修士論文

題目

資源予約ネットワークにおける経路制御に関する研究

指導教員

所 真理雄 客員教授

天野 英晴 助教授

慶應義塾大学大学院

理工学研究科 計算機科学専攻

1998 年 2 月

89624591

三輪 泰孝

修 士 論 文 要 旨

専 攻	計算機科学	学 籍 番 号	89624591	フリガナ 氏 名	ミワ ヤスタカ 三輪 泰孝

(論 文 題 名)

資源予約ネットワークにおける経路制御に関する研究

(内 容 の 要 旨)

ATM などの QoS の保証が可能なデータリンクの出現、映像や音声などの連続メディア通信あるいは超高精細画像などの大容量データ通信に対する需要の高まりを背景に、インターネット上においても個々の通信に対して QoS を提供する保証型 (guaranteed) サービスの実現が求められるようになった。

保証型サービスを実現するためには、フローの要求する様々な QoS やネットワークの特性、利用可能な資源状態を考慮した上で、経路制御を行う必要がある。本研究では、保証型サービスを提供するための資源予約ネットワークアーキテクチャを定義し、経路制御プロトコルとして **RRR (Routing for Resource Reservation)** を提案する。RRR は以下のような特徴を持つ。

- 資源予約プロトコルに対してフロー毎の経路を提供する。
- リンクステート型の経路制御情報を用いて、TOS (Type Of Service) メトリックによる経路選択を行う。
- 経路の計算は始点ノードで行い、経路上のノードに対して経路情報を配布する。
- 利用可能なネットワーク資源状態の変化に応じてネットワーク資源情報を交換し、経路選択時に考慮する。
- 代替経路を提供する。
- IP マルチキャスト経路を提供する。

RRR は、経路制御テーブルに対して経路を設定するのではなく、資源予約プロトコルに対して経路の提供を行う。経路制御テーブルは資源予約プロトコルによって管理される。RRR を経路制御テーブルから分離させることで、RRR は柔軟な経路選択を行うことができる。また資源予約プロトコルは、自らが経路の設定を行うことで、資源予約を行った経路上に確実にフローを転送することができる。また RRR は、資源予約プロトコルや他の経路制御プロトコルとの協調を考慮しており、より少ないオーバヘッドでの導入が可能である。

資源予約ネットワークにおいて、資源予約プロトコルと経路制御プロトコルは密接な関係にあり、本来これらは一体となって考えられるべきである。本研究は、その一つの指針を示すものである。

（内容の要旨は約25行程度で記入のこと）

Summary of M. S. Thesis

Major	Student No.	SURNAME, Firstname
Computer Science	89624591	MIWA, Yasutaka

Title

A Study on Routing for Resource Reservation Network

In recent years, connection oriented data links, such as ATM, have come into wide use because they are able to support QoS to each connection. Besides them, continuous media (ex. real-time audio and video) and large bulk data (ex. super high definition image) transfer are also in great demand. Then, **Guaranteed Service** with resource reservation on the Internet comes to be essential.

That, however, requires a routing scheme in which both QoS paramters of the flow and the resource available on the network are considered. In this study, we define the **Resource Reservatoion Network Architecture** for Guaranteed Service, and propose to make a new routing protocol, **RRR (Routing for Resource Reservation)**. The features it has are:

- providing per-flow-route to the resource reservation protocol.
- using link-state routing information so as to calculate the route by TOS (Type Of Service) metric.
- calculating the route on the source node and sending the route information to the down stream nodes along the paths.
- exchanging the available information of the network resources in order to select the best route.
- providing the alternative routes in case the first route is blocked by lack of the resources.
- supporting the route for IP multicast.

RRR doesn't create the route information directly on the routing table, but provides it for the resource reservation protocol. And the resource reservation protocol manages the routing table. RRR is separated from the routing table, so that it is able to select the routes quite flexibly, and the resource reservation protocol is able to fix the routes of the flows on which it reserves the resources. RRR also cooperates with other routing protocols, so that the overhead is small.

This study shows the routing protocol and the resource reservation protocol, which are closely related each other and shuold be considered as a whole.

Graduate School of Science and Technology, KEIO UNIVERSITY

(The summary of thesis should be written within 35 lines)

目 次

1 序論 **1**

1.1 はじめに 1

1.2 目的 2

1.3 本論文の構成 3

2 資源予約ネットワークと経路制御 **4**

2.1 資源予約ネットワーク 4

2.1.1 フロー，保証型サービス 4

2.1.2 参照モデル 5

2.1.3 資源予約ネットワークアーキテクチャ 7

2.2 経路制御プロトコル 10

2.2.1 インターネットにおける経路制御プロトコルの問題 10

2.2.2 資源予約ネットワークにおける経路制御プロトコルの目標と論点 11

2.2.3 提案されているプロトコル 16

2.3 経路制御テーブル 20

2.3.1 IP の経路制御テーブル 20

2.3.2 データリンク層による転送 20

2.3.3 その他 21

3 経路制御プロトコル RRR の設計 **22**

3.1 論点 22

3.1.1 必要な機能 22

3.1.2 問題点とアプローチ 22

3.2 設計 25

3.2.1 概要 25

3.2.2 経路制御情報 26

3.2.3 経路の提供 30

3.2.4 経路の消去 33

3.2.5 ネットワーク資源情報の把握と代替経路の提供 34

3.2.6 資源予約プロトコルとのインタフェース 36

3.2.7 マルチキャストについて 37

4 経路制御デーモン RRRD の実装 **40**

4.1 概要 40

4.2 RRR モジュール 42

4.2.1 起動時の動作 . . . 42
4.2.2 イベントタイマ , RRR メッセージ . . . 44
4.2.3 隣接 RRRD の監視 . . . 45
4.2.4 OSPF トポロジデータベースとの同期 . . . 46
4.2.5 経路の生成 . . . 46
4.2.6 経路の消去 . . . 52
4.2.7 マルチキャストグループ情報 . . . 53
4.2.8 アラート情報 . . . 56
4.2.9 代替経路の要求 . . . 56
4.3 OSPF モジュール . . . 57
4.4 IGMP モジュール . . . 58
4.5 RSRR モジュール . . . 59
4.6 実装上の制限 . . . 61

5 評価と検討 62
5.1 RRR の評価 . . . 62
5.1.1 経路計算 . . . 63
5.1.2 経路情報配布 . . . 65
5.1.3 経路提供 . . . 66
5.2 RRRD の評価 . . . 67
5.2.1 測定環境 . . . 67
5.2.2 測定内容 . . . 67
5.2.3 測定結果 . . . 68
5.3 今後の課題 . . . 71
5.3.1 RRR における未解決の問題及び今後の課題 . . . 71
5.3.2 資源予約ネットワークにおける経路制御についての課題 . . . 72

6 結論 74

謝辞 76

A 予備シミュレーション 77
A.1 目的 . . . 77
A.2 設定 . . . 77
A.2.1 シミュレーション条件 . . . 77
A.2.2 アルゴリズム . . . 79
A.2.3 評価項目 . . . 80
A.2.4 評価条件 . . . 80
A.3 結果 . . . 81
A.4 検討 . . . 81

B RRR パケットフォーマット 85

C RSRR 拡張メッセージフォーマット 91

参 考 文 献 95

図 目 次

2.1 統合型サービスのための実装参照モデル 5
2.2 資源予約ネットワークアーキテクチャ 8
2.3 保証型サービス提供の流れ 9
2.4 混雑の拡散 16
2.5 QOSPF におけるネットワーク資源情報 17
2.6 Explicit Routing 19

3.1 サンプルネットワーク 27
3.2 トポロジグラフによる表現 28
3.3 Type Of Service オクテット 28
3.4 OSPF エリア 30
3.5 RRR リンクステートデータベース 31
3.6 経路の提供 32
3.7 経路の消去 33
3.8 アラート発生と代替経路の提供 35
3.9 マルチキャスト経路の提供 38

4.1 資源予約ネットワークの経路制御機構の実装アーキテクチャ 41
4.2 RRRD の機能モジュール 41
4.3 RRRD 設定ファイルの例 43
4.4 隣接 RRRD の状態遷移図 46
4.5 OSPF トポロジデータベースとの同期 47
4.6 疑似ノード 48
4.7 RRR 経路エントリの状態遷移図 50
4.8 マルチキャスト経路における確認応答メッセージ 51
4.9 R5 における R1 から R5 への経路提供の例 52
4.10 R1 における R1 から M1(R3,R4) への経路提供の例 52
4.11 R5 における R1 から R5 への経路消去の例 54
4.12 R1 における R1 から M1(R3,R4) への経路消去の例 54
4.13 アラート発生及び代替経路の提供の例 57
4.14 Querier の選択に関する状態遷移図 58
4.15 マルチキャストグループ状態遷移図 59

5.1 測定ネットワーク 68
5.2 RRR 経路エントリの検索時間 69
5.3 ユニキャスト経路 (始点ノード主導) 69

5.4 ユニキャスト経路 (終点ノード主導) 70
5.5 マルチキャスト経路 (始点ノード主導) 70

A.1 ネットワークトポロジ 79
A.2 ネットワーク規模の変化 81
A.3 平均資源予約間隔の変化に対するネットワークスループット 82
A.4 平均資源予約間隔の変化に対するメッセージコスト 82
A.5 ネットワーク規模の変化に対するネットワークスループット 83
A.6 ネットワーク規模の変化に対するメッセージコスト 83

表 目 次

3.1 OSPF における TOS コード 27

第 1 章

序論

1.1 はじめに

今日のコンピュータとインターネットの普及は、我々の生活に大きな変化をもたらした。もはやコンピュータは、個人が情報端末として所有する時代となり、インターネットはコンピュータ通信網の基盤として、誰もが自由に利用できるネットワークに発展した。そこでは、電子メールやネットニュースといった文字情報を基本としたサービスから、WWW のような画像や音声などの情報が統一的に扱えるサービスまで、様々なアプリケーションレベルサービスが提供されており、新しいメディアとして確固たる地位を占めるに至っている。

インターネット普及の大きな鍵は、インターネットワーキングの技術に基づくものである。インターネットは、非均一なネットワークを相互接続する方法と通信の規約 (プロトコル) で構成され、異なる物理ネットワークや多様なコンピュータアーキテクチャなど広範囲な基盤ハードウェア技術に適応している。インターネットのアーキテクチャは、最下層のサービスとして信頼性のないコネクションレスパケット配送サービスを提供する。このサービスは IP (Internet Protocol) によって提供され、最善努力型 (**best-effort**) サービスと呼ばれる。インターネットの上位層で提供される全てのサービスは、 IP の提供するサービスの上に構築されている。

IP によって転送されるすべてのパケットは、ルータやリンクにおいて資源を共有する。故に、輻輳が生じればパケットは等しく遅延したり失われたりする。インターネットのアーキテクチャが、最下層のサービスとして最善努力型サービスを提供するのは、インターネットの基盤となるネットワークの大部分が、コネクションレスパケット配送サービスを提供するデータリンクによって構成されていることに起因する。

しかし近年、インターネットを取り巻くネットワーク環境は変化しつつある。新たなデータリンクとして、コネクション指向でコネクション毎の QoS (Quality of Service) の保証が可能な ATM (Asynchronous Transfer Mode) 技術による広域ネットワーク基盤の整備が進められている。また、映像や音声などの連続メディア伝送や対戦型ゲーム、遠隔操作などのインタラクティブシステムに必要な実時間データ通信、あるいは医療美術分野における高精細画像伝送システムや映画配送システムに必要な大容量データ通信など、新たな通信需要が高まりつつある。このような流れを受けて、インターネット上においても個々の通信のサービスの質を提供する保証型 (**guaranteed**) サービスの実現が求められるようになった。

このサービスを実現するためには、新たに

- 資源予約機構
- 経路制御機構

- ポリシ制御機構
- 流量制御機構
- パケットスケジューリング機構

などの機構が必要となる。

中でも資源予約機構と経路制御機構は、保証型サービスを行うための重要な機構である。経路制御機構は、通信を行うノード間の経路を提供し、資源予約機構は、経路上のルータやリンクの資源を確保する。

資源予約機構については、現在、資源予約プロトコル RSVP [4] がインターネット標準として仕様が公開され、実装も行われている。しかしながら経路制御機構に関しては、IETF の QoS Routing Working Group においても未だ検討中の段階に留まっており、研究途上の分野である。

1.2 目的

本研究の目的は、保証型サービスを提供するネットワークにおいて適切な経路制御機構を実現することである。

現在の経路制御機構は、基本的に IP が提供する最善努力型サービスのために設計されたものである。経路制御プロトコルは、様々なノードやネットワークへの到達可能性に関する情報を収集する。そして、宛先に対してより “近い” と思われる経路を選択し、経路制御テーブルに宛先 (ノードあるいはネットワーク) 毎のエントリとして記述する。しかし、従来の経路制御機構では、保証型サービスを提供するのに十分ではない。

保証型サービスは、基本的に個々の通信に対して独立に提供される。また個々の通信が要求する QoS は様々である。ある通信は通信遅延時間を最少化するような QoS を要求するかもしれないし、ある通信は通信帯域幅を最大化するような QoS を要求するかもしれない。すなわち、保証型サービスを提供するための経路制御機構は、QoS の要求を満たすような経路を個々の通信に対して独立に提供することが必要となる。そのために、経路制御プロトコルは、様々なネットワークの特性や利用可能な資源状態を考慮した上で、経路選択を行わなければならない。また経路制御テーブルは、個々の通信毎のエントリを記述できなければならない。

本研究では、保証型サービスを提供するネットワークを資源予約ネットワークと呼ぶことにする。そして、資源予約ネットワークに必要な様々な機構を資源予約ネットワークアーキテクチャとして定義し、それに基づく経路制御プロトコル **RRR (Routing for Resource Reservation)** を提案する。RRR は、従来のインターネットにおける経路制御プロトコルと比較して、以下の特徴を持つ。

- フローベース経路制御

  保証型サービスはフローに対して提供されるものであり、個々のフローに対応した経路を取り扱う。

- TOS メトリックによる経路選択

  フローの要求する TOS (Type Of Service) に応じた経路の選択を行う。

- ネットワーク資源情報の利用

  経路制御情報としてネットワーク資源情報を考慮し、経路の決定を行う。

- 代替経路の提供

  選択した経路において、ネットワーク資源の不足により資源予約が失敗に終った場合、可能ならば代替経路を提供する。

また本研究では、RRR を経路制御プロトコル OSPF との協調を考慮した経路制御デーモン **RRRD (RRR Daemon)** として実装し、既存のシステムとの親和性が高いことを示す。

1.3 本論文の構成

第 2章では、資源予約ネットワークのアーキテクチャと経路制御について、関連研究を交えながら概説する。第 3章では、本研究で提案する経路制御プロトコル RRR の設計について述べ、第 4章では、経路制御デーモン RRRD の実装について述べる。第 5章では、RRRD の実装を通して RRR の評価及び検討を行い、今後の課題を述べる。最後に第 6章において、本研究の結論を述べる。

第 2 章

資源予約ネットワークと経路制御

本章の前半では、本研究の基盤となる資源予約ネットワークアーキテクチャとアーキテクチャを構成する個々の技術の概説を行う。また後半では、保証型サービスを提供するための経路制御機構について、関連研究を交えながら必要な機能、特徴及び現状を述べる。

2.1 資源予約ネットワーク

本研究では、保証型サービスを提供するためのネットワークを資源予約ネットワークとして参照する。資源予約ネットワークは、あくまで現在のインターネットの延長にあたるものであり、すべての技術は既存のインターネット上に構築されることを仮定している。

2.1.1 フロー，保証型サービス

まず、本研究におけるフローと保証型サービスについて定義にする。

フロー

本研究における “フロー” の概念は、基本的に IETF IPNG Working Group が [10] で定めるところの “flow” の概念に等しいものとする。

> フローとは、特定の発信者から特定の宛先に向けて送られる一連のパケットの集合で、発信者が経路上のルータにおいて特別に扱ってほしいと考えるものである。

例えば、ある映像と音声のストリームデータに関する一連のパケットの集合や、あるホスト間での telnet セッションに関する一連のパケットの集合などがフローに相当する。

保証型サービス

Xerox の Schenker らは [37] において、IETF Integrated Services Working Group の保証型サービスについての見解を述べている。

> 保証型サービスとは、あるフローについて指定時間内にデータを転送し、指定されたトラフィックパラメータの範囲内でそのトラフィックを保証するサービスである。

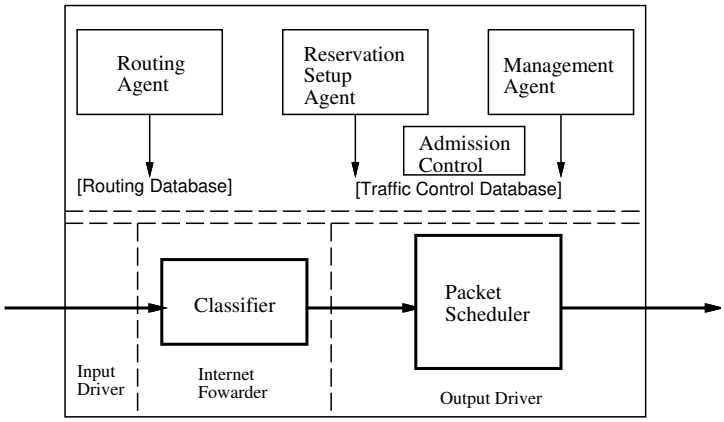


図 2.1: 統合型サービスのための実装参照モデル

トラフィックパラメータは、**QoS** パラメータあるいはフロースペック (**flow spec**) [29] とも呼ばれ、そのフローがどういう性質を持っているかを表現したり、ネットワークに対して提供してほしいサービスの質を指定したりする場合に用いる。パラメータとしては、単位時間当たりのデータ転送量、データ到着間隔の最大許容量、最大通信遅延時間、エラー率などを指定する。

このサービスは、送信者から受信者まで、ある時間内にデータが到着することを保証する必要があるアプリケーションが利用するものと考えられる。具体例としては、映像や音声といった連続メディアデータを取り扱うアプリケーションや、対戦型ゲームや遠隔操作といったのインタラクティブな制御システムなど、時間制約のあるデータを取り扱うアプリケーションなどが挙げられる。

2.1.2 参照モデル

保証型サービスを提供するために必要なネットワークアーキテクチャについては、ISI (Information Science Institute) の Braden らが [3] において述べている統合型サービス (Integrated Services) モデルがある。統合型サービスとは、インターネットにおいて、IP の提供する最善努力型サービスと同時に、実時間通信を行うための保証型サービスを提供するモデルである。統合型サービスの実装参照モデルとしては、図 2.1のようなアーキテクチャが示されている。

以下に、各モジュールの機能と研究の現状について述べる。

Packet Scheduler

キューの集合を用いて、個々の異なるフローに対して指定されたトラフィックパラメータを満たすようにパケットの転送を決定 (スケジューリング) する。この転送の決定を制御するのが、スケジューリングアルゴリズムである。スケジューリングアルゴリズムによって、ネットワークが提供できる QoS が決まるため、アーキテクチャの中でも重要な機構である。

MIT の Clark らは [7] の中で、統合型サービスを提供するパケットネットワークにおけるパケットスケジューリングについて述べており、FIFO (First In First Out) と WFQ (Weighted Fair Queueing) [11] の長所を組み合わせた FIFO+を提案している。

Classifier

パケットを幾つかの “クラス” に分類する。クラスは 1 つの抽象概念であり、ある特徴を持つパケットを 1 つの集合として取り扱いたい場合に用いる。クラスの分類は、パケットが属するフローのトラフィックパラメータの値やサービスの種類によるものであったり、パケットヘッダやパケットの内容によるものなど様々である。これらは、各ノードの Classifier がローカルに決定する。クラスという概念は、多数のフローによってリンクを共有する場合の資源管理モデルの中から生まれたものである。本来個々のフローで扱うべきものをクラスとしてまとめて扱うことで、オーバヘッドの削減と性能の向上が期待できる。

LBL (Lawrence Berkeley Laboratory) の Floyd と Jacobson は [17] の中で、統合型サービスを提供するパケットネットワークにおいて、Classifier 機構による CBQ (Class Based Queueing) をベースとした階層的なクラス分類を行う際のリンク共有のモデルについて述べている。

Admission Control

Reservation Setup Agent からの問い合わせに対して、現在利用可能なネットワーク資源の状態を調べ、あるトラフィックパラメータを持つ新しいフローをネットワークが受け入れることができるかどうかの決定を行う。そして、受け入れ可能あるいは拒否の通知を Reservation Setup Agent に対して返す。

USC (University of Southern California) の Jamin らは [22] において、トラフィックパラメータとしてトークンバケットパラメータを用いた場合の Admission Control の例を示している。

Reservation Setup Agent

フローに対する資源割り当ての管理を行う。あるフローについて資源予約の要求を受けた場合、経路上に沿ってトラフィックパラメータに基づいた資源 (帯域幅やバッファなど) の割り当てを行い、その状態を保持する。またそのフローについて通信の終了通知を受けた場合は、割り当てた資源を解放する。

提案されている資源予約プロトコルとしては、RSVP [46, 25]、ST2+ [18]、Tenet プロトコル体系 [1] における RCAP などがある。ST2+ と RCAP では、送信者が資源予約設立のメッセージを受信者に向けて送信する。資源予約設立のメッセージにはフローのトラフィックパラメータが含まれており、これに基づいて資源予約プロトコルのエージェントは、経路上のルータにおいて資源の割り当てのための処理を行う。このとき、前述の Admission Control モジュールに受け入れ不可を通知された場合、あるいは受信者によってフローの受け取りを拒否された場合は、トラフィックパラメータを変更して再度資源予約要求を行う。トラフィックパラメータの変更ができない場合、資源予約要求は失敗に終る。一方、RSVP では、受信者が送信者に対して資源予約設立のメッセージを送信し、資源予約を行う。

現在、RSVP がインターネット標準として仕様が公開され、ISI や Cisco Systems において実装が進んでいる。

Routing Agent

個々のフローについて経路の決定を行い、Routing Database を作成する。Routing Agent は、Reservation Setup Agent からの要求によりフローのトラフィックパラメータを満たすような経路を選択し、Routing Database にフローと経路を対応付けたエントリを作成し、結果を Reservation

Setup Agent に返す。フローのトラフィックパラメータを考慮する経路制御を **QoS** 経路制御という。

UCL (University College London) の Wang ら [44] によると、QoS 経路制御のように複数のパラメータ制約における経路の計算は NP 完全であることがわかっている。したがって、本質的な意味における QoS 経路制御は現在存在しない。しかしながらいくつかの制約を設けることにより、保証型サービスに適した経路制御を考えることは可能である。資源予約ネットワークにおける経路制御に関しては、次節で詳しく述べることにする。

Management Agent

Classifier や Packet Scheduler、Admission Control モジュールに対して、ネットワークのポリシにしたがって個々のフローに対する扱いを制御する。“ポリシ” という概念は、非常に曖昧かつ広範囲に渡るものである。ここではネットワークに対する制御という観点から、ポリシを “管理者がネットワークに対して適用する何らかの運用方針” として考えるものとする。例えば、利用者の権限や優先度などをフローに対して適用する。

従来ポリシ制御の研究は、経路制御と関連付けて研究されてきた。特にネットワーク管理領域間でのドメイン間経路制御の分野 [5, 13, 40, 39, 33] において、ポリシ制御の問題を取り扱っている。しかし、保証型サービスを提供するといったネットワークサービスの観点からポリシ制御の問題を取り扱った提案は、現在のところない。

2.1.3 資源予約ネットワークアーキテクチャ

本研究では、資源予約ネットワークの経路制御機構に相当する部分、つまり図 2.1の参照モデルにおける Routing Agent 及び Routing Agent と Reservation Setup Agent、Routing Database との相互作用について注目する。なお、その他のモジュールに関しては、本研究の対象外であるので、これ以降は特に触れない。しかし、基本的にこれらの構成要素は変わらないものとし、同等の機能を持つモジュールが存在することを前提として話を進めることにする。

本研究で想定する資源予約ネットワークのアーキテクチャを図 2.2に示す。このアーキテクチャにおける経路制御機構の特徴は、従来の最善努力型サービスを提供する経路制御機構と、保証型サービスを提供する経路制御機構が独立して存在することである。

従来の最善努力型の経路制御機構では、経路制御プロトコルは収集した経路制御情報に基づいて経路の選択を行い、直接経路制御テーブルに経路を記述する。しかし資源予約ネットワークにおける経路の選択は、経路制御プロトコル以外に、利用者による権限や優先度、課金などのポリシ制御機構も関係する。したがって、最終的な経路の決定者は経路制御プロトコルではなく、資源予約機構と考える方が自然である。経路制御プロトコルは、あくまで資源予約機構の要求に応じて適切と考えられる経路を提供するに過ぎない。そこで、保証型サービスの経路制御機構を別に設けて、保証型サービスを提供する経路制御プロトコルは、経路選択の結果を資源予約機構に対して提供し、直接経路制御テーブルには記述しないものとする。

また従来の経路制御機構では、経路制御テーブルは、経路を宛先 (ノードあるいはネットワーク) 毎のエントリとして記述する。しかし、保証型サービスは個々のフローに対して提供されるため、宛先毎ではなくフロー毎の経路の提供が必要となる。したがって、保証型サービスにおける経路制御機構では、経路制御テーブルは、個々のフローに対してエントリを記述するものとする。

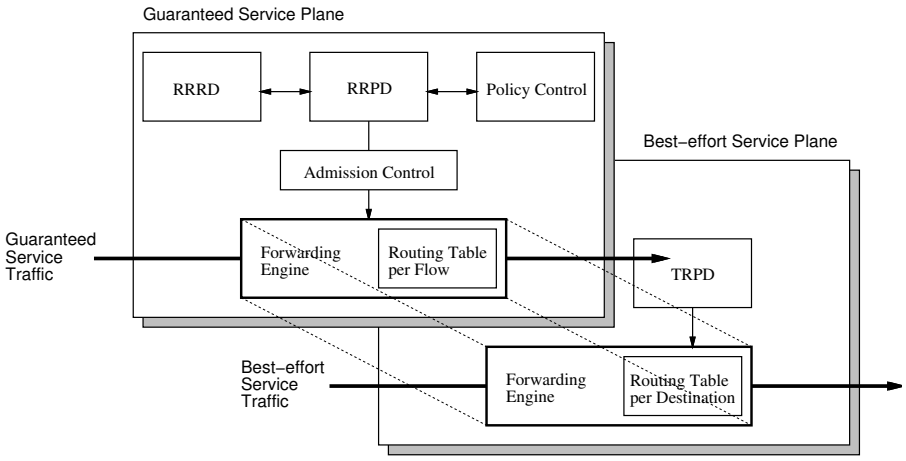


図 2.2: 資源予約ネットワークアーキテクチャ

Guaranteed Service Plane

保証型サービスを受けるパケットを取り扱う。

- Resource Reservation Protocol Daemon (RRPD)

  図 2.1における Reservation Setup Agent に相当する。RRPD は、RRR Daemon (RRRD) に対してフローについての経路を要求する。RRRD から経路を受け取ると、Policy Control モジュールや Admission Control モジュールなどを通して最終的な経路の決定を行い、必要な資源を割り当て、Forwarding Engine の Routing Table にそのフローについてのエントリを記述する。また RRPD は RRRD に対して、資源予約情報やネットワーク資源情報を提供する。

- RRR Daemon (RRRD)

  本研究で提案する経路制御プロトコル RRR のデーモンである。RRRD は図 2.1における Routing Agent に相当する。RRRD は、経路制御テーブルに対して直接エントリを記述しない。あくまで RRPD の要求に応じて経路の提供を行う。

- Forwarding Engine

  図 2.1における Packet Scheduler、Classifier に相当する。保証型サービスを受けるパケットは、フロー毎の経路制御テーブルによりフロー毎の経路制御が行われる。この経路制御テーブルは、RRRD ではなく RRPD により管理される。

- Policy Control

  図 2.1における Management Agent と同様の機能を持ち、ポリシ制御に関する情報を取り扱う。

- Admission Control

  図 2.1における Admission Control と同様の機能を持つ。

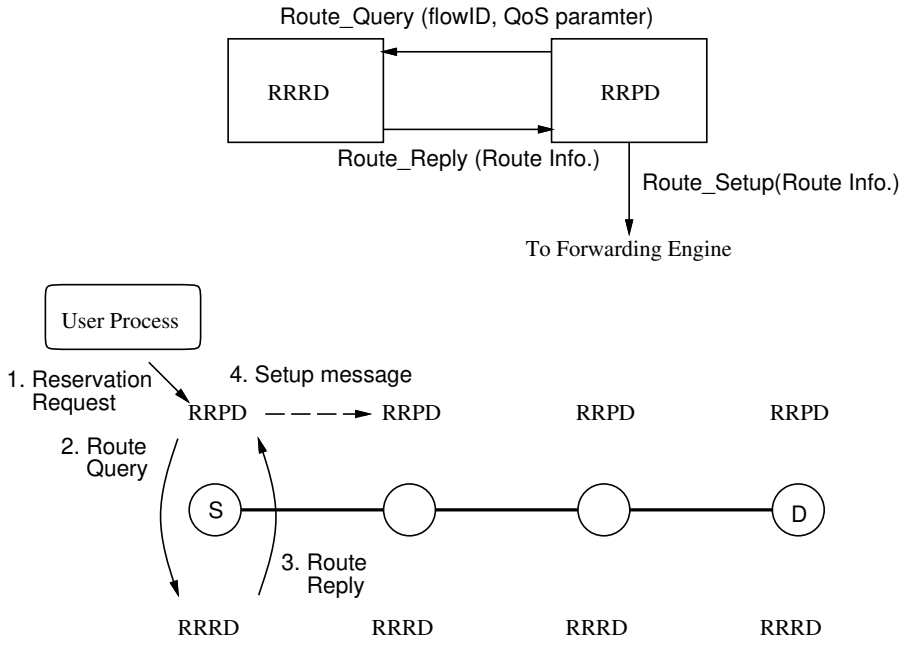


図 2.3: 保証型サービス提供の流れ

Best-effort Service Plane

最善努力型サービスを受けるパケットを取り扱う。

- Traditional Routing Protocol Daemon (TRPD)

  保証型サービスを必要としないパケットのための経路制御プロトコルのデーモンである。従来の経路制御プロトコルによるデーモンは TRPD である。TRPD で決定された経路は、直接 Forwarding Engine の経路制御テーブルに登録される。

- Forwarding Engine

  基本的に Guaranteed Service Plane と同じモジュールである。最善努力型サービスを受けるパケットは、TRPD が決定した経路制御テーブルによって宛先毎に経路制御が行われる。

保証型サービス提供の流れ

図 2.2における RRRD の動作及び RRPD との相互作用を明確にするため、実際の保証型サービス提供の流れについて触れておく。話を簡単化するため、RRPD、RRRD についての動作のみ記述する。もちろん他のモジュールが関係した場合でも基本的な動作に変更はない。

例として、ノード S からノード D に対するフローについて、保証型サービスを提供する場合を考える (図 2.3)。

1. 利用者は、必要なパラメータ (フローの識別子、QoS パラメータ、資源予約スタイルなど) を含んだ資源予約要求のメッセージを S の RRPD に伝える。

2. RRPD は、RRRD に対して要求を受けたフローの経路の問い合わせを行う。このとき必要なパラメータ (QoS パラメータなど) があれば同時に伝える。
3. RRRD は、そのフローに対して最適と思われる経路を選択し RRPD に返す。
4. RRPD は、この経路を Forwarding Engine の経路制御テーブルに登録し、経路上の次のノードの RRPD に資源予約要求メッセージを転送する。
5. 受信ノード D に到着するまで、2. ~ 4. を繰り返す。
6. 受信ノード D に到着した時点で資源予約完了する。このとき必要があれば確認応答を送信ノード S に返す。

反対に受信側から送信側に向けて資源予約要求を行う場合も同様の流れをたどる。

次節からは、資源予約ネットワークアーキテクチャの中で、本研究のテーマである経路制御機構に焦点を絞り、さらに詳しく議論を進めることにする。

2.2 経路制御プロトコル

一般に経路制御機構は、以下の 2 つの機構から構成される [23]。

- 経路制御プロトコル

  経路制御テーブルの管理を行う。ネットワークトポロジの情報など経路制御に必要な情報を他のノードと交換し、一貫性のある経路制御テーブルを構築する。

- 経路制御テーブル

  転送の際に転送機構によって参照される。インターネットのようなパケットネットワークでは、パケットの宛先アドレスに対する出力インタフェース及び次に転送すべきノードのアドレスがエントリとして記述される。

本研究では問題を切り分けるため、資源予約ネットワークにおける経路制御機構を上記の 2 つの機構に分けて取り扱う。まず本節では経路制御プロトコルについて、そして次節では経路制御テーブルについて検討する。

2.2.1 インターネットにおける経路制御プロトコルの問題

まずインターネットにおける経路制御プロトコルで、保証型サービスを提供する場合の問題点について取り上げる。

現在のインターネットにおける経路制御プロトコルのほとんどは、宛先に対する到達可能性の情報を基に経路の選択を行う。ネットワーク管理領域内で用いられる RIP [24] や OSPF [27] などの IGP (Interior Gateway Protocol) は、宛先までのホップ数あるいはリンク固有のメトリックを経路制御情報として交換し、最短と思われる経路を選択する。また複数のネットワーク管理境界間で用いられる EGP [34] や BGP [33] などの EGP (Exterior Gateway Protocol) についても、宛先までの到達情報 (到達経路) の交換により経路の選択が行われる。

これらの経路制御プロトコルは、基本的に IP で提供される最善努力型サービスを基本に設計されている。最善努力型サービスにおいては、パケットが正しく宛先のノードに到着することが重要であり、経路制御プロトコルは、ネットワークのトポロジ (一部または全部) を理解し、より宛先に対して “近い” と思われる経路を選択するだけで十分に機能することができる。もちろん資源予約ネットワークにおいても、本質的にこのような到達可能性による経路の選択は重要である。しかし、保証型サービスを提供するためには、以下のような問題が残る。

- フロー毎の扱いを行わない

  保証型サービスは、個々のフローに対して要求するトラフィックパラメータを保証するものであり、経路制御プロトコルも個々のフローに対して一意な経路の選択を行う必要がある。しかし現在の経路制御プロトコルでは、宛先アドレスによる一意な経路の選択しかできない。

- トラフィックパラメータを考慮しない

  保証型サービスを提供するための経路制御では、ネットワークのノード及びリンクが受け入れ可能なトラフィックパラメータ、すなわち利用可能なネットワーク資源を考慮した経路の選択を行うことが重要である。しかし、現在の経路制御プロトコルによる経路選択では、経路制御情報としてトラフィックパラメータを扱う仕組みがない。したがって、要求されたトラフィックパラメータを満たすような経路を明示的に選択できない。

- 代替経路の提供機構がない

  資源予約ネットワークでは、ネットワーク資源の不足やネットワーク障害以外の理由 (ポリシ制御など) により、フローの受け入れを拒否される場合が考えられる。もし選択した経路において受け入れを拒否された場合、代替経路を提供する必要が生じる。しかし、現在の経路制御プロトコルには代替経路を提供する機構はない。

2.2.2 資源予約ネットワークにおける経路制御プロトコルの目標と論点

前述した問題点を踏まえた上で、資源予約ネットワークにおける経路制御プロトコルの目標と論点について述べる。

目標

資源予約ネットワークにおける経路制御プロトコルの特徴を端的に表すと “フローの要求するトラフィックパラメータを満たすような経路を提供する” ということになる。これを一般に **QoS** 経路制御と呼ぶ。[8] には IETF QoS Routing Working Group の QoS 経路制御に関する中間報告がまとめられており、QoS 経路制御プロトコルの目標として以下の 3 つが挙げられている。

1. 経路決定の動的な対応

   フローに対する資源予約要求や資源解放要求は動的に生成されるため、ネットワークの各ノードやリンクで利用可能な資源の量も動的に変化する。したがって経路の決定も、できるだけその時点でのネットワーク資源の状態に基づき動的に行うことが望ましい。

2. 資源の効率的な使用

   ネットワーク資源は有限であるため、保証型サービスを提供できるフローの数も限られる。できるだけ多くのフローに対してサービスを提供するためには、経路の選択時になるべく余裕のある経路を選択するなど、資源の使用効率を考慮することが重要である。

3. 程良いオーバヘッド

   一般的に状態依存型 (State-Dependent) の経路制御は、状態非依存型 (State-Insensitive) に比べて、経路制御情報の更新や経路の計算などのオーバヘッドが大きく、規模拡張性や安定性に問題がある。経路制御プロトコルの性能とオーバヘッドはトレードオフの関係にあり、実現可能な範囲内で効果的な方法を見出すことが求められる。

しかし、これらの目標は、実際の経路制御プロトコルを構築する上では、極めて曖昧な意味合いしかもたない。例えば、資源の効率的な使用について考える場合、

- どのような情報が必要か

  利用可能なネットワーク資源情報 ?
  個々のフローに関する資源予約情報 ?
  ...

- 何を最適化するのか

  ネットワーク全体の使用率 ?
  サービスの提供を受けられるフローの数 ?
  フロー間の公平性 ?
  ...

など、問題は複雑である。

そこで本研究では、実際に QoS 経路制御プロトコルを構築する上で、特に重要な 3 つの論点に絞って検討を行う。

メトリック

経路制御プロトコルにおいて、ネットワークはルータやホストを節点 (node)、それを結ぶリンクを枝 (branch) としたグラフによって表現される。すなわち経路とは、ある任意の 2 つ以上の節点を含む木を構成する枝の集合である。通常枝にはメトリックという重み付けがなされる。経路の選択とは、このメトリックを基にどのような枝の集合によって木を構成するかを決定することである。例えば、ある 2 つの節点を含む木 (この場合は路とも言える) で、そこに含まれる枝のメトリックの総和が最小になるような枝の集合を求める問題は最短路問題と呼ばれ、Dijkstra のアルゴリズムにより解くことができる。それ故、メトリックをどのように設定するかは重要な問題である。

Wang らは [44] の中で、QoS 経路制御におけるメトリック選択の考慮点として、次のように述べている。

1. メトリックは、それを用いた効率的な経路計算のアルゴリズムが存在するようなものでなければならない。またアルゴリズムは、集中環境と分散環境のいずれにも対応することが望ましい。

2. メトリックは、ネットワークの基本的な特徴を反映するものでなければならない。また QoS 要求をサポートすることができる情報を含むべきである。

3. メトリックは互いに直交すべきで、冗長な情報を含むべきではない。

多くの経路制御プロトコルでは、ホップ数や管理者の恣意的な重み付けによる単一な値のメトリックを使用している。よって単一な値のメトリックで QoS 経路制御に必要なトラフィックパラメータを表現することができれば、経路計算のアルゴリズムを変えることなく対応することが可能となる。

例えば、帯域幅 B、遅延時間 D、ロス率を L とし、経路 p におけるメトリック M を以下のように設定する。

$$M = f(p) = \sum_{l \in p} \frac{B(l)}{D(l) \times L(l)}$$

この場合、M の大きな経路を選択すれば良いことになる。しかしトラフィックパラメータの複数の要素を 1 つのメトリックに集約する方法では、良い経路を選択する目安にはなるが、実際に選択した経路が、要求されたトラフィックパラメータを満たしているかを判断するのに十分な情報を含んでいるとは言えない。例えば、帯域幅 B について考えた場合、経路 p が満たす値は、経路の最小値

$$\{l \in p \mid min[B(l)]\}$$

となるが、遅延時間 D について考えた場合は、経路の和

$$\sum_{l \in p} D(l)$$

となる。またロス率 L は、積の計算

$$1 - \prod_{l \in p} (1 - L(l))$$

となる。このようにトラフィックパラメータの各要素毎に計算方法が異なるため、集約されたメトリックが、正しくその経路が満たすトラフィックパラメータを反映しているとは言えない。

そうであるとするならば、トラフィックパラメータの各要素毎にメトリックを設定し、各要素毎に計算を行うのが素直である。しかし、複数制約の条件下での経路の計算は非常に困難で、多項式計算時間によるアルゴリズムは存在しない (NP 完全問題)。

したがって、現実的にはトラフィックパラメータの各要素の中から、保証型サービスを必要とするアプリケーションにとって、基本的かつ重要と考えられるものをメトリックとして選択するのが妥当である。本研究では、保証型サービスの定義で挙げたアプリケーション、すなわち連続メディアデータや時間制約のあるデータを取り扱うアプリケーションを対象とすることから、帯域幅及び遅延時間をメトリックの主な要素とする。これは Wang らも同様の主張をしており、十分妥当性があると考えられる。遅延時間は、さらに伝搬遅延時間とキューイング (queueing) による遅延時間の 2 つの要素から構成されるが、このうちキューイングによる遅延時間は、使用できる帯域幅と実際のトラフィックで使用した帯域幅によって決定される。 よって、キューイングによる遅延時間は帯域幅の要素に含まれるものとし、帯域幅と伝搬遅延時間を要素として考えることにする。すなわち、これは経路の “幅” と “距離” に相当し、また同時に基本的なネットワーク資源を表すものである。

経路の決定

経路制御プロトコルが実際にパケットに対して経路を決定する方法として、始点経路制御とホップバイホップ (hop-by-hop) 経路制御の 2 つがある。

始点経路制御では、パケットの転送経路を送信ノードにおいて決定し、パケットヘッダに経路リストを書き込む。そして各ノードでは、パケットヘッダの経路リストをもとに転送を行う。経路の計算は送信ノードのみで行われるため、送信者による経路の選択に柔軟性があり、目的に応じて様々な経路を利用することが可能である。また経路の循環 (loop) の危険性も少ない。しかし、送信ノードは宛先までの経路に関してすべての経路制御情報を知る必要があり、パケットヘッダも大きくなる。

これに対してホップバイホップ経路制御は、パケットは経路上の各ノードにおいて転送経路が決定される。ネットワークの各ノードは、すべての宛先に対する転送すべき次のノードが書かれた経路制御テーブルを保持している。あるノードにパケットが到着すると、テーブルを検索して転送すべき次のノードを決定する。宛先までの完全な経路を知る必要がないため、経路の計算はごく限られた範囲の経路制御情報のみで行うことができる。しかし、各ノードが独立して経路の決定を行うため、すべてのノードが同じアルゴリズムで経路の決定を行わないと経路の循環などが起きる可能性がある。また全てのノードが同じ経路制御情報を持っていないと、一貫性のある経路制御ができない。

現在のインターネットでは、ホップバイホップ経路制御が主流であることからもわかるように、一般的にホップバイホップ経路制御の方が規模拡張性に優れ、分散環境に適している。しかし、QoS 経路制御を考える場合、フロー毎に QoS の要求が異なるため、フロー毎に経路の計算を行う必要がある。始点経路制御ではそのフローの送信ノードのみで経路計算を行えばよいが、ホップバイホップ経路制御では、経路上の全てのノードにおいて行う必要があり、大きなオーバヘッドになることが考えられる。またホップバイホップ経路制御では、全てのノードが同じ経路制御情報を持つことが前提となるが、ネットワーク資源情報など動的に状態が変化するものにおいて、これを満たすことは極めて難しい。始点経路制御の場合は、経路の決定は送信ノードのみであり、たとえ全てのノードが同じ経路制御情報を持っていなくても、(最適ではないにせよ) 適切な経路を選択することが可能である。

また、資源予約ネットワークにおける経路制御の場合、経路制御プロトコルは転送すべき次のノードだけではなく、宛先までどのような経路を通れば最終的に QoS の要求を満たすことができるのかを考慮する必要がある。これは本質的に始点経路制御での経路決定に等しい。またフローが要求する QoS を一種のポリシとして考えるならば、QoS 経路制御はポリシ経路制御とみることもできる。多くのポリシ経路制御プロトコルも始点経路制御を採用しており、この方針は妥当であると言える。したがって、本研究では始点経路制御を基本として考えることにする。

経路制御情報の表現

経路制御プロトコルにおける経路の決定は、様々な経路制御情報に基づいて行われる。最も基本的な経路制御情報はトポロジ情報であり、ネットワークのノードやリンクがどのように接続されているのかについての情報である。この他、ポリシ経路制御を行う場合にはポリシに関する情報が、経路制御によるネットワークトラフィックの負荷分散を行う場合には負荷情報など、それぞれの目的に応じた経路制御情報が考えられる。QoS 経路制御の場合は、現在利用可能なネットワーク資源 (リンクの帯域幅や、ルータでの平均遅延時間など) が経路制御情報として考えられる。これらの経路制御情報は、個々の目的に応じたパラメータで表現されることもあれば、論点

の 1 つ目で挙げたメトリックの形で表現されることもある。

経路制御情報の表現としては、大きく分けで距離ベクトル (distance-vector) 型とリンクステート (link-state) 型の 2 つがある。距離ベクトル型では、経路制御情報を自ノードから他のすべてのノード (あるいはサブネットワーク) までの距離 (メトリック) として表現する。一方、リンクステート型の場合は、すべてのリンクに対する接続ノードと距離 (メトリック) を経路制御情報としている。リンクステート型がネットワークのトポロジ情報をそのまま経路制御情報として表現しているのに対し、距離ベクトル型では距離という形でトポロジ情報を集約している。したがって、距離ベクトル型では、経路制御情報の交換は隣接ノード間のみで済むのに対し、リンクステート型の場合は、経路制御情報をネットワーク全てのノードに対して配布する必要がある。経路制御情報の伝搬コストを考えた場合、距離ベクトル型の方が優れている。

しかし QoS 経路制御の場合を考えてみると、距離ベクトル型では基本的にリンクやノードに付随する情報 (ネットワーク資源情報、資源予約情報など) を伝える手段がない。リンクステート型の場合には、経路制御情報は各リンクについての情報であるため、これにネットワーク資源情報や資源予約情報を付加することは容易であると考えられる。したがって本研究では、経路制御情報の表現はリンクステート型で行うことにする。

代替経路の提供

代替経路の提供に関するアルゴリズムは、回線交換ネットワークの代表である電話網における経路制御プロトコルの分野で様々な方法が提案され、実際に運用されている [38, 14, 15]。これらの経路制御プロトコルにおける代替経路提供のアルゴリズムは、いずれも比較的単純なものである。基本的には、呼 (フローの意味と等しいが回線交換ネットワークなのでこのように表現する) がブロックされた場合、予め選択した幾つかの代替経路の候補の中から、クランクバックの手法で順に代替経路を試していくというものである。

代替経路の利用で問題となるのが、混雑の拡散である (図 2.4)。あるノード間 (A–B) のリンクにトラフィックが集中し混雑し始めると、そのノード間のトラフィックは代替経路 (A–C–B, A–D–B) に回される。代替経路には、本来その経路を通常経路としているトラフィック (A–D) が存在するが、代替経路に回されたトラフィックと競合するため、代替経路自体も混雑し始める。するとさらに別の代替経路 (A–E–D) を使うことになる。このように、あるリンクの混雑は周囲に拡散していく。さらに悪いことには、代替経路は通常経路よりも冗長な経路であるため、余計な資源を消費することになる。この悪循環はネットワーク全体の使用効率の低下を招く。

この事態に対応するため、電話網においては代替経路を通る呼が使用できるリンクの容量は予め制限されている (Trunk Reservation)。また選択できる代替経路は 2 ホップまで (つまり送信ノードと受信ノードとの間の経由ノードは 1 つまで) と制限されている。この制限により、代替経路の使用による混雑の拡散を食い止め、全体の使用効率の低下を防いでいる。しかし電話網における経路制御プロトコルは、いずれも基本的には完全結合のネットワークトポロジを仮定しており、また確保する帯域幅は固定である。したがって、この方法を現在のインターネット環境に直接適用することは難しい。

パケットネットワークにおける代替経路の提供に関しては、Dijkstra のアルゴリズムを複数の経路の計算が行えるように拡張した k Shortest Path アルゴリズム [41] や ARPANET における SPF 経路制御プロトコルを拡張した SPF-EE(Emergency Exits)[42] などがある。しかし、これらはいずれも QoS 経路制御を考慮したものではない。

代替経路の提供については、設計の段階で再度検討することにする。

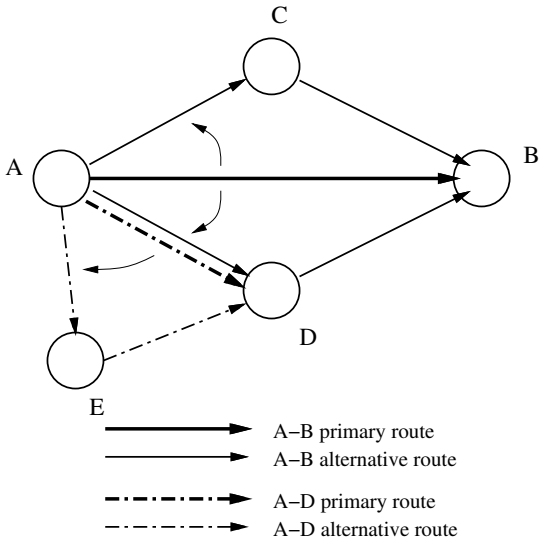


図 2.4: 混雑の拡散

これまでの議論をまとめると、

- 帯域幅と遅延時間を主なメトリックの要素とする。
- 経路の決定は始点経路制御を基本とする。
- 経路制御情報の表現はリンクステート型で行う。
- 代替経路の提供は必要であるが、方法は検討の余地がある。

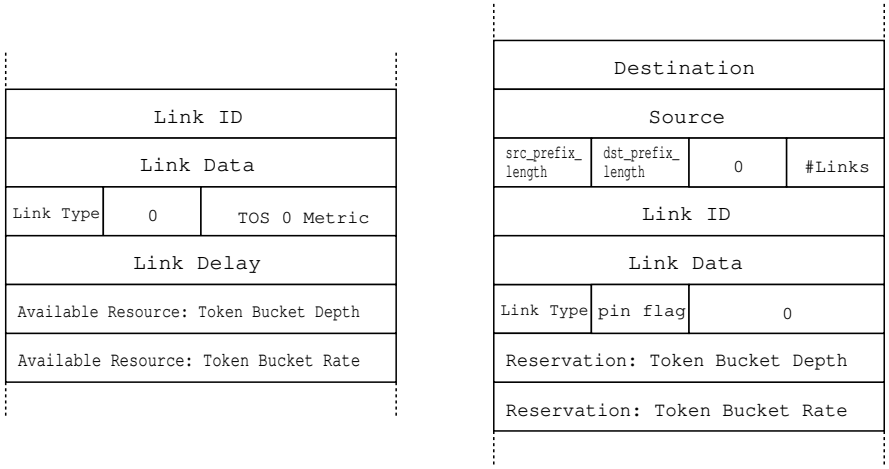
となる。これらを踏まえた上で、次章において本研究で提案する経路制御プロトコル RRR の設計について詳しく述べることにする。

2.2.3 提案されているプロトコル

QoS 経路制御に関する研究は、現在理論的アプローチに留まっており、実際的な経路制御プロトコルの提案は極めて少ない。唯一、Internet Draft としてまとめられた経路制御プロトコルに、Bay Networks の Zhang らが提案する QOSPF [47] がある。現在このドラフトは破棄されているため仕様としては無効になっているが、具体的に QoS を考慮した経路制御プロトコルを仕様としてまとめたことは、注目すべきことである。本研究で提案する経路制御プロトコルも参考にしていることから、ここでは QOSPF を提案されているプロトコルとして概説することにする。

QOSPF の概要

QOSPF は、OSPF [27] や MOSPF [26] を拡張した経路制御プロトコルで、QoS を必要とするあるフローに対して、RSVP のような資源予約プロトコルと連携して、その QoS を満たすような経路を提供する。OSPF や MOSPF では、基本的にトポロジ情報のみから経路の計算を行っ



(a) Link Resource Advertisements (part)　(b) Resource Reservation Advertisements (part)

図 2.5: QOSPF におけるネットワーク資源情報

ていたが、QOSPF ではそれに加えて帯域幅や遅延時間、資源予約状況といったネットワーク資源情報も考慮する。

QOSPF は OSPF や MOSPF に比べ、以下の 3 つの拡張を行っている。

1. ネットワーク資源情報

   QOSPF において各ルータは、トポロジ情報の他にネットワーク資源情報を経路制御情報として交換する。ネットワーク資源情報には 2 種類ある。

   - Link Resource Advertisements (RES-LSAs)

     あるリンクにおいて現在利用可能な資源を記述する。RES-LSAs は、QoS を考慮した経路の計算を行う場合、従来のトポロジ情報のみを記述した Router Links Advertisements (Router-LSAs) に代わって用いられる。RES-LSAs には従来の Router-LSA に記述されていたメトリックの他に、リンクの最大遅延時間、トークンバケットパラメータで表された利用可能な資源が記述される (図 2.5 (a))。このトークンバケットパラメータは [37] で定められているうち、トークンバケットの深さ (token bucket depth) とトークンバケットのレート (token bucket rate) が使われている。これらのパラメータは、主にリンクの帯域幅と関連付けられる。RES-LSAs は新しくリンクが生成された場合だけでなく、リンクの最大遅延時間や利用可能な資源が変化した場合にも生成 (更新) される。

   - Resource Reservation Advertisements (RRAs)

     各ルータにおいて、各々のフローの資源予約状況を表す。RRAs にはフローの識別子である送信元アドレスと宛先アドレスの組と、リンクの識別子及びそのリンクにおいてフローが予約している資源をトークンバケットパラメータで表した値が記述される (図 2.5 (b))。RRA は、資源が予約されたり解放されたりする毎に生成される。

   これらのネットワーク資源情報は、基本的にエリア内にフラッディングされる。

2. 経路の固定 (Route Pinning)

   QOSPF では、一旦フローに対して QoS を満たすことのできる経路を選択した場合、その経路を固定することができる。経路の固定は、資源予約プロトコルからの資源予約要求メッセージ (RSVP の RESV メッセージなど) により行われる。固定された経路に関しては、その後新たにより良い経路が見つかった場合でも、現在の経路がネットワーク障害などの理由により使えなくなるまでは変更されない。これは、経路変更により一時的に QoS を満たせないような状況が生じるのを防ぐためである。

3. データ駆動型 (data-driven) 経路計算

   QOSPF では、送信元アドレスと宛先アドレスの組に対して経路計算を行う。経路の計算は、外部イベント (例えば RSVP の PATH メッセージの到着) をトリガーとして行われる。つまり経路計算は、OSPF のようにネットワークの全ての宛先アドレスに対してではなく、必要な送信アドレスと宛先アドレスの組に対してしか行わない。

QOSPF の経路計算

QOSPF の経路計算は以下の要素を基に行われる。

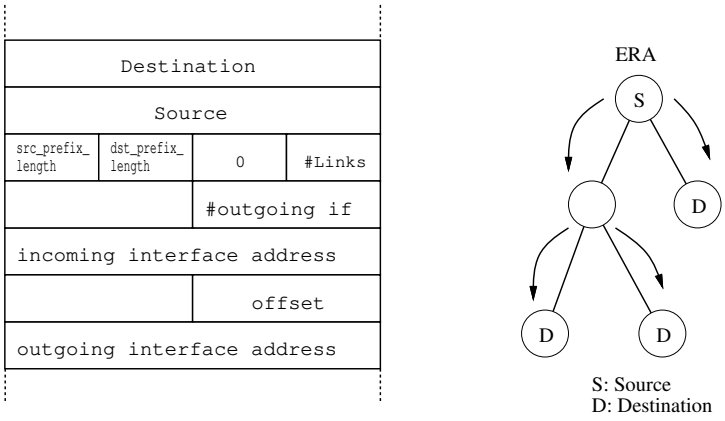
- フローの識別子 (送信元アドレスと宛先アドレスの組)
- 要求される QoS パラメータ (RSVP メッセージなどに含まれるトークンバケットパラメータ)
- ネットワーク資源情報 (RES-LSAs、RRAs)
- その他のトポロジ情報 (Network Links Advertisements (Network-LSAs))
- マルチキャストグループ情報 (Group-Membership Link State Advertisements (Group-Membership-LSAs))

基本的には OSPF や MOSPF と同様、Dijkstra のアルゴリズムによる最短経路計算を行うが、アルゴリズムに若干の変更が加えられている。

1. トポロジ情報として Router-LSAs の代わりに RES-LSAs を参照する。フローの要求する QoS パラメータ (トークンバケットパラメータ) と RES-LSAs を比較し、要求を満たすリンクのみでネットワークを構成する (つまり要求に満たないリンクはないものとして考える)。

2. 1. で構成したネットワークにおいて Dijkstra のアルゴリズムによる経路計算を行う。最短経路の計算にはメトリックではなく最大遅延時間が使われ、遅延時間の最も短い経路を選択する。

宛先アドレスがエリア外の場合も、Summary Link Advertisements (Summary LSAs) にしたがって、宛先アドレス対して最も最短の遅延時間を広告しているエリア境界ルータ (Area Border Rouer, ABR) に対して経路の計算を行う。しかしエリア境界ルータから先、すなわちエリア外における宛先アドレスまでのトポロジ情報やネットワーク資源情報は隠蔽されている。したがって選択したエリア境界ルータを経由する経路では、要求する QoS を満たせない可能性もある。この場合エリア境界ルータは、そのフローの QoS を満たす経路が存在しないことを Summary LSA によって広告する。エリア内の各ルータは、この Summary LSA によって別のエリア境界ルータを



(a) Explicit Route Advertisements (part)　　(b) ERA Distribution

図 2.6: Explicit Routing

経由する経路を選択する。AS (Autonomous System) 間の場合も、AS 境界ルータ (Autonoumous System Border Router, ASBR) が同様のことを行う。

QOSPF による QoS 経路制御は、“フローの要求するトークンバケットパラメータを満たすリンクの中で、遅延時間の最も短い経路を選択するようなアルゴリズムである” と言える。ネットワーク資源利用効率の最適化や代替経路の提供に関しては何も行われない。

Explicit Routing

QOSPF は、ホップバイホップ経路制御の経路制御プロトコルである。しかし、ホップバイホップでフロー毎の経路計算を行うことは、オーバヘッドが大きく、規模拡張性に乏しい。さらに QOSPF では、利用可能なネットワーク資源の情報や資源予約状況などの情報交換が、フローの数の増加したがって増大することも指摘されている。この問題を解決するため、QOSPF では Explit Routing (ER) という機構を導入している。

ER とは、端的に言えば QOSPF への始点経路制御の導入である。あるフローの送信元ホスト、あるいはフローの経路にあたるエリア境界ルータが、フローの下流ルータ及びホストに対する経路を計算し、その経路情報のみを経路に沿う形で配布するというものである (図 2.6 (b))。この経路情報は ERA (Explicit Router Advertisement) と呼ばれる (図 2.6 (a))。ERA には、経路情報を配布するための Installation-ERA と消去するための Flushing-ERA がある。

ER を導入することで、経路の計算を経路上のルータ毎に行う必要がなくなる。また、RRAs もフラッディングではなく、ERA の送信ルータに対して送るだけでよい。ER の導入は、オーバヘッドが大きくなりがちな QoS 経路制御においては、重要なアプローチであると考えられる。

QOSPF における問題点

QOSPF は、OSPF や MOSPF の枠組を基本として QoS 経路制御に必要な機能を追加したものであるが、実際に運用する場合を考えると、幾つかの問題がある。まず QOSPF では、ネットワーク資源情報の交換は RES-LSAs と RRAs によって行われる。特に RRAs は、資源が予約されたり解放されたりする度に生成されるため、トラフィックが増加するにつれて RRAs が大量に

生成されることになり、コストの増加が避けられない。また、RES-LSAs や RRAs をフラッディングすることによって、すべてのノードが一斉に混雑しているリンクを回避するため、経路の振動の問題が生じると考えられる。その他、明示的に代替経路を要求することもできない。

その他の提案

その他の提案としては、IBM の Guerin らが OSPF を QoS 経路制御に対応するように拡張したもの [20] がある。この QoS 経路制御では、ある帯域幅の要求を満たす最もホップ数小さい経路を選択するアルゴリズムを採用している。また Northern California State University の Salama らの提案する Delay-Constrained Unicast Routing [36] では、遅延時間制約のなかでリンクに付けられたコストの総和が最も小さい経路を選択するアルゴリズムを示している。

いずれも 2 つの要求 (制約) 条件を満たすような経路計算のアルゴリズムであるが、基本的なアルゴリズムの提案に終っており、具体的な経路制御プロトコルの構築には至っていない。

2.3 経路制御テーブル

本節では、経路制御機構のもう 1 つの機構である経路制御テーブルについて検討する。経路制御テーブルとは、パケットの転送を行う転送機構がパケットに書かれた何らかの情報 (多くの場合は宛先アドレス) を基に、転送先を決定するために参照する経路表のことである。

2.3.1 IP の経路制御テーブル

インターネットのパケット転送機構は IP であり、経路制御テーブルも IP によって参照される。IP のパケット転送アルゴリズムでは、パケットの宛先アドレスと経路制御テーブルの宛先アドレス (ホストのアドレスまたはネットワークのアドレス) のエントリとの比較を行い、最適と思われるエントリを選択する。エントリは、宛先アドレスと次に転送すべきノードのアドレス及び出力インタフェースの組で構成され、これによりパケットの経路が決定する。

パケットネットワークにおいてパケットを宛先アドレスに届けるためには、宛先アドレスによる経路制御テーブルがあれば十分である。またインターネットのアドレス体系では、アドレス中にネットワークの識別子が含まれており、経路制御テーブルの効率的な集約が可能となっている。故に、IP の経路制御テーブル及びアルゴリズムでは、宛先アドレスによる経路の選択しかできない。保証型サービスを提供するには、フロー毎の経路選択を行う必要があるため、従来の機構では不十分である。

例えば、宛先アドレスをフローの識別子として利用することができれば、既存の経路制御テーブルでフロー毎の経路制御が可能である。インターネットのアドレス体系では、アドレスは基本的にホストのネットワークインタフェースに対してつけられるため、クラス D のマルチキャストアドレスなどの特別なアドレス領域などをフロー識別子として割り当てることが考えられる。しかし、マルチキャストアドレスは、本来ホストの集合で構成されるマルチキャストグループに対するアドレスであり、この方法はフロー毎の経路制御を行うための本質的な解決とは言えない。

2.3.2 データリンク層による転送

近年、ATM ネットワークなどの高速データリンクの分野において、IP の転送機構をデータリンク層による転送機構と統合する研究が盛んに行われている [31, 21, 28, 12]。これらの研究は、

IP ソフトウェアによる転送を ATM スイッチによるハードウェア転送に置き換えることで、高速ネットワークにも対応できる高性能な IP ルータを実現するものである。

ATM スイッチの経路制御テーブルに相当する VC (Virtual Channel) テーブルは、IP 経路制御テーブルの情報を基にエントリが生成されるが、[28, 12] の機構では、VC テーブルのエントリの生成はフローに対して行われる。これは、ATM スイッチにおいてフロー毎の経路制御が行われていることに等しい。すなわち、データリンク層である ATM スイッチによる転送機構をうまく利用すれば、現在の IP の経路制御テーブルを変更することなくフロー毎の経路制御の実現が可能になると考えられる。

現在、IETF Multiprotocol Label Switching Working Group において、IP の転送機構とデータリンク層 (主に ATM) の転送機構の統合に関する議論が行われており、Label Switching [6, 35] の機構を用いた RSVP のフローのマッピング [9] の提案も行われている。データリンク層が Label Switching のアーキテクチャを持つ範囲においては、このアプローチは有効である。

また次世代インターネットアーキテクチャとして研究開発が進められている AMInet [49] においても、資源予約プロトコル ASP [50] によって保証型サービスを受けるフローを直接 ATM ネットワークの VC にマッピングする手法が採られている。

2.3.3 その他

次世代の IP である IP version 6 [10] では、IP ヘッダに新たに 24bit の Flow Label フィールドが設けられる。現在はフロー識別子として送信元アドレスと宛先アドレスの組などを利用しているが、この Flow Label により IP パケットに直接フロー識別子を指定できることになる。したがって、IP version 6 のパケット転送機構において、この Flow Label をエントリとする経路制御テーブルや経路制御アルゴリズムを組み込めば、IP によるフロー毎の経路制御が可能となる。しかし、Flow Label フィールドをどのように使うか、あるいはどのように割り当てを行うかなどの詳細については、まだ議論中の段階である。

フローに対応した経路制御テーブルについては重要な研究課題であるが、本研究では経路制御プロトコルに注目するため、これを研究の対象外とした。今後は、フローに対応した経路制御テーブルの機構が存在することを仮定して、議論を進めることにする。

第 3 章

経路制御プロトコル RRR の設計

本章では、第 2章における資源予約ネットワークアーキテクチャと経路制御機構に関する検討を踏まえて、本研究で提案する経路制御プロトコル RRR (Routing for Resource Reservation) の設計について述べる。

3.1 論点

まず RRR の設計についての論点をまとめる。

3.1.1 必要な機能

資源予約ネットワークにおいて、保証型サービスを提供するために必要な経路制御プロトコルの機能は、以下の通りである。

- フローベース経路制御

  保証型サービスはフローに対して提供されるものであり、個々のフローに対応した経路を取り扱う。

- QoS 要求の考慮

  フローの要求するトラフィックパラメータに適した経路の選択を行う。

- ネットワーク資源情報の利用

  経路制御情報としてネットワーク資源情報を利用し、動的なネットワーク状態の変化に対応した経路の決定を行う。

- 代替経路の提供

  選択した経路において、ネットワーク資源の不足により資源予約が失敗に終った場合、可能ならば代替経路を提供する。

3.1.2 問題点とアプローチ

2.2節で述べたように、QoS 経路制御を実現するための経路制御プロトコルの提案は極めて少ない。これは、以下のような理由によるものである。

- 規模拡張性に乏しい

  フローの QoS 要求に適した経路を選択するためには、利用可能なリンクの帯域幅やルータでの遅延時間などのネットワーク資源情報や資源予約情報など、従来の経路制御プロトコルに比べて膨大な量の情報が必要となる。またフローベースで経路制御を行うためには、宛先毎ではなくフロー毎に経路計算を行う必要が生じる。したがって、ネットワークの大きさやトラフィックに対する規模拡張性は期待できない。

- 経路の安定性に問題がある

  保証型サービスを提供するための経路制御プロトコルは、細粒度の状態依存型経路制御プロトコルである。経路制御情報の 1 つである資源予約情報は、フローの生成や消滅の度に更新される。経路制御情報の頻繁な更新は、経路の安定性に重大な影響をもたらすと考えられる。

- 経路の最適化が難しい

  フローが要求する QoS は、様々な要素が考えられ、どのパラメータを基準に (あるいは優先的に) 最適化を行うのが適当であるかは、個々のフローによって異なる。したがって、経路制御プロトコルとして一意な経路の最適化を適用するのは難しい。

これらの問題は、QoS 経路制御が抱える本質的な問題であり、全てを満足させるような解決は困難である。そこで本研究では、RRR の設計にあたって、運用規模、経路の計算方法、既存の経路制御プロトコルとの連携、ネットワーク資源情報把握のためのアルゴリズムを考慮し、実現可能な妥協点を求めることにする。

ネットワーク規模の限定

本質的に規模拡張性が乏しいのであるとするならば、どの程度までのネットワーク規模を考えるのが妥当なのであろうか。本研究では、RRR の運用規模として、あるネットワーク管理領域内 (Autonomous System, AS) の基幹ネットワーク (数百ノードまで) を 1 つの目安と考える。その理由は以下の通りである。

- ネットワーク管理領域間にまたがる QoS 経路制御は困難である

  ネットワーク管理領域間での経路制御情報の交換は、通常集約された情報しか交換されない。その理由は、交換する情報量を削減してオーバヘッドを少なくするほかに、ネットワークセキュリティ上の問題も存在するからである。経路制御情報はネットワーク管理上重要な情報であり、第三者に悪用されないとも限らない。QoS 経路制御には、ネットワークトポロジとネットワーク資源についての詳細な情報が必要となるが、上記の理由により管理領域間での詳細な経路制御情報の交換は難しい。また管理領域間の経路には、各々の管理領域のポリシに基づく経路制御が優先することが考えられる。したがって、ネットワーク管理領域間にまたがるような経路を考慮することは困難である。

- 基幹ネットワーク以外は冗長経路が少ない

  QoS 経路制御の利点は、現在のネットワーク資源の状態を把握して、複数の経路の候補の中から最適と思われる経路を選択できることである。しかし、冗長経路がまったくないようなネットワークトポロジにおいては、経路の選択はトポロジ情報のみがあれば十分であ

り、QoS 経路制御の機構は単なるオーバヘッドでしかない。そこでネットワーク管理領域内でも、幾つかの冗長経路が存在するような基幹ネットワークを対象として考えるのが妥当である。

既存の経路制御プロトコルとの連携と情報の共有

本研究では、RRR のオーバヘッド削減のため、既存の経路制御プロトコルとの連携と情報の共有を積極的に考える。

例えば、既存の経路制御プロトコルである OSPF は、リンクステート型の経路情報表現を扱っており、基本的なネットワークトポロジ情報を保持している。したがって、RRR が OSPF と経路制御情報を共有することで、RRR における経路制御情報の交換コストを削減することができると考えられる。

経路制御情報の共有は、RRR 導入のオーバヘッドの削減という利点の一方で、運用環境を限定してしまう欠点もある。しかし、RRR は既存の経路制御プロトコルを置き換えるものではない。RRR は、あくまで保証型サービスを必要とするフローに対して経路を提供することが目的であり、従来の最善努力型サービスのパケットに対しては、既存の経路制御プロトコルを用いることを想定している。したがってこのような連携を考慮することは、既存システムとの親和性の点で非常に重要である。

始点ノードにおける経路計算と経路配布

2.2節の検討において、QoS 経路制御には始点経路制御が有利であることを述べた。始点経路制御は、始点ノードで完全な経路の計算を行い、パケットに経路情報を含めることで宛先までの転送が行われる。しかし、インターネットにおけるパケットの大部分はホップバイホップ経路制御が行われており、資源予約を行うための資源予約プロトコルのメッセージもホップバイホップの経路制御により転送されることを前提としている。そこで、2.2節で取り上げた QOSPF の ER (Explicit Routing) の手法に注目する。ER では、始点ノードにおいて計算した経路を、経路情報として経路上のノードに配布する。これにより、始点ノードで計算した経路が経路上のすべてのノードで利用可能となる。つまり ER は、始点経路制御の利点を活かしつつ、ホップバイホップ経路制御に対応できる手法であると言える。そこで、RRR も ER と同様の手法を採り入れることにする。

ネットワーク資源情報の把握

QoS 経路制御プロトコルを設計する上での最大の問題は、正確なネットワーク資源情報の獲得と情報交換によるオーバヘッドのトレードオフをどのように考えるかである。動的に生成、消滅するフローに対応する正確なネットワーク資源情報を獲得するためには、短い周期で情報更新を行う必要がある。しかし、頻繁な情報更新は、情報交換のために余計な資源 (リンクの帯域幅や CPU 時間など) の消費を引き起こす。また経路の安定性の面からも悪影響を及ぼす。しかし逆に情報更新の頻度を下げると、実際のネットワーク資源情報の変化に追随できなくなり、適切な経路提供が困難となる。

では、どのようなアルゴリズムによりネットワーク資源情報の交換を行えば、現実的で且つ効果的な経路提供が行えるのであろうか。本研究では、ネットワーク資源情報の交換について、幾つかのアルゴリズムによる予備シミュレーションを行い、以下のような結論を得た。

- ネットワーク資源情報はフラッディングではなく、特定ホストに対して送信する方が規模拡張性に優れている。
- 一定時間毎の情報更新など時間に依存した情報更新よりも、ネットワーク資源の変化に応じた情報更新の方が効果的である。

RRR では、ネットワーク資源情報の把握に関して、予備シミュレーションに用いた Pickup Block のアルゴリズムを用いることにする。なおアルゴリズム及びシミュレーションの詳細については Appendix Aを参照されたい。

3.2 設計

3.2.1 概要

本研究では、経路制御プロトコル RRR を以下のような方針に基づき設計する。

- 資源予約プロトコルに対してフロー毎の経路を提供する。

  RRR は、あるフローに関する経路について、資源予約プロトコルの問い合わせに答えるという形で経路の提供を行う。また、経路制御テーブルは資源予約プロトコルが管理するものとし、RRR が提供する経路と実際にフローが転送される経路が同じであるかどうかについては、RRR は関知しない。

- リンクステート型の経路制御情報を用いて、**TOS** メトリックによる経路選択を行う。

  RRR の経路制御情報については、既存のリンクステート型の経路制御プロトコル OSPF との連携を行う。またネットワーク資源情報として OSPF の TOS メトリックを用いる。

- 経路の計算は始点ノードで行い、経路上のノードに対して経路情報を配布する。

  RRR における経路の計算は始点ノードのみで行う。しかし、通常の資源予約プロトコルは始点経路制御には対応していない。そこで ER と同様、始点ノードで計算した経路を経路上のノード、すなわち下流ノードに対して配布する。これにより、始点ノードで経路計算を行いつつ、始点ノード以外のすべての経路上のノードでも、資源予約プロトコルに対して経路の提供を行うことができる。

- 利用可能なネットワーク資源状態の変化に応じてネットワーク資源情報を交換し、経路選択時に考慮する。

  RRR では、利用可能なネットワーク資源の変化を把握するために “アラート (alert)” という概念を導入する。アラートとは、RRR が提供した経路において、フローの集中によるネットワーク資源の不足により、資源予約が失敗した状態を示す。資源予約プロトコルからアラートの通知を受けることで、RRR はネットワーク資源の不足が生じていることを把握し、経路の選択時に考慮する。

- 代替経路を提供する。

  RRR の選択した経路が、フローの要求するトラフィックパラメータを満たせずに資源予約が失敗した場合、RRR は資源予約プロトコルからの要求に応じて経路の再計算を行い、代替経路の提供を行う。

- **IP マルチキャスト経路を提供する。**

  IP マルチキャストに基づく、マルチキャスト経路の提供を行う。但し、動的なメンバシップの変更には対応していない。

以下、設計の詳細について解説する。

3.2.2 経路制御情報

RRR における経路制御情報の表現はリンクステート型で行う。本研究では、既存のリンクステート型経路制御プロトコルである OSPF に注目し、このプロトコルが持つ経路制御情報 OSPF トポロジデータベースを RRR の基本情報として利用することを考える。

ネットワークトポロジの表現

OSPF では、ネットワークトポロジを有向グラフにより表現する。図 3.1のようなサンプルネットワークをトポロジグラフで表現すると、図 3.2のようになる。グラフのノードは、ルータノード及びネットワークノードで構成される。ルータノードは OSPF デーモンが存在するノードを表し、ネットワークノードはサブネットを表す。リンクはルータノードを始点とするルータリンクと、ネットワークノードを始点とするネットワークリンクに分けられる。また、2 つのルータノードを結ぶリンクは、point-to-point ネットワークを表し、ルータノードとネットワークノードを結ぶリンクは、ルータがネットワークに対するインタフェースを持っていることを表す。

それぞれのノード及びリンクは、種類毎に一意となるような固有の識別子がつけられる。RRR では、これら基本的なネットワークトポロジの表現は OSPF と同じものを用いる。

メトリック

OSPF におけるメトリックは “コスト” という形で表現され、各ルータの出力側のインタフェース、すなわちルータリンクに付けられる。このコストはネットワーク管理者によって任意に設定される。通常 (リンクの帯域幅 [Mbps]) ÷ 100 で求められる整数値を指定することが多い。

2.2節の検討において、メトリックは基本的なネットワーク資源情報である帯域幅と遅延時間をそれぞれ表現することが望ましいとした。しかし、通常の OSPF のコストには明確な意味付けはなされておらず、このコストをそのまま RRR のメトリックとして用いることはできない。そこで、OSPF でサポートされている **TOS** メトリックを利用することにする (これに対して通常の OSPF のコストはデフォルトメトリックあるいは **TOS 0** メトリックと呼ばれる)。TOS メトリックとは、TOS (Type Of Service) すなわちサービスの種類毎に用いるメトリックのことである。TOS は、もともと IP パケットのヘッダフィールドの 1 つに定義されているもので、そのパケットがどのように扱われるべきか (どのようなサービスを受けたいか) を示す指標となるものである。現在では、IP パケットのヘッダフィールドとは別に [2] によって定められている (図 3.3)。OSPF では、4 ビットの TOS フィールドを 8 ビットのコードに変換して使用する (表 3.1)。

RRR では、このうち maximize throughput (8) と minimize delay (16) の TOS に対する TOS メトリックをそれぞれ帯域幅と遅延時間に対応するメトリックとして使用することにする。

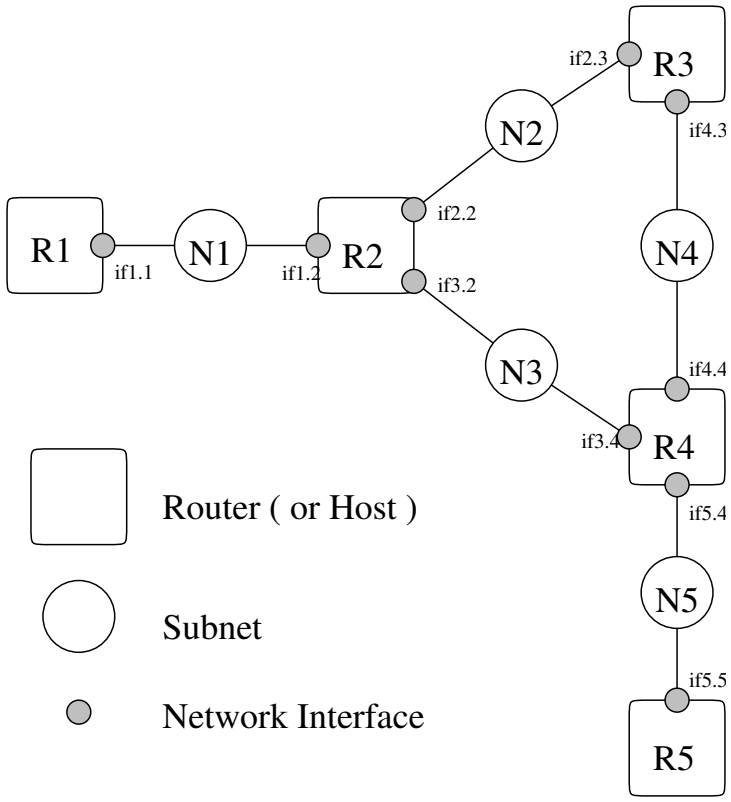


図 3.1: サンプルネットワーク

表 3.1: OSPF における TOS コード

OSPF encoding	RFC1349 TOS Values
0	0000 normal service
2	0001 minimize monetary cost
4	0010 maximize reliability
6	0100
8	0100 maximize throughput
10	0101
12	0110
14	0111
16	1000 minimize delay
18	1001
20	1010
22	1011
24	1100
26	1101
28	1110
30	1111

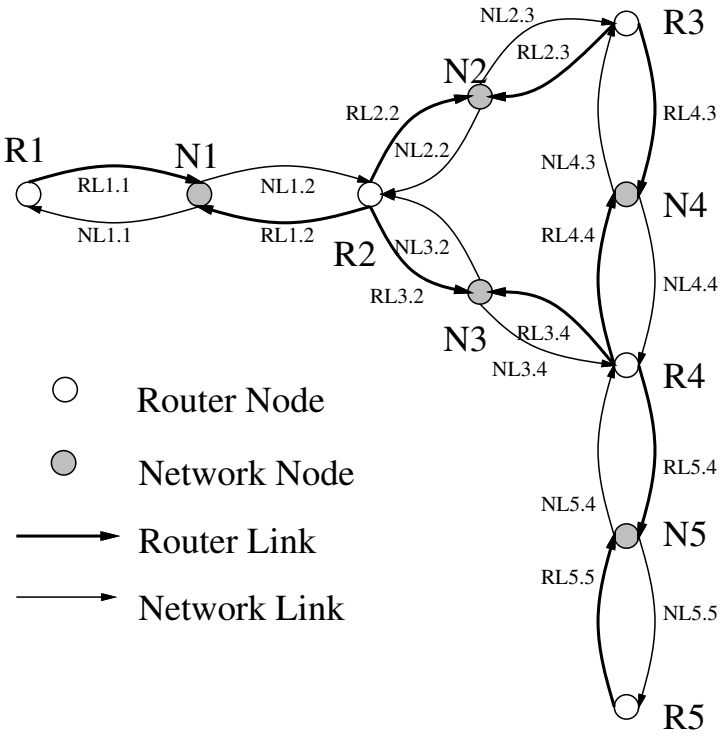


図 3.2: トポロジグラフによる表現

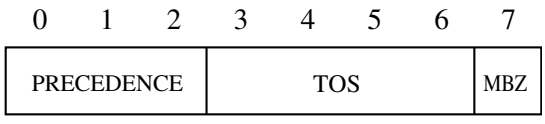


図 3.3: Type Of Service オクテット

エリア

OSPF では、1 つの AS をエリア (area) と呼ばれる複数の領域に分けることができる。1 つのエリアは隣接するサブネットやルータの集合で構成され、各エリアには、AS 内において一意なエリア識別子が付けられる。 OSPF デーモンは自ノードに関するトポロジ情報をエリア内のすべての OSPF デーモンに対して広告 (フラッディング) する。すなわち、エリア内のすべての OSPF デーモンは、エリア内に関する完全なトポロジ情報を保持することになる。一方、エリア外の情報は、複数のエリアに属するエリア境界ルータ (Area Border Router, ABR) の OSPF デーモンが、エリア外のトポロジ情報を集約した形で、エリア内の他の OSPF デーモンに広告する。集約された情報には、エリア外に存在するサブネットの IP アドレスとサブネットマスク、そしてそれに対応するメトリックだけが含まれる。すなわち、エリア外に関するトポロジ情報は隠蔽される。OSPF は、ネットワークをエリアに分割することで経路制御情報を集約し、規模拡張性を高めている。

したがって、あるルータ上の OSPF デーモンが保持する完全なトポロジ情報は、そのルータが属するエリア内だけの情報ということになる。すなわち RRR が得られるトポロジ情報もエリア内に限定される。しかし RRR で想定するネットワーク規模は、OSPF の単一エリアで十分対応できる規模であり、特に問題とはならないと考えられる。RRR においても OSPF と同様のエリアの概念を適用する。RRR は、1 つの OSPF のエリアと同一範囲内において有効な経路の提供を行う。

OSPF エリアには、一般的なエリアとエリア間を相互に接続するバックボーンエリア (識別子が 0 のエリア) の 2 つのレベルが存在する (図 3.4)。バックボーンエリアは、様々なエリアに属するエリア境界ルータの他に、AS との接続点である AS 境界ルータ (AS Border Router, ASBR) も含む。しかしいずれのエリアも、それぞれ単一のエリアとしての機能は同じであり、そのエリア内で交換されるトポロジ情報も全く同じものである。したがって RRR は、OSPF のいずれのエリアにおいても全く同様に動作する。但し OSPF と異なり、RRR においてエリア間の相互作用はなく、それぞれの RRR エリアは独立である。

RRR リンクステートデータベース

これまでに挙げた OSPF の様々な経路制御情報は、OSPF デーモンの OSPF トポロジデータベースに保持されている。RRR は、OSPF デーモンから OSPF トポロジデータベースの情報を得て、RRR リンクステートデータベースという形に再構成し保持する。RRR リンクステートデータベースは、以下の 3 種類から構成される (図 3.5)。

- ルータリンクステート ··· ルータノードに関する情報を示す。

  ルータリンクステート ： = { ルータノード識別子, (リンク情報) × 接続リンク数 }
  リンク情報 ： = { ルータリンク識別子, インタフェースの IP アドレス,
  接続先ノードの種類 (ルータノード | ネットワークノード),
  接続先ノード識別子,
  メトリック (TOS 0, TOS 1, TOS 2, ... TOS n) }

- ネットワークリンクステート ··· ネットワークノードに関する情報を示す。

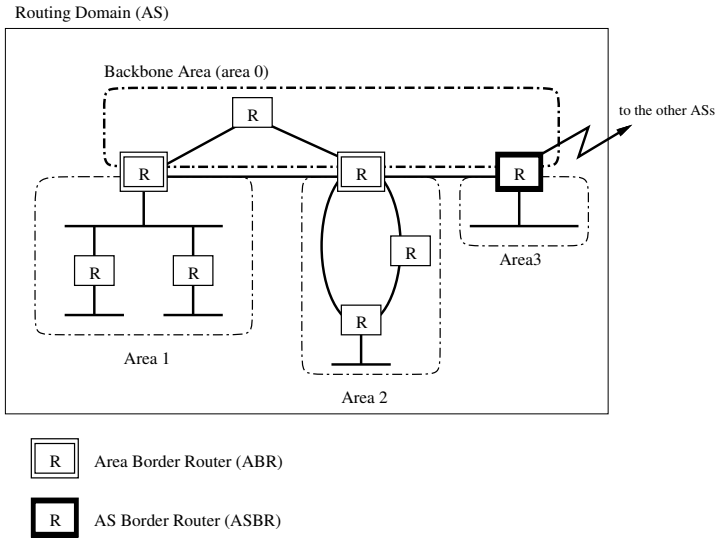


図 3.4: OSPF エリア

ネットワークリンクステート ：= { ネットワークノード識別子,
サブネット IP アドレス, サブネットマスク,
(リンク情報) × 接続リンク数 }
リンク情報 ：= { ネットワークリンク識別子, 接続先のルータノード識別子 }

- サマリリンクステート ··· エリア外及び AS 外の情報を示す。
  サマリリンクステート ：= { サブネット IP アドレス, サブネットマスク,
  メトリック (TOS 0, TOS 1, TOS 2, ... TOS n) }

3.2.3 経路の提供

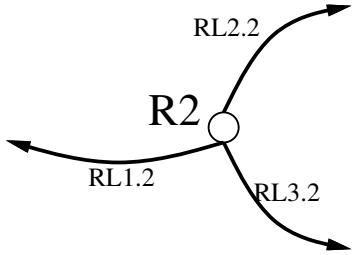
RRR における経路は、資源予約プロトコルからの問い合わせに答える形で提供される。経路の提供は、5 つのフェーズに分けられる (図 3.6)。

1. 資源予約プロトコルからの経路要求
2. 始点ノードに対する経路要求
3. 始点ノードにおける SPF ツリーの計算及び経路の決定
4. 経路上のノード (下流ノード) への経路情報の配布及び RRR 経路エントリの作成
5. 確認応答と資源予約プロトコルへの経路提供

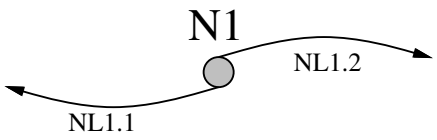
以下、それぞれのフェーズを順に解説する。

資源予約プロトコルからの経路要求

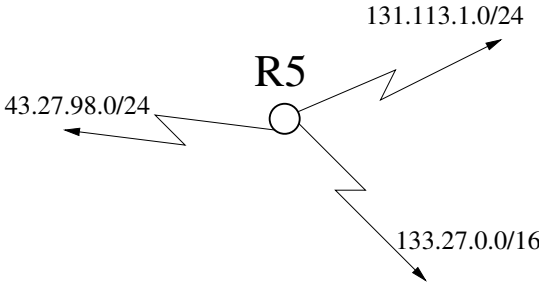
RRR デーモンは、資源予約プロトコルから経路の提供に必要な以下の情報を得る。



(a) Router Link State



(b) Network Link State



(c) Summary Link State

図 3.5: RRR リンクステートデータベース

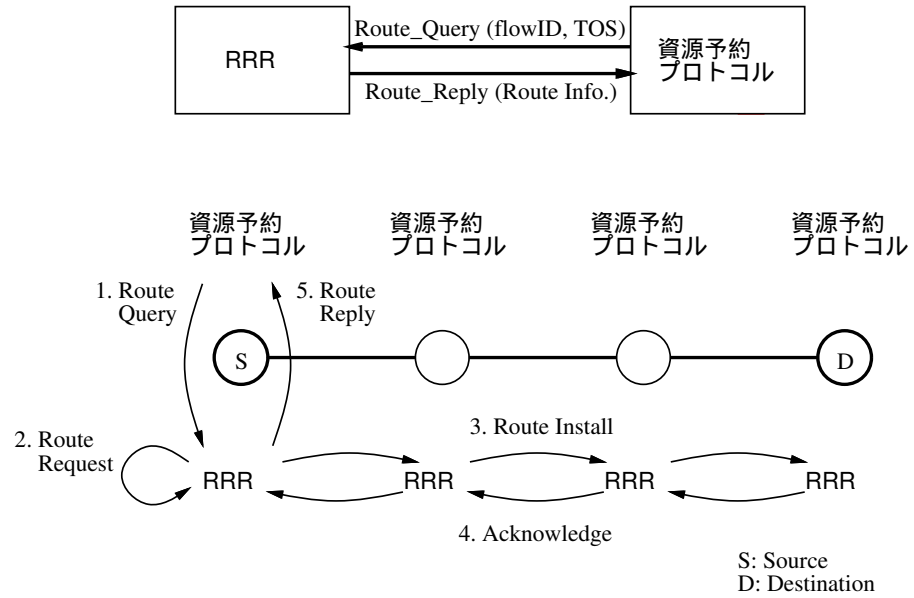


図 3.6: 経路の提供

- フローの始点アドレス及び終点アドレス

  それぞれ 32 ビットの IP アドレスにより指定する。

- フロー識別子

  そのフローを特定する一意な識別子である。RRR では以下のように定義する。

  フロー識別子 ::= { 始点 IP アドレス, 終点 IP アドレス, 始点ポート番号, 終点ポート番号 }

- フローの要求するトラフィックパラメータ

  RRR では、トラフィックパラメータを TOS コードにより指定する。

始点ノードに対する経路要求

RRR における経路計算は始点ノードで行う。したがって、資源予約プロトコルデーモンから問い合わせを受けた RRR デーモンは、まず要求のあったフローの始点アドレスのノードに対して、経路計算を要求するメッセージを送信する。始点アドレスがエリア外である場合は、その始点アドレスについてのサマリリンクステートを広告しているノード (ABR または ASBR) に対して要求を行う。もしある始点アドレスについて、複数のノードがサマリリンクステートを広告している場合、そのサマリリンクステートのメトリックで、フローの要求する TOS に対応した TOS メトリックの一番小さいノードを選択する。

始点ノードにおける SPF ツリーの計算及び経路の決定

経路計算の要求を受けた始点ノードの RRR デーモンは、Dijkstra のアルゴリズムによって SPF ツリーを計算し、経路を決定する。このときに用いるメトリックは、資源予約プロトコルが指定した TOS に対応した TOS メトリックである。宛先となる終点ノードがエリア外にある場

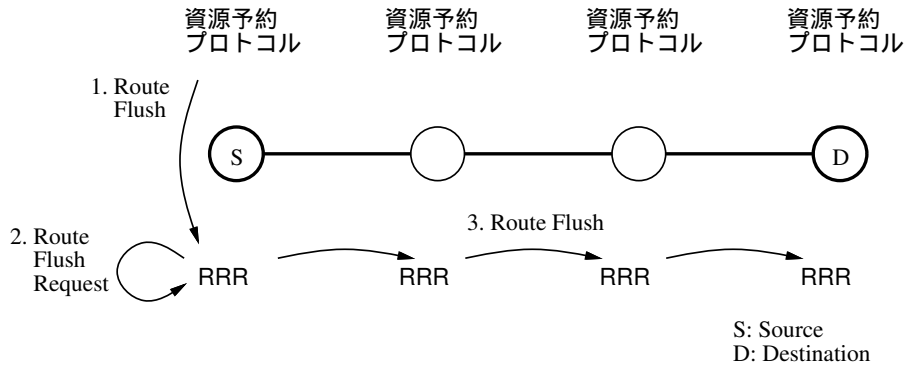


図 3.7: 経路の消去

合は、その終点アドレスについてのサマリリンクステートを広告しているノード (ABR または ASBR) を選択する。もし同じ終点アドレスについて、複数のノードがサマリリンクステートを広告している場合、そのサマリリンクステートのメトリックで、フローの要求する TOS に対応した TOS メトリックの一番小さいノードを選択する。

経路上のノード (下流ノード) への経路情報の配布及び **RRR** 経路エントリの作成

経路が決定すると、RRR デーモンは、その経路に沿って経路情報メッセージを配布する。経路情報メッセージを受け取った RRR デーモンは、RRR 経路エントリを作成する。RRR 経路エントリは、RRR が内部に持つフロー毎の経路制御テーブルである。

RRR 経路エントリ ::= { 配布元ノード (始点ノード) 識別子, (フロー情報),
上流ノード識別子, 出力インタフェース, 下流ノード識別子,
フラグ }
フロー情報 ::= { フロー識別子, 始点アドレス, 終点アドレス, TOS コード }

確認応答と資源予約プロトコルへの経路提供

終点ノードで経路情報メッセージを受け取った RRR デーモンは、始点ノードに対して確認応答メッセージを返す。確認応答メッセージは、ちょうど経路情報メッセージの逆順をたどる経路で転送される。

確認応答を受け取ったフローについて、資源予約プロトコルから問い合わせを受けていた RRR デーモンは、資源予約プロトコルに対して経路を提供する。RRR 経路エントリは、そのフローに対する経路消去が行われるまで保持される。

3.2.4 経路の消去

RRR では明示的に経路の消去を行う。あるフローについての経路消去の要求を受けた RRR デーモンは、以下の手順で経路の消去を行う (図 3.7)。

1. フローの識別子を基に RRR 経路エントリから該当するエントリを検索し、経路情報の配布元ノード識別子を得る。

2. 経路情報の配布元のノードに対して経路消去要求のメッセージを送信する。

3. 経路消去の要求を受けた配布元のノードの RRR デーモンは、経路の配布と同様の方法で経路に沿って経路消去メッセージを終点ノードへ向けて転送する。

4. 経路消去メッセージを受け取った RRR デーモンは、RRR 経路エントリから該当エントリを消去する。

RRR では、資源予約プロトコルがあるフローに対する経路が不要と判断した時点で、そのフローの経路消去を RRR デーモンに伝えることを想定している。どのような場合において経路が不要かの判断については、資源予約プロトコルの決定することであり RRR は関知しない。しかし、どのような判断であっても経路の消去を行うことは可能である。

3.2.5 ネットワーク資源情報の把握と代替経路の提供

アラート (alert)

3.1節の検討において、RRR におけるネットワーク資源情報の把握は、予備シミュレーションにおける Pickup Block のアルゴリズムを用いることとした。Pickup Block のアルゴリズムでは、フローの集中により資源不足の状態に陥ったリンクについて、資源予約要求の失敗をトリガーとしてネットワーク資源情報の交換を行う。

RRR では、資源予約要求が失敗した状態をアラート (alert) と呼ぶことにする。そして、アラートが発生したリンクの情報をアラート情報として通知する。アラート情報を受け取った RRR デーモンは、これを RRR リンクステートデータベースに記録し、経路選択の際に考慮するものとする。

アラート情報の伝達と処理

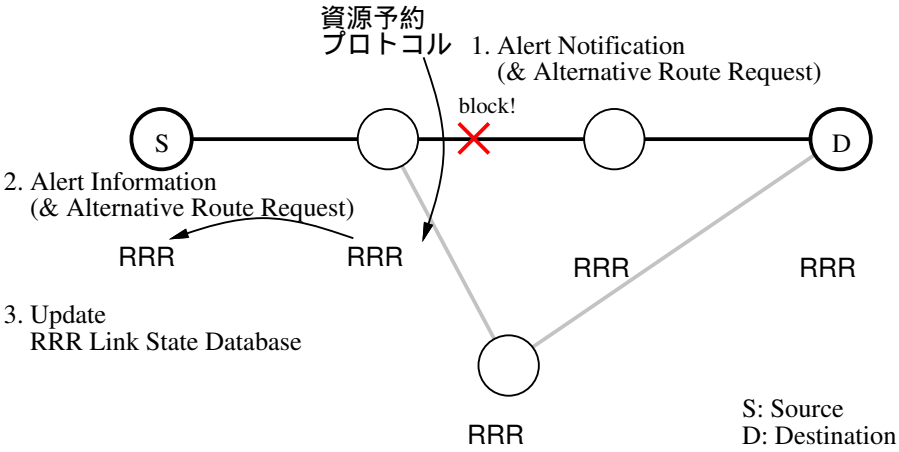
さてアラートが発生した場合、その情報をどのノードに伝えるべきだろうか。素直に考えるのであれば、エリア内の全てのノードの RRR デーモンに広告 (フラッディング) し、アラートが発生したネットワークを回避するような経路選択を促すのが妥当であると言える。しかしこのような措置を取った場合、すべてのノードの RRR デーモンがアラートが発生したネットワークを回避するような経路を選択することになり、反対に迂回経路が混雑し始める。やがて迂回経路においてアラートが発生し、また別の経路が選択される。このような経路の不安定性や振動 (oscillation) の問題は [43] などでも指摘されており、予備シミュレーションにおいても同様の結果を示している。

そこで RRR では、全ての RRR デーモンに伝えるのではなく、アラート発生の要因となった経路を選択した RRR デーモン、すなわちその経路の始点ノードの RRR デーモンに対してのみ、アラート情報を伝えることにする。

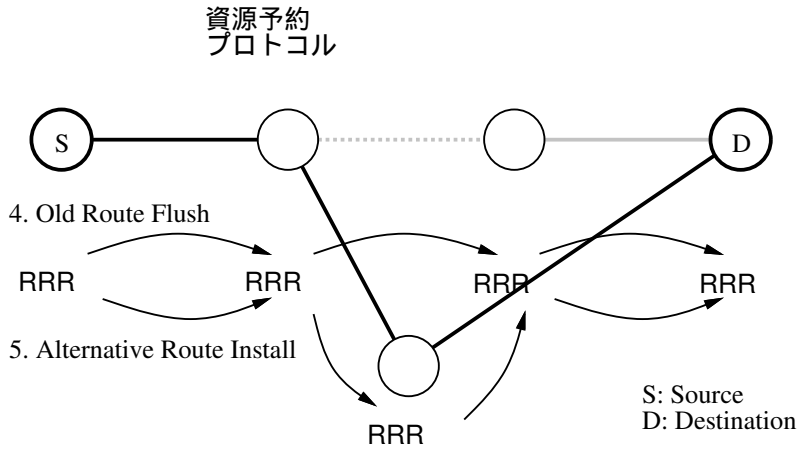
アラート発生の際の伝達と処理は、以下のように行われる (図 3.8 (a))。

1. 資源予約プロトコルは、RRR デーモンにアラート発生をフロー識別子とともに伝える。

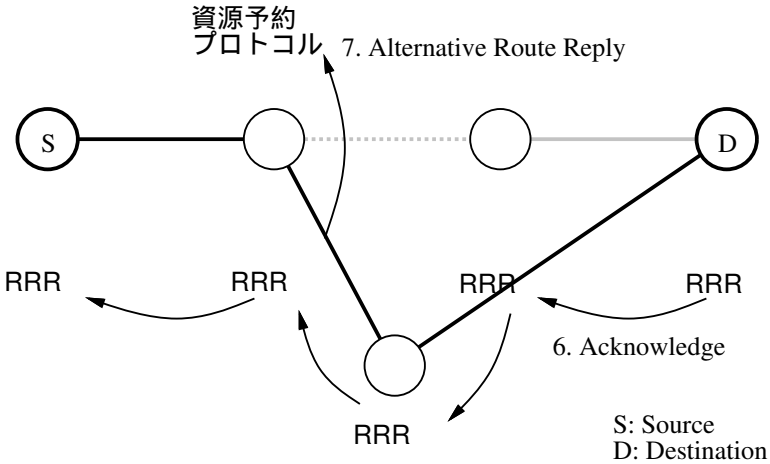
2. 資源予約プロトコルからアラート発生の通知を受けた RRR デーモンは、RRR 経路エントリからそのフロー識別子に対応するエントリを検索し、その経路情報の配布元のノード (始点ノード) に対してアラート情報を送信する。



(a) アラート情報の伝達と代替経路要求



(b) 代替経路の提供 – 1



(b) 代替経路の提供 – 2

図 3.8: アラート発生と代替経路の提供

3. アラート情報を受け取った RRR デーモンは、RRR リンクステートデータベースからアラートの発生したノードのルータリンクステートを検索し、経路に該当するルータリンクがアラートの状態であることをマークする。

このマークは、アラート情報を受け取ってから一定期間保持される。

アラート発生時の経路選択

アラートマークのついたリンクは、全てのメトリックを最大値にして SPF ツリーを計算する。すなわち RRR は、アラートの状態にあるリンクを回避するような経路選択を行う。

代替経路の提供

代替経路の提供が必要な状況とは、すなわちアラートが発生した場合である。したがって代替経路の提供も、アラートの発生と同様の手順により行う (図 3.8 (a)(b)(c))。

1. 資源予約プロトコルは、RRR デーモンにアラート発生と代替経路の要求をフロー識別子とともに伝える。

(2. ~ 3. はアラート発生と同様の処理)

4. RRR デーモンは該当フローに対する経路を消去し、経路の再計算を行い、新しい経路情報を配布する。

経路の再計算を行う際、アラート情報によって該当リンクのメトリック変更が行われるため、結果的に (もし冗長経路があれば) 代替経路が提供できることになる。

3.2.6 資源予約プロトコルとのインタフェース

RRR では、資源予約プロトコルとのインタフェースとして RSVP でサポートされている RSRR (Routing Support for Resource Reservations) [45] を利用する。しかし RSVP は図 2.2 の RRPD に相当する機能を全て持っているわけではないので、RRPD として RSVP を想定しているわけではない。しかし RRR は、RSVP に対しても RSRR による経路提供を行うこともできるよう考慮されている。

RSRR でサポートされている主なメッセージは以下の通りである。

- Reservation Daemon → Routing Daemon

  Initial_Query : = { (なし) }
  →経路に使用するインタフェースを問い合わせる
  Route_Query : = { 始点アドレス, 終点アドレス }
  →経路の問い合わせを行う

- Routing Daemon → Reservation Daemon

  Initial_Reply : = { インタフェースの数, 全インタフェースリスト }
  → Initial_Query に対する返答
  Route_Reply : = { 始点アドレス, 終点アドレス,
  入力インタフェース, 出力インタフェースリスト }
  → Route_Query に対する返答

これらの経路の要求及び提供のメッセージは、フロー識別子に対応しておらず、基本的にRSVPで使用するマルチキャスト経路の問い合わせを想定したものである。また、経路の消去やアラート発生の通知や及び代替経路要求のためのメッセージはない。そこで、RRRではRSRRを独自に拡張して、新たに以下のようなメッセージを設ける。

- Reservation Daemon → Routing Daemon

  Route_Query_Flow ::= { フロー識別子, TOSコード }
  →フローに対する経路の問い合わせを行う
  Route_Flush ::= { フロー識別子 }
  →経路の消去を行う
  Route_Alert ::= { フロー識別子, 代替経路要求フラグ }
  →アラート発生の通知及び代替経路の要求を行う

- Routing Daemon → Reservation Daemon

  Route_Reply_Flow ::= { フロー識別子, 入力インタフェース, 出力インタフェース数,
  (出力インタフェース情報) × 出力インタフェース数 }
  出力インタフェース情報 ::= { 出力インタフェースアドレス, 転送先アドレス }
  →フローに対する経路の問い合わせに答える

資源予約プロトコルデーモンは、同じノード上に存在するローカルなRRRデーモンに対してRSRRメッセージを送信し、RRRデーモンもRSRRメッセージにより返答を行う。

なおRSRR拡張メッセージのフォーマットはAppendix Cに記載した。

3.2.7 マルチキャストについて

RRRでは、IPマルチキャストによるマルチキャスト経路を提供する。

マルチキャストグループ情報

RRRは、IGMP (Internet Group Management Protocol) [16]を用いて独自にIPマルチキャストグループ情報(どのマルチキャストアドレスのメンバが存在するか)を収集する。収集した情報は、マルチキャストグループ情報としてエリア内の全てのRRRに広告(フラッディング)される。マルチキャストグループ情報は以下の内容で構成される。

マルチキャストグループ情報 ::= { 広告ノード識別子, IPマルチキャストアドレス,
インタフェース }

これらの情報は、RRRデーモンのRRRグループデータベースに保持される。RRRグループデータベースの内容は以下の通りである。

RRRグループデータベース ::= { IPマルチキャストアドレス,
(マルチキャストグループ情報) × 広告ノード数 }

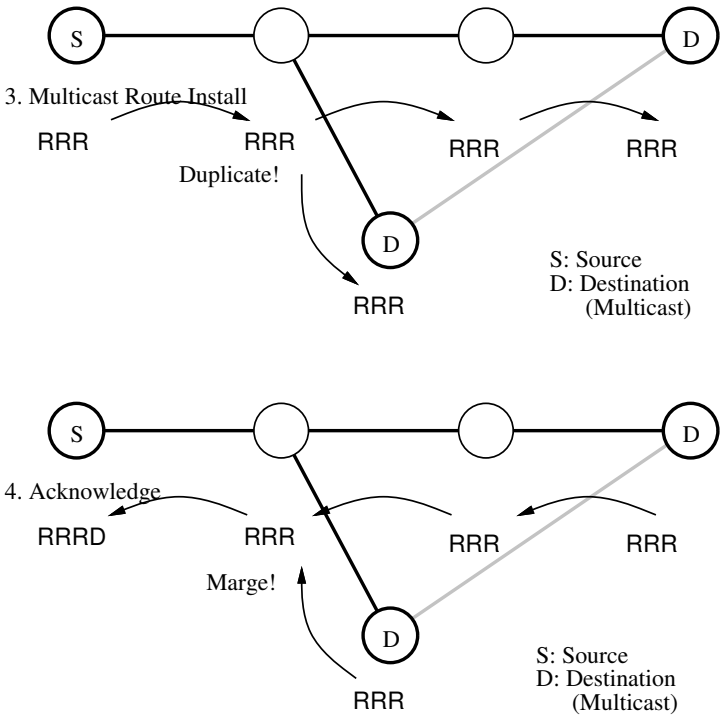


図 3.9: マルチキャスト経路の提供

経路の計算と配布

経路の計算及び経路の配布については、以下の通りである (図 3.9)。

1. まず始点ノードにおいて SPF ツリーを構築する。
2. RRR グループデータベースに基づき、終点アドレスに指定されたマルチキャストアドレスのグループに参加しているノードだけのツリーに枝刈り (pruning) を行い、経路を決定する。
3. マルチキャスト経路に沿って、すなわちツリーのルート (root) ノードからリーフ (leaf) ノードに向けて経路情報の配布を行い、各ノードで RRR 経路エントリを作成する。経路が枝別れしているノードでは、経路情報メッセージをそれぞれの枝に対して複製して転送する。
4. 経路情報メッセージがリーフノードまで到達したら、ルートノードに対して確認応答メッセージを返す。経路が枝別れしているノードでは、それぞれの枝からの確認応答メッセージを統合して、1 つの確認応答メッセージとしてルートノードに送る。

RRR はフロー毎の経路提供を行うことから、あるマルチキャストアドレスに対する経路は、始点アドレス毎のツリーの集合で構成されることになる (マルチキャストグループによるツリーの共有は行わない)。

マルチキャスト経路における制限

RRR は始点ノードからの経路情報の配布によって、各ノードの RRR 経路エントリを生成するため、RRR が提供するマルチキャスト経路は、その経路を生成した時点におけるメンバに対する経路となる。したがって、経路を配布した後で新たなノードがマルチキャストグループに加わった場合や、マルチキャストグループの参加ノードが離脱した場合など、動的なメンバシップの変更に対する経路の変更ができない。この場合、一旦そのマルチキャスト経路を消去して、再度マルチキャスト経路の計算と配布を行うことになる。マルチキャストアドレスを用いて定期的に資源予約プロトコルのメッセージを転送している RSVP などでは、問題が起きる可能性がある。この制限に関しては、5.3節で詳しく述べる。

本研究では、本章で設計した経路制御プロトコル RRR を、経路制御デーモン RRRD (RRR Daemon) として実装を行った。次章では、RRRD について述べる。

第 4 章

経路制御デーモン RRRD の実装

本章では、経路制御プロトコル RRR を実装した経路制御デーモン **RRRD (RRR Daemon)** について、その詳細を述べる。

4.1 概要

本研究では、RRRD を 4.4BSD Lite ベースの FreeBSD* 2.2.1-Release 上に構築した。コードは全て C で書かれており、ネットワーク API としてバークレーソケットインタフェースを利用している。RRR では OSPF のトポロジデータベースを利用する関係から、OSPF デーモンとして University of Michigan の Merit GateDaemon Project†が実装した経路制御デーモン GateD Release 3.5.5 を一部改造して使用している。また、資源予約プロトコルとのインタフェースである RSRR は、USC の ISI‡が実装した RSVP デーモン rsvpd Release 4.2a1 を参照した。なお RRRD を含む資源予約ネットワークの経路制御機構の実装アーキテクチャを図 4.1に示した。

RRRD の機能モジュール

RRRD は、機能別に 4 つのモジュールから構成されている (図 4.2)。

- RRR モジュール
- OSPF モジュール
- IGMP モジュール
- RSRR モジュール

これらのモジュールは、外部からのメッセージの到着、他のモジュールからの呼び出し、イベントスケジューラからの呼び出しによって起動される。次節から、RRRD の機能及び動作についてモジュール別に解説する。

*FreeBSD Inc. `http://www.freebsd.org/`
†Merit GateD Cosortium `http://www.gated.org/`
‡USC Information Sciences Institute RSVP Project `http://www.isi.edu/rsvp/`

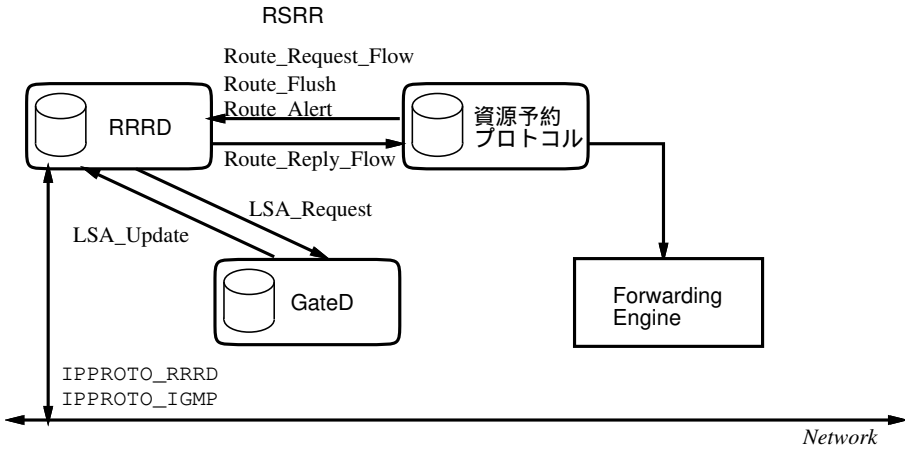


図 4.1: 資源予約ネットワークの経路制御機構の実装アーキテクチャ

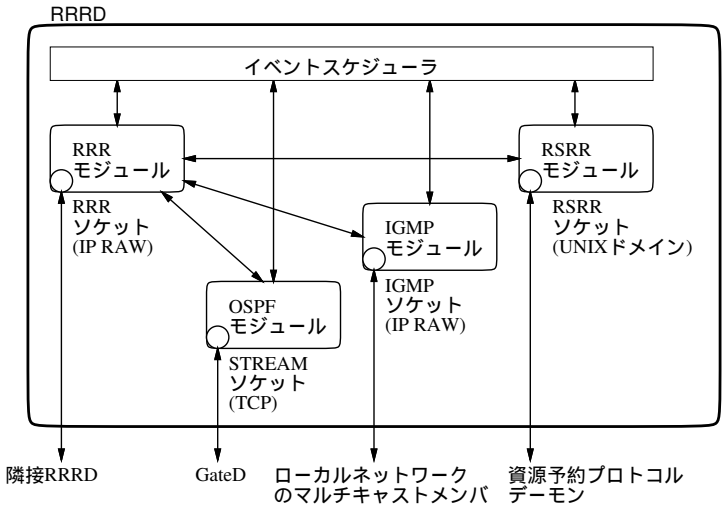


図 4.2: RRRD の機能モジュール

4.2 RRR モジュール

RRR モジュールは RRRD の中でも根幹を成すモジュールである。RRR メッセージを用いて他の RRRD と通信を行い、以下のような役割を担う。

- 隣接 RRRD の監視

  隣接 RRRD と生存 (Hello) メッセージを交換し、正しく起動しているか、あるいは正しく通信が行うことができるかを確認する。

- RRR リンクステートデータベースの管理

  OSPF モジュールから得た OSPF トポロジデータベースや、他の RRRD と交換するアラート情報を基に、RRR リンクステートデータベースを構成する。

- 経路の生成、消去

  他の RRRD と経路情報を交換し、RRR 経路エントリに経路を生成、消去する。

- RRR グループデータベースの管理

  他の RRRD とマルチキャストグループ情報を交換し、RRR グループデータベースに登録、削除する。

4.2.1 起動時の動作

起動時には、以下の手順により初期設定及び各データベースの初期化を行う。

1. 設定ファイルの読み込み
2. RRR リンクステートデータベースの初期化
3. デフォルトツリーの作成
4. RRR ソケットの作成
5. RRR 経路エントリの初期化
6. RRR グループデータベースの初期化
7. タイマの起動

設定ファイルの読み込み

はじめに RRRD 設定ファイル (図 4.3) を読み込んで以下の情報を得る。

- エリア識別子

  RRR のエリア識別子は、OSPF と同様 32 ビットの 2 進数で表現される。このエリア識別子を用いて、GateD から OSPF トポロジデータベースを得る。したがって、RRR のエリア識別子は、OSPF のエリア識別子と同じになるように設定する。

```
#
# サンプルネットワークの R2 における RRRD 設定ファイルの例
#
#<書式>
#ospfarea       エリア識別子                          # エリアの設定
#interface      IP アドレス       サブネットマスク    # インタフェースの設定
#neighbor       IP アドレス                           # 隣接 RRRD の設定
#(neighbor は対応する interface の設定の直後に書く)

ospfarea        172.16.0.0

interface       172.16.32.84    255.255.255.0     # if1.2
                neighbor        172.16.32.86      # R1 - if1.1
interface       172.16.33.84    255.255.255.0     # if2.2
                neighbor        172.16.33.83      # R3 - if2.3
interface       172.16.34.84    255.255.255.0     # if3.2
                neighbor        172.16.34.82      # R4 - if3.4
```

図 4.3: RRRD 設定ファイルの例

- インタフェース情報

  RRR で使用するインタフェースを登録する。インタフェース情報は、IP アドレス、サブネットマスク、隣接 RRRD の IP アドレスから構成される。登録しないインタフェースや隣接 RRRD については単に無視される。

RRR リンクステートデータベースの初期化

設定ファイルから得たエリア識別子を OSPF モジュールに渡し、OSPF トポロジデータベースを得て、RRR リンクステートデータベースを構築する。

デフォルトツリーの作成

初期化された RRR リンクステートデータベースに基づき、デフォルトツリーを作成する。デフォルトツリーとは、TOS 0 メトリックによって作成される SPF ツリーのことで、経路要求メッセージや経路消去要求メッセージなど一部の RRR メッセージの転送経路に用いられる。

RRR ソケットの作成

隣接 RRRD との通信は、RRR ソケットを介して行われる。RRR ソケットは IP RAW ソケットを使用し、プロトコル番号は 190 番を割り当てている。

RRR 経路エントリの初期化

RRRD は起動時に、エリア内のすべての RRRD に対して、自ノードが配布した経路情報による RRR 経路エントリに関して削除を要求する。これは、RRRD の一時的な DOWN による RRR

経路エントリの混乱を防ぐためである。通常、ある RRRD が DOWN したことを隣接 RRRD が検知すると、DOWN した RRRD が配布したすべての経路について経路消去が行われる。しかし、もし一時的な DOWN を隣接 RRRD が検知できなかった場合、DOWN した RRRD によって配布された経路情報による古い経路エントリがいつまでも残り、経路情報の一貫性が保てなくなる。そこで、起動時に全ての隣接 RRRD に対して特別な経路消去メッセージ (後述) を送信することで、明示的に古い経路エントリの削除を行う。

RRR グループデータベースの初期化

経路エントリ同様、RRR グループデータベースに関しても一貫性を保つために、起動時にエリア内の全ての RRRD に対して、自ノードが広告したマルチキャストグループ情報に関して削除を要求する。また同時に、隣接 RRRD に対して保持しているグループデータベース情報を自ノードに送るように要求する。これにより、現在エリア内に存在する最新のマルチキャストグループ情報を得る。これらの要求は、起動時に全ての隣接 RRRD に対して、特別なグループ情報メッセージ (後述) を送信することで行う。

タイマの起動

RRR モジュールで使用する Hello タイマ及び隣接 RRRD 状態更新タイマをイベントスケジューラに登録する。

4.2.2 イベントタイマ , RRR メッセージ

起動時の一連の処理が終ると、RRR モジュールは、RRR メッセージの到着、他のモジュールからの呼び出し、またはタイマによるイベントスケジューラからの呼び出しまで待機状態に入る。

イベントタイマ

RRR モジュールは 2 種類のイベントタイマを使用する。

- Hello タイマ

  生存 (Hello) メッセージを隣接 RRRD に対して送信するためのタイマ。現在、Hello タイマは 30 秒に設定されている。

- 隣接 RRRD 状態更新タイマ

  隣接 RRRD の状態 (UP もしくは DOWN) の更新を行うためのタイマ。現在は 10 秒に設定されている。

RRR メッセージ

RRR モジュールが他の RRRD と交換するメッセージは、以下の 7 種類である。

- 生存 (Hello) メッセージ
- 経路要求/経路消去要求/代替経路要求メッセージ
- 経路情報メッセージ

- 確認応答メッセージ
- 経路消去メッセージ
- グループ情報メッセージ
- アラートメッセージ

またこの他に、GateD から RRRD に対して OSPF トポロジデータベース更新の通知を行う特別なメッセージがある。

- OSPF トポロジデータベース更新メッセージ

RRR メッセージの詳細なフォーマットについては、Appendix Bに記載した。

4.2.3 隣接 RRRD の監視

RRRD では、隣接 RRRD が動作しているか、あるいは隣接 RRRD と正しく通信を行うことができるかを確認するために、Hello プロトコルを用いて監視を行う。

生存 (Hello) メッセージ

Hello タイマによってイベントスケジューラに呼び出された RRR モジュールは、生存 (Hello) メッセージを生成して、すべての隣接 RRRD に送信する。

生存 (**Hello**) メッセージ　: =　{ Hello 間隔, 最大生存時間 }

隣接ノードから生存メッセージを受け取った RRRD は、メッセージを受け取った時刻及び最大生存時間を記録する。

隣接 RRRD の状態更新

隣接 RRRD の状態の更新は、生存メッセージを受け取ったとき、または隣接 RRRD 状態更新タイマによってイベンスケジューラから呼び出されたときに行われる。更新時には、各々の隣接 RRRD について、最後に生存メッセージを受け取った時刻と現在時刻を比較する。そして、その間隔が最大生存時間より短ければ UP 状態、長ければ DOWN 状態となる。隣接 RRRD の状態遷移図を図 4.4に示した。状態が遷移したとき、以下のような処理が行われる。

- DOWN → UP

  隣接 RRRD が UP に遷移したときには、デフォルトツリーの再計算を行う。また、Hello タイマによる呼び出しを待たずに、直ちに生存メッセージを隣接 RRRD に返す。

- UP → DOWN

  隣接 RRRD が DOWN に遷移したときも、デフォルトツリーの再計算を行う。そして、RRR 経路エントリをチェックし、DOWN したノードが配布した経路、もしくは DOWN したノードを通過する経路が存在する場合、経路消去メッセージによって経路の消去を行う。このときに経路消去メッセージのタイプフィールドを用いて転送方向を指定する。DOWN した

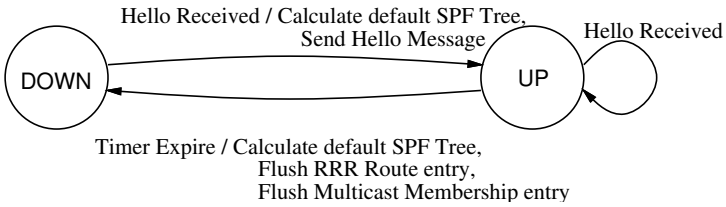


図 4.4: 隣接 RRRD の状態遷移図

ノードが経路の下流ノードにあたる場合は、経路の上流方向に経路消去メッセージを転送する。逆に、DOWN したノードが経路の上流にあたる場合は、経路の下流方向に転送する。また、DOWN したノードが配布したマルチキャストグループ情報に関しても消去を行うため、タイプフィールドが Flush のグループ情報メッセージをエリア内のすべての RRRD に対してフラッディングする。

4.2.4 OSPF トポロジデータベースとの同期

RRR リンクステートデータベースの基本となる OSPF トポロジデータベースは、GateD によって適宜更新されるため、同期を取る必要がある。そこで、GateD の OSPF トポロジデータベースの更新処理の中に、RRRD に対して RRR メッセージを送信し、OSPF トポロジデータベース更新の通知を行うように変更を加えた。

OSPF トポロジデータベース更新メッセージ

OSPF トポロジデータベースの更新があった場合、GateD は以下のメッセージによって RRRD に通知する (図 4.5の 1)。

OSPF トポロジデータベース更新メッセージ ::= { エリア識別子 }

このメッセージを受け取った RRR モジュールは、メッセージ中のエリア識別子と RRR エリア識別子を比較する。同じであるならば、OSPF モジュールを呼び出す (図 4.5の 2)。そして OSPF モジュールから新しい OSPF トポロジデータベースを得て、RRR リンクステートデータベースを更新する (図 4.5の 5, 6)。

4.2.5 経路の生成

経路の生成は、経路要求メッセージ、経路情報メッセージ、確認応答メッセージの 3 種類のメッセージによって行われる。

経路要求メッセージ

RSRR モジュールからフローについての経路の要求を受けた RRR モジュールは、経路要求メッセージを生成する。

経路要求メッセージ ::= { 経路要求ノード, 宛先ノード, フロー識別子, TOS コード }

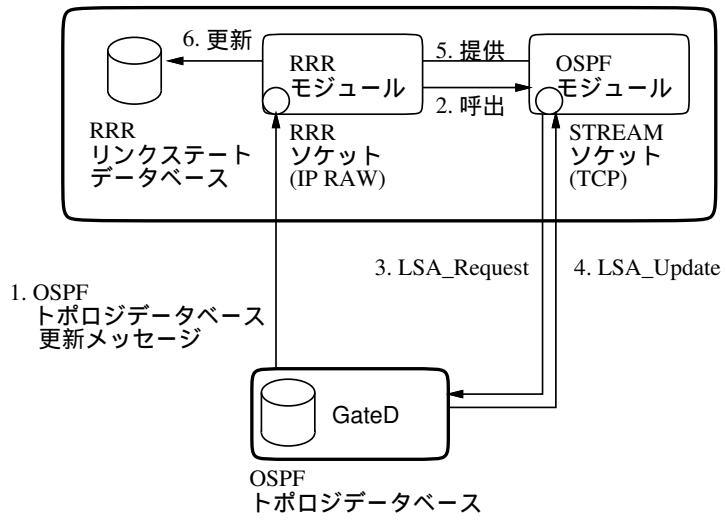


図 4.5: OSPF トポロジデータベースとの同期

生成された経路要求メッセージの宛先ノード (すなわち経路の始点ノード) は、RRR リンクステートデータベースを以下のような手順で検索を行い、決定する。

1. ルータリンクステートの検索

   フローの始点アドレスとルータリンクステートが持つインタフェースアドレスが同じであれば、そのルータノードを宛先ノードとする。

2. ネットワークリンクステートの検索

   まずフローの始点アドレスに、ネットワークリンクステートが持つネットマスクを適用する。そして、ネットワークリンクステートと同じサブネットアドレスであれば、そのネットワークノードを宛先ノードとする。

3. サマリリンクステートの検索

   まずフローの始点アドレスに、サマリリンクステートが持つネットマスクを適用する。そして、同じサブネットアドレスを持つサマリリンクステートの中で、フローが要求する TOS に対する TOS メトリックが最も小さいものを広告したルータノードを宛先ノードとする。

4. 1.~3. で適合しない場合、RRR リンクステートデータベースには経路要求を受けたフローの始点アドレスに関する到達情報がないことになる。したがって、この時点で処理を中断し、RSRR モジュールに対してエラーを通知する。

ここで宛先がネットワークノードとなる場合とは、始点アドレスのホストがエリア内のサブネットに接続しているが、RRRD (及び GateD) が存在しないホストである場合に生じる (図 4.6 の H1)。RRR では、RRRD が存在しないホストを “疑似ノード” として処理することで経路の提供を行う。実際には、宛先のネットワークノードに隣接するルータノード (図 4.6の R1 もしくは R2) を代替の宛先ノードとして経路要求メッセージを転送する。

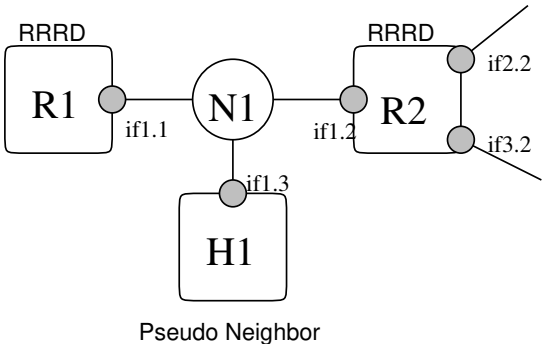


図 4.6: 疑似ノード

宛先ノードが決定した経路要求メッセージは、各ノードのデフォルトツリーに基づいて転送される。

経路情報メッセージ

宛先ノードが自ノードである経路要求メッセージを受け取った RRRD は、メッセージ中の TOS コードに対応する TOS メトリックによる SPF ツリーを計算し経路を決定する。このときフローの始点アドレスが “疑似ノード” である場合は、ローカルな RRR リンクステートデータベースに “疑似ノード” を作成する。求められた SPF ツリーはキャッシュされ、次回からはこのキャッシュにある SPF ツリーが参照される。このキャッシュは、RRR リンクステートデータベースが更新されるまで保持される。

次に経路情報メッセージの生成にあたって、終点ノードを以下のような手順で決定する。

1. ルータリンクステートの検索

   フローの終点アドレスとルータリンクステートが持つインタフェースアドレスが同じであれば、そのルータノードを終点ノードとする。

2. ネットワークリンクステートの検索

   まずフローの終点アドレスに、ネットワークリンクステートが持つネットマスクを適用する。そして、ネットワークリンクステートと同じサブネットアドレスであれば、そのネットワークノードを終点ノードとする。

3. サマリリンクステートの検索

   まずフローの終点アドレスに、サマリリンクステートが持つネットマスクを適用する。そして、同じサブネットアドレスを持つサマリリンクステートの中で、フローが要求する TOS コードに対する TOS メトリックが最も小さいものを広告したルータノードを終点ノードとする。

4. 1.~3. で適合しない場合、RRR リンクステートデータベースには、経路要求を受けたフローの終点アドレスに関する到達情報がないことになる。したがって、この時点で処理を中断し、RSRR モジュールに対してエラーを通知する。

ここで、宛先がネットワークノードとなる場合は、経路要求メッセージと同様に “疑似ノード” として処理し、隣接するルータノードを代替宛先ノードとする。
フローの終点アドレスがマルチキャストアドレスの場合は、RRR グループデータベースから、そのマルチキャストアドレスに関するマルチキャストグループ情報を広告したすべてのノードを終点ノードとする。

終点ノードが決定すると、SPF ツリーを自ノードから終点ノードまでたどり、経路情報メッセージを生成する。経路情報メッセージは、経路上の各ノードに対する経路情報の集合で構成される。

経路情報 ::= { ノード識別子, 出力インタフェース数,
(出力インタフェースアドレス, 下流ノード) × 出力インタフェース数 }

そして、経路上の全ノードの経路情報をリストにして、1 つの経路情報メッセージを作る。

経路情報メッセージ ::= { 経路配布ノード, 経路識別子, フロー識別子,
(経路情報) × 経路上の全てのノード数 }

メッセージ中の経路識別子とは、経路情報を参照するための識別子である。経路配布ノードの識別子と経路識別子の組み合わせにより、一意な経路を特定する。経路情報メッセージは、メッセージ自身に書かれた経路にしたがって転送される。マルチキャスト経路のように 2 つ以上の下流ノードが存在する場合は、経路情報メッセージを複製してそれぞれの下流ノードに転送する。

経路情報メッセージを受け取った RRRD は、メッセージの中から自ノードに対する経路情報を検索し、それを基に RRR 経路エントリを作成する。このとき経路エントリの状態は、確認応答メッセージが未到着である NACK となる (図 4.7)。

確認応答メッセージ

RRRD は受け取った経路情報メッセージが以下のような場合について、確認応答メッセージを生成する。

確認応答メッセージ ::= { 経路配布ノード, 経路識別子, フラグ }
フラグ ::= { Success | Unreachable | Fatal Error }

- 自ノードが終点ノード (出力インタフェース数 0) である

  経路情報メッセージは終点ノードまで正しく転送されている。したがって、フラグが Success (成功) の確認応答メッセージを生成し、始点ノードに対して送信する。

- 下流ノード (転送先ノード) が DOWN の状態にある

  これ以上、下流ノードに経路情報メッセージを転送することができない。したがって、フラグが Unreachable (到達不可能) の確認応答メッセージを生成し、始点ノードに対して送信する。

- 自ノードに関する経路情報がない

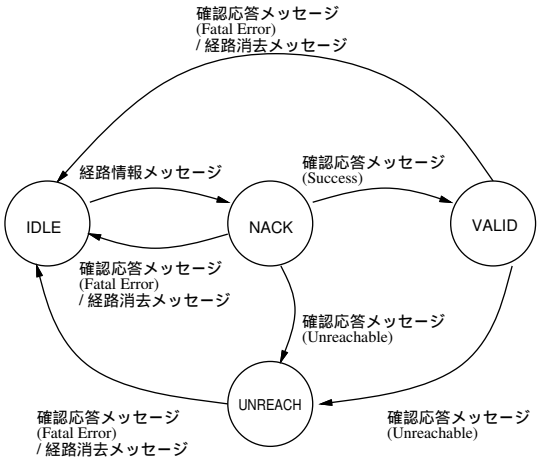


図 4.7: RRR 経路エントリの状態遷移図

- 経路情報に記述された下流ノードが存在しない

  経路計算に何らかのエラーが発生している可能性がある。したがって、フラグが Fatal Error (致命的な誤り) の確認応答メッセージを生成し、始点ノードに対して送信する。

確認応答メッセージは、経路情報メッセージの配布経路の逆順をたどって始点ノードまで転送される。

確認応答メッセージを受け取った RRRD は、フラグにしたがって該当する経路エントリの状態を変更する。図 4.7に RRR 経路エントリの状態遷移図を示した。図において状態の遷移が起こらない場合 (自己ループ) は省略してある。RSRR モジュールから経路要求を受けていた場合、確認応答メッセージを受け取りエントリの状態が NACK からいずれかの状態に遷移した時点で、RSRR モジュールを呼び出して経路提供 (もしくはエラーの通知) を行う。

マルチキャスト経路における確認応答メッセージ

マルチキャスト経路では、経路が枝別れしているノードにおいて、それぞれの下流ノードから複数の確認応答メッセージを受け取ることになる。RRRD では、すべての下流ノードから確認応答メッセージを受け取った後、確認応答メッセージのフラグを統合して、1 つの確認応答メッセージとして上流ノードに返す。フラグの統合は、Success < Unreachable < Fatal Error の順で、上位のフラグの確認応答メッセージが 1 つでもあった場合、そのフラグが上流ノードに送信する確認応答メッセージのフラグとなる。

このときフラグの統合によって、下流からの確認応答メッセージのフラグと上流ノードに送信する確認応答メッセージのフラグが異なる場合がある。この場合、マルチキャスト経路の枝によって同じフローに対する RRR 経路エントリの状態に違いが生じる。これを防ぐため、RRRD はフラグの統合を行ったノードから下流方向に特別な確認応答メッセージを送信することで、同じフローに対する経路エントリの状態を一意に保つ (図 4.8)。

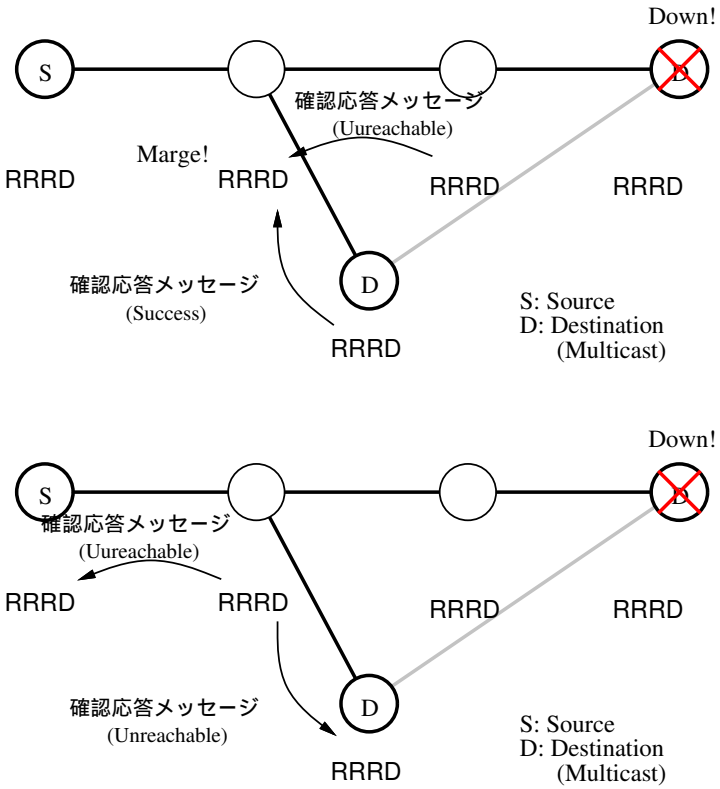


図 4.8: マルチキャスト経路における確認応答メッセージ

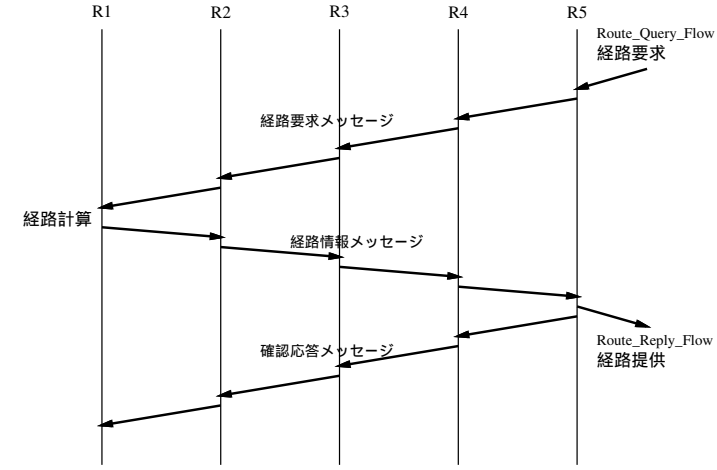


図 4.9: R5 における R1 から R5 への経路提供の例

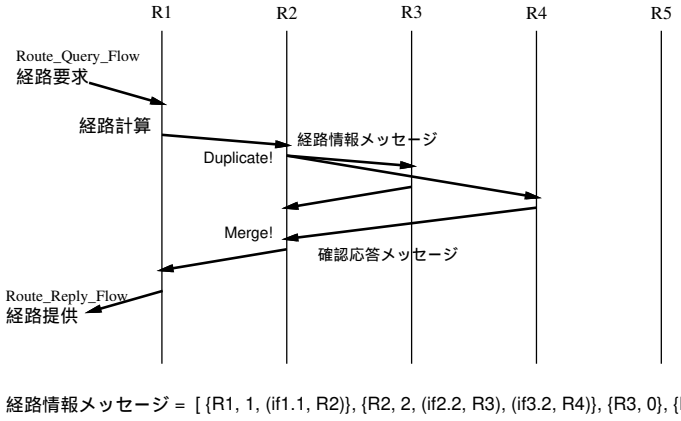


図 4.10: R1 における R1 から M1(R3,R4) への経路提供の例

サンプルネットワークにおける例

図 3.1のサンプルネットワークを用いて、経路の生成時における動作及びメッセージ交換の例を示す。R5 において、R1 から R5 に対するフローについての経路提供の要求が出た場合は図 4.9のようになる。また、R1 において、R1 から R3 と R4 が所属するマルチキャストグループ M1 に対するフローについての経路提供の要求が出た場合は図 4.10のようになる。

4.2.6 経路の消去

経路消去要求メッセージ

RSRR モジュールから経路消去の要求を受けた RRR モジュールは、経路消去要求メッセージを生成する。

経路消去要求メッセージ ::= { 経路要求ノード, 宛先ノード, フロー識別子 }

経路消去要求メッセージの宛先ノードの決定や転送方法は、経路要求メッセージと同様に行う。

経路消去メッセージ

宛先ノードが自ノードである経路消去要求メッセージを受け取った RRRD は、RRR 経路エントリを検索して該当するエントリの存在を確認し、経路消去メッセージを生成する。

経路消去メッセージ ::= { 経路配布ノード, 経路識別子, タイプ }
タイプ ::= { Initialize | Upstream | Downstream }

タイプは転送の方法を指定する。通常は Downstream (下流) を指定する。この場合、経路消去メッセージは、経路の下流方向へと転送される。

経路消去メッセージを受け取った RRRD は、RRR 経路エントリを検索し、該当する経路エントリを消去する。もし該当する経路エントリがない場合は、単にこのメッセージを無視する。なお、経路消去メッセージに対する確認応答は行わない。

起動時の経路消去メッセージ

起動時の動作で述べた特別な経路消去メッセージとは、タイプフィールドが Initialize の経路消去メッセージを指す。このメッセージを受け取った RRRD は、以下のような処理を行う。

1. RRR 経路エントリを検索し、メッセージ中に記述された経路配布ノードによって生成された経路エントリを全て消去する。
2. もし 1 つ以上経路エントリを消去した場合は、全ての隣接 RRRD に対して経路消去メッセージを転送する (フラッディング)。

サンプルネットワークにおける例

図 3.1のサンプルネットワークを用いて、経路の消去時における動作及びメッセージ交換の例を示す。R5 において、R1 から R5 に対するフローについての経路消去の要求が出た場合は図 4.11 のようになる。また、R1 において、R1 から R3 と R4 が所属するマルチキャストグループ M1 に対するフローについての経路消去の要求が出た場合は図 4.12のようになる。

4.2.7 マルチキャストグループ情報

RRRD におけるマルチキャストグループ情報の交換は、RRR モジュールが生成するグループ情報メッセージによって行われる。

グループ情報メッセージ ::= { グループ情報広告ノード, マルチキャストアドレス, インタフェースアドレス, タイプ }
タイプ ::= { Join | Leave | Flush | Solicit }

タイプフィールドが Join と Leave のメッセージは IGMP モジュールからの呼び出しによって生成される。また Flush と Solicit のメッセージは、RRRD 起動時の RRR グループデータベースの初期化に使用する。

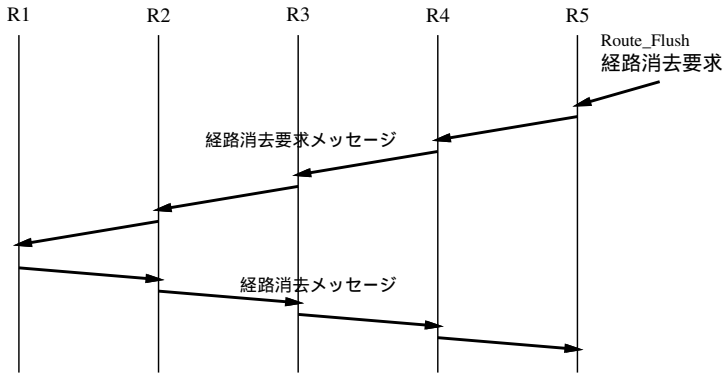


図 4.11: R5 における R1 から R5 への経路消去の例

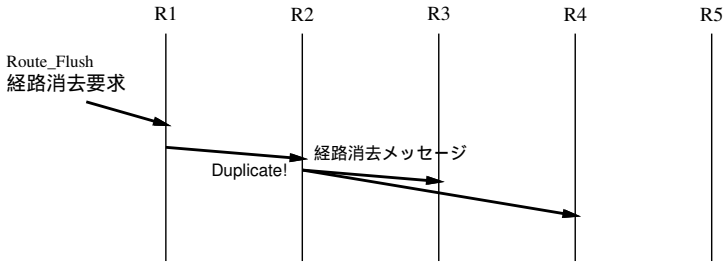


図 4.12: R1 における R1 から M1(R3,R4) への経路消去の例

IGMP モジュールからの呼び出し

IGMP モジュールから新たなマルチキャストグループメンバの発生を伝えられた RRR モジュールは、タイプフィールドが Join のグループ情報メッセージを生成する。グループ情報メッセージは、基本的にエリア内のすべての RRRD に対してフラッディングされる。このメッセージを受け取った RRRD は、以下のような処理を行う。

1. RRR グループ情報データベースから、メッセージに該当するマルチキャストグループ情報を検索する。
2. 該当する情報を発見できなかった場合、メッセージを新しい情報としてグループ情報データベースに追加し、メッセージをすべての隣接 RRRD に転送する。
3. 既にメッセージと同じマルチキャストグループ情報を保持していた場合は、単にこのメッセージを無視する。

また IGMP モジュールからマルチキャストグループメンバの消滅を伝えられた場合は、タイプフィールドが Leave のグループ情報メッセージを生成する。このメッセージを受け取った RRRD は、以下のような処理を行う。

1. RRR グループ情報データベースから、メッセージに該当するマルチキャストグループ情報を検索する。
2. 該当する情報を発見できた場合、そのマルチキャストグループ情報をグループ情報データベースから削除し、メッセージをすべての隣接 RRRD に転送する。
3. 該当する情報を発見できなかった場合、単にこのメッセージを無視する。

起動時のグループ情報メッセージ

起動時の動作の項目で述べた特別なグループ情報メッセージとは、タイプフィールドが Flush と Solicit のグループ情報メッセージである。Flush により古いマルチキャストグループ情報を消去し、Solicit によって最新のグループ情報を得る。

- Flush のグループ情報メッセージを受け取ったとき
  1. RRR グループ情報データベースを検索し、メッセージ中に記述されたグループ情報広告ノードによって広告されたマルチキャストグループ情報を全て消去する。
  2. もし 1 つ以上のマルチキャストグループ情報を消去した場合には、全ての隣接 RRRD ノードに対してメッセージを転送する (フラッディング)。
- Solicit のグループ情報メッセージを受け取ったとき

  現在保持している RRR グループ情報データベースの全てのマルチキャストグループ情報を、Solicit のグループ情報メッセージを送信したノードに対して返信する。

4.2.8 アラート情報

RSRR モジュールからあるフローについてアラートの発生を通知された場合、RRR モジュールは、RRR 経路エントリからそのフローについての経路エントリを検索し、アラートメッセージを生成する。

アラートメッセージ ::= { アラート発生ノード, 経路配布ノード, 経路識別子, インタフェースアドレス × 出力インタフェース数 }

メッセージ中のインタフェースアドレスは、アラートが発生したインタフェースを表す。このメッセージは、経路配布ノードまで転送される。

アラートメッセージを受け取った経路配布ノードは、以下のような処理を行う。

1. メッセージ中のアラート発生ノードを参照し、RRR リンクステートデータベースから該当するルータリンクステートを検索する。
2. メッセージ中のインタフェースアドレスを参照し、1. で検索したルータリンクステートの中の該当するインタフェースアドレスのリンクに対して、アラート状態であることをマークする (更新)。
3. 現在キャッシュに格納されている SPF ツリーを破棄する。

このとき同時に、一定時間後にアラートの解除を行うため、イベントスケジューラに対してアラートの解除の呼び出しを登録する。現在、アラートの保持時間は 30 秒に設定されている。

4.2.9 代替経路の要求

RSRR モジュールからあるフローについてアラートの発生と同時に代替経路の要求を通知された場合、RRRD はアラートメッセージと共に代替経路要求メッセージを生成、送信する。

代替経路要求メッセージ ::= { 経路要求ノード, 宛先ノード, フロー識別子 }

このメッセージ宛先の決定や転送方法は、経路要求メッセージと同様に行う。

宛先が自ノードである代替経路要求メッセージを受け取った RRRD は、RRR 経路エントリを検索して、フロー識別子に該当する経路エントリの存在を確認する。そしてまず経路消去メッセージを生成、送信してアラートの発生した経路の消去を行う。続けて経路情報メッセージを生成、送信して新しい経路、すなわち代替経路の配布を行う。この経路情報メッセージによって提供される新しい経路は、アラート情報メッセージによって RRR リンクステートデータベースが更新された後に計算される経路であり、結果的に代替経路の提供と同じ意味を持つことになる。

サンプルネットワークにおける例

図 3.1のサンプルネットワークを用いて、アラート発生及び代替経路要求の動作及びメッセージ交換の例を示す。R1 から R5 に対するフローの経路 (R1 - R2 - R4 - R5) の R2 においてアラートが発生し、代替経路の要求が行われた場合、図 4.13のようになる。

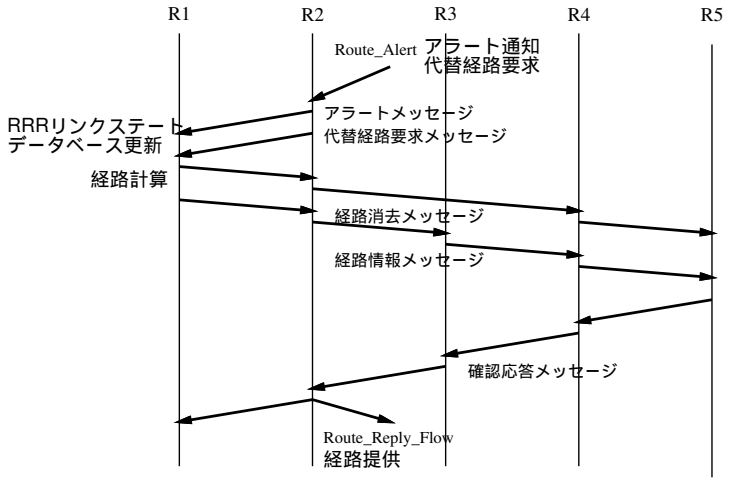


図 4.13: アラート発生及び代替経路の提供の例

4.3 OSPF モジュール

OSPF モジュールは GateD との通信を行うモジュールで、RRR モジュールからの呼び出しに応じて、GateD から OSPF トポロジデータベースを得る。

OSPF トポロジデータベースの獲得

GateD では、パケットタイプ 0 の特別な OSPF パケットが、内部状態を得るためのインタフェースとして用意されている。RRR モジュールから呼び出し (図 4.5の 2) を受けた OSPF モジュールは、OSPF トポロジデータベース要求のためのパケットタイプ 0 の OSPF パケットを生成する。

トポロジデータベース要求パケット ::= { エリア識別子, ポート番号 }

エリア識別子は、要求する OSPF トポロジデータベースのエリア識別子を記述する。ポート番号は、GateD からトポロジデータベースを受信するため (TCP) ポート番号を記述する。生成された OSPF パケットは、OSPF プロトコル (プロトコル番号 89) の IP RAW ソケットを通して、同じノード上のローカルな GateD に送信される (図 4.5の 3)。

OSPF トポロジデータベースの受信

GateD からのデータの受信は、ストリームソケット (TCP) を介して行われる。OSPF モジュールはストリームソケットを生成し、前述のパケットに記述したポート番号でコネクション設立を待つ。一方、トポロジデータベース要求パケットを受け取った GateD は、パケットに記述されたポート番号に対してコネクションを設立し、OSPF トポロジデータベースを送信する (図 4.5の 4)。

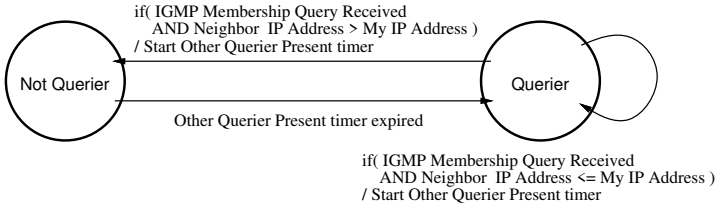


図 4.14: Querier の選択に関する状態遷移図

4.4 IGMP モジュール

IGMP モジュールは、IGMP を用いて自ノードが接続しているサブネットのマルチキャストグループ情報を収集する。IGMP の実装は基本的に IGMP Version 2 [16] に準拠している。

起動時の動作

起動時には、以下の手順で初期設定を行う。

1. IGMP プロトコル (プロトコル番号 2) の IP RAW ソケットを作成し、カーネルに登録する。
2. RRR で使用する全てのインタフェースを全マルチキャストルータグループ (224.0.0.2) に Join し、カーネルに VIF (Virtual InterFace) として登録する。これによりすべてのマルチキャストアドレスに対するパケットを受信することができる。
3. 30 秒後に Query タイマが起動するようにイベントスケジューラに登録する。

初期設定が終ると、IGMP モジュールは Query タイマによるイベントスケジューラからの呼び出し、もしくは IGMP パケットの到着まで待機状態に入る。

Query タイマ

Query タイマは、IGMP Membership Query メッセージを送信するためのタイマである。現在、Query タイマは 125 秒に設定されている。タイマによってイベントスケジューラから呼び出された IGMP モジュールは、RRR で使用する全てのインタフェースを通して全マルチキャストホストグループ (224.0.0.1) に対して IGMP Membership Query メッセージを送信する。

Querier の選択

通常、あるサブネットにおいて IGMP Membership Query メッセージを送信するマルチキャストルータは 1 つあればよい。したがって、同じサブネット内に複数の RRRD が存在する場合、IGMP モジュールは、最も IP アドレスの大きい隣接 RRRD (自ノードも含む) を Query メッセージを送信する Querier として選択する。Querier でない RRRD の IGMP モジュールは、Query タイマによる呼び出しがあっても、Query メッセージを送信しない。なお Querier の選択に関する状態遷移図を図 4.14に示した。図において、他の Querier についての生存期間を表す Other Querier Present タイマは 260 秒に設定されている。

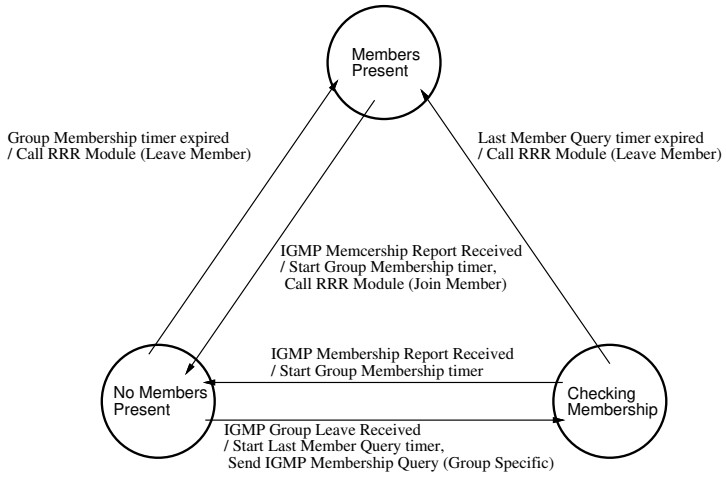


図 4.15: マルチキャストグループ状態遷移図

マルチキャストグループ状態遷移図

IGMP におけるマルチキャストグループのメンバシップ状態遷移図を図 4.15に示した。IGMP Version 2 の仕様では、IGMP Group Leave メッセージを受け取ったときに送信する特定グループへの IGMP Membership Query について、再送タイマによる再送が行われるが、この実装では省略している。図において、IGMP Membership Report が有効な期間を表す Group Membership タイマは 260 秒、Checking Membership 状態の継続時間を表す Last Member Query タイマは 1 秒に設定されている。

RRR モジュールの呼び出し

RRR モジュールの呼び出しは、以下の場合に行う。

- 新たなメンバの発生 (Join)

  No Members Present → Members Present

- メンバの消滅 (Leave)

  Members Present → No Members Present
  Checking Membership → No Members Present

4.5 RSRR モジュール

RSRR モジュールは、資源予約プロトコルと RRRD のインタフェースである RSRR メッセージを処理する。

起動時の動作

RSRR メッセージは、データグラムソケット (UNIX ドメイン) を介して送受信される。RSRR モジュールは、起動時に RSRR メッセージ用のソケットを作成する。

各メッセージの処理

RSRR の各メッセージの処理は以下の通りである。

- Initial_Query

  RRR で使用するインタフェース情報から Initial_Reply メッセージを返送する。

- Route_Query / Route_Query_Flow

  1. RRR 経路エントリを検索し、メッセージ中のフロー識別子に対応する経路が存在するかどうかを調べる。このとき Route_Query メッセージには、フロー識別子の始点ポート及び終点ポートに該当する項目がないので、これを 0 として検索する。
  2. もし経路が存在するのであれば、Route_Reply / Route_Reply_Flow メッセージを返送する (終了)。
  3. 経路が存在しない場合は、RRR モジュールに対して経路要求を行う。このとき Route_Query メッセージの場合の TOS コードは 0 を指定する。
  4. Route_Query / Route_Query_Flow メッセージを RSRR Reply キューに登録し、RRR モジュールからの経路提供を待つ。このとき、タイムアウトのための Reply Deadline タイマを起動する。現在 Reply Deadline タイマは 5 秒に設定されている。

- Route_Flush

  RRR 経路エントリを検索し、メッセージ中のフロー識別子に対応する経路が存在するかどうかを調べる。存在する場合は RRR モジュールに対して経路消去を要求する。

- Route_Alert

  RRR 経路エントリを検索し、メッセージ中のフロー識別子に対応する経路が存在するかどうかを調べる。存在する場合は RRR モジュールに対してアラート発生を通知する。また、メッセージ中の代替経路要求フラグが立っていた場合は、同時に代替経路の要求を行う。

RRR モジュールからの呼び出し

RRR モジュールから経路提供 (またはエラー通知) の呼び出しがあった場合、以下の処理を行う。

1. RSRR Reply キューから、該当する Route_Query / Route_Query_Flow メッセージを検索する。
2. キューの中にメッセージが存在した場合は、Route_Reply / Route_Reply_Flow メッセージを送信する。このとき該当する経路エントリの状態が UNREACH の場合、もしくは該当するエントリがない (IDLE) の場合は、メッセージ中のエラーフラグを立てる。
3. RSRR Reply キューからそのメッセージを削除する。

なお 1. において、RSRR Reply キューの中に該当する Route_Query / Route_Query_Flow メッセージがなかった場合は、既にタイムアウトしたとみなして、そのまま処理を終了する。

Reply Deadline タイマ

Reply Deadline タイマによって呼び出された RSRR モジュールは、以下の処理を行う。

1. RSRR Reply キューから、該当する Route_Query / Route_Query_Flow メッセージを検索する。
2. もしメッセージが存在する場合は、タイムアウトとみなして Route_Reply / Route_Reply_Flow メッセージをエラーメッセージ (エラーフラグを立てる) として送信する。
3. RSRR Reply キューからそのメッセージを削除する。

なお 1. において、RSRR Reply キューの中に該当する Route_Query / Route_Query_Flow メッセージがなかった場合は、既に正常に Route_Reply / Route_Reply_Flow メッセージが送信されたとみなして、そのまま処理を終了する。

4.6 実装上の制限

GateD において OSPF の TOS メトリック扱う実装が一部なされていないため、現在、RRRD で扱える TOS メトリックは 1 種類 (帯域幅もしくは遅延時間のいずれか) に制限される[§]。

[§]通常の TOS 0 メトリック (16 ビット) のうち上位 8 ビットを TOS メトリック、下位 8 ビットを TOS 0 メトリックとして使用している。したがって、各リンクに設定できるメトリックの最大値は 255 に制限される。

第 5 章

評価と検討

本章では、本研究で提案する経路制御プロトコル RRR 及び経路制御デーモン RRRD の評価及び検討を行う。まず始めに RRR の評価の議論を行い、次に RRRD の実装についての測定結果を示す。最後に、RRR を含めた資源予約ネットワークにおける経路制御についての今後の課題を述べる。

5.1 RRR の評価

RRR の目的は、保証型サービスを要求するフローに対して適切な経路を提供することである。したがって RRR の評価としては、実際に保証型サービスを提供する資源予約ネットワークにおいて RRR を運用し、個々のフローに提供する経路が適切であるかどうかを調査するのが望ましいと言える。しかし、現在のインターネットにおいて、保証型サービスを提供するネットワークは実験用のテストベッドを除いて存在しない。シミュレーションによる評価を行うとしても、ネットワークの規模やトポロジ、リンクの特性、フローのトラフィックパターンなど、実際に資源予約ネットワークの構築や運用を通じてその特徴を明らかにしないと決定できない要素が多く含まれる。また、図 2.2で示した資源予約ネットワークアーキテクチャのうち、RRRD 以外の構成要素についてどのような機構のモジュールを用いるかによっても、得られる結果が大きく異なることが予想される。さらに、様々なトラフィックパラメータを持つフローに対して、RRR が提供する経路が適切であるかどうかについて、単純な評価を行うことは難しい。

RRR の設計のアプローチは、従来の経路制御プロトコルや資源予約プロトコルとの連携を前提に置き、負荷と性能のトレードオフを考慮しながら、実現可能な範囲でどのような経路提供が行えるのかを模索するものである。RRR は、提供する経路が必ず最適な経路を提供することを保証するものではない。あくまで保証型サービスを実現するための経路制御プロトコルの一つの形態に過ぎない。しかし、RRR が従来の経路制御プロトコルにはない機能を持っていることは、既に述べた通りである。

そこで本研究では、RRR のような機構を導入することによって、どの程度新たな負荷が生じるのかを中心に評価を行うことにする。

一般に経路制御プロトコルの負荷に関する項目は、経路制御情報の交換に関する負荷と経路計算に関する負荷に大別される。しかし RRR の基本的な経路制御情報は、OSPF の OSPF トポロジデータベースから得ているため、経路制御情報の交換に関する部分は評価の対象から除外した。その代わりに、RRR の特徴である始点ノードから経路上のノードに対する経路情報配布と、資源予約プロトコルに対する経路提供に関する部分を評価対象に加えた。したがって、RRR の

負荷についての評価及び検討は、

- 経路計算
- 経路情報配布
- 経路提供

の 3 項目について行うこととする。

5.1.1 経路計算

RRR では、フローの始点ノードにおいて RRR リンクステートデータベースから Dijkstra のアルゴリズムを用いて SPF ツリーを計算し、経路を決定する。経路計算に関する評価項目として以下のものを取り上げる。

- メモリ使用量 ... RRR リンクステートデータベースの容量
- 計算量 ··· SPF ツリーの計算量
- 計算頻度 ··· RRR リンクステートデータベースの更新頻度
- 安定性 ··· RRR リンクステートデータベースの更新頻度

メモリ使用量

n 個のルータが、各ルータ毎にそれぞれ k個のリンクを介して k個のルータと接続しているようなネットワークを考える。各リンクにはそれぞれ 2 個のルータが接続していることになるので、サブネット数は$\frac{n \times k}{2}$となる。すなわち、RRR リンクステートデータベースには

- k個のネットワークリンクを持った n 個のルータノードリンクステート
- 2 個のルータリンクを持った$\frac{n \times k}{2}$個のネットワークノードリンクステート

が存在することになる。1 つのルータノードリンクステートは $58 + 16 \times k$ バイト、1 つのネットワークノードリンクステートは 66 バイトである。ネットワーク規模として、ルータ数 1,000、各ルータの接続リンク数 4 のネットワークを仮定した場合、RRR リンクステートデータベースの大きさは以下のようになる。

$$(58 + 16 \times 4) \times 1000 + 66 \times \frac{1000 \times 4}{2} = 122000 + 132000 = 254000(byte)$$

この程度のメモリの消費は、現在のコンピュータ環境においては全く問題にならない。またこの他に、RRR エリア外の情報であるサマリリンクステートが加わるが、その大きさは、ルータノードリンクステートやネットワークノードリンクステートに比べて十分小さいと考えられる。

計算量

ノード数を N、各ノードの持つリンク数を K としたネットワークにおいて、Dijkstra のアルゴリズムを計算するには、一般にノード数 N の対数とネットワークの総リンク数 $((N \times K)/2)$ の積に比例した大きさ、つまり $O(N \times K \times logN)$ の計算量が必要である。この中で、$logN$ という因数は、経路が決定していないノードを二分探索を用いてコスト順の並べるために生じる負荷である。したがって、コストがある範囲に限定されているならば、適切なハッシュ関数を用いることで、この負荷を取り除くことができる。つまり、Dijkstra の SPF ツリーを計算するのに必要な計算量は $O(N \times K)$ となる [32]。

単純にこの計算量を見る限りは、ネットワークの規模に対して計算量が飛躍的に増大することが考えられる。しかし、RRR の運用で想定するネットワークの規模は、現在同じ Dijkstra のアルゴリズムにより経路計算を行う OSPF が運用されている範囲と同等であり、十分に現実的な範囲の計算量に収まるものと思われる。

計算頻度

これまでに述べたメモリ使用量や計算量は、OSPF トポロジデータベースを基本としていることや、Dijkstra のアルゴリズムを使用していることなどから、ほぼ OSPF と同じ程度であった。RRR が OSPF と異なるのは、その計算頻度である。RRR の SPF ツリーの計算頻度は、RRR リンクステートデータベースの更新頻度に依存する。RRR リンクステートデータベースには、ネットワークの資源状態を把握するためのアラート情報が含まれるため、その更新頻度はネットワーク全体における資源状態の変化、すなわち資源予約要求の頻度やネットワークの規模に依存する。RRR では、Appendix Aに示した予備シミュレーションの結果を参考に、資源予約要求頻度の増加やネットワーク規模の拡大に対して不要な RRR リンクステートデータベースの更新や SPF ツリーの計算を避けるため、以下のような機構を採り入れた。

- 経路要求メッセージと始点ノードによる経路計算及び決定

  RRR では、始点ノードによる経路計算及び決定を行っており、あるフローに関する経路については、始点ノード以外で行う必要がない。また経路決定に必要な SPF ツリーの計算は、経路要求メッセージが到着してはじめて計算される。これにより、RRR リンクステートデータベースを更新があったとしても、経路要求メッセージが到着しない限りは経路計算を行われないため、不要な経路計算を回避することができる。

- アラートメッセージの処理

  RRR では、ネットワーク資源状態の変化をアラートメッセージにより通知している。このアラートメッセージは、アラート発生の要因となったフローの始点ノードに対して送信されるため、RRR リンクステートデータベースが更新されるのも、その始点ノードのみである。また、RRR リンクステートデータベース中のアラートは、一定時間保持されるようになっている。もしその一定時間中に、同じリンクに関するアラートメッセージが再度到着した場合、RRR はアラートマークの保持時間の更新のみを行い、実際の RRR リンクステートデータベースの更新はしない。これにより、キャッシュに存在する SPF ツリーを破棄することなく経路決定に使用できるため、不要な経路計算を回避することができる。

安定性

経路の安定性も計算頻度と同様に、リンクステートデータベースの更新頻度に依存する。更新が頻繁に起きると、SPF ツリーの再計算により提供する経路が不安定になり、経路の循環 (loop) や振動 (oscillation) といった問題が生じる。しかし、RRR では始点ノードにおいて経路決定を行うため、経路循環の発生を完全に除去することができる。また先ほど述べたアラートの保持時間を適切に設定することにより、RRR リンクステートデータベースの更新を抑え、経路の振動を抑制することができると考えられる。

計算頻度や安定性に対する定量的評価に関しては、実際に RRR の運用を通して、経路要求メッセージやアラートメッセージの発生頻度に関する詳細な調査を待つ必要があると思われる。

5.1.2 経路情報配布

RRR では、始点ノードで決定された経路を経路情報メッセージによって経路上のノードに配布し、RRR 経路エントリを作成する。経路情報配布に関する評価項目として以下のものを取り上げる。

- ネットワーク帯域幅 ··· 経路情報メッセージの交換量
- メモリ使用量 ··· RRR 経路エントリの容量

ネットワーク帯域幅

経路情報配布に関して消費されるネットワーク帯域幅は、経路情報メッセージによるものと、確認応答メッセージによるものの 2 つがある。始点ノードから終点ノードまで k ホップの経路を考えた場合、1 つの経路情報メッセージの大きさは $52 + 16 \times k$ バイトとなる。また確認応答メッセージは 48 バイトである。仮にあるリンクにおいて、平均 10 ホップの経路の経路情報メッセージが毎秒 100 フロー分流れると考えると、IP ヘッダ 20 バイトを含めたこれらのメッセージが消費するネットワーク帯域幅は、

- 上流ノード→下流ノード (経路情報メッセージ)

$$((20 + 52 + 16 \times 10) \times 100) \times 8 = 185.6(kbps)$$

- 下流ノード→上流ノード (確認応答メッセージ)

$$((20 + 48) \times 100) \times 8 = 54.4(kbps)$$

となる。この帯域幅を多いと見るか少ないと見るかは、そのリンクの帯域幅に依る。経路情報メッセージは、資源予約プロトコルからの経路提供の要求、すなわち資源予約要求と連動しているため、帯域幅が十分に広いリンク (基幹ネットワーク) ではより多くのメッセージが流れる可能性があるが、帯域幅の狭いリンク (末端ネットワーク) ではそれほどメッセージは流れないものと考えられる。また資源予約ネットワークのリンクは、一般的に百数十 Mbps から数 Gbps の帯域幅を持つものと考えられるため、経路情報メッセージによる帯域幅の消費量はごく小さいものと考えられる。ただし、確実なメッセージの転送を行うためには、これらの RRR メッセージ用の帯域を予め確保しておくことは重要である。

メモリ使用量

RRR は、内部に RRR 経路エントリという独自のフロー毎の経路テーブルを持ち、経路情報メッセージによって、この RRR 経路エントリを作成する。1 つの経路エントリはユニキャスト経路で 43 バイトである。したがって、フロー数が 1,000 の場合は 43,000 バイト、10,000 でも 430,000 バイトである。この程度のメモリの消費は現在のコンピュータ環境においては、全く問題にならない。

5.1.3 経路提供

RRR は、資源予約プロトコルに対して経路を提供する。ここでは、特に資源予約プロトコルが経路提供の要求を出してから RRR が返答するまでの応答時間を評価項目として取り上げる。応答時間は、RRR 経路エントリの検索の結果によって 2 通りに分かれる。

A. RRR 経路エントリに経路がある場合

そのまま経路エントリに基づいて資源予約プロトコルに返答する。

B. RRR 経路エントリに経路がない場合

フローの始点ノードに対して経路要求を行う。そして、経路情報メッセージ及び確認応答メッセージによって経路エントリが確立された時点で、資源予約プロトコルに返答する。

A. と B. のどちらの場合であるかによって応答時間は大きく異なる。以下、それぞれの場合について評価する。

RRR 経路エントリに経路がある場合

この場合、資源予約プロトコルへの応答時間は、基本的に RRR 経路エントリの検索にかかる時間のみである。実装にも依存するが、RRR 経路エントリに登録されているフロー数 n とすると、検索時間は $O(logn)$ から $O(n)$ となる。

RRR 経路エントリに経路がない場合

この場合、資源予約プロトコルへの応答時間は、以下の項目の和となる。

1. 経路要求メッセージが自ノードから始点ノードまで転送される時間
2. 経路情報メッセージが始点ノードから終点ノードまで転送される時間
3. 確認応答メッセージが終点ノードから自ノードまで転送される時間

もし要求を受けたノードがフローの始点ノードである場合は 2. と 3. の和、また終点ノードにおいて要求を受けた場合は 1. と 2. の和となる。これらはすなわち、フローの RTT (Round Trip Time) に近い値となるものと考えられる。

一般に RTT は 2 つのノード間の距離、すなわちホップ数が増えるほど増加する傾向にあるが、トラフィックの状況により様々に変化する。RRR メッセージ用の帯域を予め確保しておくことにより、他のトラフィックの影響による RTT の増加を回避することもできる。しかし、RRR メッ

セージ自体が輻輳状態を起こした場合は、RTT の増加は避けられない。RRR では、RRR メッセージ自体の輻輳に対する機構は持ち合わせていない。

応答時間について RTT が関与するような機構は、予測ができないため望ましくない性質である。しかし、資源予約プロトコルに対する経路提供という観点から考えた場合、応答時間に RTT を要することはそれほど問題とはならないと考えられる。それは以下のような理由による。

- RRR 経路エントリに経路がない場合は、そのフローに関して最初に経路の問い合わせを受けたノードに限定される

  資源予約プロトコルからの経路提供の要求によって、一旦 RRR により経路情報が配布されると、経路上の全てのノードの RRR 経路エントリにそのフローの経路が登録される。したがって、他のノードにおける資源予約プロトコルへの応答時間は、A. の RRR 経路エントリの検索にかかる時間のみとなる。

- 保証型サービスを必要とするフローの通信時間は十分に長い

  連続メディア通信やインタラクティブなシステムにおける通信など、保証型サービスを必要とするフローの通信時間は、一般的に十分な継続時間を持つものと考えられる。つまり、通信時間に対して経路を確立するための RTT は十分に小さく、実用上は問題ないと考えられる。実際、多くの資源予約プロトコルはユーザから要求を受けてから資源予約を完了するまで、RTT の時間を要する。

5.2 RRRD の評価

本節では、RRRD の実装についての評価を行う。RRRD の評価については、その測定環境 (ルータの性能、ネットワークの規模、トラフィックパターン) によって結果が大きく異なることが予想される。したがってこの評価の目的は、あくまで RRR を導入することによって生じる新たな負荷についての実際の目安を得ることである。

5.2.1 測定環境

測定は、図 5.1に示したようなネットワークにおいて行った。各ルータは、IBM PC/AT 互換機 (Pentium 166MHz, 64MB RAM) に ATM NIC (PCI バス) を挿したもので構成されている。また各ルータ間は OC-3 (155.52Mbps) の光ファイバによって接続されている。OS は FreeBSD-2.2.1-RELEASE を用いており、測定はシングルユーザモードで行った。

5.2.2 測定内容

測定は、以下の 2 つについて行った。

測定 1. 各ルータ上において RRRD が RRR 経路エントリの検索にかかる時間

RRRD では、経路の生成や消去時、資源予約プロトコルからの経路の問い合わせ時に、RRR 経路エントリを検索する。RRR 経路エントリは、現在その RRRD が保持している全てのフローについての経路情報が登録されており、フローの増加につれて検索時間も増すものと考えられる。そこで、RRR 経路エントリに予めある数の経路情報を登録しておき、経路エントリ数の増加に

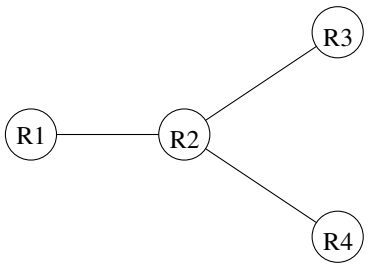


図 5.1: 測定ネットワーク

対する検索時間の変化を測定した。RRRD では、RRR 経路エントリの検索にハッシュテーブルを利用していることから、ハッシュテーブルサイズを 3 種類変化させて、それぞれについて測定を行った。

測定は、RRR 経路エントリに登録されている経路情報の中から、ランダムに 100 フロー分についての検索を行った。そしてこれを 10 回繰り返し、計 1,000 回の試行における 1 フロー当たりの平均検索時間を求めた。

測定 **2.** 各ルータ上において **RRRD** がメッセージの処理にかかる時間

次に、実際に経路提供を行う場合に、始点ノード、中間ノード、終点ノードにおいて RRRD が各メッセージの処理にかかる時間を測定した。これには測定 1. の RRR 経路エントリの検索時間や登録時間、転送先ノードの決定、メッセージの作成やチェックサム計算にかかる時間などが含まれる。

測定は、以下の 3 通りの場合について行った。

1. ユニキャスト経路 (始点ノード主導)

   R1 から R3 へのユニキャストフローについて、始点ノード R1 において資源予約プロトコルから経路提供の要求を受けた場合について測定した。

2. ユニキャスト経路 (終点ノード主導)

   R1 から R3 へのユニキャストフローについて、終点ノード R3 において資源予約プロトコルから経路提供の要求を受けた場合について測定した。

3. マルチキャスト経路 (始点ノード主導)

   R1 から R3 及び R4 へのマルチキャストフローについて、始点ノード R1 において資源予約プロトコルから経路提供の要求を受けた場合について測定した。

上記それぞれの場合について、1 秒間に 10 回の割合で RSRR インタフェースを介して計 100 フロー分の経路提供の要求を行った。そしてこれを 10 回繰り返し、計 1,000 回の試行における 1 フロー当たりの平均時間を求めた。

5.2.3 測定結果

測定 1. の結果を図 5.2に、測定 2. の結果を図 5.3、図 5.4、図 5.5に示した。図中に示された値は、すべて平均値である。

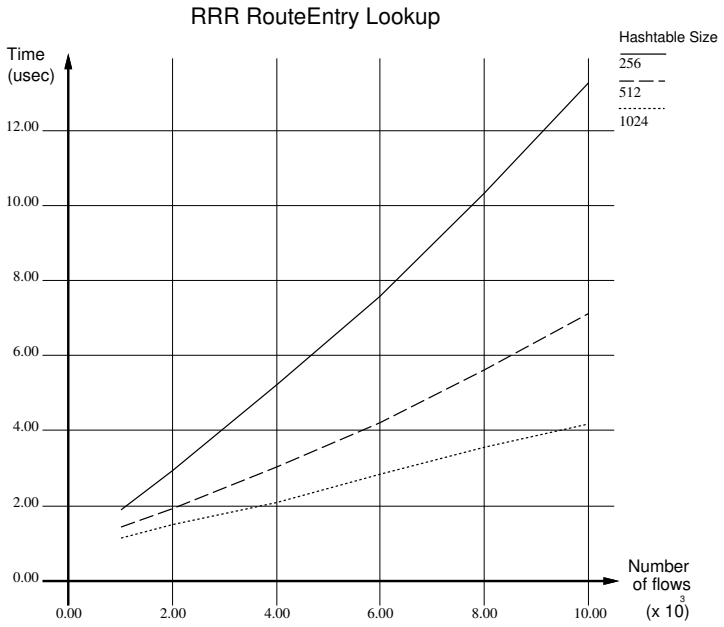


図 5.2: RRR 経路エントリの検索時間

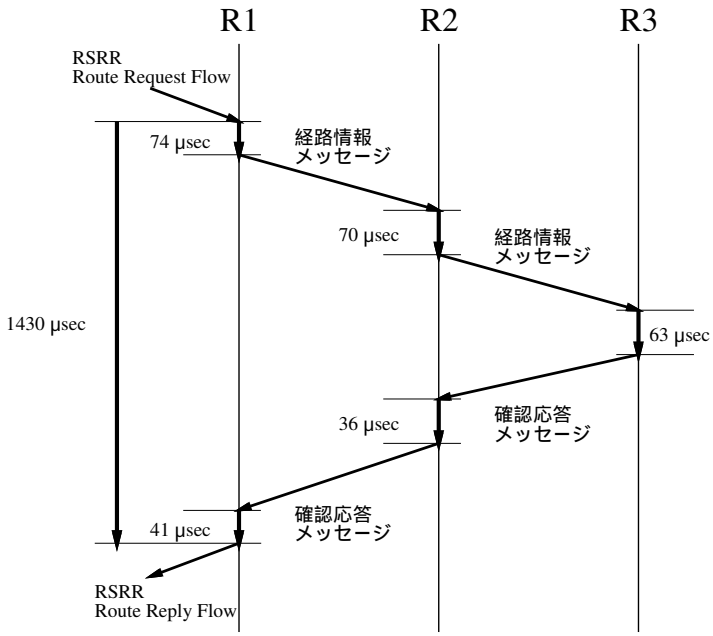


図 5.3: ユニキャスト経路 (始点ノード主導)

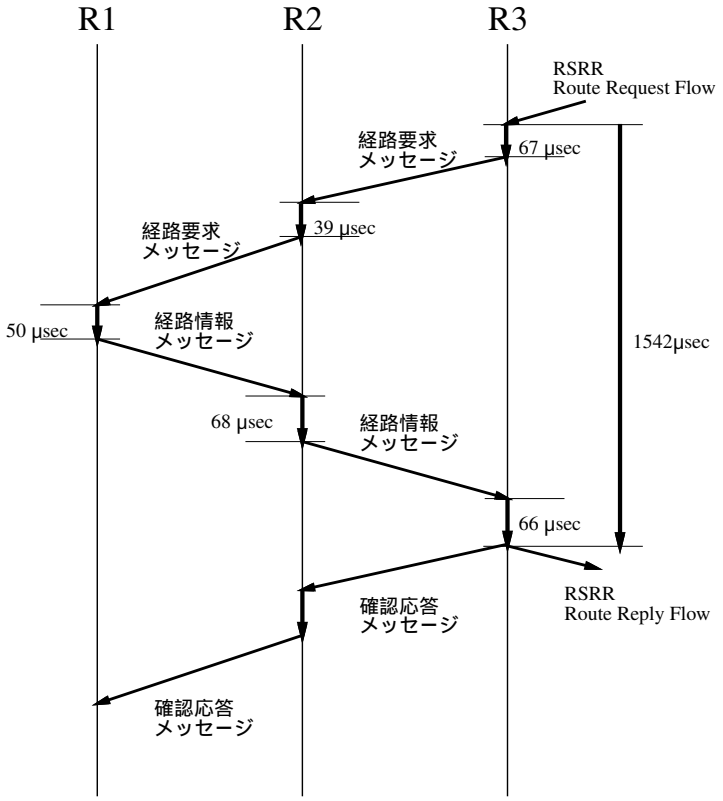


図 5.4: ユニキャスト経路 (終点ノード主導)

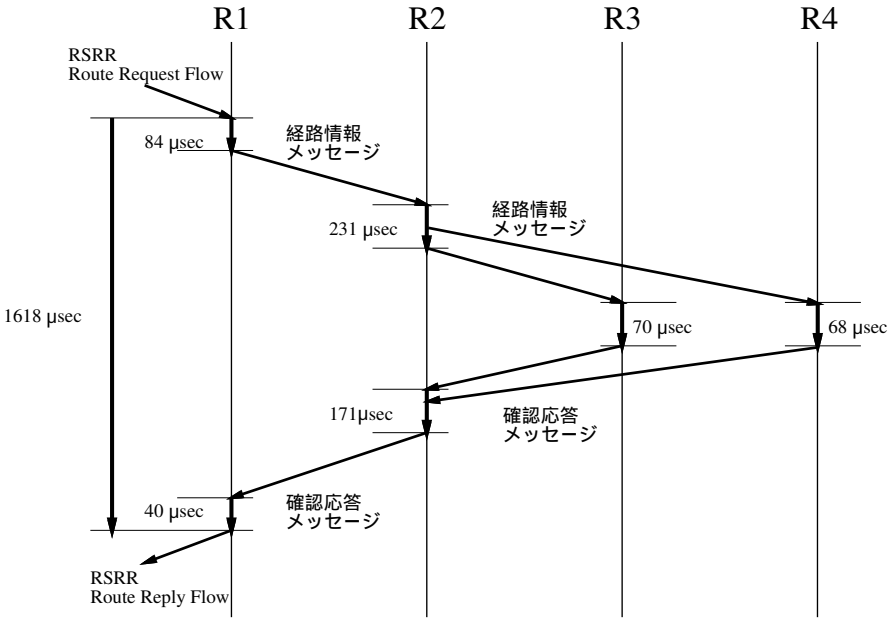


図 5.5: マルチキャスト経路 (始点ノード主導)

測定 1. の結果について

フロー数 (経路エントリ数) の増加に対して検索時間は、ほぼ $O(n)$ で増加している。またハッシュテーブルサイズが大きいほど、検索時間の増加は緩やかである。ハッシュテーブルが 1024 の場合、10000 フローの中から経路情報検索するのに、約 $4\mu sec$ ほどかかっている。測定 2. において、RRR のメッセージ処理にかかる時間が数十μsec 前後であることを考えると、フロー数の増加による RRR 経路エントリの検索時間の増加が与える影響はごく小さいものと考えられる。

測定 2. の結果について

経路提供に関するそれぞれのメッセージの処理について、各ノードにおいて数十μsec 前後の時間がかかっていることがわかる。このうち、RRR メッセージのチェックサム計算にかかる時間が $30\mu sec$ 前後である。また、図 5.5のマルチキャスト経路の場合は、R2 のように枝分かれする箇所において、若干の時間がかかっている。これは、枝にあたるそれぞれのリンクに対して個別にメッセージを送る必要があるのと、複数の確認応答メッセージを待ち合わせるために生じる時間である。

同時に測定した資源予約プロトコルに対する応答時間を測定したが、いずれの場合も $1.5msec$ 前後の時間を要している。各ノードにおける処理の合計が $200 \sim 300\mu sec$ であることを考えると、メッセージ転送自体に要する時間は、RRRD のメッセージ処理にかかる時間の数倍かかっている。すなわち、実際に資源予約プロトコルに対して経路提供を行う場合の応答時間は、始点ノードと終点ノード間の RTT が支配的であると言える。

5.3 今後の課題

本節では、前半で本研究で提案した RRR に関して未解決の問題及び今後の課題を述べ、後半では RRR の設計及び RRRD の実装を通して資源予約ネットワークにおける経路制御についての今後の課題を議論する。

5.3.1 RRR における未解決の問題及び今後の課題

ネットワーク資源情報

RRR では、リンクの帯域幅及び遅延時間の表現として、固定された OSPF TOS メトリックを用いており、トラフィックによる利用可能なネットワーク資源の動的な変化については、アラート情報のみで対応している。しかし、各ルータの CPU の処理速度の向上やメモリの大容量化、ネットワーク帯域幅の増加などネットワーク環境が変化すれば、より細粒度なネットワーク資源情報の把握も可能になると考えられる。ただし経路選択を行う上で、どのようなネットワーク資源情報がどの程度有効に機能するかについては、資源予約ネットワークにおけるトラフィックパターンの詳細な調査が必要であり、今後資源予約ネットワークの構築、運用を通じて明らかになるものと考えられる。

マルチキャスト

現在、RRR ではマルチキャスト経路として IP マルチキャストに対する経路をサポートする。しかし、RRR は始点ノードからの明示的な経路情報の配布によって経路を生成するため、提供

するマルチキャスト経路は経路を生成した時点におけるマルチキャストメンバに対してのみの経路となり、動的なメンバシップの変化に対応できない。これは、動的なメンバシップの変化に対応するマルチキャスト経路制御を前提として資源予約メッセージ (PATH メッセージ) を送信している RSVP などでは問題となると考えられる。

マルチキャスト通信は、保証型サービスを要求する連続メディア通信などにおいて、ネットワーク資源を有効に利用するために欠かせない要素であり、資源予約機構に対してどのようにマルチキャスト経路を提供していくべきか、今後の課題として検討が必要である。

耐障害性

RRR では、Hello プロトコルによって隣接ノード及び隣接ノード間のリンクの状態について監視を行っている。しかし、隣接ノードとの RRR メッセージの交換においてはメッセージのチェックサム計算のみで、個々のメッセージに対する確認応答の機構は取り入れていない。したがって、何らかの原因によりメッセージそのものが喪失した場合には、それを検知することができない。特に経路を生成するための経路情報メッセージや確認応答メッセージが失われると、RRR 経路エントリの一貫性が保てなくなる。

資源予約プロトコル ASP では、資源予約情報の一貫性の検証のため、SIP (Status Inquiry Protocol)[48] を用いて隣接ノード間で定期的に資源予約情報の突合を行っている。そして、矛盾のある情報については、それを解消する (ガベージコレクション) ような機構が取り入れられている。そこで RRR においても、隣接ノード間で定期的に RRR 経路エントリの突合を行い、矛盾のある経路エントリに関してはこれを消去するような機構を導入する必要があるものと思われる。

セキュリティ

現在、RRR メッセージの認証や暗号化などのセキュリティ対策は考慮されていない。RRR メッセージによって認証なしに経路エントリの生成や消去、あるいはアラートの通知を行うことができるため、悪意のあるユーザによって RRR メッセージが生成された場合、ネットワーク全体に混乱を引き起こすことになる。OSPF ではメッセージヘッダに認証のためのフィールドが用意されおり、GateD の実装においては MD5 を用いた秘密鍵共有方式の認証が行われている。RRR でも同様の手法によって秘密鍵共有方式の認証を行うことで、セキュリティ対策を行う必要があるものと思われる。

5.3.2 資源予約ネットワークにおける経路制御についての課題

本研究では、資源予約ネットワークにおける経路制御について、単なる経路制御プロトコルの提案ではなく、資源予約ネットワークのアーキテクチャを定義し、資源予約機構や経路制御機構など様々なモジュールの関係を考慮した上で RRR の設計を行った。ここでは、RRR の設計及び RRRD の実装を踏まえた上で、資源予約ネットワークアーキテクチャについて再度検討する。

資源予約機構と経路制御機構の関係

本研究では、経路制御機構において経路制御プロトコルと経路制御テーブルを全く独立したものとして考え、資源予約機構と経路制御機構の関係を見直した。本研究で提案した経路制御プロトコル RRR は、経路制御テーブルに対して経路を設定するのではなく、資源予約機構に対して経路の提供を行う。そして、経路制御テーブルは資源予約機構によって管理される。これにより

RRR は、既存の資源予約フローに対する経路を気にすることなしに、アラート情報により RRR リンクステートデータベースの更新を行い、その時点での最適な経路選択を行うことができる。また資源予約プロトコルは自ら経路設定を行うことで、 ネットワーク資源の変化による経路制御プロトコルの経路変更に左右されずに、資源予約を行った経路上に確実にフローを転送することができるのである。

現在提案されている資源予約プロトコルは、必ずしも経路制御に関して十分な考慮がなされているとは言えない。資源予約プロトコルとのインタフェースを規定しているのは、RSRR をサポートする RSVP のみである。また RSRR においても Route Query に際して、フロー識別子やフローのトラフィックパラメータを指定するようなインタフェースは存在しない。しかし最近の RSVP の経路制御インタフェースに関する議論 [19] では、RSRR の Route Query にトラフィックパラメータを渡すような改善が提案されており、今後 RRR のようなフローの識別子やフローのトラフィックパラメータに対応した経路制御プロトコルへの適応が期待される。また、資源予約プロトコルが直接経路制御テーブルに対して経路を設定するという考えについても、2.3節で述べた Label Switching の機構を用いたアプローチが議論されており、今後の動向が注目される。

資源予約プロトコルのもう 1 つの問題は、資源予約プロトコルのメッセージの転送方法である。IP は基本的にホップバイホップの経路制御機構を持つため、資源予約プロトコルのメッセージもホップバイホップで各ノードの経路制御テーブルを参照して転送される。しかし、トラフィックパラメータを考慮するような柔軟な経路選択を行うためには、始点ノードで完全な経路を決定する始点経路制御の手法を用いるほうが有利であり、RRR もこの機構を採り入れている。しかし、提案されている資源予約プロトコルはメッセージを始点経路制御により転送することを考慮していない。したがって、ER や RRR で行っているように、経路情報を経路上の全てのノード配布することで対応せざるを得ない。もし資源予約プロトコルがメッセージの始点経路制御のメッセージ転送をサポートすれば、この経路情報の配布のオーバヘッドを削減することができると考えられる。

従来の経路制御機構との関係

本研究では、従来の最善努力型サービスに対する経路制御機構と保証型サービスに対する経路制御機構を独立したものとして考えた。したがって本研究で提案した経路制御プロトコル RRR は、既存の経路制御プロトコルと完全な共存が可能である。さらに RRR は、経路制御プロトコル OSPF の経路制御情報を利用することで、少ないオーバヘッドによる実現を可能としている。資源予約ネットワークにおける経路制御機構に求められるものは、従来の経路制御機構の置き換えではなく、あくまで保証型サービスを提供するための機能の拡張である。よって、従来の経路制御機構との協調を考えることは、実現可能性の観点から大変重要である。

本研究では、経路制御プロトコル RRR について、RRRD という独立した経路制御デーモンとして実装を行った。しかし、OSPF デーモンとして使用した GateD は、個々の経路制御プロトコルをモジュールとして扱い、複数の経路制御プロトコルを 1 つの経路制御デーモンとして実装することが可能である。したがって、RRR をモジュール化して GateD に組み込むことで、さらに効率の良い実装も可能であると考えられる。

第 6 章

結論

保証型サービスを提供する資源予約ネットワークにおいては、様々なフローの要求やネットワークの特性、ネットワークの資源状態を考慮した上で経路制御を行う必要がある。本研究では、まず資源予約ネットワークのアーキテクチャを定義し、保証型サービスに対する経路制御機構を従来の最善努力型サービスに対する経路制御機構から独立させた。保証型サービスに対する経路制御機構は、以下のような特徴を持つ。

- 経路制御プロトコルは、資源予約機構に対して経路を提供し、資源予約機構が経路制御テーブルを管理する。
- 経路制御テーブルは、フロー毎に経路を記述する。

そして、このアーキテクチャに基づく経路制御プロトコルとして RRR を提案した。RRR は、以下のような特徴を持つ。

- 資源予約プロトコルに対してフロー毎の経路を提供する。
- リンクステート型の経路制御情報を用いて、TOS メトリックによる経路選択を行う。
- 経路の計算は始点ノードで行い、経路上のノードに対して経路情報を配布する。
- 利用可能なネットワーク資源の変化に応じてネットワーク資源情報を交換し、経路選択時に考慮する。
- 代替経路を提供する。
- IP マルチキャスト経路を提供する。

RRR は、経路制御テーブルに対して経路を設定するのではなく、資源予約プロトコルに対して経路の提供を行う。経路制御テーブルは資源予約プロトコルによって管理される。RRR を経路制御テーブルから分離させることで、RRR は柔軟な経路選択を行うことができる。また資源予約プロトコルは、自らが経路の設定を行うことで、資源予約を行った経路上に確実にフローを転送することができる。

また RRR は資源予約プロトコルや他の経路制御プロトコルとの協調を考慮しており、より少ないオーバヘッドで新たな経路制御プロトコルの導入が可能である。資源予約ネットワークにおいて、資源予約プロトコルと経路制御プロトコルは密接な関係にあり、本来これらは一体となって考えられるべきである。本研究は、その 1 つの指針を示したと言える。

本研究では、RRR を RRRD として実装を行ったが、実際に資源予約ネットワークが運用されていない現在、その評価は基本的な部分のみに留まっている。今後、資源予約ネットワークが構築、運営されていく過程で、RRR の有効性や問題点について、さらに評価検討を行う必要があるものと考えられる。

謝辞

本研究の機会を与えられ、また日頃から懇切丁寧な御指導をいただきました慶應義塾大学理工学部 客員教授 所 真理雄 博士、慶應義塾大学理工学部 助教授 天野 英晴 博士に深く心から感謝致します。

また本研究に関し、貴重な御示唆を頂いた (株) ソニーコンピュータサイエンス研究所の塩野崎 敦 博士、慶應義塾大学大学院理工学部研究科 計算機科学専攻博士課程 福田 健介 氏、同専攻修士課程 久曽神 宏 氏、矢崎 啓之 氏、そして慶應義塾大学理工学部所研究室の皆様に深く感謝致します。

そのほか、(株) ピーデーの川俣 晶 様には実験機材を御提供いただき、また白百合女子大学大学院文学研究科 児童文学専攻博士課程の鈴木 宏枝 様には、英文の査読を引き受けていただきました。この場を借りて御礼を申し上げます。

最後に、学生生活を支えてくれた家族、そして公私に渡り多くの励ましをくださってすべての方々に感謝の意を捧げます。

1998 年 2 月

Appendix A

予備シミュレーション

本研究では、経路制御プロトコル RRR の設計にあたって、どのようなアルゴリズムによってネットワーク資源情報を把握するのが妥当であるかを調査するため、予備シミュレーションを行った。

A.1 目的

利用可能な帯域幅や平均遅延時間などの QoS 経路制御に必要なネットワーク資源情報は、トラフィックの変化によって逐次状態が更新されるため、従来の経路制御情報に比べて更新頻度は非常に高い。一般にコンピュータネットワークなどの分散環境において、更新頻度の高い情報を交換することは、帯域幅や CPU 資源の消費などのオーバヘッドを伴う。しかし、オーバヘッドを抑えるために情報交換の頻度を低くすると、正確な情報の把握が難しくなる。

すなわち QoS 経路制御において、正確なネットワーク資源情報の獲得と情報交換に必要なコストはトレードオフの関係にあり、どのようにネットワーク資源情報を交換するかは、資源予約ネットワークにおける経路制御プロトコルの性能を左右すると言っても過言ではない。本研究で提案する経路制御プロトコル RRR の設計においても、ネットワーク資源情報の交換についての問題は大変重要である。

本来であれば、実際の資源予約ネットワークのトポロジやトラフィックパターンから、詳細な検討を行うのが望ましい。しかし、現在資源予約ネットワークはまだ研究段階であり、実際のネットワークにおけるトポロジやトラフィックパターンなどのデータを収集することは不可能である。そこで本研究ではシミュレーションにより、ネットワーク資源情報を把握するための幾つかのアルゴリズムを、適切な経路選択と情報交換のコストの観点から比較検討を行った。

A.2 設定

A.2.1 シミュレーション条件

資源予約要求

本研究で対象とする資源予約ネットワークでは、通信の開始時に資源予約要求を行なう。したがって、シミュレーションに用いるトラフィックパターンとは、資源予約要求の発生パターンということになる。資源予約ネットワークにおける資源予約要求の発生モデルとして、どのようなモデルを想定するのが適切であるか、大変難しい問題である。本シミュレーションでは、資源予約

ネットワークと同様に、通信の開始時に回線の確保を行う回線交換ネットワークである電話網の呼の発生モデルを参考にしながら、以下のような条件で資源予約要求を発生させることとした。

- 資源予約要求の発生間隔

  資源予約要求の発生間隔は指数分布に従うものとし、平均発生間隔をパラメータとして用いる。

- 資源予約の継続時間

  資源予約の継続時間は指数分布に従うものとし、平均継続時間をパラメータとして用いる。

- 資源予約要求のトラフィックパラメータ

  資源予約要求のトラフィックパラメータは、帯域幅に関するパラメータのみで、すべての資源予約要求は同じ帯域幅 1 単位を確保するものとした。したがって、本シミュレーションの経路制御において考慮するネットワーク資源も、リンクの帯域幅のみとする。

- 資源予約要求の分布

  資源予約要求は単方向のユニキャストに限定する。また資源予約を行うフローの送信ノードと受信ノードの組は、全てのノードからランダムに選択する。ただし、送信ノードと受信ノードが同じノードになる資源予約要求はないものとする。

- 資源予約要求の再試行

  ネットワーク資源の不足により資源予約要求が失敗しても、再試行は行わない。

なお資源予約要求はフローの始点ノードにおいて発生し、始点ノードから終点ノードに向けてホップバイホップで資源予約が行われるものとした。

ネットワークトポロジ

資源予約要求の発生パターンと同様、資源予約ネットワークがどのようなネットワークトポロジになるのかを想定することは非常に難しい。ネットワークトポロジの選択によっては、シミュレーションの結果に大きな影響を与える。本シミュレーションでは、既存インターネットの延長として資源予約ネットワークを捉えるという観点から、RRR が対象とする既存の基幹ネットワークに近いトポロジを想定するのが望ましいと考えた。そこで、インターネットの基幹ネットワークの 1 つである WIDE*バックボーンのトポロジを参考にすることとした。シミュレーションを行う基本ネットワークトポロジを図 A.1に示した。図中のリンクに示された値は、シミュレーションで設定する帯域幅である。それぞれのリンクは双方向リンクであり、各方向が独立に資源予約されるものと考える。またノード間のメッセージ交換には、1 ホップにつき 1 単位時間かかるものとした。

経路制御アルゴリズム

本シミュレーションで用いる経路制御アルゴリズムは RRR に準じ、リンクステート型の経路制御情報によりネットワークのトポロジ情報を把握し、始点ノードにおいて SPF ツリーを計算し

*WIDE Project http://www.wide.ad.jp/

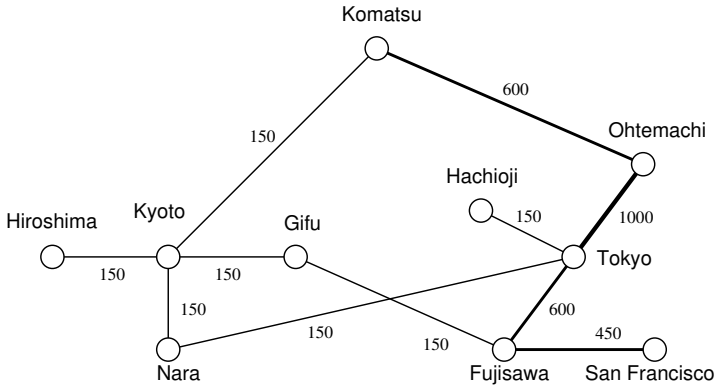


図 A.1: ネットワークトポロジ

経路の決定を行う。リンクのメトリックは、$1000 \div$ (現在リンクで使用可能であると考えられる帯域幅) として設定する。また始点ノードから終点ノードに向けて、経路上の各ノードに対して経路情報の配布を行う。経路情報が終点ノードに到達した時点で、終点ノードは確認応答を始点ノードに返すものとし、始点ノードに確認応答が到着した時点で、経路が利用できるものとする。

A.2.2 アルゴリズム

本シミュレーションでは、あるリンクにおいて現在利用可能な帯域幅をネットワーク資源情報とし、その収集に関して以下の 4 つのアルゴリズムを比較対象とした。

- Birds Eye

  ある時点におけるすべてのネットワーク資源情報を完全に把握することができる (故にネットワーク資源情報の交換は行わない)。これは、他のアルゴリズムとの比較のためのアルゴリズムとして設定している。

- Reservation State Flooding

  資源予約要求の発生毎に、各ノードが他のすべてのノードに対して、自ノードの隣接リンクのネットワーク資源情報をフラッディングする。すなわち情報更新の頻度は、資源予約要求の頻度に依存する。

- Timer State Flooding

  一定時間毎に各ノードが他のすべてのノードに対して、自ノードの隣接リンクのネットワーク資源情報をフラッディングする。すなわち情報更新の頻度は、時間に依存する。なおシミュレーションでは、フラッディング間隔について 300, 1,500, 3,000 単位時間の 3 種類を設定した。

- Pickup Block

  ネットワーク資源の不足により資源予約要求が失敗した場合のみ、失敗を検知したノードが資源予約要求を行った始点ノードに対して、自ノードの隣接リンクのネットワーク資源

情報を送信する。すなわち情報更新の頻度は、ネットワーク資源の変化 (不足状態) に依存する。

A.2.3 評価項目

本シミュレーションでは、適切な経路選択と情報交換のコストの観点から比較検討を行うため、評価項目として以下の 2 項目を取り上げた。

- ネットワークスループット

  ネットワーク全体で、実際にどの程度資源予約が行うことができたかを示す。成功したすべての資源予約について、(資源予約経路のホップ数) × (資源予約帯域幅) × (資源予約継続時間) の総和を求めた。適切な経路を選択した場合は、ネットワークのスループットは向上し、ネットワークが効率良く使われていることがわかる。逆に不適切な経路を選択により資源予約の失敗が多くなると、ネットワークのスループットは低下することになる。

- メッセージコスト

  ネットワークの全てのノードで、ネットワーク資源情報及び経路情報や確認応答を交換したメッセージの数を示す。隣接ノードに対して 1 つのメッセージを送信した場合を 1 として、その総和を求めた。メッセージコストが低いほど、ネットワーク資源情報や経路情報の交換によってノードやリンクに与える負荷は小さいと考えられる。

A.2.4 評価条件

本シミュレーションでは、以下の 2 つの条件が変化した場合についての評価を行った。

- 平均資源予約間隔の変化

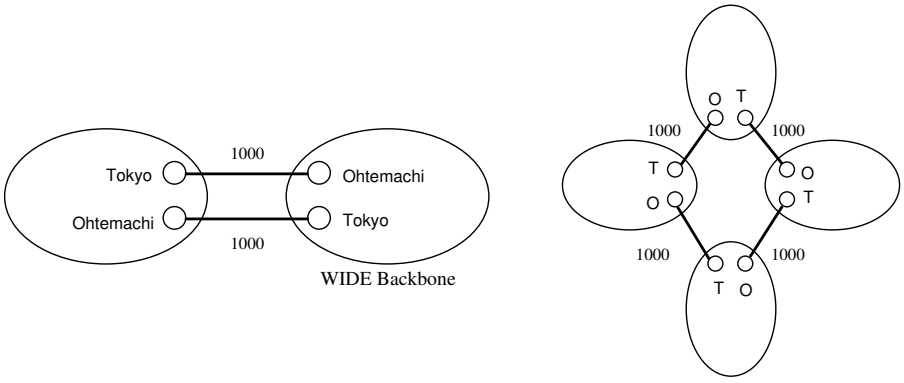
  資源予約の平均継続時間を固定し、平均発生間隔 (平均資源予約間隔) を変化させた場合の、それぞれのアルゴリズムにおけるネットワークスループット及びメッセージコストを比較した。これは、トラフィックに対する規模拡張性を評価するものである。

  シミュレーションでは、資源予約の平均継続時間を 10,000 単位時間、平均発生間隔を 100, 120, 140, 160, 180, 200 単位時間としたときの資源予約要求列を 100,000 単位時間分生成し、この資源予約要求列をそれぞれのアルゴリズムに適用して結果を求めた。

- ネットワーク規模の変化

  資源予約の平均継続時間及び平均発生間隔を固定し、ネットワーク規模を変化させた場合の、それぞれのアルゴリズムにおけるネットワークスループット及びメッセージコストを比較した。ネットワーク規模は、図 A.1に示した基本ネットワーク (10 ノード) を、図 A.2 のように幾つか組み合わせることで変化させた (20 ノード、40 ノード)。これは、ネットワークトポロジに対する規模拡張性を評価するものである。

  シミュレーションでは、資源予約の平均継続時間を 10,000 単位時間、平均発生間隔を 200 単位時間としたときの資源予約要求列を、ネットワークトポロジ毎に 100,000 単位時間分生成し、この資源予約要求列をそれぞれれのアルゴリズムに適用して結果を求めた。



(a) 20 nodes (b) 40 nodes

図 A.2: ネットワーク規模の変化

A.3 結果

平均資源予約間隔の変化させた場合のシミュレーション結果を図 A.3、図 A.4に、ネットワーク規模を変化させた場合のシミュレーション結果を図 A.5、図 A.6に示した。ネットワークスループットに関するグラフは、Birds Eye の値を 1 とした場合の各アルゴリズムの値の比を示している。なお Timer State Flooding 項目の括弧内に書かれた数字は、フラッディングの時間間隔を示している。また図 A.4の Reservation State Flooding の項目は、比較できないほど値が大きいために省略している。

A.4 検討

平均資源予約間隔を変化させた場合について

ネットワークスループットについては、フラッディング間隔が最も短い 300 単位時間の Timer State Flooding が最も良い結果となった。ネットワーク資源情報のフラッディング間隔が短いほど、 広告されたネットワーク資源情報と実際のネットワーク資源状態との差が少なくなるため、適切な経路を選択できるものと考えられる。しかしメッセージコストをみると、当然フラッディング間隔が短い Timer State Flooding ほど値は大きい。さらに平均資源予約間隔が短くなるにつれて、急速にメッセージコストが上昇していることがわかる。Timer State Flooding では、ネットワーク資源情報は資源予約要求に関係なく一定時間毎にしか交換されないので、このメッセージコストの上昇分は、ネットワーク資源情報の交換によるものではなく、経路変更に伴う経路情報の配布によるものであることがわかる。ネットワーク資源情報を頻繁に更新すると、当然経路も頻繁に更新される。特に資源予約間隔が短くなりトラフィックの変動が激しくなると、その影響を強く受けるものと考えられる。これは経路の安定性を考えた場合、好ましくない。

一方、Pickup Block に注目してみると、非常に少ないメッセージコストで、3000 単位時間の Timer State Flooding と同じかそれよりもよいネットワークスループットになっている。Pickup Block は、資源予約要求が失敗した時点で初めてネットワーク資源情報を交換する。したがって、ネットワーク資源に余裕があるときは、ネットワーク資源情報を交換しないため、メッセージコ

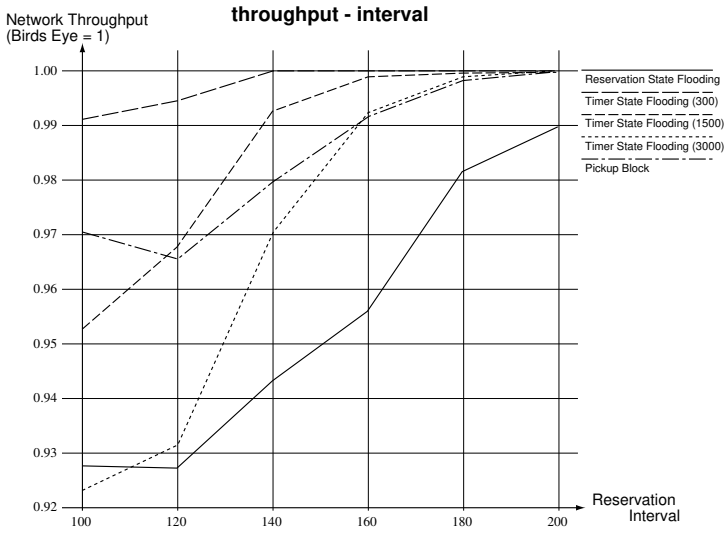


図 A.3: 平均資源予約間隔の変化に対するネットワークスループット

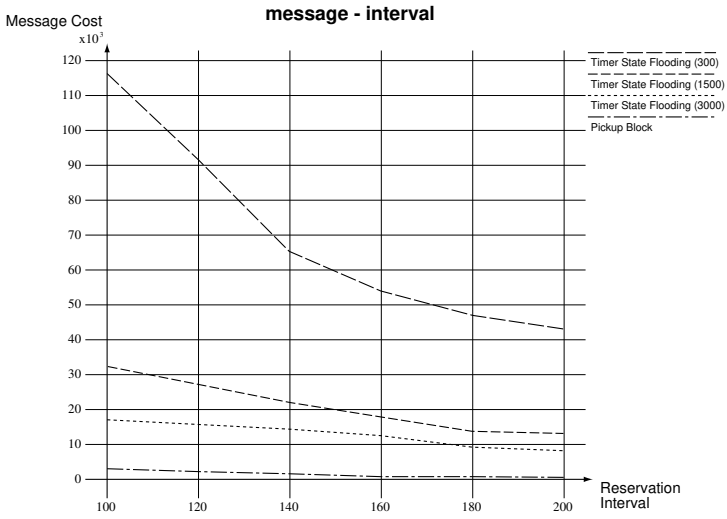


図 A.4: 平均資源予約間隔の変化に対するメッセージコスト

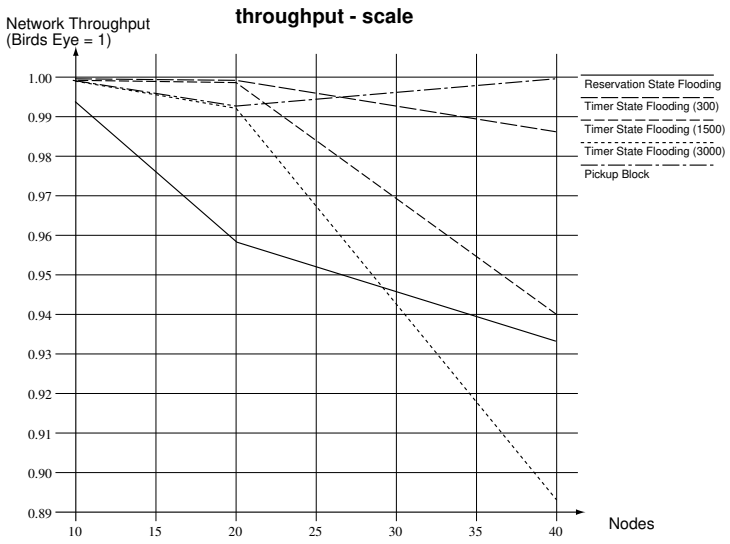


図 A.5: ネットワーク規模の変化に対するネットワークスループット

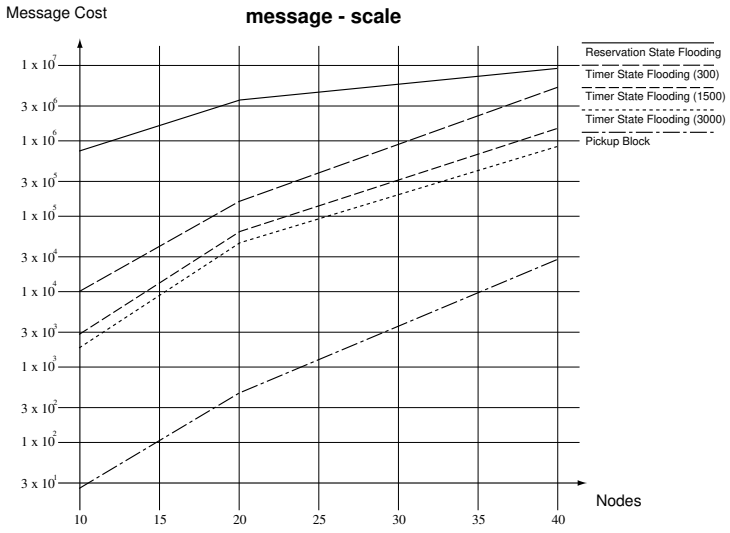


図 A.6: ネットワーク規模の変化に対するメッセージコスト

ストが抑えられる。逆に Timer State Flooding のように一定時間毎にネットワーク資源情報を交換するようなアルゴリズムの場合、前回に広告したネットワーク資源情報と変化があるかどうかに関わらず新しいネットワーク資源情報の広告を行なうので、無駄が多いと言える。また Pickup Block では、ネットワーク資源情報をフラッディングするのではなく、資源予約要求を行った始点ノードに対してのみ送信するので、ネットワーク資源情報の更新の影響を受けるノードが限定される。したがって、経路変更を行うノードも限定されることになり、経路情報の配布によるメッセージコストの上昇も抑えることができるものと考えられる。

ネットワーク規模を変化させた場合について

ネットワークスループットについてみると、全般に Timer State Flooding はネットワークの規模が大きくなるにつれて、他のアルゴリズムよりもスループットが低下が著しい。これは、ネットワーク資源情報のフラッディングによるトラフィックの集中が起きているためと考えられる。Timer State Flooding では、ネットワーク資源情報をすべてのノードに対してフラッディングするため、すべてのノードが空いているリンクを経路として選択してしまう可能性がある。ネットワーク規模が大きくなればなるほど、多くのノードが同じリンクを経路として選択するため、フローが集中し資源予約要求が失敗する可能性が高くなる。一方、Pickup Block においてスループットが低下しにくいのは、ネットワーク資源情報のフラッディングを行わないため、経路の選択を分散させ、フローの集中をうまく避けることができるためと考えられる。すなわち、Pickup Block のアルゴリズムは、ネットワーク規模の変化に対して適応しやすいと考えられる。

なお Reservation State Flooding については、いずれの場合もメッセージコストが大きするため、現実的なアルゴリズムとしては考えにくいと言える。

まとめ

以上の検討をまとめると、

- ネットワーク資源情報はフラッディングではなく、特定ホストに対して送信する方が規模拡張性に優れている。
- 一定時間毎の情報更新など時間に依存した情報更新よりも、ネットワーク資源の変化に応じた情報更新の方が効果的である。

と言える。

Appendix B

RRR パケットフォーマット

本研究で提案した経路制御プロトコル RRR のパケットフォーマットを解説する。

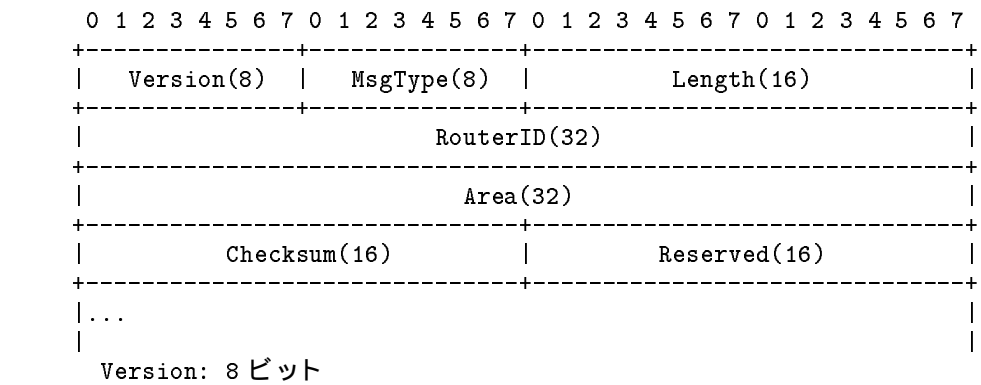
メッセージ

RRR のパケットには、以下の 8 種類のメッセージがある。

- 生存 (Hello) メッセージ
- 経路要求/経路消去要求/代替経路要求メッセージ
- 経路情報メッセージ
- 確認応答メッセージ
- 経路消去メッセージ
- グループ情報メッセージ
- アラートメッセージ
- OSPF トポロジデータベース更新メッセージ

それぞれのメッセージは、共通ヘッダと各メッセージ毎のペイロードに分けられる。以下、詳細を示す。

共通ヘッダ



RRR のバージョン番号を示す。現在のバージョンは 1 である。

MsgType: 8 ビット

メッセージの種類を示す。現在、以下の値が定義されている。

OSPF	(0)	OSPF トポロジデータベース更新メッセージ
HELLO	(1)	生存 (Hello) メッセージ
INSTALL	(2)	経路情報メッセージ
INSTALL_ACK	(3)	確認応答メッセージ
FLUSH	(4)	経路消去メッセージ
ALERT	(5)	アラートメッセージ
REQUEST	(6)	経路要求/経路消去要求/代替経路要求メッセージ
MGROUP	(7)	グループ情報メッセージ

Length: 16 ビット

ヘッダ部分を含めたメッセージの全体の長さをオクテット単位 (~ 65535) で記述する。

RouterID: 32 ビット

メッセージ送信ノードの識別子を記述する。

Area: 32 ビット

RRR エリア識別子を記述する。

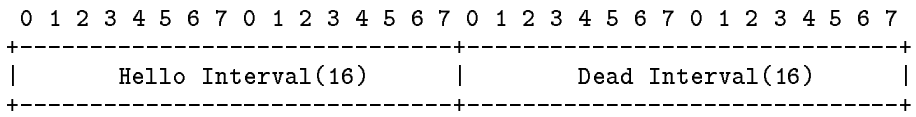
Checksum: 16 ビット

ヘッダ部分を含めたメッセージ全体のチェックサムで、IP ヘッダのチェックサムと同様のルーチンにより生成する。

Reserved: 16 ビット

(未使用)

生存 (Hello) メッセージ



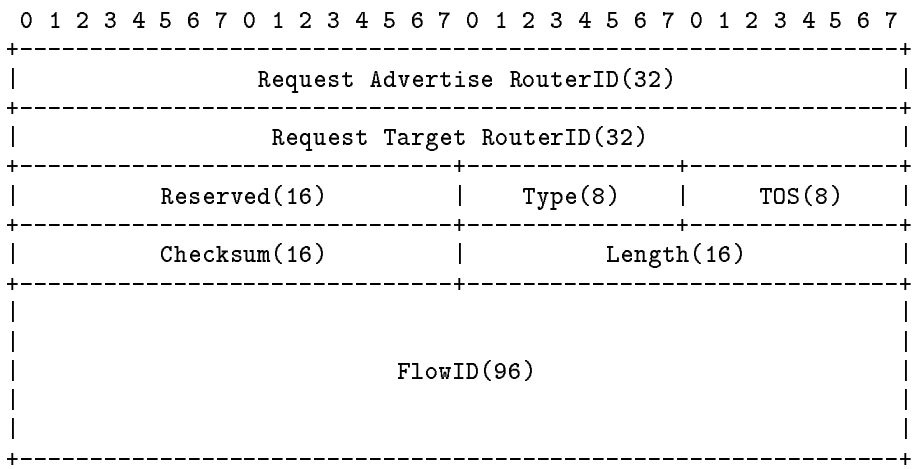
Hello Interval: 16 ビット

生存メッセージの送信間隔を秒単位で記述する。

Dead Interval: 16 ビット

生存メッセージの最大生存時間を秒単位で記述する。

経路要求/経路消去要求/代替経路要求メッセージ



Request Advertise RouterID: 32 ビット

メッセージを生成したノードの識別子を記述する。

Request Target RouterID: 32 ビット

メッセージの宛先ノードの識別子を記述する。

Reserved: 16 ビット

(未使用)

Type: 8 ビット

メッセージの種類を示す。現在、以下の値が定義されている。
INSTALL (1) 経路要求メッセージ
FLUSH (2) 経路消去要求メッセージ
REDIRECT (3) 代替経路要求メッセージ

TOS: 8 ビット
経路要求の際の TOS コードを記述する。

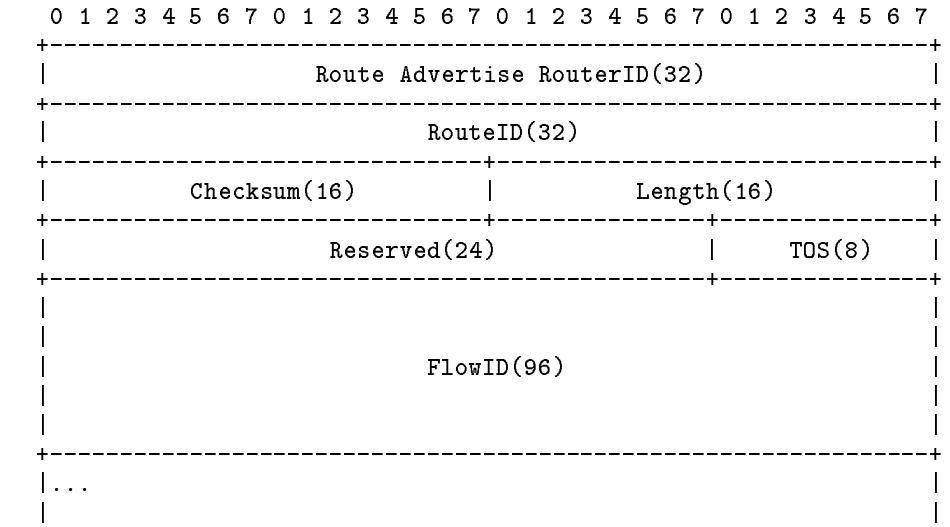
Checksum: 16 ビット
ペイロード部分のチェックサムで、IP ヘッダのチェックサムと同様のルーチンにより生成する。

Length: 16 ビット
ペイロード部分の長さをオクテット単位 (~ 65535) で記述する。

FlowID: 96 ビット
フロー識別子を記述する。

経路情報メッセージ

<基本情報> : 経路情報メッセージのヘッダ



Route Advertise RouterID: 32 ビット
経路情報メッセージを生成したノードの識別子を記述する。

RouteID: 32 ビット
経路識別子を記述する。

Checksum: 16 ビット
ペイロード部分のチェックサムで、IP ヘッダのチェックサムと同様のルーチンにより生成する。

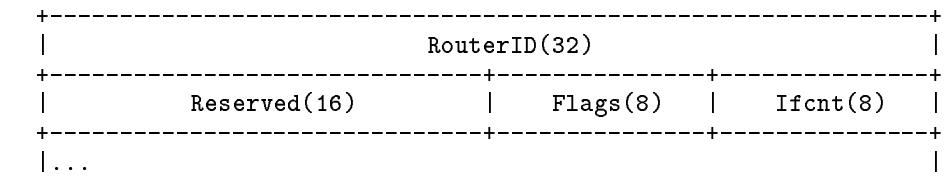
Length: 16 ビット
ペイロード部分の長さをオクテット単位 (~ 65535) で記述する。

Reserved: 24 ビット
(未使用)

TOS: 8 ビット
経路の TOS コードを記述する。

FlowID: 96 ビット
フロー識別子を記述する。

<ノード情報> : 経路上の各ノードに対する情報



```
|                                                               |
```

RouterID: 32 ビット
    ノード識別子を記述する。
Reserved: 16 ビット
    (未使用)
Flags: 8 ビット
    現在、未使用である。
Ifcnt: 8 ビット
    出力インタフェース数を記述する。

<インタフェース情報> : 経路上の各ノードにおける出力インタフェース情報

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---------------------------------------------------------------+
|                      Interface Address(32)                    |
+---------------------------------------------------------------+
|                      Neighbor RouterID(32)                    |
+---------------------------------------------------------------+
```

Interface Address: 32 ビット
    出力インタフェースの IP アドレスを記述する。
Neighbor RouterID: 32 ビット
    出力インターフェースに対する転送先の隣接ノード識別子を記述する。
    もし隣接ノードが疑似ノードの場合は 0 が記述される。

確認応答メッセージ

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---------------------------------------------------------------+
|                  Route Advertise RouterID(32)                 |
+---------------------------------------------------------------+
|                           RouteID(32)                         |
+-------------------------------+-------------------------------+
|          Checksum(16)         |          Length(16)           |
+-------------------------------+---------------+---------------+
|                  Reserved(24)                 |    Flags(8)   |
+-----------------------------------------------+---------------+
```

Route Advertise RouterID: 32 ビット
    確認応答メッセージの対象となる経路情報メッセージを生成したノードの識別子を
    記述する。
RouteID: 32 ビット
    経路識別子を記述する。
Checksum: 16 ビット
    ペイロード部分のチェックサムで、IP ヘッダのチェックサムと同様のルーチンに
    より生成する。
Length: 16 ビット
    ペイロード部分の長さをオクテット単位 (~ 65535) で記述する。
Reserved: 24 ビット
    (未使用)
Flags: 8 ビット
    確認応答の状態を示す。現在、以下の値が定義されている。

ACK	(0)	成功
UNREACH	(1)	到達不可能
FATAL	(2)	致命的な誤り

経路消去メッセージ

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---------------------------------------------------------------+
|                  Route Advertise RouterID(32)                 |
+---------------------------------------------------------------+
```

```
|                            RouteID(32)                          |
+---------------------------------+---------------------------------+
|          Checksum(16)           |          Length(16)             |
+---------------------------------+---------------+---------------+
|                   Reserved(24)                  |     Type(8)   |
+-------------------------------------------------+---------------+
```

Route Advertise RouterID: 32 ビット
: 経路消去の対象となる経路情報メッセージを生成したノードの識別子を記述する。

RouteID: 32 ビット
: 経路識別子を記述する。

Checksum: 16 ビット
: ペイロード部分のチェックサムで、IP ヘッダのチェックサムと同様のルーチンにより生成する。

Length: 16 ビット
: ペイロード部分の長さをオクテット単位 (~ 65535) で記述する。

Reserved: 24 ビット
: (未使用)

Type: 8 ビット
: メッセージの種類を示す。現在、以下の値が定義されている。

  INITIALIZE (1) RRR 経路エントリ初期化
  DOWNSTREAM (3) 下流方向へ転送
  UPSTREAM   (4) 上流方向へ転送

グループ情報メッセージ

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---------------------------------------------------------------+
|             Multicast Group Advertise RouterID(32)            |
+---------------------------------------------------------------+
|                  Multicast Group Address(32)                  |
+---------------------------------+---------------+---------------+
|           Reserved(16)          |    Type(8)    |  If Type(8)   |
+---------------------------------+---------------+---------------+
|                       Interface Address(32)                   |
+---------------------------------+---------------------------------+
|          Checksum(16)           |          Length(16)             |
+---------------------------------+---------------------------------+
```

Multicast Group Advertise RouterID: 32 ビット
: グループ情報メッセージを生成したノードの識別子を記述する。

Multicast Group Address: 32 ビット
: IP マルチキャストアドレスを記述する。

Reserved: 16 ビット
: (未使用)

Type: 8 ビット
: メッセージの種類を示す。現在、以下の値が定義されている。

  JOIN   (1) マルチキャストグループへ参加
  LEAVE   (2) マルチキャストグループから離脱
  FLUSH   (3) グループ情報データベース初期化 (消去)
  SOLICIT (4) グループ情報データベース初期化 (収集)

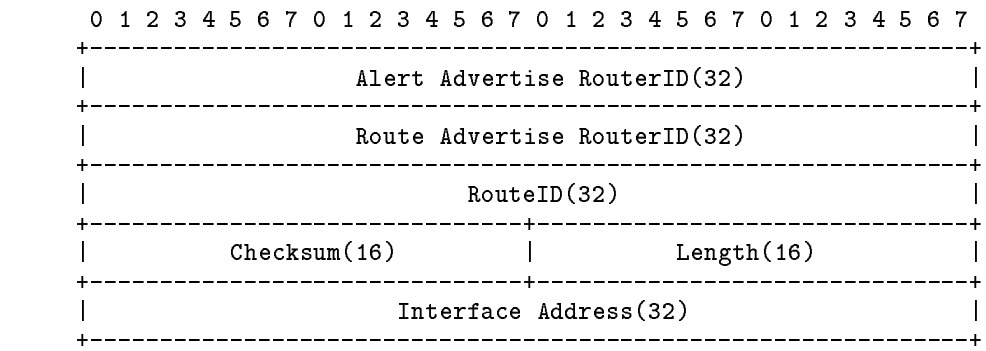
If Type: 8 ビット
: マルチキャストグループに参加するインタフェースの種類を示す。

Interface Address: 32 ビット
: マルチキャストグループに参加するインタフェースの IP アドレスを示す。

Checksum: 16 ビット
: ペイロード部分のチェックサムで、IP ヘッダのチェックサムと同様のルーチンにより生成する。

Length: 16 ビット
: ペイロード部分の長さをオクテット単位 (~ 65535) で記述する。

アラートメッセージ



Alert Advertise RouterID: 32 ビット
: アラート情報メッセージを生成したノードの識別子を記述する。

Route Advertise RouterID: 32 ビット
: アラートが発生した経路の経路情報メッセージを生成したノードの識別子を記述する。

RouteID: 32 ビット
: 経路識別子を記述する。

Checksum: 16 ビット
: ペイロード部分のチェックサムで、IP ヘッダのチェックサムと同様のルーチンにより生成する。

Length: 16 ビット
: ペイロード部分の長さをオクテット単位 (~ 65535) で記述する。

Interface Address: 32 ビット
: アラートの発生した経路のインタフェースの IP アドレスを記述する。

OSPF トポロジデータベース更新メッセージ

ペイロード部分はもたない。

Appendix C

RSRR 拡張メッセージフォーマット

本研究で提案した経路制御プロトコル RRR は、資源予約プロトコルとのインタフェースとして RSRR を拡張して使用している。ここでは [45] で定義されている以外に独自に拡張した RSRR メッセージについて、メッセージフォーマットを示す。

RSRR 拡張メッセージ

RRR で使用するために独自に拡張した RSRR メッセージは、以下の通りである。

- Route Query Flow

  フローに対する経路を問い合わせる。

- Route Reply Flow

  フローに対する経路を提供する。

- Route Flush

  経路を消去する。

- Route Alert

  アラート発生の通知及び代替経路の要求を行う。

それぞれのメッセージは、共通ヘッダと各メッセージ毎のペイロードに分けられる。以下、詳細を解説する。

RSRR 共通ヘッダ

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---------------+---------------+---------------+---------------+
| Version(8)    | Type(8)       | Flags(8)      | Num(8)        |
+---------------+---------------+---------------+---------------+
|...                                                            |
|                                                               |
```

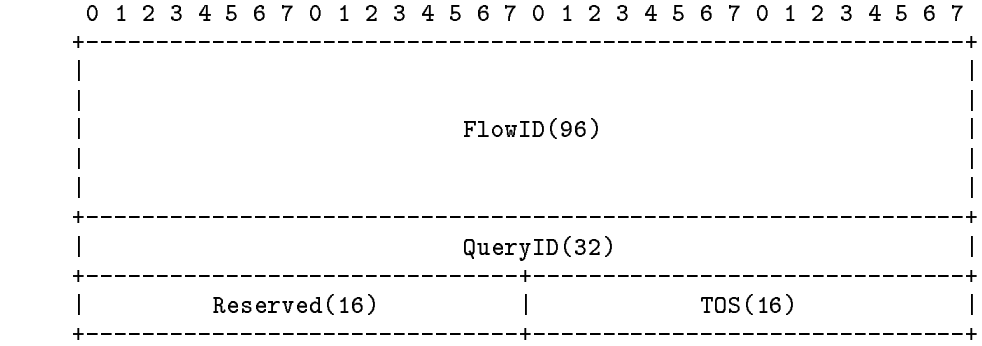
Version: 8 ビット
RSRR のバージョン番号を示す。現在のバージョンは 1 である。
Type: 8 ビット
メッセージの種類を示す。拡張メッセージの値は以下の通りである。
Route Query Flow (5)

```
       Route Reply Flow (6)
       Route Flush      (7)
       Route Alert      (8)
```
Flags: 8 ビット
　各メッセージ毎に定義される。
Num: 8 ビット
　各メッセージ毎に定義される。

Route Query Flow

<共通ヘッダ>
Flags: 8 ビット
　（未使用）
Num: 8 ビット
　（未使用）

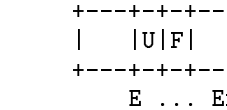
<ペイロード>



FlowID: 96 ビット
　フロー識別子を記述する。
QueryID: 32 ビット
　資源予約プロトコルのための問い合わせ識別子を記述する。
Reserved: 16 ビット
　（未使用）
TOS: 16 ビット
　経路要求の際の TOS コードを記述する。

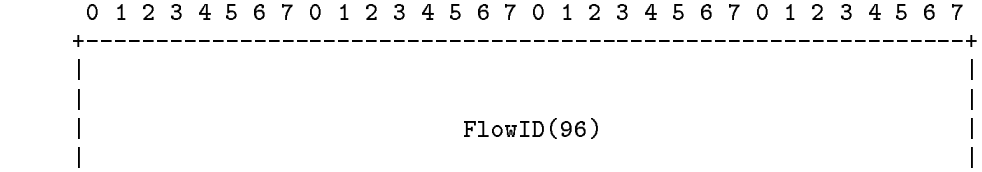
Route Reply Flow

<共通ヘッダ>
Flags: 8 ビット



　E ... Error Bit エラー通知を示す。
　F ... Fatal Error Bit RRRD での致命的な誤り
　U ... Unreachble Bit 終点ノードアドレスに到達不可能
Num: 8 ビット
　出力インタフェースの数を記述する。

<ペイロード - 基本情報>



```
|                                                               |
+---------------------------------------------------------------+
|                           QueryID(32)                         |
+---------------------------------------------------------------+
|                Incoming Interface Address(32)                 |
+---------------------------------------------------------------+
|...                                                            |
|                                                               |
```

FlowID: 96 ビット
フロー識別子を記述する。
QueryID: 32 ビット
資源予約プロトコルのための問い合わせ識別子を記述する。
Incoming Interface Address: 32 ビット
入力インタフェースの IP アドレスを記述する。

<ペイロード - インタフェース情報>

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---------------------------------------------------------------+
|                 Outgoing Interface Address(32)                |
+---------------------------------------------------------------+
|                     Next Hop Address(32)                      |
+---------------------------------------------------------------+
```

Outgoing Interface Address: 32 ビット
出力インタフェースの IP アドレスを記述する。
Next Hop Address: 32 ビット
出力インタフェースに対応する転送先ノードの IP アドレスを記述する。

Route Flush

<共通ヘッダ>
Flags: 8 ビット
(未使用)
Num: 8 ビット
(未使用)

<ペイロード>

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---------------------------------------------------------------+
|                                                               |
|                                                               |
|                           FlowID(96)                          |
|                                                               |
|                                                               |
+---------------------------------------------------------------+
```

FlowID: 96 ビット
フロー識別子を記述する。

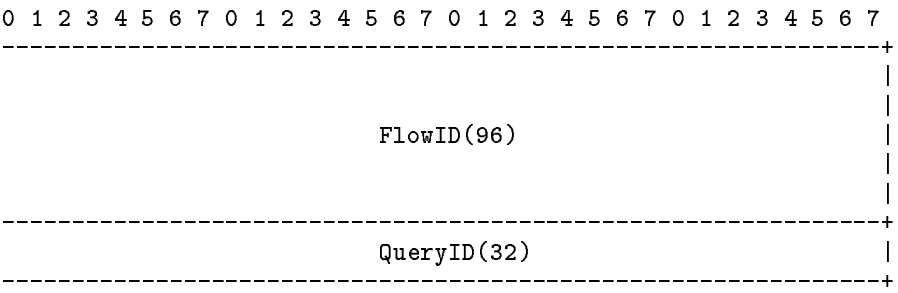
Route Alert

<共通ヘッダ>
Flags: 8 ビット

```
 0 1 2 3 4 5 6 7
+-------------+-+
|             |A|
+-------------+-+
```

A ... Alternative Route Bit 代替経路要求を示す。
Num: 8 ビット
(未使用)

<ペイロード>



FlowID: 96 ビット
フロー識別子を記述する。
QueryID: 32 ビット
資源予約プロトコルのための問い合わせ識別子を記述する。

参 考 文 献

[1] A. Banerjea and D. Ferrari and B. A. Mah and M. Moran and D. C. Verma and H. Zhang. The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences. Technical Report TR-94-059, University of California at Berkeley and International Computer Science Institute, Nov. 1994.

[2] P. Almquist. Type of Service in the Internet Protocol Suite, Jul. 1992. Internet RFC1349.

[3] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview, Jul. 1994. Internet RFC1633.

[4] R. Braden, L. Zhang, L. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, Sep. 1997. Internet RFC2205.

[5] L. Breslau and D. Estrin. Design of Inter-Administrative Domain Routing Protocol. In *Proceedings of SIGCOMM '90*, Sep. 1990.

[6] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A Framework for Multiprotocol Label Switching, Nov. 1997. Internet-Draft draft-ietf-mpls-framework-02.txt *work in progress.*

[7] D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Application in an Integrated Services Packet Networl: Architecture and Mechanism. In *Proceedings of SIGCOMM '92*, Aug. 1992.

[8] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QOS-based Routing in the Internet, Jul. 1997. Internet-Draft draft-ietf-qosr-framework-01.txt *work in progress.*

[9] B. Davie, Y. Rekheter, E. Rosen, A. Viswanathan, and V. Srinivasan. Use of Label Switching With RSVP, Nov. 1997. Internet-Draft draft-davie-mpls-rsvp-01.txt *work in progress.*

[10] S. Deering and R. Hidden. Internet Protocol, Version 6 (IPv6) Specification, Jul. 1997. Internet-Draft draft-ietf-ipngwg-ipv6-spec-v2-00.txt *work in progress.*

[11] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proceedings of SIGCOMM '89*, Sep. 1989.

[12] H. Esaki, M. Ohta, and K. Nagami. High Speed Datagram Delivery over Internet using ATM Technology. *IEICE Trans. Communicasions, Vol.E78-B, No.8*, Aug. 1995.

[13] D. Estrin, Y. Rekhter, and S. Hotz. Scalable Inter-Domain Routing Architecture. In *Proceedings of SIGCOMM '92*, Aug. 1992.

[14] G. R. Ash et al. Real-Time Network Routing in a Dynamic Class-of-Service Network. In *Proceedings of the 13th Int'l Teletraffic Congress*, 1991.

[15] Inoue et al. A State- and Time-Dependent Dynamic Routing Schemes for Telephone Networks. In *Proceedings of the 13th Int'l Teletraffic Congress*, June 1991.

[16] W. Fenner. Internet Group Management Protocol, Version 2, Nov. 1997. Internet-Draft draft-ietf-idmr-igmp-v2-08.txt *work in progress.*

[17] S. Floyd and V. Jacobson. Link-sharing and Resource Management Models for Packet Networks. *IEEE Trans. on Networking*, Aug. 1995.

[18] ST2 Working Group. Internet Stream Protocol, Version 2 (ST-2) Protocol Specification - Version ST2+, Aug. 1995. Internet RFC1819.

[19] R. Guerin, S. Kamat, and S. Herzog. QoS Path Management with RSVP, Mar. 1997. Internet-Draft draft-guerin-qospath-mgmt-rsvp-00.txt *work in progress.*

[20] R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions, Mar. 1997. Internet-Draft draft-guerin-qos-routing-ospf-01.txt *work in progress.*

[21] F. Hoymany and D. Mosse. Switch-Borne Router for High Performance Packet Forwarding of Connectionless Traffic in ATM Networks. In *Proceedings of ICCCN '96*, Oct. 1996.

[22] S. Jamin, S. Shenker, L. Zhang, and D. D. Clark. An Admission Control Algorithm for Predictive Real-Time Service. In *Proceedings of 3rd International Workshop on Netowrok and Operating System Support for Digital Audio and Video*, Nov. 1992.

[23] S. Keshav. *An Engineering Approach to Computer Networking.* Addison-Wesley Publishing Company, 1997.

[24] C. Medrick. Routing Information Protocol, Jun. 1988. Internet RFC1058.

[25] D. J. Mitzel, D. Estrin, S. Shenker, and Lixia Zhang. A Study of Reservation Dynamics in Integrated Services Packet Networks. In *Proceedings of IEEE Infocom '96*, Mar. 1996.

[26] J. Moy. Multicast Extensions to OSPF, Mar. 1994. Internet RFC1584.

[27] J. Moy. OSPF Version 2, Mar. 1994. Internet RFC1583.

[28] P. Newman, T. Lyon, and G. Minshall. Flow Labelled IP: A Connectionless Approach to ATM. In *Proceedings of IEEE Infocom '96*, Mar. 1996.

[29] C. Partridge. A Proposed Flow Specification, Sep. 1992. Internet RFC1363.

[30] C. Partridge. *Gigabit Networking.* Addison-Wesley Publishing Company, 1994.

[31] G. Parulkar, D. C. Schmidt, and J. S. Turner. IP on ATM: a Strategy for Integrating IP with ATM. In *Proceedings of SIGCOMM '95*, Aug. 1995.

[32] R. Perlman. *Interconnections*. Addison-Wesley Publishing Company, 1992.

[33] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4), Mar. 1995. Internet RFC1771.

[34] E. C. Rosen. Exterior Gateway Protocol, Oct. 1982. Internet RFC827.

[35] E. Rosen, A. Viswanathan, and R. Callon. A Proposed Architecture for MPLS, Aug. 1997. Internet-Draft draft-ietf-mpls-arch-00.txt *work in progress*.

[36] H. F. Salama, D. S. Reeves, and Y. Vinitotis. A Distributed Algorithm for Delay-Constrained Unicast Routing. In *Proceedings of IEEE Infocom '97*, Apr. 1997.

[37] S. Shenker and J. Wroclawski. General Characterization Parameters for Inegrated Service Network Elements, Sep. 1997. Internet RFC2215.

[38] R. R. Stacey and D. J. Songhurst. Dynamic Alternative Routing in the British Telecom Trunk Reservation. In *Proceedings of Int'l Switching Symposium '87*, 1987.

[39] M. Steenstrup. An Architecture for Inter-Domain Policy Routing, Jun. 1993. Internet RFC1478.

[40] M. Steenstrup. IDPR as a Proposed Standard, Jul. 1993. Internet RFC1477.

[41] D. M. Topkis. A k Shortest Path Algorithms for Adaptice Routing in Communication Networks. *IEEE Trans. on Communications*, Jul. 1988.

[42] Z. Wang and J. Crowcroft. Shortest Path First with Emergency Exits. In *Proceedings of SIGCOMM '90*, Sep. 1990.

[43] Z. Wang and J. Crowcroft. Analysis of Shortest-Path Routing Algorithms in a Dynamic Network Environment. *Computer Communication Review Vol.22 No.2*, Apr. 1992.

[44] Z. Wang and J. Crowcroft. Quality of Service Routing for Supporting Multimedia Applications. *IEEE Journal on Selected Areas in Communications*, Sep. 1996.

[45] D. Zappala. RSRR: A Routing Interface For RSVP, Nov. 1996. Internet-Draft draft-ietf-rsvp-routing-01.txt *work in progress*.

[46] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP:A New Resource ReSerVation Protocol. *IEEE NETWORK MAGAZINE*, Sep. 1993.

[47] Z. Zhang, C. Sanchez, B. Salkewics, and E. Crawley. Quality of Service Extension to OSPF or Quality Of Service Path First Routing (QOSPF), Jun. 1996. Internet-Draft draft-zhang-qos-ospf-00.txt *work in progress*.

[48] 筒井章博, et al. ASP 仕様書. AMInet Project, Jan. 1998.

[49] 所真理雄, 小野定康, 太田直久, 寺岡文男. AMInet：次世代ネットワークアーキテクチャ. 日本ソフトウェア科学会第 14 回全国大会論文集, Oct. 1997.

[50] 塩野崎敦, 三輪泰孝, 寺岡文男. AMInet における ATM と IP の統合によるリアルタイム通信の実現. 日本ソフトウェア科学会第 14 回全国大会論文集, Oct. 1997.