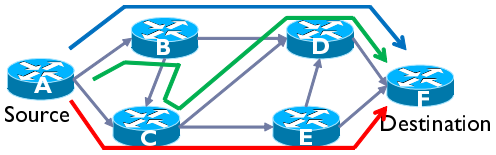

図 4.1: 元の経路

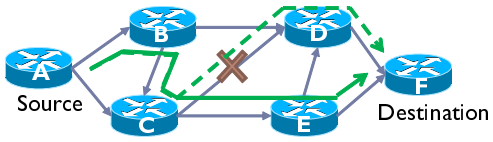

図 4.2: 故障時の経路

図 4.1 は、MARA によって計算される経路の例であり、1 つの宛先に対して複数の経路が利用できる。このようなネットワークで、ノード C とノード D の間のリンクが切断されてしまった場合の経路を図 4.2 に示す。このように、複数の経路を持つことで何らかの原因でリンクが切断されても、すぐに別の経路に切り替えられる。MARA は、多数の経路を計算することで、より高い耐障害性を持ったルーティングを可能にする。

第5章　本研究の提案

5.1　マルチパスルーティングを適用したアドホックネットワーク

アドホックネットワークには、通信障害が発生しやすいという問題があるため、ルーティングでは耐故障性が必要になる。耐故障性に優れたルーティングとして、マルチパスルーティングがある。しかし、既存の手法では代替経路を最短経路の予備としてしか扱っておらず、障害を検知した時にしか利用しない。そのため、計算した代替経路を十分に活用できていない。

本研究では、既存のアルゴリズムである Maximum Alternative Routing Algorithm(MARA) をアドホックネットワーク上のルーティングに適用することを提案する。MARA は代替経路の数を増やし、その代替経路の選択を障害検知システムに依存せずに送信元で選択できる。これにより、トラフィックの負荷分散や、耐故障性の向上が期待できる。しかし、MARA をアドホックネットワークで利用する方法やその有効性などは検証されていない。そこで、実際にアドホックネットワークにマルチパスルーティングを適用し、その性能を評価する。

評価は、無線 LAN エミュレータを使用して行う。本研究では、、無線 LAN エミュレータとして QOMET を利用する。エミュレータで評価することによって、シミュレータによる実験より現実に近い結果を得ることができる。

5.2　既存のカーネルでの問題点

カーネルとは、オペレーティングシステム (OS) の基本機能を実装したソフトウェアのことである。OS の中核部分として、ディスクやメモリなどの資源の管理など、OS としての基本機能を提供する。現在、サーバ等のネットワーク機器の OS として、様々なサーバ向け OS が開発されている。代表的な OS として、windows 系、LINUX 系そして UNIX 系がある。

それぞれの特徴を説明する。まず、windows 系のサーバ OS とは、Microsoft 製の OS で Windows と同じ画面構成、操作方法であり、GUI が充実してるといった特徴がある。代表的なものとして、Windows server2008 シリーズがある。操作が容易ではあるが、攻撃

を受けやすい。また、ライセンスが必要であるため、数を増やすと導入コストが高額になる。

次に、LINUX 系あるいは、UNIX 系のサーバ OS とは、どちらも UNIX 風のシステム体系を持った OS のことを指す。UNIX ベースのカーネル、あるいは UNIX 互換のカーネルを使用している OS のことであり、非常に多くの派生 OS が開発されている。それらのうち、いくつかは無償で提供されており、導入コストは非常に低額である。また、安定性と堅牢性にも優れている。しかし、操作は基本的にコマンドラインで行うため、運用に高度なネットワークの知識や LINUX の操作経験が必要になる。

ルーティングでは、これらのカーネルに実装されたルーティングテーブルに対して経路情報を追加したり、削除したりする。そして、パケット転送はルーティングテーブルに基づいて行われる。そのため、マルチパスルーティングが可能かどうかは、カーネルのルーティングテーブルの実装に依存している。しかし、既存のカーネルではマルチパスをルーティングテーブルに持たせることはできるが、そこから自由に経路を選択することができない。

5.3 既存のカーネルによるマルチパスルーティング

本研究では、実験で無線 LAN エミュレータの QOMET を使用するため、QOMET の動作実績のある FreeBSD [20] を利用した。FreeBSD は BSD UNIX をベースとした OS であり、本研究では FreeBSD 8.0-RELEASE を用いる。しかし、現在のカーネルにはルーティングでマルチパスを使用することを想定しておらず、ルーティングテーブルの中に複数のネクストホップを持たせて、それらを利用してフォワーディングを行う機能が実装されていない。そのため、既存のカーネルではマルチパスを扱うルーティングを動かすことができない。例えば、既存のルーティングプロトコルの OSPF [10] [11] は、イコールコストマルチパス (ECMP) [4] というマルチパスを利用できるが、一般的なカーネル上では扱うことができない。

5.4 マルチパス対応ルーティングテーブルの提案

本研究では、マルチパスルーティングを適用するために、カーネルのルーティングテーブルでマルチパスを実装する。カーネルでマルチパスを利用できるようにすることで、すべてのアプリケーションでマルチパスを利用できるようになる。また、FreeBSD のカーネルのルーティングテーブルツリーのソースコードは非常に複雑になっており、経路情報の構造体にも変更を加えることを考慮すると、書き換えるよりも入れ替えた方が開発コストが低いと考え、新しいルーティングテーブルツリーを実装した。このとき、カーネルの既存の API をそのまま利用できるようにすることで、変更部分をルーティングテーブル周りのみにした。

第6章　マルチパスルーティング対応のカーネル設計

本章では、評価対象である新しいマルチパスルーティングの設計と、その実装方法について説明する。

6.1　既存のルーティングテーブル

各ルータでは、カーネルの中に宛先 IP アドレスが所属するネットワークとゲートウェイの関係をまとめたルーティングテーブルを持っている。パケット転送時は、IP パケットの宛先アドレスからルーティングテーブルを参照して、ゲートウェイを決定する。このとき、目的の経路情報の検索を効率化するために、ルーティングテーブルではラディックスツリーを利用した基数探索を行う。基数探索とは、探索に用いるキーを R 進数の数とし

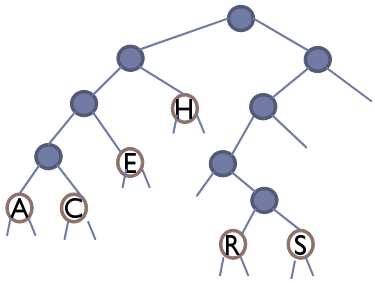

図 6.1: ラディックスツリー

て、その数の桁毎に比較を行う探索方法である。ルーティングテーブルでは基数となる R は 2 であるため、2 分探索木になる。N 個のキーで構成されたラディクッスツリーの探索で必要な比較回数は、平均 $logN$ 回であり、最悪でも N 回ですむ。また、ラディックスツリーでは、探索木の内部ノードにはキーを置かず、キーはすべて外部ノード (葉) に置く。そのため、2 つのキーに対して最低 1 つの内部ノードが必要になるため、平均で約 1.4N 個のノードをもつ。この基数探索によって、ビット比較という単純な処理のみで、高速な探索を可能にしている。

ルーティングテーブルでは、基数探索のキーとして宛先 IP アドレスを使用して一致した外部ノードからルーティングエントリを得る (図 6.2)。しかし、既存のエントリで扱う

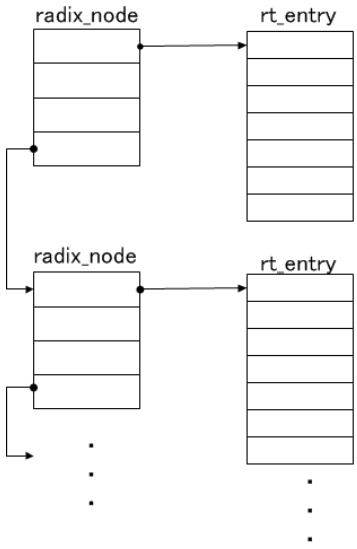

図 6.2: 既存の rtentry

ことのできる経路情報は、同じ宛先に対して一つである。オプションとして複数登録することができる場合があるが、既存のカーネルの実装ではその複数の経路情報の使用方法が定義されていない。そのため、複数の経路情報を登録しても一番最初に追加した経路情報しか参照できず、その経路情報が使えなくなって、初めて次の経路情報が参照できるようになる。つまり現状では、最初のゲートウェイが使用できなかった際の予備経路としてしか利用できない。既存のカーネルでは、経路を各ルータが複数の経路から選択して決定するような、本研究で対象となっているマルチパスルーティングに対応できない。

6.2 マルチパスルーティングテーブルの設計

本研究で評価する新しいマルチパスルーティングでは、できる限り多くの経路を保持し、コネクションごとにプライオリティに関わらずに選択する。このルーティングプロトコルを正しく動作させるには、非常に多くの経路情報を効率的に管理できるルーティングテーブルと登録された経路情報を同一のプライオリティで扱える機能が必要である。

本研究で実装した、マルチパスルーティングを動かすためのカーネルの設計を、図 6.3 に示す。マルチパスのルーティングアプリケーションは、ネットワークから経路情報を受け取り、マルチパスを計算する。そして、計算したマルチパスをカーネル上のルーティングテーブルの中で管理できるようにする。そのために必要な、マルチパスの登録方法、検索方法、そしてパケット転送の際に使用する経路の選択方法について説明する。

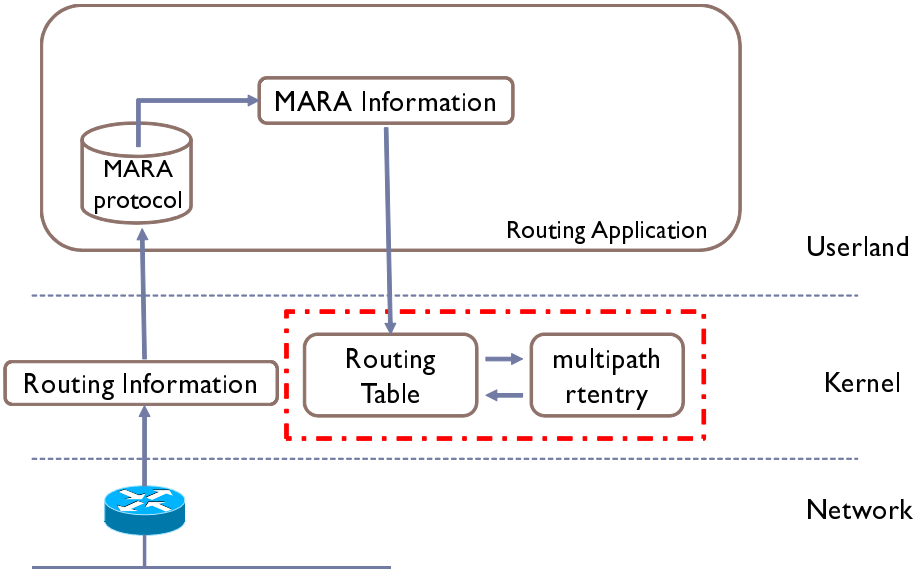

図 6.3: マルチパスに対応したルーティングシステム

6.2.1 マルチパスの登録

既存のカーネル上に実装されているルーティングテーブルでは、マルチパスの使用を想定していない。そのため、既存のカーネルのままではマルチパスルーティングが正しく動作しない。本研究では、マルチパスルーティングが正しく動作できるようにするために、カーネル自体にマルチパスを利用する機能を追加した。

まず、複数の経路情報を格納するための配列をルーティングエントリの構造体に追加した。この配列は、それぞれルーティングエントリの構造体へのポインタを格納しており、インデックス番号を指定することによって、任意の経路情報を選択できるようになっている。複数のエントリのうち、配列の一番最初にあるエントリをデフォルトのエントリとして、そこに配列へのポインタを記憶させておく。このデフォルトエントリは、ルーティングテーブルを探索して一致したノードに記録されているエントリで、特に指定がない場合はこのエントリを使用するものとする。

6.2.2 マルチパスの検索

ルーティングテーブルの検索木として、パトリシアトライを実装して、既存のラディクッスツリーと入れ替えた。パトリシアトライとは、ラディックスツリーに存在する一方向分岐をなくしてより効率的に探索が行えるようにしたものである。このパトリシアトライを用いた探索に必要なビット比較の数は平均 $logN$ 回であり、ラディックスツリーと同等である。ただし、パトリシアトライを構成するのに必要なノードの数が N 個で済む。これは、ラディクッスツリーの平均的な数の約 7 割程度である。

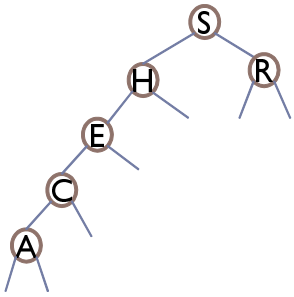

図 6.4: パトリシアトライ

表 6.1: 平均ノード数

Radix	Patricia
1.44N[a]	N

[a]N 個のランダムなキーで構成した場合

つまり、パトリシアトライはラディックスツリーよりも効率的に探索木を構築できる。この特性は、多くの経路情報を管理する必要があるマルチパスルーティングにおいて、メモリの節約が可能であるため、ラディックスツリーよりも適切であると考えられる。

6.2.3 マルチパスを用いたパケット転送

既存のカーネルでは、複数の経路情報を、ラディックスツリーノードの中に片方向リストとして登録することができた。しかし、登録後は、片方向リストの一番上にあるノードをデフォルトの経路情報として扱うように設計されており、2 番目以降の経路情報は有効に活用されていない。マルチパスの性能を最大限に生かせるようにするためには、これらの複数の経路情報を扱えるようにする必要がある。

本研究で使用するマルチパスルーティングは、フローごとに経路を選択可能にする必要がある。そこで、経路の決定の際には、それぞれのコネクションごとに持つ ID によって決定するようにした。このときの ID とは、IPv6 パケットの flowlabel や、IPv4 パケットの tos フィールドのような、フローごとに固有な数字のことである。

第7章　実装

7.1　マルチパスのルーティングエントリ

マルチパスを格納する配列について説明する。

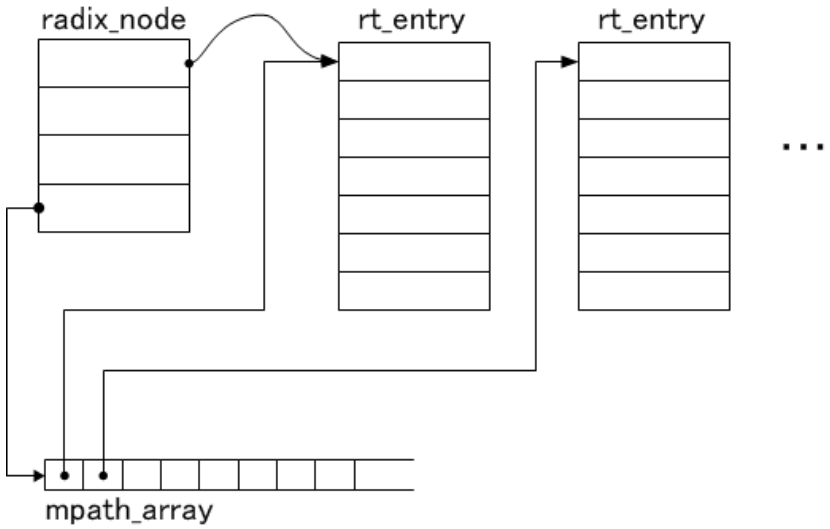

図 7.1: 実装した rtentry

本研究で実装した、複数の経路情報を管理するための配列を図 7.1 に示す。パトリシアトライのノードは、ルーティングエントリへのポインタの他に、ルーティングエントリのポインタを格納する配列を持っている。各ルーティングエントリを配列に格納することによって、任意の経路情報へのアクセスを容易にした。

7.2　マルチパスの追加・削除

マルチパスの追加と削除の実装について説明する。

FreeBSD にはルーティングテーブルの経路情報を操作するための関数として rtrequest(図 7.2) が実装されている。本研究では、この関数にいくつか関数を追加してマルチパスを追加・削除できるようにした新しい rtrequest(図 7.3) を実装した。

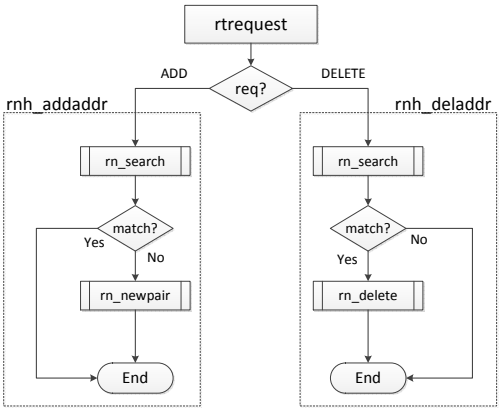

図 7.2: 既存の rtrequest

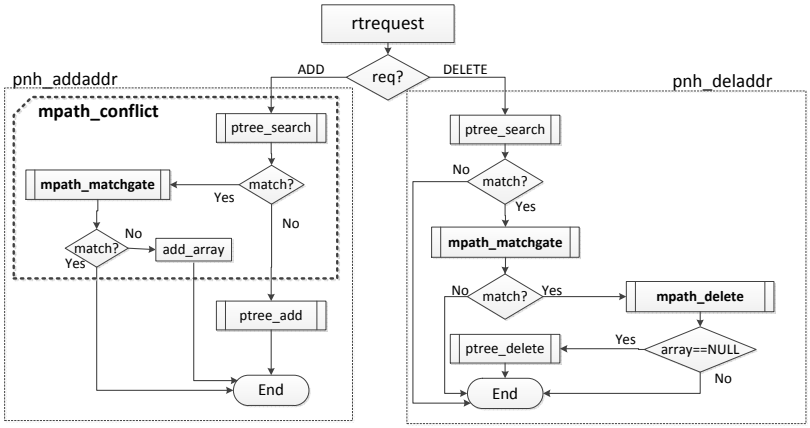

図 7.3: 実装した rtrequest

新しい機能を持った関数は、mpath_conflict、mpath_matchgate、mpath_delete である。ptree_add、ptree_delete、ptree_search は、ルーティングテーブルへの追加、削除および検索のための関数である。利用している探索木がラディックスツリーからパトリシアトライに変更されている以外は同じ機能であるため説明は省く。

mpath_conflict は、ルーティングテーブルに経路情報を追加しようとした時にすでにエントリが存在していないかを確認する。引数として追加したい経路情報を持ったエントリと、宛先 IP アドレス、ネットマスクを渡す。宛先 IP アドレスでルーティングテーブルを検索する。すでにエントリがある場合、そのエントリの配列に格納されている複数の経路情報を、これから追加しようとしている新しい経路情報とを比較して重複がないことを確認する。重複が確認された場合は 1 を返す。重複が確認されなければ 0 を返し、処理を続行して新しい経路情報を追加する。

経路情報を追加するとき、ルーティングツリーへは新しいエントリを追加せずに既存のエントリが持っている配列の中に新しい経路情報へのポインタを追加している。このときの処理 (add_array) では、realloc を使用して配列のサイズを動的に確保している。この方法によって、従来のように経路情報ごとにツリーノードを複製する必要がなくなり、メモリを節約できるようになった。

mpath_matchgate は、経路情報の配列から指定したゲートウェイを検索する。配列に格納されている経路情報のゲートウェイを参照し、検索対象と一致した場合、その経路情報のポインタを返す。一致するゲートウェイがない場合は NULL ポインタを返す。この関数は、経路情報の配列の重複を確認するときや、指定した経路情報んだけを削除するときに利用される。

mpath_delete は、配列内の経路情報から、指定した経路情報の削除を行う。削除したいルーティングエントリと、そのエントリが格納された配列を持ったエントリを引数として渡す。配列の中から指定した経路情報へのポインタを削除して、0 を返す。ただし、もし配列が空になった場合のみ 1 を返す。これは、配列が空になったときにのみルーティングテーブルからエントリを削除する必要があるためである。よって、mpath_delete が 1 を返した場合のみ ptree_delete を実行する。

7.3 マルチパスのパケット転送

マルチパスを利用したパケット転送（図 7.4）について説明する。FreeBSD にはパケット転送を行うための関数として ip_output が実装されている。ip_output は、IP ヘッダの宛先アドレスを参照して、その宛先アドレスに対する経路情報を rtalloc 関数によって得る。通常は、このときの経路情報をそのまま利用して、パケットの転送先を決定している。この転送方式をデフォルト（レガシー）とし、それ以外に 2 つの新しい転送方式を追加した。

まず、1 つめはランダムに転送先を決定する方式である。これは、イコールコストマルチパス [4] による標準の転送方式であり、この方式を実装したことにより従来のイコール

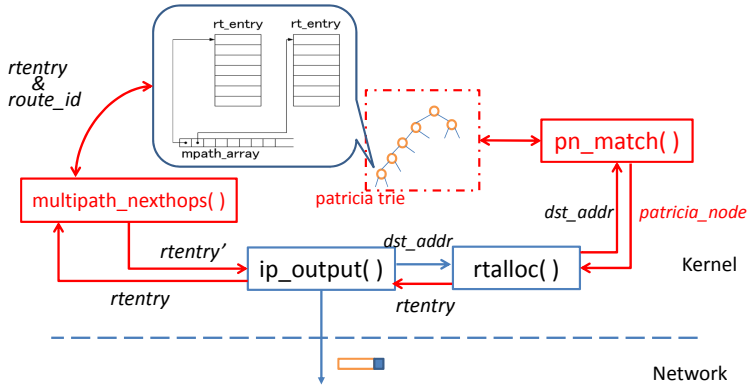

図 7.4: マルチパスパケット転送

コストマルチパスを利用したルーティングプロトコルが動かせるようになった。

２つめは、送信元が付加した情報によって転送先を決定する方式であり、そのために新しく multipath_nexthops という関数を追加した。この multipath_nexthops 関数は、rtalloc によって得られたルーティングエントリの中にマルチパスの配列が存在していた場合に呼び出される。multipath_nexthops 関数は、配列を持ったルーティングエントリと、IP ヘッダの情報（送信元アドレス、tos フィールドなど）から作成された ID（32bit）を渡すと、与えられた ID に基づいて配列からルーティングエントリを選択して返す。

7.4　ルーティングテーブルの表示

UNIX のコマンドである netstat によって、ネットワークに関する統計情報を参照できる。このコマンドは、ルーティングツリーを読み込んでツリーの全探索を行い、すべての経路情報を表示するものであり、ルーティングの研究において有用なコマンドである。しかし、netstat はカーネルの API を使用せずに直接ルーティングテーブルにアクセスしているため、本研究のカーネルのようにルーティングテーブル周りを変更すると既存の netstat は動作しなくなる。そこで、新しいルーティングツリーの構造体に合わせて netstat のプログラムを書き換えた。それによって、パトリシアトライのルーティングテーブルの経路情報を参照できるようになった。また、複数経路が登録されているときも、どの宛先に対するゲートウェイなのかを表示できるようになった。

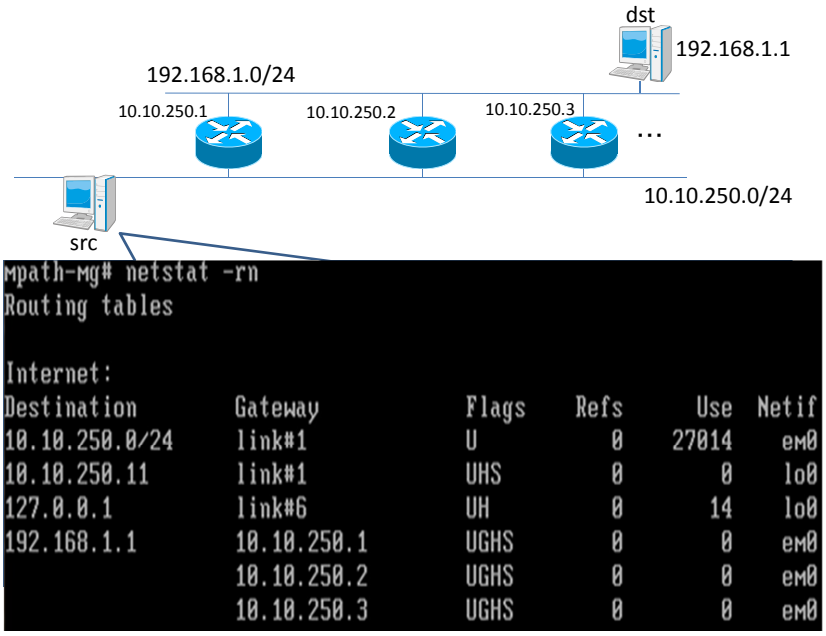

図 7.5: マルチパス対応の netstat

第8章　実験・評価

本実験では、実装したカーネルの動作確認を行う。実際にマルチパスルーティングのランニングコードを用いて、複数の経路が登録され、それらが使用可能であることを確認した。

8.1　実験1

本実験では、実装したカーネルの動作確認と、マルチパスルーティングのランニングコードが動作することの確認を行う。

そのために、図8.1のようなネットワークを構築し、そこでルーティングを行った。各ノードのネットワークアドレスは10.10.230.0/24であり、ホストアドレスはノード番号である。

その結果が図8.2〜8.11である。すべての宛先に対して複数の経路が作られていることがわかる。

実線線で表されている経路が最短経路で、点線の矢印がMARAによって計算された経路である。例えば、ノード1からノード2への経路を見てみる（図8.3)。最短経路(1-10-6-3-2)の他に、MARAによって計算された経路(1-10-9-8-4-5-2)が存在することが分かる。

またカーネルの中の経路表を確認したところ、実際に一つの宛先に対して複数のゲートウェイが追加されていることを確認できた。

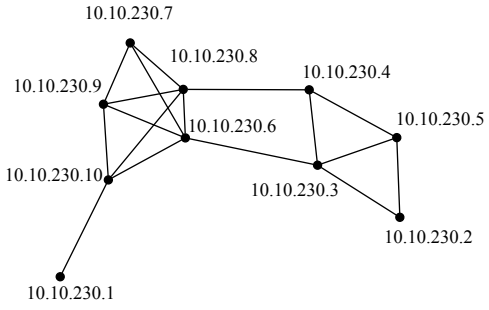

図 8.1: ネットワークトポロジ

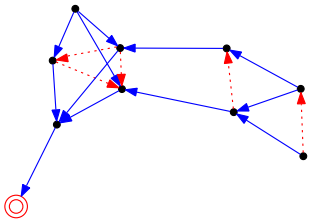

図 8.2: ノード 1 への経路

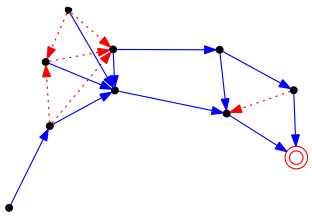

図 8.3: ノード 2 への経路

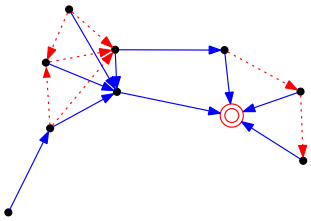

図 8.4: ノード 3 への経路

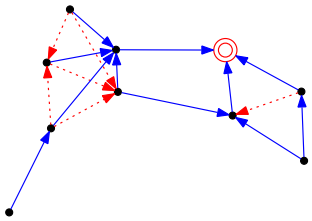

図 8.5: ノード 4 への経路

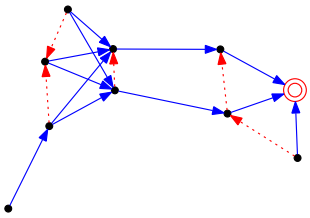

図 8.6: ノード 5 への経路

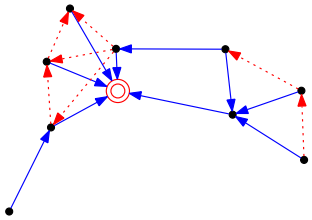

図 8.7: ノード 6 への経路

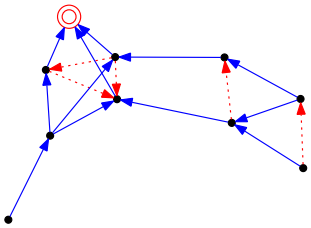

図 8.8: ノード 7 への経路

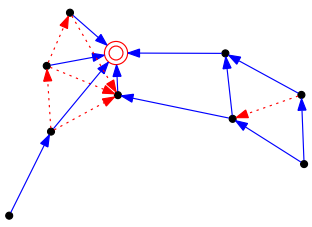

図 8.9: ノード 8 への経路

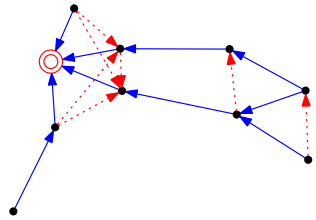

図 8.10: ノード 9 への経路

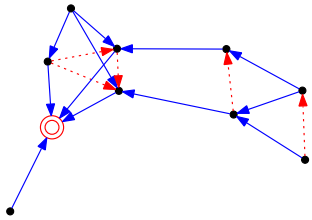

図 8.11: ノード 10 への経路

8.2 実験 2

本研究では、モバイルアドホックネットワークにおける性能を評価するために、QOMET を使用した。以下に、実験に使用した設定を示す。

8.2.1 ネットワークモデル

実験環境は 100m × 100m の領域として ノイズ強度は-100dB とする。これは、特に障害物のない屋外を想定している。この領域内に 10 個のノードをランダムに配置することによって構築されるアドホックネットワークに対して評価を行う。本実験におけるノードとは移動無線端末のことであり、領域内を動き回り、ネットワークは動的に変化する。また、ノードの無線の通信規格は、IEEE802.11 のアドホックモードを使用する。

8.2.2 ノードの動作モデル

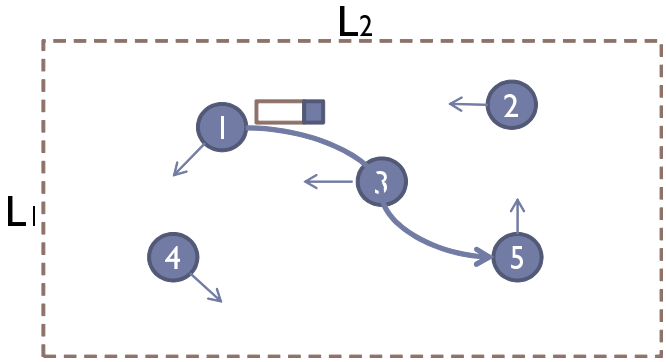

図 8.12: 動作モデル

本実験では、移動端末で構成されたアドホックネットワークでのルーティングの性能を評価する。その際、ノードの移動を再現するための移動移動モデルとして、Random walk

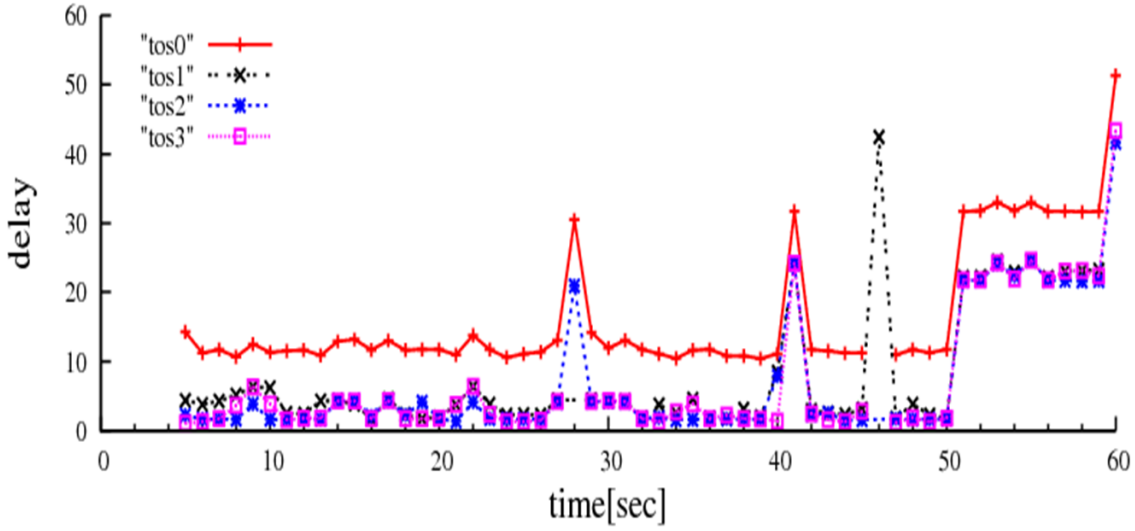

図 8.13: 遅延

を使用する。Random walk とは、一定時間ごとにランダムな方向に移動するモデルであり、このときの移動速度はランダムに決定される。本実験では、速度の最低は 0m/s、最高は 10m/s、方向を変化させる間隔を 10 秒間とする。これは、人間の移動を想定している。

8.2.3 通信モデル

それぞれのノードはアドホックモードの無線通信を使用しており、通信規格は EEE 802.11b で、送信電力は 20dBm とする。通信の測定に ping コマンドを用いる。ping コマンドとは ICMP echo パケットを送信し続け、ICMP echo reply が届くたびに往復時間などの情報を表示するプログラム。このときのパケットサイズは 56 バイトであり、1 秒間隔で送信される。また、IPv4 パケットにおける ToS フィールドの値 ([0x0000〜0xffff]) をオプションによって決定できる。今回の実験では、一つの宛先に対して複数の ToS を同時に送信して測定する。

8.3 実験結果

ノード 1 からノード 5 に対して、ToS の値が 0〜4 の場合について ping を実行した。実験を 60 秒間実行して、そのときのパケットの遅延と到達率を測定した。

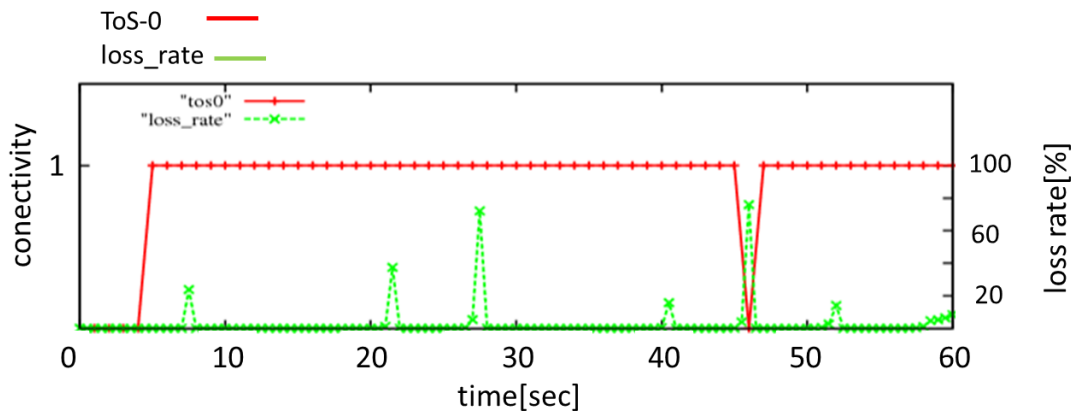

図 8.14: シングルパスの場合のパケット到達率

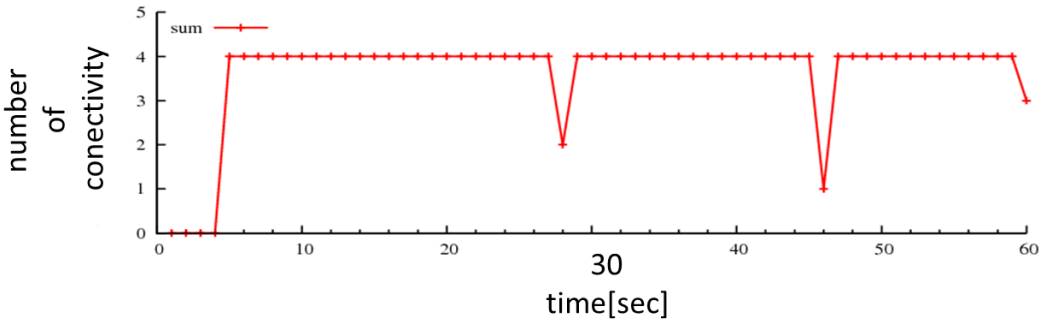

図 8.15: マルチパスの場合のパケット到達率

8.3.1 遅延

パケットの遅延の結果を図 8.13 に示す。横軸が経過時間 [sec] で、縦軸が遅延 [msec] を表す。応答がなかった場合は空白にしてある。

8.3.2 パケット到達率

シングルパスの場合のパケット到達率の結果と、そのとき設定されていたパケットロス率を図 8.14 に、マルチパスの場合のパケット到達率の結果を図 8.15 に示す。横軸が経過時間 [sec] で、縦軸は到達性を 0 か 1 で表したものである。マルチパスのグラフの場合は、マルチパス全体での到達率を見るために、すべての ToS の結果を足し合わせたものをグラフにしている。

シングルパスの場合（図 8.14）のグラフをみると、ちょうどパケットロス率が高くなっている 45 秒のあたりで一度パケットをロスしていることが分かる。それに対して、マル

チパスの場合（図 8.15）のグラフを見てみると、何度かパケットロスしている箇所があるが、マルチパス全体で見れば到達性が完全に 0 になっている瞬間は存在しないことが分かる。

第9章　おわりに

カーネルにマルチパスを扱う機能を実装したことにより、マルチパスルーティングのランニングコードが動作する環境を作成した。この実装により、実環境によるマルチパスルーティングの実験が可能になったため、シミュレーションでは評価が難しかったマルチパスを適用したアドホックネットワークの耐故障性の評価を実環境で行えるようになった。

本研究の実装を利用して、実際にマルチパスルーティングのランニングコードが動作することを確認した。その際に複数の経路が計算され、追加されたことを確認した。

また、QOMET によってモバイルアドホックネットワークを模倣して、不安定なネットワークでのパケットの到達性を測定した。マルチパス全体での到達性が、シングルパスでの到達性よりも高いことを確認した。

今後の課題として、もっといろんな条件で実験を行う必要がある。また、他のルーティングプロトコルとの比較を行う必要がある。

謝辞

研究を行うにあたり、主指導教員である知念賢一特任准教授には多くの御指導や御助言をいただきました。深く感謝し、心よりお礼申し上げます。また、主テーマ審査員である篠田陽一教授、丹康雄教授、副テーマ指導教員である飯田弘之教授に感謝いたします。

本学 小原泰弘助教には適切なご指導と多大な御協力をいただきました。心より感謝いたします。

情報通信機構の研究員である三輪信介氏、宮地利幸氏、中井浩氏、Razvan BEURAN 氏には StarBED の利用の際に御助言や御協力をいただきました。心より感謝いたします。

研究室の皆様には様々な場面で御協力いただきました。

最後に、研究や生活を支えてくれた家族に感謝いたします。

参考文献

[1] Razvan Beuran, Junya Nakata, Takashi Okada, Lan Tien Nguyen, Yasuo Tan, and Yoichi Shinoda. A multi-purpose wireless network emulator: Qomet. *AINAW 2008*, 2008.

[2] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), oct 2003.

[3] Z.J. Haas, M.R. Pearlman, and P.Samar. The zone routing protocol (zrp) for ad hoc, 2002.

[4] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992 (Informational), nov 2000.

[5] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), feb 2007.

[6] Eddie Kohler. The click modular router. *Massachusetts Institute of Technology*, 2000.

[7] Eddie Kohler. Click for measurement. *UCLA Computer Science Department Technical Report*, 2006.

[8] Eddie Kohler, Robert Morrisy, and Benjie Chen. Programming language optimizations for modular router congurations. *ICSI Center for Internet Research and yMIT Lab for Computer Science*, 2002.

[9] Sung-Ju Lee and Mario Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. *IEEE*, 2001.

[10] J. Moy. OSPF Version 2. RFC 1583 (Draft Standard), mar 1994. Obsoleted by RFC 2178.

[11] J. Moy. OSPF Version 2. RFC 2328 (Standard), apr 1998. Updated by RFC 5709.

[12] NICT. Starbed project. `http://www.starbed.org/`, 2011.

[13] ns 3 project. The ns-3 network simulator. `http://www.nsnam.org/`.

[14] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684 (Experimental), feb 2004.

[15] Yasuhiro Ohara, Shinji Imahori, and Rodney Van Meterr. Mara: Maximum alternative routing algorithm. *IEEE INFOCOM*, 2009.

[16] Yasuhiro Ohara, Hiroyuki Kusumoto, Osamu Nakamura, and Jun Mura. Drouting architecture: Improvement of failure avoidance capability using multipath routing. *IEICE Transactions*, 2008.

[17] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), jul 2003.

[18] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. *IEEE*, 2010.

[19] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, and Mahesh K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications*, 2001.

[20] The FreeBSD Documentation Project. Freebsd handbook. `http://www.freebsd.org/doc/ja_JP.eucJP/books/handbook/`.

[21] rchertov. The click modular router project. `http://read.cs.ucla.edu/click/click`.

[22] Luigi Rizzo. Dummynet home page. `http://info.iet.unipi.it/~luigi/dummynet/`.

[23] Jiazi Yi, Eddy Cizeron, Salima Hamma, and Benoit Parrein. Simulation and performance analysis of mp-olsr for mobile ad hoc networks. *IEEE WCNC*, 2008.

付 録A

本研究で作成したFreeBSD 8.0-Rのソースコードをgithubにて公開している。
github (`https://github.com/m-hashimoto/mpath-project`)
git repository (`git@github.com:m-hashimoto/mpath-project.git`)