通信状況に応じて柔軟に経路再選択を行う マルチパスルーティングの提案

北陸先端科学技術大学院大学

情報科学研究科

可児友邦

2018 年 3 月

修 士 論 文

通信状況に応じて柔軟に経路再選択を行うマルチパスルーティングの提案

1410018　　可児友邦

主指導教員　篠田陽一
審査委員主査　篠田陽一
審査委員　丹康雄
　　知念賢一

北陸先端科学技術大学院大学
情報科学研究科

提出年月: 2018 年 2 月

概 要

インターネット上では, 各種組織の公式 Web サイト, 電子掲示板, 検索エンジン, YouTube に代表される動画共有サイト, Facebook と Twitter に代表される SNS, BitTorrent に代表されるファイル共有ソフトウェア, Business to Business（B2B）, ブロックチェーン, Internet of Things（IoT）, Machine 2 Machine（M2M）, Artificial Intelligence（AI）など, より高度な情報処理を行うために新しいサービスが展開され続けている. この状況に合わせてインターネットインフラにおける回線容量増加の試みは続けられているが, 依然としてインターネット上は輻輳が多発している状況にある. インターネットの全域で用いられているシングルパスルーティングは, 複数の端末間で経路決定においてリンクを共有する可能性が高いため, トラフィックの集中を起こしやすく, 輻輳を生じさせやすい性質を持っていると言える.

その問題に対してマルチパスルーティングに関するアルゴリズムが幾つか提案されて来てはいるが, ネットワーク全体で同期して経路を決定する前提を置いているため, 計算量オーダが大きい, 収束が遅い, 経路選択の自由度が低いなどの問題を抱えてしまっている. それと同時に, 負荷に適応して経路そのものを再決定することも難しくなっている.

従って, 本研究ではポテンシャルルーティングの手法を応用することで, 負荷の変化に応じて通信経路を柔軟に再決定することが可能なマルチパスルーティングを提案する. 提案手法は Multipath Potential Based Routing（MP-PBR）と名付ける. MP-PBR では負荷に反応して持ち上がるような挙動をする拡散方程式により, 負荷適応的かつネットワーク全体で非同期的に協調しながら計算を行いネットワーク全体にメトリック面を形成する. 従って MP-PBR では, 負荷の変動に応じた局所的な経路再計算が可能になっている. そして, 各ルータにおいては, メトリック面の勾配を考慮しながらパケットを分散転送する. このような手法のため MP-PBR では, 既存のマルチパスルーティングで支持されているようなネットワーク全体に対して厳格に経路を決定してデータパケットを転送するパラダイムから, データパケットが柔軟に形状が変化する曲面の上を自由に転がり落ちるようなパラダイムに転換していると言える.

目 次

第1章　はじめに

この章では, 本研究の背景, 目的, 本論文の構成を述べる.

1.1　背景

インターネットは史上初めて民間に開放された 1990 年代初頭から, 各種組織の公式 Web サイト, 電子掲示板, 検索エンジン, YouTube に代表される動画共有サイト, Facebook と Twitter に代表される SNS, BitTorrent に代表されるファイル共有ソフトウェア, Business to Business（B2B）, ブロックチェーン, Internet of Things（IoT）, Machine 2 Machine（M2M）, Artificial Intelligence（AI）などの新たなサービスが次々と展開されている状況下にある. 従って, 継続的にインターネット上のデータ転送量が増加している状況下にある.

インターネットの全域では Routing Information Protocol（RIP） [1], Open Shortest Path First（OSPF） [2] [3], Border Gateway Protocol（BGP） [4] に代表されるように, 経路の合計コストを最小化する単一の通信路を設けるシングルパスルーティングに関するルーティングプロトコルが主に用いられている. 上記のルーティングプロトコルの基礎設計は, インターネット上の通信量が非常に少なかった 1990 年代初頭までに完了している. ネットワークトポロジのみを考慮して最適経路を算出して利用するシングルパスルーティングには, 経路計算に関わる負荷を少なく抑えられる, 経路切替によるパケットリオーダリングが生じ難くなるという利点があるものの, 複数の端末ペアで決定される通信路が重複しやすく, インターネット内に膨大なトラフィックが生じると複数の経路が重複して使用するリンクにトラフィックが集中することで, 輻輳が起きやすくなるという欠点も存在する.

シングルパスルーティングを採用することで生じる制約を根本的に回避する方法として, 複数の経路を設けて同時に使用するマルチパスルーティングが提案されている. 実用化

されているマルチパスルーティングの手法には, 複数の最適経路を同時に使用する Equal Cost Multi Path（ECMP）とネットワーク管理者の運用ポリシーを反映するトラフィックエンジニアリングがある. しかし, ECMP に関しては, RIP, OSPF, BGP 等の既存のルーティングプロトコルで利用可能な追加機能として幅広いルータに実装されているが, インターネット内部で発見できる最適経路の数が非常に限られており, 十分に負荷分散が行えないという欠点を抱えている. また, トラフィックエンジニアリングに関しては, トラフィック分散の観点から言えばネットワーク管理者自身が全体最適を考慮して経路を決定しなければならないため, ネットワークの規模拡大に伴って運用コストが飛躍的に増大するという欠点を抱えている. 実際のインターネット上においては, これらの手法は非常に限定的にしか導入されておらず, 通信負荷分散において主流の手法にはなり得ていない.

手動で複数の経路を決定する事により生じる運用面での負担増大に関する問題を解決するため, OSPF-OMP [5], MLB ルーティング [6], MPATH [7], MDVA [8], YAMR [9], MARA [10], AOMDV [11], MP-OLSR [12] などの動的なマルチパスルーティングアルゴリズムが提案され, 一部のアルゴリズムに関しては実用化を見据えた実験的な運用を行っている最中にある. しかし, どのマルチパスルーティングアルゴリズムに関しても, 計算量オーダが大きい, 経路収束が極めて遅い, パケットリオーダリングへの対策が不十分, 経路選択の自由度が低いなど, 実運用を行う上で解決しなければならない問題が未だに残されており, 動的なマルチパスルーティングの実用化は難しい段階にあると言える. 動的なマルチパスルーティングの研究の中には, 負荷適応型の経路制御も含めて動的なマルチパスルーティングを実現しようとする研究 [5], [6] がある. 従って, 動的なマルチパスルーティングに関する研究は, 経路振動の問題から実用化が難しいと言われている負荷適応型の経路制御の実用化に繋がる可能性もある.

1.2 目的

シングルパスルーティングでは複数の端末ペア間で経路の重複が起きやすく, データパケットが特定のリンクに集中することで輻輳が起きやすい. また, 既存の動的なマルチパスルーティングにおいては, ネットワーク全体が同期して計算することを前提に置いており, 計算量オーダが大きい, 収束が遅い, パケットリオーダリングへの対策が不十分, 経路選択の自由度が低いなどの致命的な問題がある. 従って, 本研究においては, 計算量と収

束にかかる時間を抑えた上で, 通信状況の変化に応じて柔軟に通信経路の再計算を行うアルゴリズムを提案する.

1.3 本論文の構成

第 2 章ではマルチパスルーティングの概要と目的を述べ, 続いて既存のマルチパスルーティングに関する代表的な事例を挙げながら, それらの諸特性を説明する. 第 3 章でポテンシャルルーティングの動作機構を説明した後, 第 4 章では提案手法としてポテンシャルルーティングをマルチパスルーティングに拡張した Multipath Potential Based Routing（MP-PBR）に関して説明する. 第 5 章では提案手法を実装する前提となる ns-3 [13] と提案手法の実装に関して説明する. 第 6 章では, 提案手法の実装に関して, 既存手法との実験結果の比較を示す. 第 7 章では実験結果を分析し, 本手法が適用可能な状況に関して定性的に述べ, 続いてパケットリオーダリングを防ぐ手法の概要を述べる.

第2章　既存のマルチパスルーティングアルゴリズム

マルチパスルーティングは負荷分散や耐障害性の向上を目的として, 複数の経路を用意して同時に使用する手法である. この章では, 負荷分散を目的とするマルチパスルーティングに先駆けて行われていた, シングルパスの負荷適応型ルーティングアルゴリズムに関する研究 [14] [15] [16] を挙げた後に, 既存のマルチパスルーティングに関する代表的な事例として MLB ルーティング [6], MDVA [8], AOMDV [11], OSPF-OMP [5], MP-OLSR [12] を挙げ, その手法の概要を説明する. 既存の手法は何れも, 1 回の経路計算内でネットワーク全体に明確に経路を決定する. このようにネットワーク全体に明確に経路を決定する場合, 経路の形状を変更する際には各ルータがネットワーク全体に経路情報を送信する必要が生じるため, ネットワーク上の局所的な負荷の変動に応じて柔軟に経路の形状を再決定することは難しくなる. そのため, 負荷に適応するように局所的な経路計算のみで局所的に経路形状を変更することでトラフィック分布を最適化するアルゴリズムを提案する余地が存在すると言える.

2.1　シングルパスルーティングの概要

インターネットでは, ルータ数の増大を考慮して, 運用者が手動で通信経路を決定するのではなく, ルータに最適経路を自動で発見させるシングルパスルーティングを考案し, 実運用を行ってきている. これまでに, 経路情報の伝播の過程で各ルータが分担して経路計算を実行するディスタンスベクタ型アルゴリズム, 管理者のポリシーに基づいて分担して経路計算を実行するパスベクタ型アルゴリズム, 各ルータがネットワーク全体のトポロジ情報を利用して経路計算を行うリンクステート型アルゴリズムが考案され実運用されてきている. 具体的なプロトコルとしては順に RIP [1], OSPF [2] [3], BGP [4] が挙げられ

る. このような最適経路のみを利用するルーティングプロトコルでは, 端末間で決定される経路が重複しやすく, インターネット上の特定の箇所にトラフィックが集中しやすくなり, 経路の故障による通信障害に弱くなる点が問題となる. 従って, 負荷に適応して単一の経路を設ける負荷適応型ルーティングアルゴリズムに関する研究が開始された.

2.1.1 負荷適応型ルーティングアルゴリズム

輻輳を回避するために, 各ルータのメトリック値の設定において, リンクやルータに掛かっている通信負荷を考慮する負荷適応型ルーティングアルゴリズムが考案されている.

ARPANET における最初のルーティングアルゴリズム

挙動に着目した場合に負荷適応型ルーティングアルゴリズムと呼べる最初のアルゴリズムは, 1970 年に発表された, ARPANET 上の Interface Message Processor（IMP）に実装されたトランジットタイムを最小化するディスタンスベクタ型のルーティングアルゴリズム [14] である. これは 1966 年の B. Boehm と R. Mobley による先行研究 [17] で提示されていた分散コミュニケーションシステムにおける動的ルーティングの手法を応用したものである. このアルゴリズムは 2[s] 毎に各ルータが各送信先までのトランジットタイムを最小にする経路を評価し, 0.5[s] 毎にトランジットタイムの最小化を達成する経路を経路表に書き込むルーティングアルゴリズムとして設計された. しかし, これはメトリック値の設定においてトランジットタイムを考慮することによる副次的な効果に過ぎず, 本来的な意味では異なる目的のアルゴリズムである. 送信元から送信先までのトランジットタイムの合計のみを最小化するように最適経路を計算する場合, 物理的な距離により生じるトランジットタイムの効果が大きく働くため, パケットが物理的に最短となる経路を通る可能性が非常に大きくなり, 負荷分散は非常に限定的にしか行うことが出来ない.

BBN ARPANET ルーティングアルゴリズム

本格的な負荷適応型ルーティングアルゴリズムとして提案された最初の主要な事例としては Bolt Beranek and Newman（BBN）社から 1980 年に提案された BBN ARPANET ルーティングアルゴリズム [15] がある. BBN ARPANET ルーティングアルゴリズムは,

メトリック値の設定において通信負荷を考慮するリンクステート型のルーティングアルゴリズムである. このルーティングアルゴリズムに関しては ARPANET 上で実験的な運用も試みられていた. しかし, 輻輳を避けるために大きく経路を変更すると別の場所で輻輳が始まり再び元の経路に戻る過程を繰り返す経路振動の問題が発生し, ネットワーク全体として通信負荷の集中と分散を繰り返し続ける状態に陥ることが判明し, この試みは失敗に終わった. 暫く後の 1989 年に, メトリック値の変動の範囲を制限することで経路振動が抑えられ, ネットワーク全体で通信負荷の平均化を行えることは分かった [16] が, その段階に至っても実運用において安全であるという確証は得られず, 実運用に至る事は無かった.

現在の状況

インターネットの歴史の初期において前述のような試行錯誤が行われ, 経路制御を破綻させるような重大な問題が発覚して実用化が頓挫した事例があるため, 現在に至るも負荷適応型ルーティングアルゴリズムに関しては経路振動の発生が懸念され続けており, 研究そのものが下火になってしまっている. しかし, 未だに経路振動に関して安全性を担保した負荷適応型ルーティングアルゴリズムに関する研究 [5] [18] [6] や実験的な運用は続けられている模様である. 従って, 負荷適応型ルーティングアルゴリズムに関して実用化されている手法は現時点では存在していないと言える. 負荷適応型ルーティングアルゴリズムに関する研究からポテンシャルルーティング [18] という新たなパラダイムが登場し, Disruption-Tolerant Networking（DTN）におけるルーティングの研究に大きく貢献しているため, 次の節で説明を行う.

ポテンシャルルーティング

負荷適応型か否かを問わず, 既存のルーティングアルゴリズムの殆ど全てはネットワーク全体で同期しながら経路計算を行うことを前提に置いている. ネットワーク全体で同期して経路を計算する場合, 局所的な負荷の変動に適応して局所的に経路を変更する場合でもネットワーク全体のルータに経路再計算の負荷が掛かる事が問題となり得る. そのため, ネットワーク全体で非同期に負荷適応型ルーティングアルゴリズムを行う事ができるポテンシャルルーティング [18] が提唱されている. ポテンシャルルーティングの場合, 明

確な経路を設けるというよりも, 曲面の上をパケットが転がり落ちるようなパラダイムに転換しており, 負荷に反応して持ち上がるような挙動をする拡散方程式による局所的な経路再計算により, 曲面の形状を局所的に変更することで, 柔軟な負荷適応型ルーティングアルゴリズムを実現している. ポテンシャルルーティングにおいても負荷の平均化に成功しているが, 拡散方程式により算出されるメトリック値の範囲が限られていることから, 前述の [16] と同様の対策がポテンシャルルーティングにおいて暗黙の内に織り込まれた結果と言える. 負荷適応型ルーティングアルゴリズムにおける経路振動発生の恐れが払拭出来ないため, ポテンシャルルーティングに関しては, 現在は非同期な経路再計算の側面にのみ注目が集まっており, ネットワークトポロジが不安定である Disruption-Tolerant Networking（DTN）（代表的な事例としては惑星間インターネットやセンサネットワーク等が挙げられる）におけるルーティングアルゴリズムとして研究（例えば [19] [20] [21]）が継続されている.

2.2 マルチパスルーティングの概要

シングルパスルーティングにおける課題を根本的に解決するために, 端末間で複数の経路を決定して併用するマルチパスルーティングが考案され, 様々なアルゴリズムに関する性能検証が行われている. 具体的には下記のようなアルゴリズムが存在する.

2.2.1 ディスタンスベクタ型アルゴリズム

ルータ間で送信先に関する経路情報を伝播させる過程で, メトリックを増加させるようにして経路を計算する. 各ルータで経路計算に関わる負荷を分散して端末間の経路を決定する事が可能である.

MLB ルーティング

MLB ルーティング [6] では非集計分析により, ネットワーク全体のトラフィック分布の最適化を行っている. 非集計分析は「個人が交通行動の基本的な意志決定単位であるとし, 個人はある選択状況の中から効用を最大化する選択肢を選ぶように行動する」として, 理

想的な交通人の行動を確定項と誤差項によりモデル化して分析する手法である. 最も多用されているモデルはロジットモデルであり, MLB ルーティングでもそのモデルを採用している. MLB ルーティングでは誤差項のみをモデル化し, 下記のような式 (2.1), 式 (2.2) で表記する.

$$P_r^{od} = \frac{\exp\left[-\gamma C_r^{od}\right]}{\sum_{r\in\Phi^{od}} \exp\left[-\gamma C_r^{od}\right]} \tag{2.1}$$

$$C_r^{od} \stackrel{\text{def}}{=} c_{ov_1} + \sum_{i=1}^{\Lambda-1} c_{v_1 v_{i+1}} + c_{v_\Lambda d} \tag{2.2}$$

上記の式 (2.1) により示される確率 P_r^{od} は, パケットが $o \to v_1 \to v_2 \to \cdots \to v_\Lambda \to d$ の経路を転送される確率である. 式 (2.2) により示されるコスト C_r^{od} は経路に沿ったリンクコストの合計である. r はルータ o とルータ d の間の経路集合 Φ^{od} に含まれる経路のうちの 1 本である. γ は確率分布の形状を調節するためのパラメータである.

実際の交通行動における誤差項の分布がガンベル分布, 正規分布のどちらであるかは分かっていないため, 離散的な行動データからロジットモデルのパラメータを回帰して決定しておき, 回帰して求めた連続的な効用値を用いて, 将来予測の際に離散的行動を判別する必要がある. MLB ルーティングでもロジットモデルにおける自動的なパラメータ調節に関する研究が行われている. MLB ルーティングの実現方法はディスタンスベクタ型, リンクステート型を問わないが, 提唱者らはディスタンスベクタ型のルーティングアルゴリズムとして実証実験を行っている. ディスタンスベクタ型として実現する場合, 上記のロジットモデルを下記の式 (2.3), 式 (2.4) で記述されるマルコフ連鎖として書き直す.

$$p\left(j|i\right) = \exp\left[-\gamma c_{ij}\right] \frac{W_{jd}}{W_{id}} \tag{2.3}$$

$$W_{id} \stackrel{\text{def}}{=} \sum_{r\in\Phi^{id}} \exp\left[-\gamma C_r^{id}\right] + \delta_{id} \tag{2.4}$$

δ_{id} はクロネッカーのデルタである．そして, ルータ間を経路情報が転送される過程で, 各ルータで下記の式 (2.5) のような行列計算が行われる．

$$W = I + A[I - A]^{-1} \tag{2.5}$$

上記の式 (2.5) において, W は自ルータから各隣接ルータに向かうパケット転送の比率を決定する重み行列であり, 各要素は前述のマルコフ連鎖で見られた確率 $p(j|i)$ である．A は隣接ルータから転送されてきた重み行列であり, こちらも各要素は $p(j|i)$ である．数学的には正しくない表記であるが, 直感的には経路情報がルータを通過する毎に, 下記の式 (2.6) のような再帰的な計算が実行されるイメージである．

$$W = I + AW \tag{2.6}$$

ネットワークに参加するルータの個数を N と置いたとき, A は次数 N の正方行列である．MLB ルーティングでは, 経路計算において逆行列計算 $[I - A]^{-1}$ を実行し, その計算が最も負荷が大きいため, 計算量オーダは $O(N^3)$ である．

MDVA

MDVA [8] ではディスタンスベクタ型ルーティングアルゴリズムであるベルマンフォード法に対してマルチパス拡張を行っている．MDVA ではネットワーク全体に経路情報を送信する過程で Directed Acyclic Graph（DAG）を計算し, 経路間でリンクやルータを共有しない（disjoint 性を満たす）マルチパスを用意する．その 1 つの例が図 2.1 に示されている．ルータの次数を K, ネットワークに参加するルータの個数を N と置いたとき, ストレージサイズは $O(K * N)$ で増加する．計算量に関してはネットワーク中の全てのリンクに関するコストの変更が収束するまで増大する．

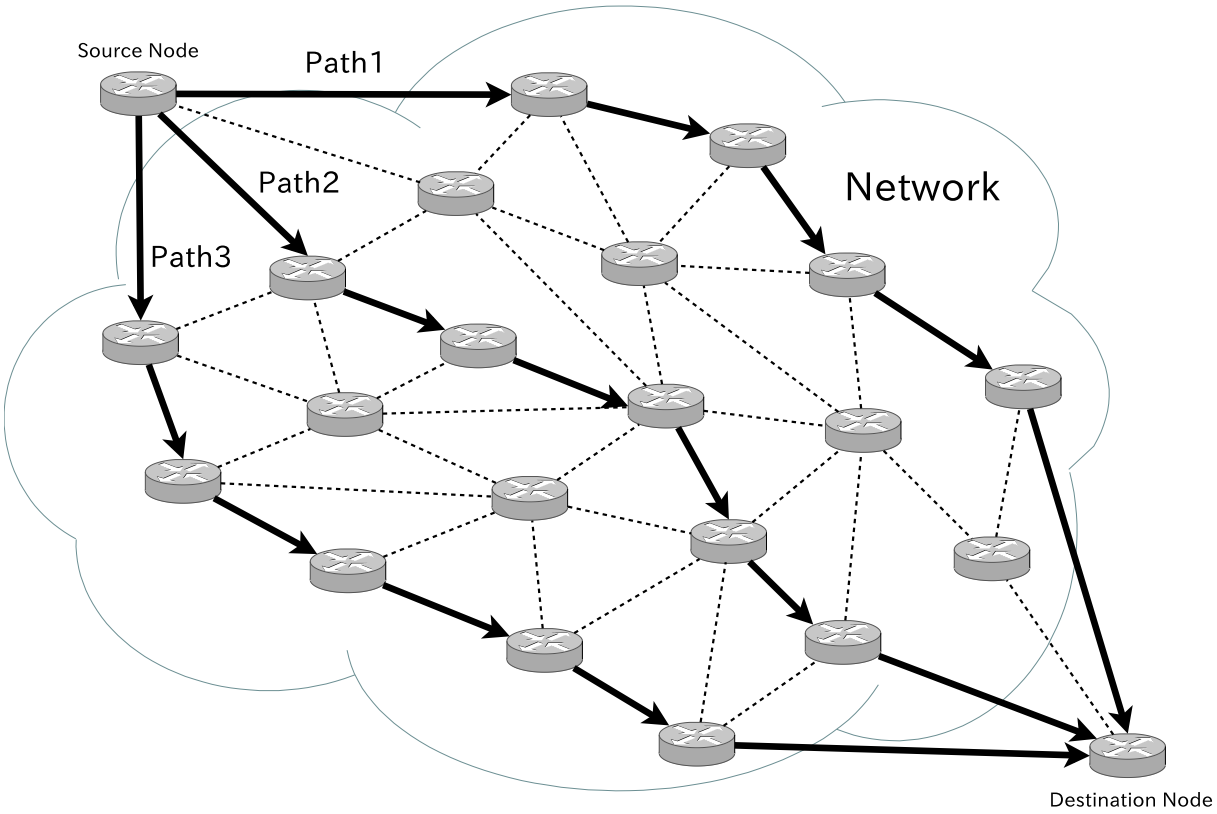

図 2.1: disjoint 性を満たすマルチパスの例

AOMDV

AOMDV [11] では AODV [22] に対してマルチパス拡張を行っている. AODV は端末間の通信開始時に経路を形成するリアクティブ型かつ, 各ルータ間で経路情報を伝播させる過程で経路計算が進行するディスタンスベクタ型のルーティングプロトコルである. AODV では送信元がシーケンスナンバー付きの経路情報である Route REQuest（RREQ）をネットワーク全体にフラッディングし, 送信先が最初に到達した RREQ に対して Route REPly（RREP）を返信することで端末間の経路を作成する.

AOMDV ではネットワーク全体に経路情報を送信する過程で, 送信端末の隣接ルータを経路の起点として, リンクやルータを共有しない（disjoint な）経路を複数用意する. 具体的には, 1 つの送信元から送られたシーケンスナンバーが異なる RREQ 全てに対して送信先が RREP を返信するが, 送信先において到着した RREQ のシーケンスナンバーを確認し, 重複したシーケンスナンバーを持つ RREQ に関しては最初に届いたもののみに RREP を送り返すようにする. こうすることで, リンクを共有しない disjoint な経路を作成することが可能になる. 計算量に関してはネットワーク中の全てのリンクに関するコストの変更が収束するまで増大する.

2.2.2　リンクステート型アルゴリズム

各ルータがネットワーク全体のトポロジを把握して経路を計算する方式である.

OSPF-OMP

OSPF-OMP [5] では各ルータが定期的にネットワーク全体のリンクの負荷状態を収集しながら, 最大流問題を解くようにして複数の経路を計算する. 各ルータは全ての隣接ルータに向かう経路のコストを計算し, そのコストの比率に合わせてパケットを分散転送する. その様子は図 2.2 に示されている. 従って, 各ルータは, 定期的にネットワーク全体からリンク負荷に関する情報を収集してトポロジマップを作成し, 各リンクに対してコストを設定し, ダイクストラ法を隣接ルータの個数回だけ実行することになる. 実用化に最も近い負荷適応型のマルチパスルーティングアルゴリズムであると言われていたが, 前述のような計算方法のため計算量オーダが, ルータの次数を K, ネットワークに参加するルータの個数を N と置いたとき, $O\left(K * N^2\right)$ と極めて大きくなり, ネットワーク全体を常に監視し続けるため転送すべき経路情報量が非常に多くなり, 経路収束も極めて遅いため, 実用化されなかった.

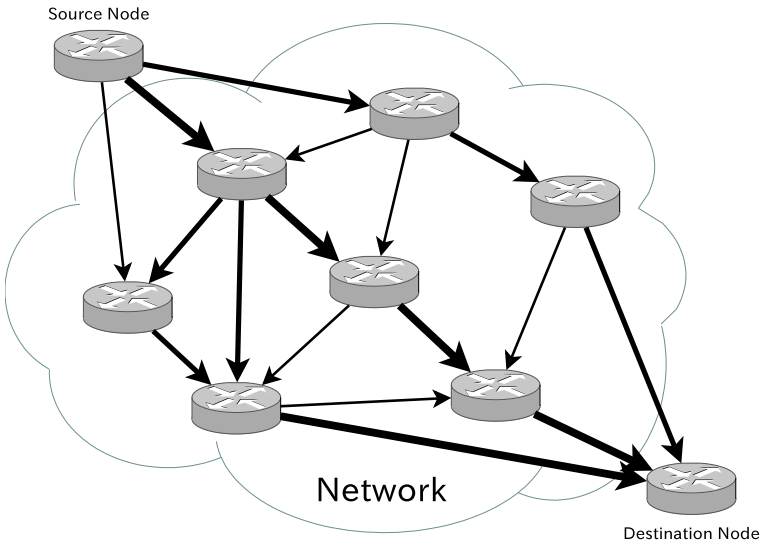

図 2.2: OMP によるトラフィックの最適化

MP-OLSR

MP-OLSR [12] では OLSR [23] [24] に対してマルチパス拡張を行っている．OLSR は, 常時経路を維持するプロアクティブ型かつ, 各ルータがネットワーク全体からトポロジ情報を収集して経路を計算するリンクステート型のルーティングアルゴリズムである．無線ネットワークにおけるリンクステート型のルーティングプロトコルでは, 各ルータがフラッディングを行うことでネットワーク全体のトポロジ情報を収集して経路を計算する．OLSR では, マルチホップ無線ネットワークで効果的にフラッディングを実行するために, MultiPoint Relay（MPR）集合と呼ばれる必要最小限の数の再送信ルータの集まりを規定している．OLSR では, 各ルータが MPR 集合を発見し, MPR 集合に基づきフラッディングを行ってトポロジ情報を収集し, 続いて収集したトポロジ情報に対してダイクストラ法を実行することで最適経路を発見する．その MPR 集合を用いたフラッディングに関する 1 つの例が図 2.3 に示されている．

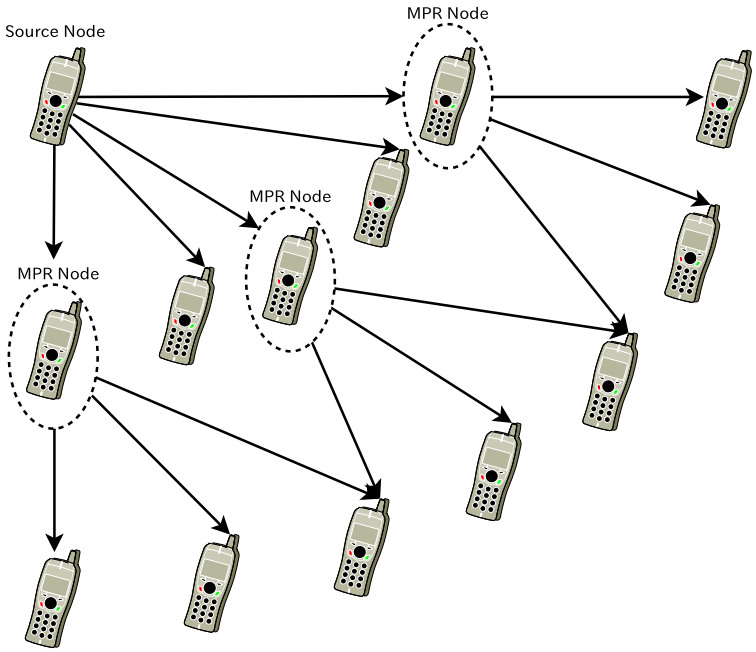

図 2.3: MPR 集合を用いたフラッディング

MP-OLSR の場合は, 前述の OLSR の手順に基づき各ルータがネットワーク全体からトポロジ情報を収集した後, リンクやルータを共有しない（disjoint な）経路を複数用意するために, 既に経路決定に用いたリンクのコストを増大させながら, 作成したい経路の本数回だけダイクストラ法を実行する．従って, 作成する経路の本数を R, ネットワークに参

加するルータの個数を N と置いたとき, 計算量オーダーは $O\left(R * N^2\right)$ となる. MP-OLSR はソースルーティングを行うが, ソースルーティングはパケットヘッダ内に経路情報を含めるため, ホップ数が増加するとともに経路情報のサイズが大きくなり, 各ルータにおける計算量も飛躍的に増大してしまう. 従って, MP-OLSR はスケーラビリティに関する問題を抱えている.

第3章　ポテンシャルルーティングの概要

この章ではポテンシャルルーティングの動作機構に関して述べる.

3.1　ポテンシャルルーティングの定義

ポテンシャルとは, 物理的な場が物体に潜在的に与える働きのことである. ポテンシャルルーティングはネットワーク上に場を形成し, その場の勾配がパケットの転送方向を決定するようなルーティングの手法のことである. 下記の図 3.1 にその概要を示す. 図 3.1 には, メトリック面の上にパケットが置かれており, メトリック面の勾配に応じてパケットに力が働き, その力の方向にパケットが転送される様子が示されている.

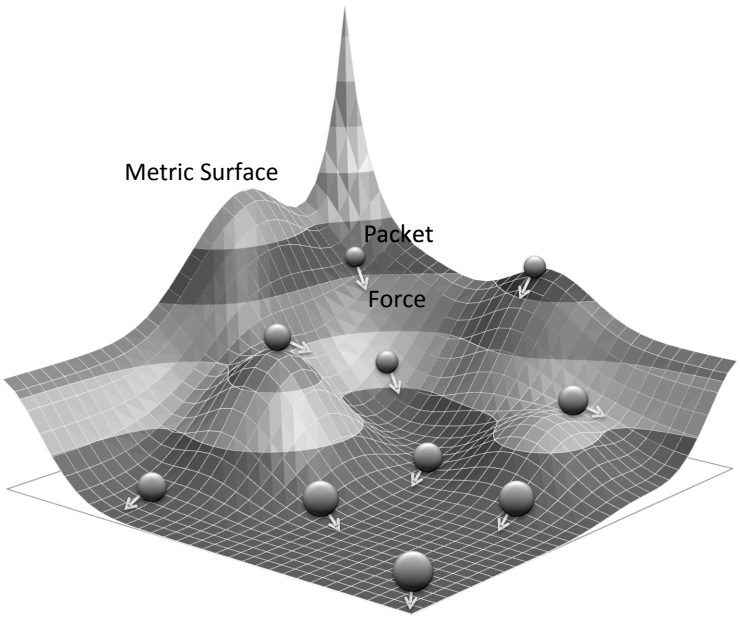

図 3.1: ポテンシャルルーティング概要図

3.2 有限要素法による分散計算と局所最適解の探索

ポテンシャルルーティングとは, 有限要素法による拡散方程式の数値解析をルーティングに応用したものである. 拡散方程式は熱や物質の拡散現象を数式でモデル化したものに当たり, その解は正規分布の形となる. この拡散方程式の解析領域における数値解がポテンシャルルーティングにおけるメトリックである.

拡散方程式は式 (3.1) のようになる.

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2} \tag{3.1}$$

また, 拡散方程式の差分化を行うと式 (3.2) のようになる.

$$\frac{f\left(x, t+\Delta t\right) - f\left(x,t\right)}{\Delta t} = D\frac{f\left(x+\Delta x, t\right) - 2f\left(x,t\right) + f\left(x-\Delta x, t\right)}{\Delta x^2} \tag{3.2}$$

上記の式では隣接関係にある要素 $(x+\Delta x)$, $(x-\Delta x)$ のみが現在の位置 x に対して影響を与える事が読み取れる. 従って, 全ての差分の間隔を 1 とした上でグラフ上にマッピングすると, 式 (3.3) になる.

$$f\left(n, t+\Delta t\right) - f\left(n,t\right) = D\sum_{k\in \mathrm{nbr}(n)}\left\{f\left(k,t\right) - f\left(n,t\right)\right\} \tag{3.3}$$

関数表記 f をポテンシャル表記 V に置き換え, 現在のポテンシャルの値 $V\left(n,t\right)$ を右辺に移項すると, 式 (3.4) になる.

$$V\left(n, t+\Delta t\right) = V\left(n,t\right) + D\sum_{k\in \mathrm{nbr}(n)}\left\{V\left(k,t\right) - V\left(n,t\right)\right\} \tag{3.4}$$

考慮する指標に応じて項を追加する. 例えば送信先からのホップ数を考慮してパケットを転送する場合には, 正の定数 Hop を加え, 式 (3.5) にする.

$$V(n, t+\Delta t) = V(n,t) + D \sum_{k \in \mathrm{nbr}(n)} \{V(k,t) - V(n,t)\} + \mathrm{Hop} \tag{3.5}$$

数値解析を行うに当たり, 初期条件として $V(n,0) = 0$, 境界条件として送信先ノード d において $V(d,t) = 0$ とする。拡散方程式の数値解の上で, データパケットに働く力を式 (3.6) のように定義する.

$$F = V(n,t) - V(k,t), k \in \mathrm{nbr}(n) \tag{3.6}$$

この力が最大となる方向にパケットを配送する. 各ノードでデータパケットに働く最大の力は式 (3.7) のように定義される.

$$F_{\max} = \max_{k \in \mathrm{nbr}(n)} \{V(n,t) - V(k,t)\} \tag{3.7}$$

この時, 隣接ノードから自ノードに向けて経路情報が入って来ない場合には, メトリック値の変動が止まり次第, 計算ループをサスペンドする. 更に, 経路計算パケットの送出に関して下記の式 (3.8) を満たした時, 経路情報の送出を停止する. この条件を追加することで, 局所的な経路再計算による局所的な経路変更が可能になる. 下記の条件において $V(n,t)$ は現在の自ノードのメトリック値, $V\left(n, t_{\mathrm{last}}\right)$ は最後に経路情報を送出した時の自ノードのメトリック値である. MetricThreshold は経路情報を送出し始めるメトリック値の変動量である.

$$|V(n,t) - V\left(n, t_{\mathrm{last}}\right)| \leq \mathrm{MetricThreshold} \tag{3.8}$$

ここまで説明した配送規則により, 最急降下法によるパケット配送が実現されている. その様子が図 3.2 に示されている.

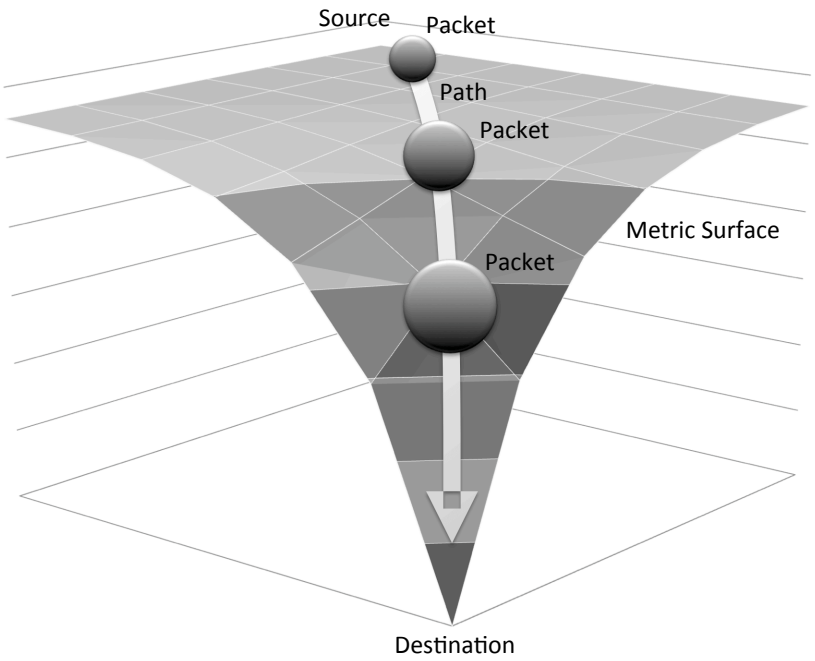

図 3.2: ホップ数のみを考慮したメトリック計算とパケット転送

3.3 負荷への適応

ネットワークに掛かる様々な負荷に適応して経路を制御する際には, 差分化した拡散方程式に対して, 式 (3.9) のように負荷を考慮する新たな項 $\beta L(n,t)$ を追加すると良い. この項の内, β は項の効果の大きさを調節する係数であり, $L(n,t)$ は自ノードの負荷, 自ノードに接続されるリンクの負荷の大きさに比例して大きくなる関数値である.

$$V(n,t+\Delta t) = V(n,t) + D \sum_{k \in \mathrm{nbr}(n)} \{V(k,t) - V(n,t)\} + \alpha \mathrm{Hop} + \beta L(n,t) \qquad (3.9)$$

このような項を加えることで, 負荷の周辺でメトリック値が持ち上がるため, 最急降下法により負荷を避けるようにしてパケットが配送される仕組みである. その様子は図 3.3 に示されている.

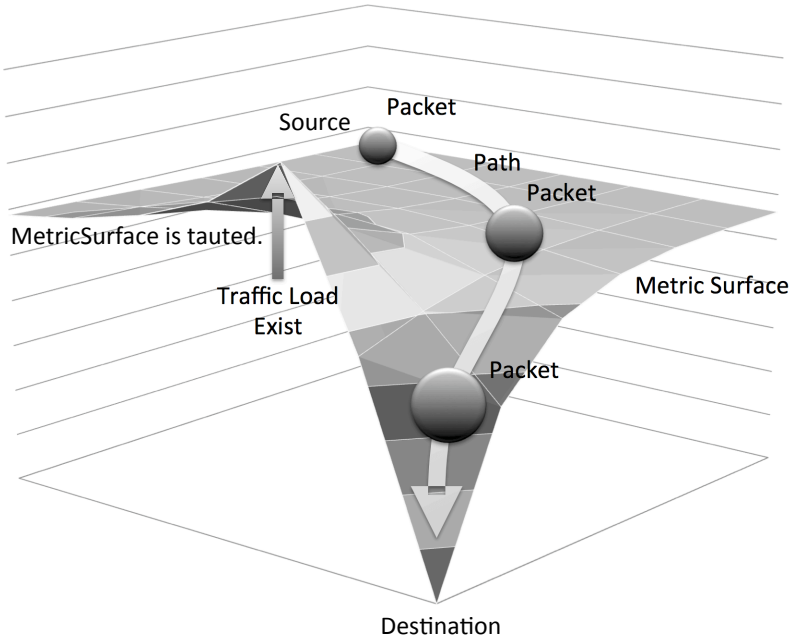

図 3.3: ホップ数とトラフィック負荷の両方を考慮したメトリック計算とパケット転送

第4章　柔軟な経路再選択を可能にするマルチパスルーティングの実現

この章では, ポテンシャルルーティング上でマルチパスルーティングを実現する方法に関して提案を行う. 今回提案するマルチパスルーティングアルゴリズムでは, 経路計算に必要な計算量と経路収束にかかる時間を抑えた上で, トラフィック状況に応じた柔軟な経路再選択を実現する. 今回の提案はポテンシャルルーティングのメトリック計算式の再定義と, ポテンシャルルーティング上のパケットの分散規則の定義である.

4.1　通信負荷の計測とパケット転送経路の適応

ポテンシャルルーティングではメトリック値の計算に用いる拡散方程式を差分化し, 考慮すべき指標に応じて項を追加する. 今回の提案手法における拡散方程式は式 (4.1) のように構成する. 今回はホップ数, リンクコスト, ラウンドトリップタイム（RTT）を同時に最小化するような経路を算出するようにするため, 本来の拡散方程式と比較して 3 種類の項が追加されている. また, 各評価指標の合計に比例して増加する形状とするため, 拡散項は最小値 $\min_{k\in\mathrm{nbr}(n)}\{V(k,t)-V(n,t)\}$ のみを取るように変更した.

$$\begin{aligned} V(n,t+1) = V(n,t) & \\ & + D \min_{k\in\mathrm{nbr}(n)} \{V(k,t)-V(n,t)\} \\ & + \alpha\mathrm{Hop} \\ & + \beta\mathrm{Cost}(n,t) \\ & + \gamma\mathrm{RTT}(n,t) \end{aligned} \tag{4.1}$$

上記の式の右辺の各項は, 前から順番に, 現在のメトリック値, 拡散項, ホップ項, リンクコスト項, ラウンドトリップタイム（RTT）項である. 各項の詳細は下記で説明する. これらの項を拡散方程式に加算することで, 各指標を同時に考慮した経路の最適化が可能になる.

拡散項

拡散項は隣接関係にあるノード間でメトリック値の挙動を牽制させ合うために必要である. この項がない場合, 各ノードは完全に独立にメトリック値を設定できてしまうため, メトリックの場に大きな凸凹を生じ, 送信先へ向かってメトリックが減少するような勾配も生じず, ルーティング自体が成立しなくなってしまう. あまり正確な例えではないものの, ネットワーク全体に生じるメトリックの場にゴムシートのような性質を持たせることが可能になる.

拡散項は自ノードの現在のメトリック値 $V(n,t)$ と隣接ノードの現在のメトリック値の内で最小の値 $\min_{k\in \mathrm{nbr}(n)}\{V(k,t)\}$ との差 $\min_{k\in \mathrm{nbr}(n)}\{V(k,t)-V(n,t)\}$ を取り, 拡散項の効果を調節する拡散係数 D を掛けて最終的な拡散項の値とする. 自ノードと隣接ノードで正負が逆になっている理由は, 自ノードの現在のメトリック値に対して, 自ノードと隣接ノードのメトリック値の差を抑える牽制力を作用させるためである. なお, 拡散係数 D の有効な値域は $0 < D \leq 1$ であり, D がこの範囲を外れると拡散方程式の数値解析の誤差が収束せず, 数値解析自体が破綻してしまう. 経路収束の高速化のためには $D = 1$ と設定するべきである.

ホップ項

ホップ項は各ノードに送信先までのホップ数に比例して高くなるメトリック値を設定するために必要である. この項はメトリック場全体を正の定数 Hop だけ牽引し続ける操作を行い, 拡散項により生じる牽制力とこの項の牽引力が釣り合った段階でメトリックの上昇が停止し, 送信先からのホップ数に比例して高くなるメトリックの場が形成される. ホップ項の効果の強さを調節する係数 α を掛け, 最終的なホップ項の値とする.

リンクコスト項

リンクコスト項は, ネットワーク上に存在するリンクの帯域から算出されるコストの合計値を最小化する経路を発見するために必要である. 算出の仕組みは下記の通りである.

まず各ノード n において, 接続された全てのリンクの空き帯域 BW_k $(k \in \mathrm{nbr}\,(n))$ [bps] を取得する. そして, 式 (4.2) のように, 収集した BW_k $(k \in \mathrm{nbr}\,(n))$ の平均値 $\mathrm{BW}_{\mathrm{ave}}$ を求める.

$$\mathrm{BW}_{\mathrm{ave}} = \frac{\sum_{k \in \mathrm{nbr}(n)} \mathrm{BW}_k}{\left|\mathrm{nbr}\,(n)\right|} \tag{4.2}$$

次に回線の空き帯域を用いてリンクコストを計算する. リンクコストの定義は式 (4.3) の通りである.

$$\mathrm{Cost}\,(n, t) = \frac{1}{\log_b\left(\mathrm{BW}_{\mathrm{ave}} + 1\right)} \tag{4.3}$$

ここで, 対数の底 b はリンクの空き帯域の平均値に対するリンクコストの大きさを制御するための定数として扱う.

最後に, リンクコスト項の効果の強さを調節する係数 β を掛け, 最終的なリンクコスト項の値とする.

ラウンドトリップタイム（RTT）項

ラウンドトリップタイム項は, ラウンドトリップタイムの合計を最小化する経路を発見するために必要である. この項の計算において, 次のような処理を行う. 各ノードは現在時刻を書き込んだ RTT 計測用のパケットを隣接ノード宛に送信する. 各ノードは RTT 計測用のパケットを受け取ると, 受け取ったパケットを送信した隣接ノードに向けて再度送り返す. RTT 計測用パケットの往復が終了した段階で, 各ノードは RTT 計測用パケットに書き込まれた送信時刻と現在時刻の差を取って各リンクの RTT とする. この RTT に効果調節用の γ を掛け, ラウンドトリップタイム項とする. この項を拡散方程式に足し合わせることで, RTT の増加に比例してメトリックを増加させる力が大きく働くようになり,

拡散項と合わせると送信先までの RTT に比例して大きくなるようなメトリック場が形成でき, 端末間の RTT を最小化するような経路選択が可能になる. ラウンドトリップタイム項の効果の強さを調節する係数 γ を掛けて, 最終的なラウンドトリップタイム項の値とする.

4.2 メトリック計算式の空間差分を利用したマルチパス拡張

ポテンシャルルーティングにおけるマルチパスルーティングを実現するため, 各ノードが隣接ノードに向かって生じる, メトリックの負の勾配を利用してパケットを分散配送する手法を提案する. 今回提案するアルゴリズムをアルゴリズム 1 に示す. アルゴリズム中の表記法は表 4.1 に示す.

表 4.1: アルゴリズム中の表記

記号	定義
i	ノードを区別する添字 $i \in I$
n	ノード n
$\mathrm{nbr}\,(n)$	ノード n における隣接ノードの集合
k	隣接ノード $k \in \mathrm{nbr}\,(n)$
d	送信先ノード
Sum	メトリックの勾配の合計値
Force	メトリックの勾配によりパケットに働く力
Rnd	乱数値
Random (Min, Max)	Min から Max までを出力する乱数関数
D	ノード n が保持するメトリック値の集合
D_i	ノード i におけるメトリック値
Threshold	パケット転送可否を判断する閾値
$K_{\mathrm{candidated}}$	転送先の候補となる隣接ノードの集合
$K^i_{\mathrm{candidated}}$	$K_{\mathrm{candidated}}$ に属す i 番目の隣接ノード
$R_{\mathrm{candidated}}$	転送先の候補となる隣接ノードに対応した値域を保存する集合
$R^i_{\mathrm{candidated}}$	$R_{\mathrm{candidated}}$ に属す i 番目の値域

Algorithm 1: Gradient-aware Random Packet Dispatch（GRPD）

Procedure `GRPD(`n,$nbr(n)$,d,D`)`

Data: n is a node n. $\mathrm{nbr}(n)$ is a set of the neighbors around the node n. d is a destination. D is a set of the distances holded by the node n.

Result: Determine one of the $k \in \mathrm{nbr}(n)$ as a Next Hop

begin

`/* Part 1 */`

$\mathrm{Sum} \longleftarrow 0.0$;

forall $k \in nbr(n)$ **do**

$\mathrm{Force} \longleftarrow D_n - D_k$;

if *Force* $>$ *Threshold* **then**

$\mathrm{Sum} \longleftarrow \mathrm{Sum} + \mathrm{Force}$;

$R_{\mathrm{candidated}} \longleftarrow R_{\mathrm{candidated}} \cup \mathrm{Sum}$;

$K_{\mathrm{candidated}} \longleftarrow K_{\mathrm{candidated}} \cup k$;

else if $k = d$ **then**

return k;

endif

end

`/* Part 2 */`

if $|K_{candidated}| = 0$ **then**

return NULL;

endif

`/* Part 3 */`

$\mathrm{Rnd} \longleftarrow \mathrm{Random}(0.0, \mathrm{Sum})$;

for $i = 0$ *to* $|K_{candidated}|$ **do**

if $Rnd < R^i_{candidated}$ **then**

return $K^i_{\mathrm{candidated}}$;

endif

endfor

end

end

Data

入力データとして, ノード n, ノード n における隣接ノードの集合 $\mathrm{nbr}\,(n)$, 送信先ノード d, ノード n が保持するメトリック値の集合 D が与えられる.

Result

出力結果として, ネクストホップとなる隣接ノード $k \in \mathrm{nbr}\,(n)$ の 1 つを返す.

Part 1

メトリックの勾配の合計値 Sum を 0.0 に初期化する. 隣接ノード $k \in \mathrm{nbr}\,(n)$ を順次走査し, メトリックの勾配によって隣接ノードに向かって生じている力 Force のうち, Threshold 以上の値を持つ力を Sum に足す. その際に, 加算中の Sum を参照中の隣接ノードに対応した確率変数の区間として保存する. この過程で隣接ノード $k \in \mathrm{nbr}\,(n)$ が送信先ノード d であればその隣接ノードをネクストホップとして返す.

Part 2

転送先の候補となる隣接ノードが存在しなければネクストホップとして NULL を返して終了する.

Part 3

0.0 から Sum までの確率変数の区間で乱数値 Rnd を 1 つ生成する. 転送先の候補となる隣接ノード集合 $K_{\mathrm{candidated}}$ を $i = 0$ から $i = |K_{\mathrm{candidated}}|$ まで順次走査し, 乱数値 Rnd がノードに割り当てられた確率変数の上限値 $R^i_{\mathrm{candidated}}$ を下回っていた場合に参照中の隣接ノード $K^i_{\mathrm{candidated}}$ をネクストホップとして返す.

負荷適応型のポテンシャルルーティングの上でアルゴリズム 1 を実行した結果として形成される経路は下記の図 4.1 のようになっている. 図 4.1 には, パケットが送信先に向かって, 水の流れのように広がりながらメトリック面の勾配を下るように転送されて行く様子が描かれている.

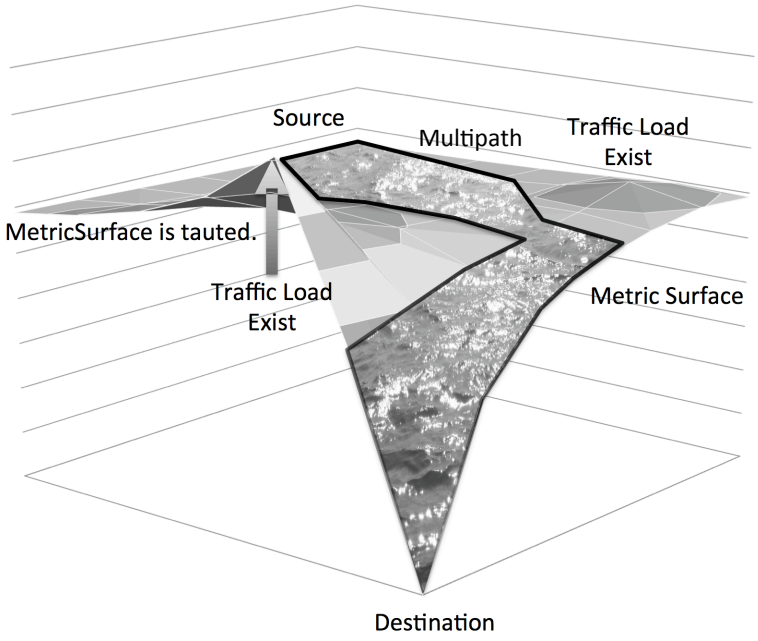

図 4.1: 負荷適応型のポテンシャルルーティングにおけるマルチパスルーティングの様子

第5章　実装

この章では提案手法である MP-PBR の実装に関して述べる. 主に実装の前提となるネットワークシミュレータである ns-3 [13] と MP-PBR の実装モジュール構成に関して説明する.

5.1　ネットワークシミュレータ ns-3 の概要

この項ではネットワークシミュレータ ns-3 [13] の内部構造に関して説明する.

5.1.1　ns-3 のシミュレーションシナリオの記述で用いる概念

ns-3 を用いたシミュレーションシナリオの記述では下記の概念を意識する必要がある. ns-3 内部のソフトウェアスタックでは, 下記抽象概念の各々が C++を用いて単一のクラスとして実装されている. これらのクラスを継承しカスタマイズすることで, 具体的な機能を定義してシミュレーション内で用いることが出来るようになっている.

Node

ネットワークに参加するノードを指す. ns-3 における Node と呼ばれる概念は, インターネット・プロトコル・スタックに密接に関連付けられているわけではなく, どちらかと言えばグラフ理論におけるノードに近いイメージである. ns-3 におけるルータはノードに持たせる機能の 1 つとして定義されている.

Application

コンピュータ・ソフトウェアのアプリケーションに該当する．現実のコンピュータシステムであれば, システム・ソフトウェア側のトラップの発生によるアプリケーションの特権レベルの変更が実装されているが, ns-3 では現実のオペレーティングシステムの再現は目指しておらず, 特権レベルの変更やシステムコールは実装していない．ns-3 における Application の動作はどちらかと言えば, 現実の空間で複数の人間により同時並行して行われる作業のイメージに近い．シミュレーション空間内では ns-3 の Node 上で動作する．

Channel

現実世界ではコンピュータをネットワークに接続することができる．この時, ネットワークにおいてデータが流れる媒体をチャンネルと呼ぶ事が多い．ns-3 では Node を通信チャンネルを表現するオブジェクトに接続する．ns-3 でも基本的な通信サブネットの抽象はチャンネルと呼ばれている．

NetDevice

コンピュータをネットワークに接続する際に必要となる Network Interface Controller（NIC）を抽象化したものである．現実の NIC はハードウェアをコントロールするためのソフトウェアドライバ無しでは動作しない．そのため, ns-3 における NetDevice ではソフトウェアドライバとハードウェアを統合して抽象化している．NetDevice は Node に組み込まれ, NetDevice に Channel を接続することで, Channel を介して他の Node と通信を行うことを可能にする．

Topology Helpers

現実のネットワークには, NIC を追加したコンピュータが存在する．ns-3 では NetDevice を追加した Node が相当する．従って, 大規模なシミュレーションネットワークを構築するためには, Node, NetDevice, Channel, IP アドレスなどの多数の組み合わせを設定しなければならない．しかし, シミュレーションネットワークの構築では共通の作業プロセス

を経ることが多いため, 煩雑な作業を抽象化し, 無駄な作業の繰り返しを避け, 作業を簡単に済ませるために Topology Helper が用意されている.

5.1.2 ns-3 のスタック構成

ns-3 は表 5.1 のようなスタックで構成されている.

表 5.1: ns-3 のソフトウェアスタック

helper	
application	
internet	
network	mobility
core	

これらの各々に関して説明する.

helper 層

シミュレータ内部の詳細な実装を隠蔽し, 前述の抽象的な概念のみを用いてシミュレーションシナリオの記述を行うためのラッパーを実装する. 例えばルータ, ネットワークトポロジ, アプリケーションなどに対応するヘルパークラスが実装されている.

application 層

application 層には Node 上で動作するアプリケーションに関するクラスが実装される. OnOff アプリケーションや Sink アプリケーションなど, サーバやクライアントの動作を模擬する物が多い.

internet 層

internet 層にはインターネット・プロトコル・スタックに関するクラスが実装される. 例えば, TCP [25], UDP [26], IP [27] [28], RIP [1], OSPF [2] [3], DSR [29], DSDV [25], OLSR [25] などが実装されている.

mobility 層

mobility 層には通信中に移動する Node の挙動を制御するためのクラスが実装される. 等速度移動, ランダムウォーク, ガウスマルコフモデルによる移動などが存在する.

network 層

network 層には基本的なネットワークの機構に関するクラスが実装される. ノード, Network Interface Card（NIC）, IP アドレス, キュー, ソケットなどの定義が含まれている.

core 層

core 層には離散事象シミュレータとしての基礎的な機構が実装される. core 層ではイベントスケジューラが動作しており, シミュレーション内で登録した順番に沿ってイベントが実行される仕組みである. 上位層において Simulate::Schedule() などの関数を呼び出す事によって, core 層のイベントスケジューラに対してシミュレーション内で実行するイベントを登録できる.

5.1.3 ns-3 におけるルーティングの実装

ns-3 では internet 層にルーティングのための機構が存在する. ns-3 においては大まかには図 5.1[30] のようにルーティングが実装されている.

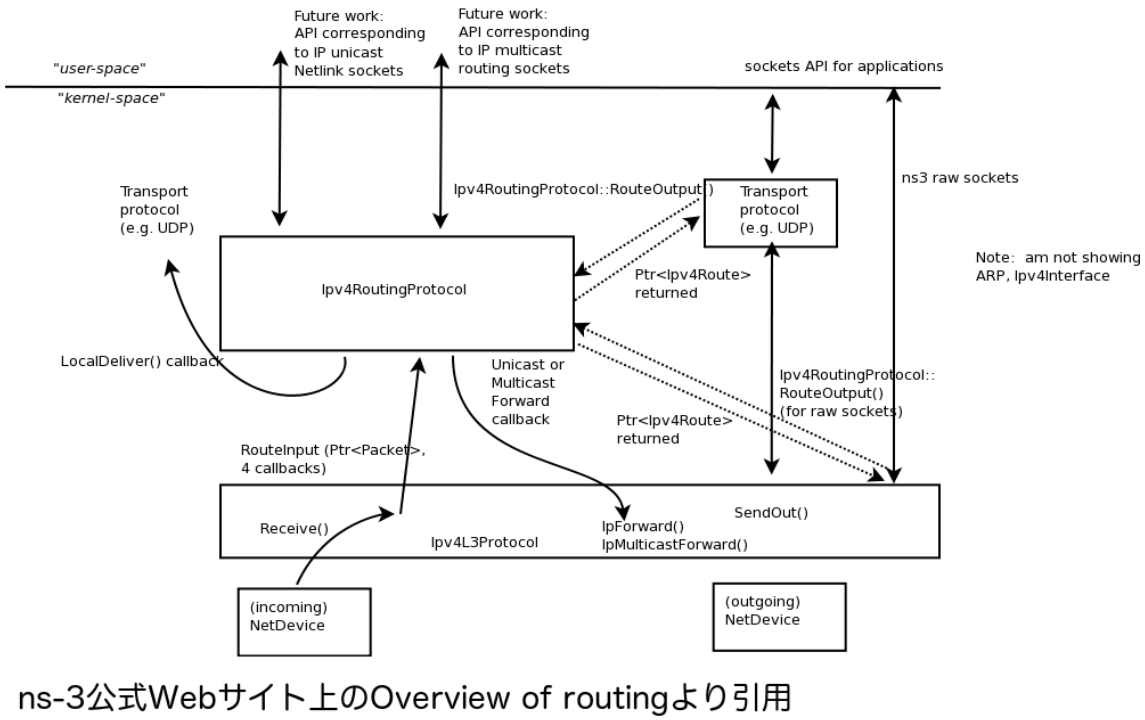

図 5.1: ns-3 におけるルーティング機構

NetDevice よりパケットが受信されると, IP パケットの転送を実行する Ipv4L3Protocol にパケットが渡る. 次に, Ipv4L3Protocol からルーティングプロトコルが実装されている Ipv4RoutingProtocol を継承して作成されたルーティングクラスにパケット転送処理用の 4 種類のコールバック（ユニキャスト転送の UnicastCallback, マルチキャスト転送の MulticastCallback, ローカルホスト向けに転送する LocalDeliverCallback, 転送失敗時の処理を行う ErrorCallback）と共にパケットを渡す. ルーティングクラス内部では送信先の IP アドレスを元にルーティングテーブルを検索し, IP アドレスの種類ごとに別々のコールバックを呼び出して, 検索結果のルーティングテーブルエントリと共にパケットを Ipv4L3Protocol に渡す. Ipv4L3Protocol は呼び出されたコールバックに応じて, ルーティングテーブルエントリに書かれている NetDevice へとパケットを渡す.

ルーティングデーモンがユーザーランドで経路計算を行った結果を用いて, カーネルラ

ンドに存在する Forward Information Base（FIB）エントリを書き換える事が普通である Linux 等と比較すると, ルーティングに関わる機構の全てがカーネルランドで動作しており, 構造に相違がある. しかし, ルーティングに必要な機構の全てをカーネルランドに収めることで, ルータの実装を少数のクラスに単純化することが可能になっている.

5.2 MP-PBR の実装モジュール

ns-3 ではプロトコル毎にモジュール化して実装する. この項では MP-PBR の実装モジュールに関して説明する. ns-3 におけるルーティング機構のうち, Ipv4RoutingProtocol を具体化して実装した. 全体像は下記の図 5.2 のようになっている.

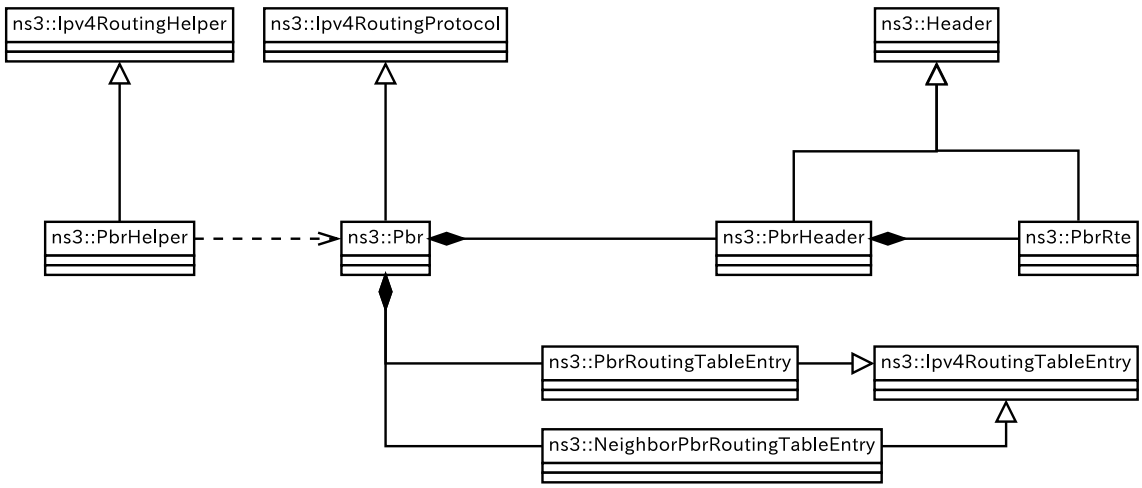

図 5.2: MP-PBR の実装におけるクラス構成

ルーティングテーブルを定義するクラスには自ルータが算出した送信先までのメトリックの値を保持するルーティングテーブルに関するクラスと, 隣接ルータが算出した送信先までのメトリックの値を保持するルーティングテーブルに関するクラスの 2 種類がある. この 2 つは前述のメトリック計算式における自ルータと隣接ルータの集合に対応するものである. この他に, ルーティングプロトコルで送受信するパケットを定義するクラスが存在する. 本体であるルーティングプロトコルのクラスでは, 前述のクラスの全てをオブジェクトコンポジションにより組み込んで内部で使用するようになっている. また, シミュレーションシナリオの記述を単純化するために, 別途ヘルパークラスを実装している.

5.2.1 経路情報パケットの定義

経路情報には, コマンド, プロトコルのバージョン, パディング領域, パケット作成時刻, ルーティングテーブルエントリのリストが含まれる.

表 5.2: 経路情報パケットの定義

Offset [bit]	0-7	8-15	16-23	24-31
0	Command	Version	Must Be Zero	
32	Transmitted Time			
64+	List of Routing Table Entries			

Command は隣接ルータに行うべき処理を知らせるコマンド領域である. Version はプロトコルのバージョンである. Must Be Zero は 0 でパディングする領域である. Transmitted Time はリンクの RTT を計算する際に用いるパケット送信時刻である. List of Routing Table Entries は隣接ルータに送信する経路表エントリのリストを格納する領域である.

経路情報パケットに含まれるルーティングテーブルエントリの定義は下記の通りである.

表 5.3: ルーティングテーブルエントリの定義

Offset [bit]	0-7	8-15	16-23	24-31
0	Address Family Identifier	Route Tag	Route Attribute	
32	Destination Address			
64	Destination Network Mask			
96	Metric			

Address Family Identifier はネットワーク層のアドレスの種別を示す識別子である. Route Tag は経路の種類を示すタグである. Route Attribute は経路の属性である. Destination Address は送信先アドレスである. Destination Network Mask はサブネットマスクである. Metric は送信先までのメトリックである.

経路情報パケットの実装におけるクラス構成は下記の通りである.

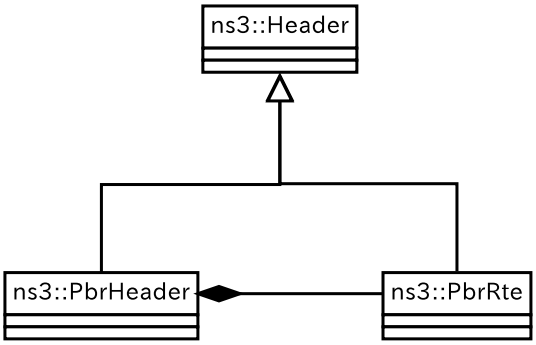

図 5.3: 経路情報パケットの実装におけるクラス構成

ns-3 に標準で実装されている ns3::Header を継承し, パケットを定義するクラスである ns3::PbrHeader と, 経路表エントリを定義するクラスである ns3::PbrRte を実装した. そして, ns3::PbrHeader 内に ns3::PbrRte のリストを組み込んでいる.

5.2.2 ルーティングテーブル

ルーティングテーブルには自ルータのメトリックを管理する PbrRoutingTableEntry の集合によるものと隣接ルータのメトリックを管理する NeighborPbrRoutingTableEntry の集合によるものが存在する.

表 5.4: PbrRoutingTableEntry の定義

Offset [bit]	0-7	8-15	16-23	24-31
0	Destination Address			
32	Destination Network Mask			
64	Gateway			
96	Output Interface			
128	Time Out Delay			
192	Garbage Collection Delay			
256	Address Family Identifier	Route Tag	Route Attribute	
288	Metric			
352	Metric at Last Transmitted			
416	Route Status			
448	Condition Flag for Route Updated			
480	Permission Flag for Update Cancel			
512+	List of NeighborPbrRoutingTableEntry			

Destination Address は送信先アドレスである. Destination Network Mask はサブネットマスクである. Gateway はネクストホップとなる隣接ルータのアドレスである. Output Interface はデータパケットを送出するインターフェースである. Time Out Delay は送信先アドレスに関する経路情報が到達しない状況において経路が無効になるまでの時間である. Garbage Collection Delay は経路が無効となった場合に経路表から削除されるまでの時間である. Address Family Identifier はネットワーク層のアドレスの種別を示す識別子である. Route Tag は経路の種類を示すタグである. Route Attribute は経路の属性で

ある．Metric は送信先までのメトリックである．Metric at Last Transmitted は経路表エントリが最後に送出された瞬間のメトリックである．Route Status は経路の状態である．Condition Flag for Route Updated は経路表エントリの更新状態を示すフラグである．Permission Flag for Update Cancel は経路表エントリの更新処理の中断に関する権限である．List of NeighborPbrRoutingTableEntry は隣接ルータのメトリックを管理する集合を表すリストである．

表 5.5: NeighborPbrRoutingTableEntry の定義

Offset [bit]	0-7	8-15	16-23	24-31
0	Address Family Identifier	Route Tag	Route Attribute	
32	Metric			
96	Route Status			
128	Condition Flag for Route Updated			

Address Family Identifier はネットワーク層のアドレスの種別を示す識別子である．Route Tag は経路の種類を示すタグである．Route Attribute は経路の属性である．Metric は送信先までのメトリックである．Route Status は経路の状態である．Condition Flag for Route Updated は経路表エントリの更新状態を示すフラグである．

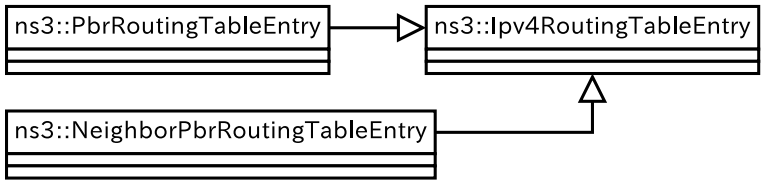

図 5.4: ルーティングテーブルの実装におけるクラス構成

ns-3 に標準で実装されている ns3::Ipv4RoutingTableEntry を継承し, 経路表エントリを定義するクラスである ns3::PbrRoutingTableEntry と, 隣接ルータのメトリックを管理する集合のエントリである ns3::NeighborPbrRoutingTableEntry を実装した．

5.2.3　ルーティングプロトコルの実装

PbrHeader::ROUTE_RESPONSE のコマンドが書き込まれた経路情報パケットを受け取った場合はパケット内部のルーティングテーブルエントリのリストを参照し, 隣接ルータに関するメトリック値を更新する. 自ルータではメトリック計算を定期的に実行して前回経路情報を送出した時のメトリック値から閾値以上メトリック値が変動した場合に, PbrHeader::ROUTE_RESPONSE のコマンドを書き込み経路情報を載せたパケットを隣接ルータに向けて送出する. データパケットを受け取った場合にはデータパケットの送信先アドレスを元にルーティングテーブルを検索し, 送信先に関する経路を発見した場合は送信先 IP アドレスの種別に応じてユニキャスト, マルチキャスト, ローカル転送に関するコールバックを呼び出す. 経路が発見できなかった場合はエラーコールバックを呼び出す. PbrHeader::ROUTE_REQUEST のコマンドが書き込まれた経路情報パケットを受け取った場合, PbrHeader::RTT_RESPONSE のコマンドを書き込んだ経路情報パケットを返す. PbrHeader::RTT_RESPONSE のコマンドが書き込まれた経路情報パケットを受け取った場合は経路情報パケットに書き込まれた作成時刻と受信時刻の差を取り, 該当リンクの RTT とする.

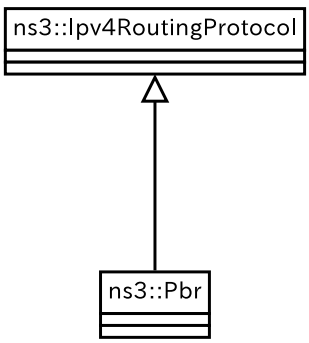

図 5.5: ルーティングプロトコルの実装におけるクラス構成

ns-3 に標準で実装されている ns3::Ipv4RoutingProtocol を継承して拡張する方法で ns3::Pbr を実装した.

5.2.4　シナリオ記述用のヘルパークラス

ルーティングプロトコルの実装を隠蔽し, シナリオ上で使用できるようにするクラスである.

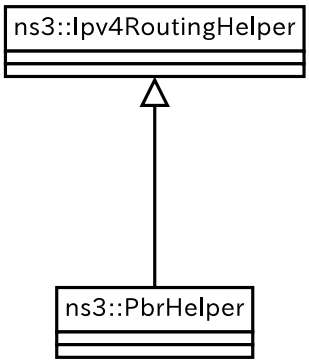

図 5.6: ヘルパークラスの実装におけるクラス構成

ns-3 に標準で実装されている ns3::Ipv4RoutingHelper を継承して拡張する方法で ns3::PbrHelper を実装した.

第6章　実験

この章ではMP-PBRを評価するための実験に関して述べる. 今回の実験は, 第4章で提案したように, マルチパスルーティングにおいて柔軟に経路形状の変更を行えるようにしたことで, 端末間のスループットが向上したかどうかを確認する. また, ルーティングアルゴリズムとしての基本的な諸特性であるプロトコルオーバーヘッド, 経路計算オーバーヘッド, メトリック計算の挙動に関しても確認する. ネットワークシミュレーションにおいて, インターネットトポロジを模したネットワークトポロジを設定して端末間でトラフィックを流す手法を採用する.

6.1　実験シナリオ

6.1.1　実験ネットワークトポロジ

実験ネットワークトポロジは実在のインターネットトポロジを参考に, AS の配置をノードに対応させ, AS 間の接続関係をリンクに対応させたネットワークトポロジを採用した. 参考にした実際のトポロジは, Sprintlink 社が構築・運用している Tier1 インターネットバックボーン内に存在する, 米国カリフォルニア州アナハイム区域の AS レベルのトポロジである, Rocketfuel 上の pop1239/AnaheimCA [31] を参考にした. このネットワークは, ネットワークへのデータ流入量とネットワークからのデータ流出量が均衡しているネットワークである. 実験ネットワークトポロジでは周縁部分に存在する AS を取り除き, 中心部分のみの近似となっている. 実験ネットワークトポロジは下記の図 6.1 の通りである.

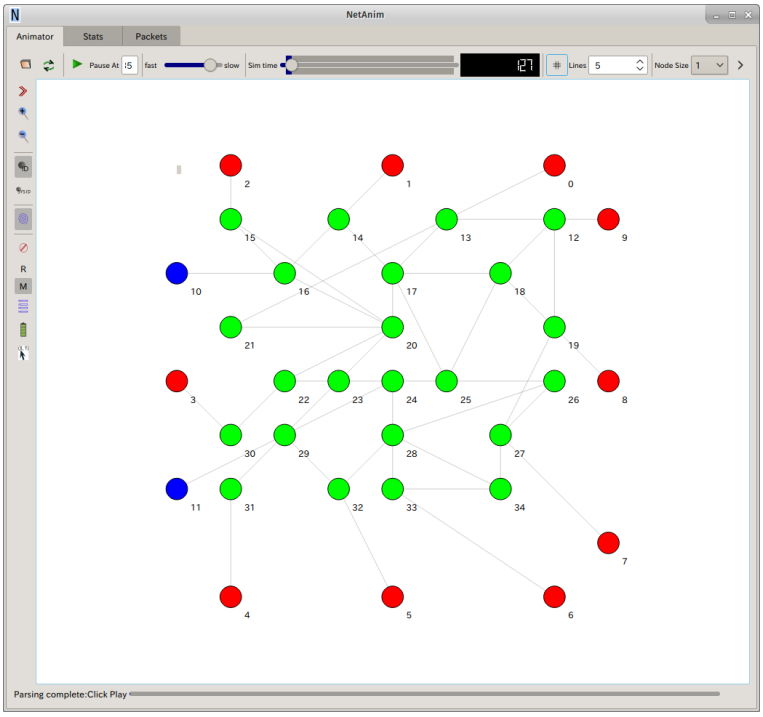

図 6.1: 実験ネットワークトポロジ

図 6.1 中には実験ネットワークトポロジが描かれている. ノード間に張られた線はリンクである. 各ノードに添えられた数字はノード番号である. 0, 1, 2 番の端末は送信端末であり, 4, 5, 6 番の端末は受信端末である. 今回の実験においては端末にもルータの機能を持たせている. それ以外のノードは全て純粋なルータである. 全てのノードにルータの機

能を持たせることで, シミュレータ内のノード番号がルータに対して 1 対 1 で対応するようにしている. 全てのリンクは 1[Mbps] の回線帯域を持っている. また, 全てのリンクの遅延は 1[ms] である.

6.1.2　データ送受信タイミング

端末のデータ送受信タイミングは下記の図 6.2 の通りである．一旦, 経路が安定な状態になるまで待つため, シミュレーション時間 $t = S$ の半分 $t = S/2$ が経過した段階から端末間のデータ転送を開始するように設定した．データは 1[Mbps] で転送される．

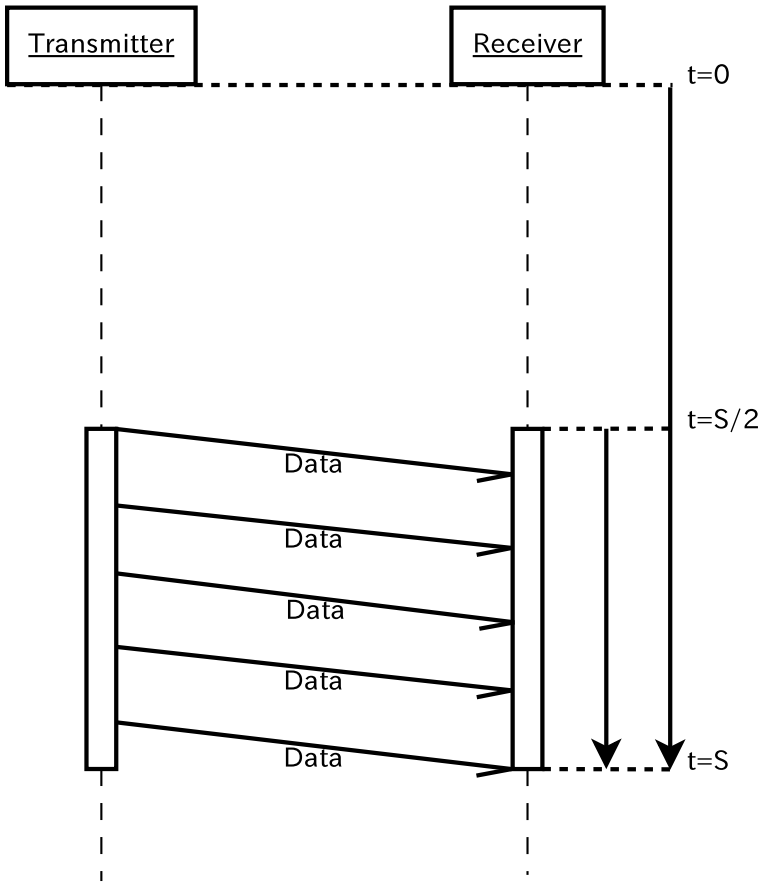

図 6.2: 端末間の送受信シーケンス

6.2 既存手法と提案手法の性能比較

既存手法として OSPF 上で ECMP を行う OSPF-ECMP を用意し, 提案手法である MP-PBR と性能比較を行った結果を示す.

6.2.1 端末間スループットの比較

ns-3 にはシミュレーション世界の状況を計測するための Tracing Subsystem と呼ばれる機構が存在する. Tracing Subsystem では, シミュレーション世界に存在する TracedValue をシミュレーション世界の外側に存在するシナリオ上で読み出してデータファイルに記録する. 今回はその Tracing Subsystem を経由して端末間スループットの計測を行った. 送信端末のデータ送信量はノードのカーネルランドにある ns3::Ipv4L3Protocol に存在する TracedValue である Tx を, 受信端末のデータ受信量はノードのカーネルランドにある ns3::Ipv4L3Protocol に存在する TracedValue である Rx を計測した.

合計比較

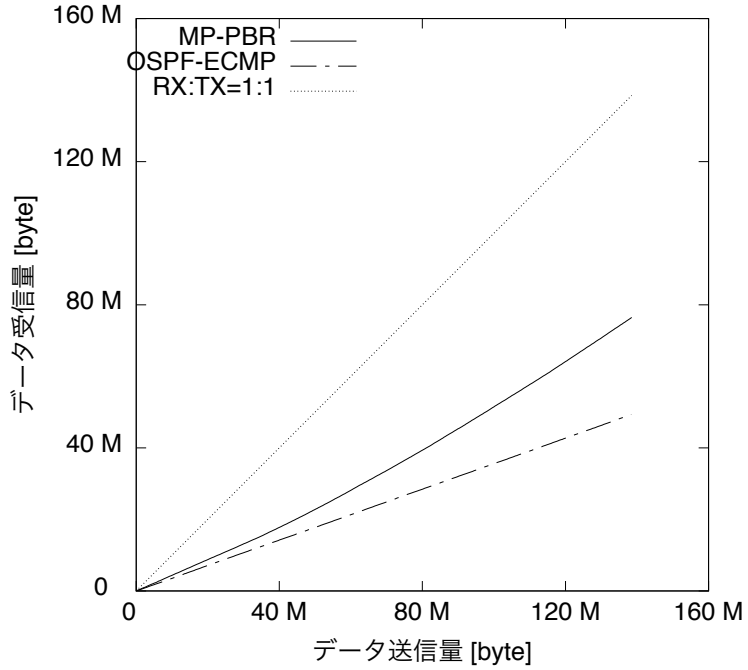

図 6.3: OSPF-ECMP と MP-PBR のデータ送受信量比率の比較

図 6.3 は OSPF-ECMP と MP-PBR のデータ送受信量比率の比較を行ったグラフである. グラフの縦軸はデータ受信量 [byte] であり, 横軸はデータ送信量 [byte] である. グラフ中では左下から右上に向かってシミュレーション時間が進んでいる. 各々の軸の表記に関しては, 数字の桁数を短縮するため Mega(グラフ中では M) 表記となっている. 凡例にはプロトコル名が表記されている. 但し, RX:TX=1:1 はデータ受信量の合計とデータ送信量の合計が一致した場合の理想的な直線である. RX:TX=1:1 の下に存在する直線状の線に関しては OSPF-ECMP であり, OSPF-ECMP におけるデータ受信量の合計とデータ送信量の合計の比率を示している. OSPF-ECMP におけるデータ受信量の合計に関しては, 経路上でパケットロスを起こしているためデータ送信量の合計を遥かに下回る値となっている. RX:TX=1:1 の下に存在する曲線状の線に関しては MP-PBR であり, MP-PBR におけるデータ受信量の合計とデータ送信量の合計の比率を示している. MP-PBR におけるデータ受信量の合計に関しては, パケットロスを起こしているためデータ送信量の合計を遥かに下回る値となっているが, トラフィックに対する経路の最適化が行われ, データ受信量の合計が増加して行っており, OSPF-ECMP のデータ受信量の合計と比較した場合でも

上回る結果となっている. 今回のシミュレーションではOSPF-ECMPとMP-PBR共に3つある送信端末からそれぞれ1[Mbps]ずつデータを送信しているが, 経路上に存在する1つ1つのリンクの容量が全て1[Mbps]となっているため経路上でパケットロスが発生し, シミュレーションの初期段階における受信端末では約3分の1程度しかデータを受信できていないという結果になった. しかし, MP-PBRにおいてはシミュレーションが進むに連れてメトリック値の最適化も進み, シミュレーション終了段階ではMP-PBRのデータ受信量の合計がOSPF-ECMPよりも多くなるという結果になった.

OSPF-ECMP における端末別のデータ転送量

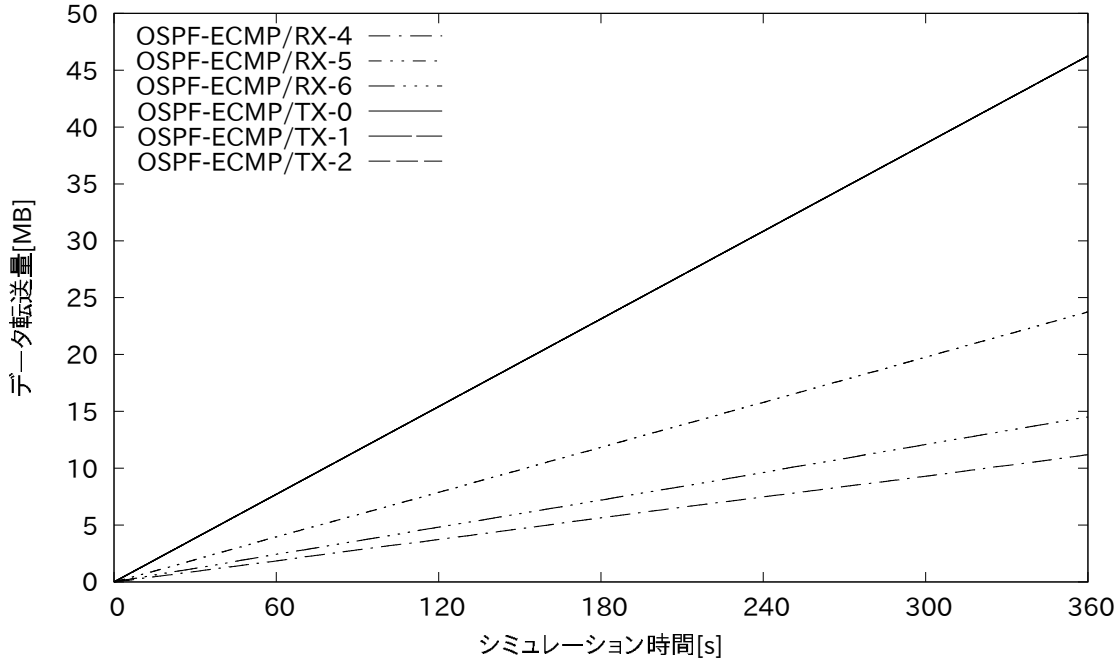

図 6.4: OSPF-ECMP における端末別のデータ転送量

図 6.4 は OSPF-ECMP における端末別のデータ転送量を示したグラフである. 凡例の表記法としては, { プロトコル名 }/{ 受信量であれば RX, 送信量であれば TX}-{ 計測を行ったノードの番号 } のフォーマットとする. 最も上に存在する比例直線は OSPF-ECMP/TX-0, OSPF-ECMP/TX-1, OSPF-ECMP/TX-2 の重なりである. この比例直線からは, 送信端末においてはパケットロスを起こしていないことが読み取れる. その下に存在する直線状の線は OSPF-ECMP/RX-4, OSPF-ECMP/RX-5, OSPF-ECMP/RX-6 であり, OSPF-ECMP を実行する受信端末におけるデータ受信量を示している. OSPF-ECMP におけるデータ受信量に関しては, 経路上でパケットロスを起こしているためデータ送信量を遥かに下回る値となっている. OSPF-ECMP/RX-5 が OSPF-ECMP/RX-4, OSPF-ECMP/RX-6 の倍近くの受信量を示している. 従って, 各端末間で見た場合に不平等なトラフィックの分配が行われてしまっている.

MP-PBR における端末別のデータ転送量

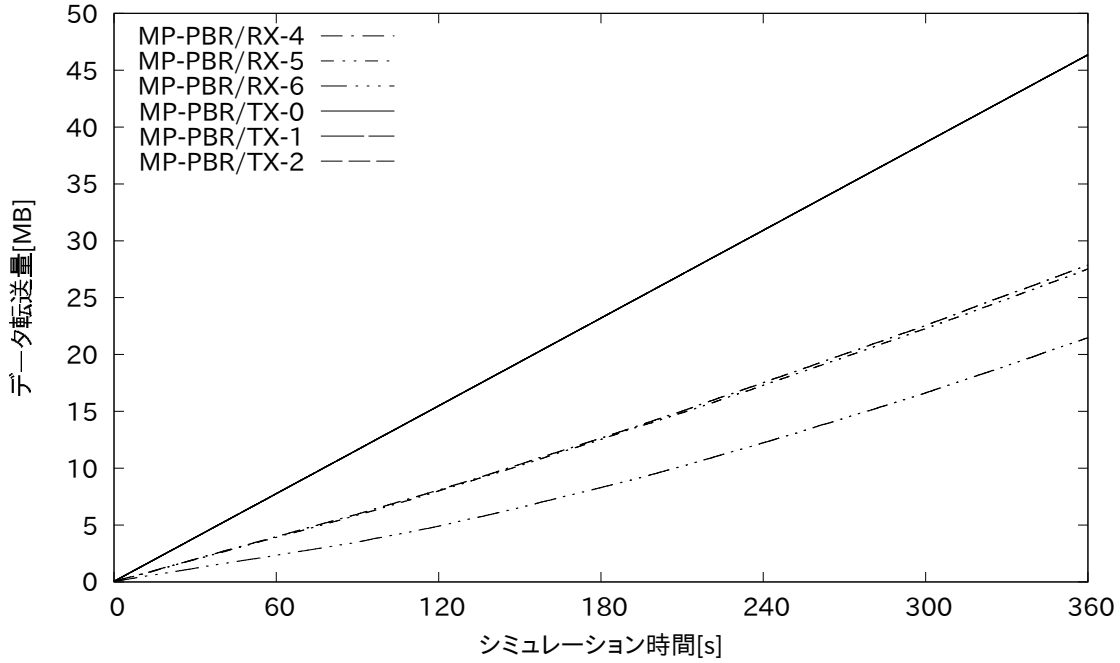

図 6.5: MP-PBR における端末別のデータ転送量

図 6.5 は MP-PBR における端末別のデータ転送量を示したグラフである．凡例の表記法としては, { プロトコル名 }/{ 受信量であれば RX, 送信量であれば TX}-{ 計測を行ったノードの番号 } のフォーマットとする．最も上に存在する比例直線は MP-PBR/TX-0, MP-PBR/TX-1, MP-PBR/TX-2 の重なりである．この比例直線からは, 送信端末においてはパケットロスを起こしていないことが読み取れる．その下に存在する曲線状の線は MP-PBR/RX-4, MP-PBR/RX-5, MP-PBR/RX-6 であり, MP-PBR を実行する受信端末におけるデータ受信量を示している．MP-PBR におけるデータ受信量に関しては, パケットロスを起こしているためデータ送信量を遥かに下回る値となっているが, トラフィックに対する経路の最適化が行われ, データ受信量が増加して行っている．MP-PBR/RX-4, MP-PBR/RX-5 がほぼ同じ受信量を示し, MP-PBR/RX-6 はそれらより少し低い受信量を示している．従って, 各端末間で見た場合に不平等さは残るものの, 図 6.4 と比較した場合には平等さの面で優れた結果を示していると言える．

6.2.2 プロトコルオーバーヘッドの比較

ネットワーク上の各ノードで送受信される経路情報パケットの転送量を計測し, 全ノードの転送量の合計値を比較する.

OSPF-ECMP

OSPF-ECMP において送受信される経路情報のサイズは理論的に計算可能である.

まず 1 つのルータから送出される経路情報のサイズを計算する. その計算式を式 (6.1) に示す. 今回の実験ネットワークトポロジではルータが直接的に相互接続されているため, 経路計算において Router-LSA を送受信するものとして仮定する. 各ルータ上で生成される Router-LSA の固定部分のサイズは 24[Bytes] である. そして, ルータに接続されたインターフェース 1 つに付き 12[Bytes] のリンクに関する情報が追加される. 作成された Router-LSA は複製され, 全てのインターフェースから送出される. 従って, 1 個のルータから送出される Router-LSA のサイズは下記の式で求めることが出来る. 下記の式 (6.1) において $|\mathrm{nbr}\,(n)\,|$ は, ノード n に隣接するノードの個数である. 今回の実験では, ノードとルータは 1 対 1 で対応するとしている. 本節における n はシミュレータ内のノード番号とし, $n \in I \subset \mathbb{N}$ とする. また, 計算結果の単位は [Bytes] である.

$$|\mathrm{nbr}\,(n)\,|\,(24 + 12|\mathrm{nbr}\,(n)\,|) \tag{6.1}$$

これをネットワークに参加する全てのノードに対して繰り返し実行し, 各計算結果の総和を取ったものがネットワーク全体に送信される経路情報のサイズとなる. 下記にその計算式である式 (6.2) を示す. 下記の式において N はネットワーク内に存在するノードの個数を示す. 下記の式 (6.2) においても計算結果の単位は [Bytes] である.

$$\sum_{n=0}^{N-1}\{|\mathrm{nbr}\,(n)\,|\,(24 + 12|\mathrm{nbr}\,(n)\,|)\} \tag{6.2}$$

OSPF-ECMP の場合は, 上記の理論的な計算によれば, 全ノードの経路情報を合計して 9972[Bytes] が一斉に送信される計算になる.

MP-PBR

MP-PBR の場合は, 経路更新時における全ノードの経路情報送信量を合計した結果, 6276[Bytes] が一斉に送信されていた.

6.2.3 経路計算オーバーヘッドの比較

OProfile により消費した CPU カウントを関数毎に計測した結果を示す．OProfile は統計的プロファイラであり, 100000 クロックサイクル毎にプログラムカウンタのサンプリングを行い, プログラムカウンタに明記された実行中の関数に関するサンプル数を加算している．ソースコード 6.1, ソースコード 6.2 共に左端に並んだ数字がプログラムカウンタから得られた各関数のサンプル数である．

OSPF-ECMP

表 6.1: OSPF-ECMP のプロファイル結果

サンプル数	実行した関数
44	ns3::GlobalRouteManagerImpl::SPFNext()
19	ns3::GlobalRouteManagerImpl::SPFIntraAddTransit()
18	ns3::GlobalRoutingLSA::GetLinkRecord()
9	ns3::GlobalRoutingLSA::GetNLinkRecords()
9	ns3::GlobalRoutingLinkRecord::GetLinkType()
8	ns3::GlobalRouteManagerImpl::SPFIntraAddRouter()
5	ns3::GlobalRouteManagerImpl::SPFCalculate()
5	ns3::GlobalRoutingLinkRecord::GetLinkData()
2	ns3::GlobalRouteManagerImpl::SPFNexthopCalculation()
2	ns3::GlobalRouter::GetRouterId()
1	_Rb_tree<>::_M_erase()
1	ns3::GlobalRouteManagerImpl::SPFProcessStubs()
1	ns3::GlobalRouter::ClearLSAs()
1	ns3::GlobalRouter::GetRoutingProtocol()
1	ns3::GlobalRouter::NetDeviceIsBridged(ns3::Ptr¡ns3::NetDevice¿)
1	ns3::GlobalRoutingLSA::GetLSType()
1	ns3::GlobalRoutingLSA::GetLinkStateId()
1	ns3::GlobalRoutingLSA::GetNetworkLSANetworkMask()
1	ns3::GlobalRoutingLSA::GetStatus()
1	ns3::Ipv4GlobalRouting::AddNetworkRouteTo()
1	ns3::Ipv4GlobalRouting::DoDispose()
1	ns3::Ipv4GlobalRouting::Ipv4GlobalRouting()
1	ns3::SPFVertex::SPFVertex()

上記表 6.1 内のサンプル数を合計すると 133 となる. リンクステート型であるため, トポロジの変更が起きない限り, 計算量がこれ以上に増えることはない.

MP-PBR

表 6.2: MP-PBR のプロファイル結果

サンプル数	実行した関数
181	ns3::Pbr::UnsolicitedCalculation()

上記表 6.2 内のサンプル数を合計すると 181 となる. ディスタンスベクタ型であるため, 継続的に経路計算を行う. 従って, これ以上に計算量が増加する可能性がある.

6.2.4　メトリック計算の挙動比較

MP-PBR に関しては, Node6 において, 6 ホップ離れた Node0 に向かう経路のメトリック値の変動を計測した. OSPF-ECMP に関しては, ns-3 上の OSPF-ECMP モジュールが経路情報をシミュレーション上のネットワークに送出せず, 理論的に経路計算を行うため, 実験ネットワークトポロジから推測する.

OSPF-ECMP

OSPF-ECMP の場合, リンクステート型であるため, 経路計算は 1 回のみで完了する. まず OSPF-ECMP の経路情報の量は 9972[Bytes] である. 今回の実験ネットワークトポロジにおける各リンクのトランジットタイムは全て同一であり 1[ms] である. 各ルータ内部での経路計算時間に関しては, ns-3 ではシミュレーション時間 0[s] で各ルータ内部の経路計算処理が終了するとしているため 0[s] とする. また, 各リンクのトランジットタイムは 1[ms] である. 更に, 実際のルータでは新たなリンクの存在を知らせる経路情報が僅かな時間の差で複数個入ってきた場合に CPU 使用率が大幅に上昇しないようにするために, SPF 計算に間隔を設けており, 多くの場合は管理者が標準設定の 5[s] に設定している. 従って, この実験ネットワークトポロジにおいては 5.006[s] で経路計算が完了するものと見られる.

MP-PBR

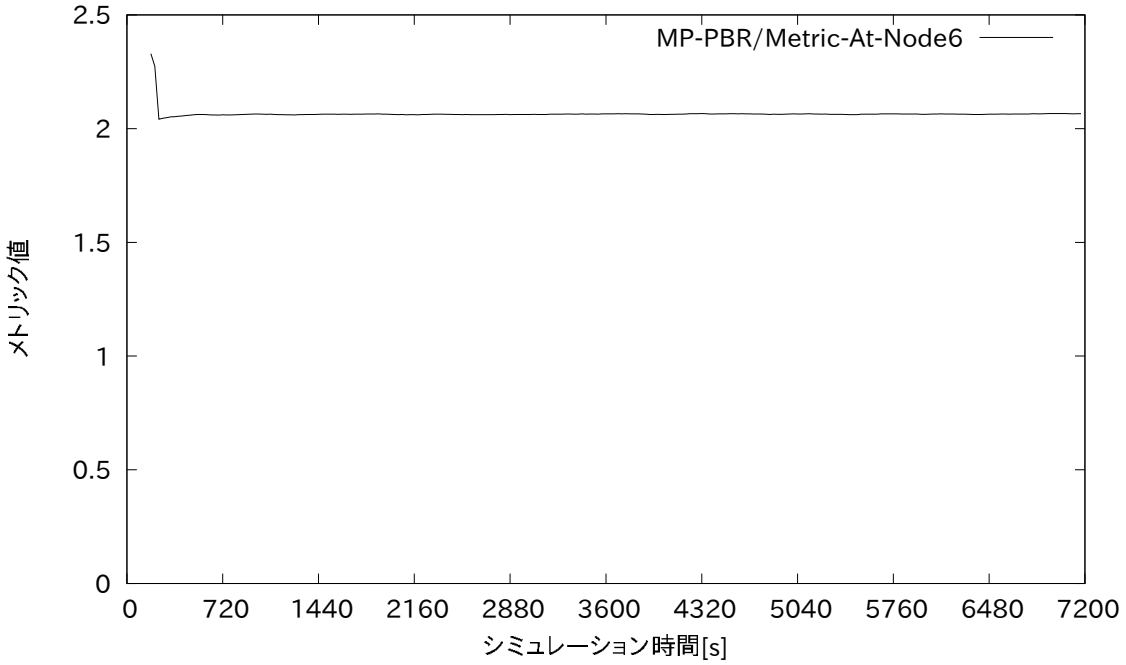

図 6.6: MP-PBR におけるメトリック値の変動の様子

図 6.6 の横軸はシミュレーション内の経過時間であり, 単位は [s] である. 図 6.6 の縦軸はメトリック値であり, メトリック値は無次元量である. 今回の実験では経路の更新間隔は 30[s] とした. 図 6.6 を見ると, シミュレーション開始時の 0[s] からシミュレーション終了時の 7200[s] までが示されている. Node0 に関する経路情報が初めて Node6 に到達した 180[s] からメトリック値の変動に関する記録が開始されており, メトリック値の記録を開始した直後の部分以外では, メトリック値の変動がほぼ平坦になる様子が示されている. 次に, メトリック値の変動が激しい部分を拡大したグラフを示す.

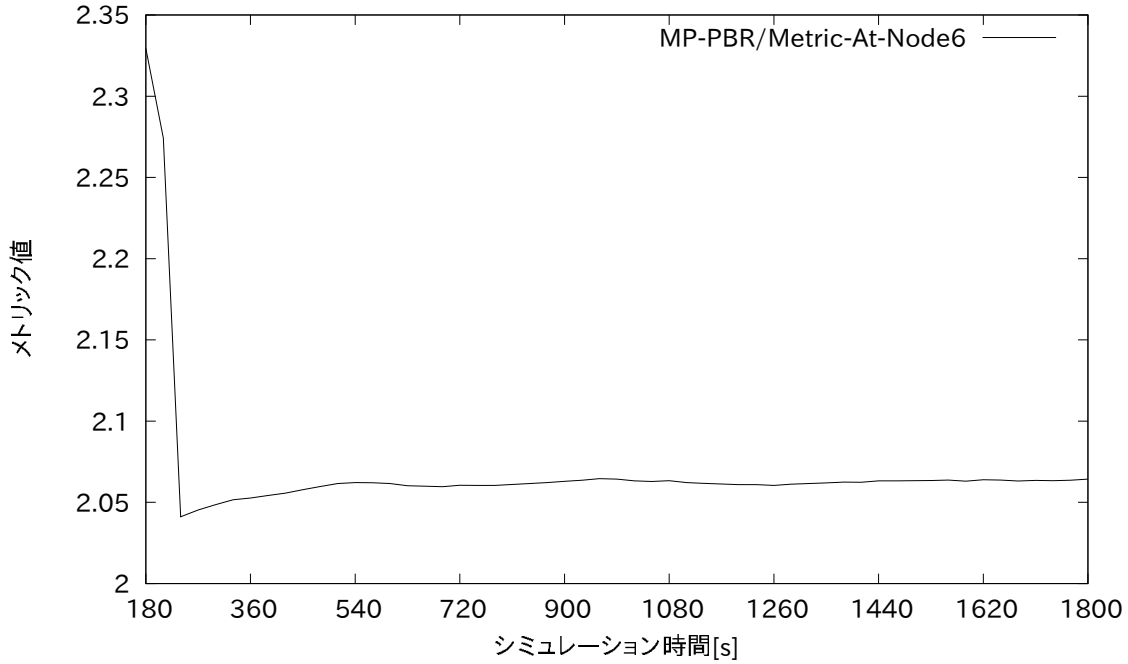

図 6.7: MP-PBR におけるメトリック値の変動が激しい部分の拡大

図 6.7 の横軸はシミュレーション内の経過時間であり, 単位は [s] である. 図 6.7 の縦軸はメトリック値であり, メトリック値は無次元量である. 30[s] 間隔で経路計算を行っている. メトリック値の変動を見ると, Node0 に関する経路情報が初めて Node6 に到達した 180[s] から, メトリック値の変動が平坦になる 1800[s] までが示されている. まず, シミュレーション開始から 240[s] までメトリック値がオーバーシュートするような挙動が見られている. 次に, 240[s] から約 540[s] まででメトリック値が殆ど横ばいになって行く傾向が見られる. 滑らかにメトリック値が移り変わる様子が見られるが, これは拡散方程式が現在の値を用いて次の時間ステップの値を求めるため, IIR フィルタに近い特性を持つためである.

第7章　考察

この章では実験結果を分析し, MP-PBR の適用可能性とパケットリオーダリング問題への対処方法に関して述べる.

7.1　MP-PBR の適用可能性に関する考察

実験結果を見る限りでは, スループットに関しては OSPF-ECMP のパケット受信量を MP-PBR のパケット受信量が上回っており, OSPF-ECMP よりも MP-PBR の方が優位に立っている. しかし, その他の指標であるプロトコルオーバーヘッド, 経路計算オーバーヘッド, メトリック計算の挙動に関しては何れも OSPF-ECMP よりも MP-PBR の方が劣る結果となった. また, 今回提案した, メトリック面の負の勾配の比率で重み付けした乱数によるデータパケットの分散転送は内部に隣接ルータを順に検査するループを含む. 今回の実験では, 1 つ 1 つのデータパケットに対してこの処理を行っているため, 処理負荷が非常に高くなってしまう欠点が生じてしまう. そのため, 実際には経路キャッシュを設けて, 複数個のパケットを纏めて同じ隣接ルータに転送する計算量削減の工夫が必要になる. しかし, このようにした場合, 各ルータにおけるパケットの転送方向に偏りが生じやすくなるデメリットが生じる.

今回提案した MP-PBR は通信負荷の分散のみに焦点を置いて開発されている. 従って, パケットリオーダリングがトランスポート層に制御撹乱を引き起こす問題を別とすれば, データセンタのような膨大なデータが転送されるネットワークに適しているものと思われる. 逆に, メトリック面の負の勾配の比率で重み付けした乱数によるデータパケットの分散転送を行うとパケットリオーダリングが生じやすくなるため, リアルタイムな音声・画像通信, リアルタイムなアクチュエータの遠隔操作などが主たる利用用途であるネットワークには基本的に不向きであると言える.

メトリック面の負の勾配の比率で重み付けした乱数によるデータパケットの分散転送

はパケットリオーダが生じやすい欠点がある. しかし, マクロに見ればメトリック面がすり鉢状に形成されるため, 勾配は常に送信先へ近付く方向に下るように生じる傾向にある. 従って, 今回のようなデータパケットの分散転送を行ってもデータパケットが非常に大きく迂回してから送信先に届くことはない（例えば, メトリック計算のパラメータの設定にもよるが, 最適経路と比較してホップ数が非常に大きく増えるような経路は選択されない）. また, TCP においても多少のパケットリオーダであれば Selective ACKnowledgment（SACK） [32] で対処可能である. しかし, SACK は本来的にはパケットリオーダ向けに開発された物ではないため, パケットリオーダによりアプリケーションのパフォーマンスに悪影響が生じる可能性は依然として残ると言える. 次の項ではその対策に関して議論を行うことにする.

7.2 パケットリオーダリング問題に関する考察

今回の MP-PBR では各ルータでランダムにパケットを分散させるため, 原理的にパケットリオーダリングが避けられない. 従って, パケットリオーダリング問題への対策を考える必要がある. トランスポート層に TCP を用いた場合, パケットリオーダリングが生じても受信端末側のバッファリングと SACK によってある程度吸収可能ではある. しかし, それでも吸収できなかった場合には, TCP がウィンドウサイズを下げることでアプリケーション側のスループットが下がる問題を生じるため, その問題への対処方法に関して述べる.

7.2.1 アプリケーション層における対処

デッドラインミスに対する扱いで情報処理システムを区分すると, デッドラインミスが起きるとシステムが破綻するハードリアルタイムシステム, デッドラインミスが起きたタスクの価値が即座に 0 になるファームリアルタイムシステム, デッドラインミスが起きたタスクの価値が時間とともに減少するソフトリアルタイムシステム, タスクに対するデッドラインそのものが存在しない非リアルタイムシステムが存在する. 今回提案したルーティングプロトコルではパケットリオーダリングが原理的に避けられないため, このルーティングプロトコル上に実装可能なシステムはファームリアルタイムシステム, ソフトリアルタイムシステム, 非リアルタイムシステムのみである. 上記のファームリアルタイム

システムに該当するアプリケーションは画像・音声のリアルタイム処理である. ソフトリアルタイムに該当するアプリケーションは端末の遠隔操作などである. 非リアルタイムシステムに該当するアプリケーションはメールやチャットなどのテキストデータの転送や, ファイルの転送などである.

ハードリアルタイムシステムはデータの欠落が起きた時点でシステムが破綻し, ファームリアルタイムシステムはリオーダリングが起きた時点でタスクの価値が 0 になるため, パケットが所定の時間内に届かなかった場合にはパケットロスと判断して, 欠落したデータを各アプリケーションに適した方法で推測して補完し処理を続行する他にない. 非リアルタイムシステムの場合は, パケットの遅延が常軌を逸して大きい（1000ms 以上等）場合にはパケットロスと判断し, 再送制御を行うことで対処する. ソフトリアルタイムシステムにおいては, 暫く受信者を待機させる事を許容しているため, 閾値以上遅延が大きくなるまでは待機し, それ以上に遅延が大きくなればパケットロスと判断して, 欠落したデータを推測して補完し処理を継続する.

第8章　おわりに

この章では本研究内容の総括と, 本研究内容の発展可能性について述べる.

8.1　まとめ

本研究では負荷に応じた柔軟な経路再選択を行うマルチパスルーティングを可能にするために, 拡散方程式に基づき, 負荷に適応してメトリック計算を行う Potential Based Routing（PBR）をマルチパスルーティングに拡張し, 負荷適応型のマルチパスルーティングである Multipath Potential Based Routing（MP-PBR）を提案した. そして, ns-3 上に MP-PBR を実装した後, インターネットトポロジの一部を模したネットワークを ns-3 のシミュレーションシナリオ上に構築し, OSPF-ECMP と MP-PBR の間でスループット, プロトコルオーバーヘッド, 経路計算オーバーヘッド, メトリック計算の挙動に関する比較検討を行った. 結果として端末間スループットのみにおいて OSPF-ECMP よりも MP-PBR に優位性があることが分かった. 従って, MP-PBR は端末間スループットが必要なネットワークに向いている事が分かった.

8.2　今後の課題

MP-PBR は依然としてパケットリオーダリングへの対策が不充分である問題や, メトリック面の負の勾配の比率で重み付けした乱数によるデータパケットの分散に関する計算量の問題を抱えている. パケットリオーダリングに関する問題に対しては, TCP 上で動くアプリケーションの大部分に関しては SACK で対処可能としたが, パケットリオーダリングの程度によっては対処不能となることも十分に考えられる. 従って, TCP の SACK の適用範囲を拡張する方法を考案する必要があるものと思われる. また, UDP 上で動作し, 通信のリアルタイム性を要求するアプリケーションの通信に関しては, そもそもパケットリ

オーダリングを起こさないようにフローベースで経路制御を行う必要がある. しかし, フローベースの経路制御の場合, 単一の端末間通信における負荷分散は不可能になる. 従って, ルータ側で通信の特性を監視してパケットベース, フローベースを切り替える方法を考案する必要がある. メトリック面の負の勾配の比率で重み付けした乱数によるデータパケットの分散に関する計算量の問題に関しては, 複数のパケットをまとめて同じ隣接ルータに転送する経路キャッシュを導入し, 計算間隔を間引くことで解決可能であるが, その具体的な検討を行う必要がある.

謝辞

本研究を行うにあたり, 多数の方からご助言やご助力を頂きました.

本研究を進めるにあたり, 主指導教員である篠田陽一教授には研究室における研究活動に関する全面的なご指導を賜りました. 心から深く感謝致します. また, 主テーマ審査員である丹康雄教授は審査時に研究内容に関して各種指摘をして頂き誠に有難うございました. 副テーマ指導教員である赤木正人教授も副テーマ研究の内容に関して丁寧に助言して頂き誠に有難うございました.

本研究室の知念賢一特任准教授, 宇多仁助教には, 既存研究の調査, 理論構築, 研究で用いるツール群, 実験で収集した計測値の解釈, ネットワーク構築・運用等に関して全面的なご指導やご助力を賜りました. 心から感謝致します.

情報通信研究機構（NICT）北陸 StarBED 技術センターの高野祐輝氏, 安田真悟氏, 三浦良介氏, 井上朋哉氏, 明石邦夫氏には既存研究の調査, 理論構築, 研究で用いるツール群, 実験で収集した計測値の解釈, ネットワーク構築・運用等に関して多大なご指導, ご助力を頂きました. 心から感謝致します.

本研究室の博士後期課程の太田悟史氏, 阿部博氏には, 実務経験を持つ立場からゼミにおける議論, 研究で用いるツール群, ネットワーク構築・運用に関して多大なご指導, ご助力を頂きました. 心から感謝致します.

本研究室修了生の大野夏希氏, 岩橋紘司氏, 加藤邦章氏, 園田真人氏, 八木辰弥氏, 岩本裕真氏, 廣瀬真人氏にはプロポーザル作成, 修士論文作成, ゼミにおける議論, プレゼンテーションにおけるログ取り, 技術的課題解決, 研究生活を送る上で必要な各種雑務に関して様々なご指導, ご助力を頂きました. 心から感謝致します.

本研究室の博士前期課程の押川侑樹氏, 橋本光世氏, 村上正樹氏, 淺葉祥吾氏, 広瀬太志氏, 三島航氏, 宮崎駿氏には修士論文作成, ゼミにおける議論, プレゼンテーションにおけるログ取り, 技術的課題解決, 研究生活を送る上で必要な各種雑務に関して様々なご助力を頂きました. 心から感謝致します.

最後に, 学業を支えてくれた家族へ心から感謝致します.

参考文献

[1] G. Malkin. RIP Version 2, RFC2453. November 1998.

[2] J. Moy. OSPF Version 2, RFC2328. April 1998.

[3] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. OSPF for IPv6, RFC5340. July 2008.

[4] Y. Rekhter, Ed., T. Li, Ed., S. Hares, and Ed. A Border Gateway Protocol 4 (BGP-4), RFC4271. January 2006.

[5] Curtis Villamizar. OSPF Optimized Multipath (OSPF-OMP), INTERNET-DRAFT. February 1999.

[6] 本間 裕大, 会田 雅樹, 下西 英之, 岩田 淳. 非集計ロジットモデルを用いたマルチパスルーティング制御技術の提案. 電子情報通信学会技術研究報告. NS, ネットワークシステム, November 2009.

[7] S Vutukury, J.J Garcia-Luna-Aceves. MPATH: a loop-free multipath routing algorithm. *Microprocessors and Microsystems*, Vol. 24, No. 6, pp. 319–327, October 2000.

[8] S Vutukury, J.J Garcia-Luna-Aceves. MDVA: a distance-vector multipath routing protocol. *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, April 2001.

[9] Igor Ganichev, Bin Dai, P. Brighten Godfrey, Scott Shenker. YAMR: Yet Another Multipath Routing Protocol. *ACM SIGCOMM Computer Communication Review*, Vol. 40, No. 5, pp. 14–19, October 2010.

[10] Y. Ohara, S. Imahori, R. Van Meter. MARA: Maximum Alternative Routing Algorithm. *IEEE INFOCOM 2009 proceedings*, April 2009.

[11] Mahesh K. Marina, Samir R. Das. Ad hoc on-demand multipath distance vector routing. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 6, No. 3, pp. 92–93, July 2002.

[12] Jiazi Yi, Asmaa Adnane, Sylvain David, Benoit Parrein. Multipath optimized link state routing for mobile ad hoc networks. *Ad Hoc Networks*, Vol. 9, No. 1, pp. 28–47, January 2011.

[13] ns-3 project. ns-3. https://www.nsnam.org/.

[14] F. E. HEART, R. E. KAHN, S. M. ORNSTEIN, W. R. CROWTHER, D. C. WALDEN. The interface message processor for the ARPA computer network. *AFIPS '70 (Spring) Proceedings, spring joint computer conference*, pp. 551–567, May 1970.

[15] J. McQuillan, I. Richer, E. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, Vol. 28, No. 5, pp. 711–719, May 1980.

[16] A. Khanna, J. Zinky. The revised ARPANET routing metric. *SIGCOMM '89 Symposium proceedings on Communications architectures & protocols*, September 1989.

[17] B. Boehm, R. Mobley. Adaptive Routing Techniques for Distributed Communications Systems. *Rand Corporation Memorandum RM-4781-PR*, 1966.

[18] Anindya Basu, Alvin Lin, Sharad Ramanathan. Routing using potentials: a dynamic traffic-aware routing algorithm. *SIGCOMM '03 Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 37–48, August 2003.

[19] 落合 秀也, 江崎 浩. DTN 環境を想定したトポロジ変化に強いメッセージルーティング. 情報処理学会論文誌, Vol. 50, No. 9, pp. 2312–2326, September 2009.

[20] 小南 大智, 加嶋 健司, 橋本 智昭, 村田 正幸. 管理型自己組織化制御に基づくネットワークを目指したモデル予測制御を用いたポテンシャルルーティングの提案. 電子情報通信学会技術研究報告, Vol. 113, No. 473, pp. 205–210, March 2014.

[21] 豊永 慎也, 小南 大智, 菅野 正嗣, 村田 正幸, 畠内 孝明. 無線センサネットワークにおけるポテンシャルルーティングに基づく下り方向通信実現手法の提案と評価. 研究報告ユビキタスコンピューティングシステム, Vol. 2012-UBI-34, No. 12, pp. 1–2, May 2012.

[22] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing, RFC3561. July 2003.

[23] T. Clausen, Ed., P. Jacquet, and Ed. Optimized Link State Routing Protocol (OLSR), RFC3626. October 2003.

[24] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. The Optimized Link State Routing Protocol Version 2, RFC7181. April 2014.

[25] J. Postel. Transmission Control Protocol, RFC793. September 1981.

[26] J. Postel. User Datagram Protocol, RFC768. August 1980.

[27] J. Postel. Internet Protocol, RFC791. September 1981.

[28] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification, RFC1883. December 1995.

[29] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4, RFC4728. February 2007.

[30] ns-3 project. Routing overview — Model Library - ns-3. https://www.nsnam.org/docs/models/html/routing-overview.html.

[31] Department of Computer Science & Engineering University of Washington. pop1239/AnaheimCA, Rocketfuel: An ISP Topology Mapping Engine. http://research.cs.washington.edu/networking/rocketfuel/interactive/pop1239/AnaheimCA.b.gif.

[32] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options, RFC2018. October 1996.