# 次世代インターネット経路制御に関する研究


**指導教員**
**村井 純**

**慶應義塾大学 政策・メディア研究科**
89932543
**小原 泰弘**


2001年 7月 2日

# 修士論文要旨　　２００１年度(平成１３年度)

# 次世代インターネット経路制御に関する研究

本研究では、次世代インターネットの方向性を示し、基盤技術である経路制御機構の実装および広域テストベッドでの運用を行い、その有用性を示した。IPv6によって多くの接続ノードが自由な接続形態でインターネットに接続された場合において、インターネットが強靭な情報基盤として機能する経路制御機構を実現した。本研究の成果は、フリーソフトウェアであるzebraの一部として公開されており、現在世界中の多くのIPv6テストベッドで利用されている。

インターネットはこれからも急速に拡大し、社会基盤として多くの人に使われるようになると予測される。社会基盤としてのインターネットの規模の拡大は、接続ノード数の増加と接続回線の多様化によるネットワークトポロジの複雑化を招き、既存のネットワークの基盤部分に大きな影響を与えることとなる。

特に、最も重要な機能である経路制御機構への影響は非常に深刻で、軽視することはできない。現在の経路制御機構は、故障経路を回避するような障害復旧能力が大幅に不足している。そこで、次世代インターネットを予測し、ネットワークモデルを提案することにより、その状況下における経路制御機構が実現するべき機能を定義した。

次に、モデルにそったネットワークの規模において、経路収束の速度がどれほどになるのかを、現在のネットワークの実測値を用いて考察した。この考察結果から、次世代インターネットの経路制御機構 であるOSPFv3を実装し、改善を行なった。

本論文では、次世代インターネットにおけるネットワークモデルを提案し、経路制御機構の機能について述べた。また、比較と計測ならびに考察により、妥当な経路制御プロトコルを選択し、設計・実装・改善を行った。本研究の成果は、次世代インターネット経路制御の方向性を示すものとなった。

キーワード

1. 次世代インターネット 2. 経路制御 3.OSPF

慶應義塾大学 大学院 政策・メディア研究科

小原 泰弘

# Abstract of Masters's Thesis Academic Year 2001

## Development of Routing Mechanism for Next Generation Internet

In this research, a general outline of next generation Internet is suggested, and its basic technology, a new routing mechanism was designed, implemented and then was experimented on a wide area testbed network. Its efficiency was confirmed by the experiment. By this implementation, the most important technology for IPv6 based Internet was realized. This research is installed as a part of zebra implementation, which is a free software, and is utilized by numerous IPv6 testbed network all over the world.

Internet is projected to expand greatly and be used by many people as social infrastructure. As results, large number of nodes will be connected and various datalinks will be used, which calls complexity of network topology. Existing network technology will be effected greatly by this change.

Current routing mechanism lacks the ability to recover from network failure. This research considers next generation Internet, suggests a new network model, and defines function for the new routing mechanism.

Actual measurement of existing network was used to consider the convergence speed of suggested routing protocol. A new routing mechanism, OSPFv3 was implemented and modified according to this consideration.

As conclusion, a new network model for next generation Internet was suggested, and the requirement for its routing mechanism was defined. Valid routing protocol was designed and implemented through comparison and simulation. Consequently, this research directs the routing mechanism for next generation Internet.

Keywords

1.Next generation Internet 2.Routing Protocol 3.OSPF

Keio University Graduate School of Media and Governance

Yasuhiro Ohara

# 目 次

# 図 目 次

# 表 目 次

# 第1章

# はじめに

本章では、まず本研究の前提となる背景について述べ、次に、本研究の目的を述べる。最後に本論文の構成を述べる。

## 1.1 本研究の背景

インターネットは、IPv4 [1] の持つ 40 億の IP アドレスを枯渇させるほど発展してきた。この問題を解決するために IPv6 [2] が開発され、充分な数の IP アドレスを提供することが可能となった。

最近では、IPv6 の広大な アドレススペースを利用して、携帯電話、自動車、冷蔵庫など、すべてのものに対してインターネットへの接続が考慮される傾向にある。また、インターネット電話やインターネット TV などに見られるように、これまでインターネットとは独立に動作していたシステムのインターネット接続も考えられている。これらのことはすべて、インターネットがこれまで以上の速度で拡大することを示している。

規模の拡大とは、具体的には接続されるノード数の増加と、様々な種類のデータリンクによるネットワークトポロジの複雑化のことである。次世代インターネットを構築する際には、このトポロジの複雑化による経路制御機構への影響を考慮しなければならない。そして、次世代のインターネットはどのように接続され、どのような規模を持つのかを予測し、現在の経路制御機構が次世代のインターネットに適しているのかどうかを検討する必要がある。

さらに、電話網や銀行取り引きのシステムがインターネットを利用して構築されるとき、インターネットの社会基盤としての責任は重大になる。このことから、通信経路上の故障を即座に回避する能力が必要である。経路制御機構の現在の故障回避能力を調べ、次世代インターネットモデルに適するように改善を行なう必要がある。

## 1.2　本研究の目的

前述の通り、次世代インターネットは日常生活のライフラインとして利用される。ライフラインとして利用するためには、ネットワークの信頼性を向上させる必要がある。すなわち、ネットワークは現状より冗長な構成を持ち、複雑に接続される。また、従来の計算機だけではなく、様々な機器が、様々なリンクを用いて接続されると予想する。これが次世代インターネットモデルである。

そこで、本研究では、次世代インターネットモデルを定義し、定義した次世代インターネットモデルにおいて、経路制御機構が有効に機能するかどうかを検証する。そして、次世代インターネットモデルに適した経路制御技術を明確にし、実装・改善ならびに実証実験を行なう。具体的には、実際に経路制御を行なうソフトウェアを開発し、次世代インターネットにおいて起こり得る経路制御機構の問題と求められる機能、性能を考慮し、次世代インターネットに適応する経路制御機構を実装し、運用実験を行なう。これらの結果を総合して、社会基盤としての次世代インターネットにおいても、有効に機能することのできる経路制御機構を作ることを目的とする。

## 1.3　本論文の構成

本論文では、第 2 章にて現状のインターネットを分析した。主に規模性、信頼性、経路制御の観点から分析を行なった。次に、第 3 章にて次世代インターネットモデルを定義し、次世代インターネットにおいて経路制御に必要とされる要素に関して考察を行なった。その後、第 5 章にて次世代インターネット経路制御機構に関する考察と分析を行なった。第 5 章の分析結果を元に、次世代インターネットのための経路制御機構を実装した。この設計と実装を第 6 章にて述べる。最後に第 7 章にて本研究の評価を行なった。

# 第2章

# 現状のインターネットと経路制御機構の分析

本章においては、現状のインターネットを主に経路制御の観点から分析する。これによって、次世代インターネットモデルを定義するにあたっての要素事項となる、ネットワークの接続形態や、経路制御状況を明確にする。

## 2.1　現在のインターネットにおける経路制御

今日のインターネットがここまで発展してきたのは、経路制御技術の開発によるものが大きい。世界の端と端をつなぐ通信が実現できるのは、通信経路上の中継ルータが、一貫した経路情報を保持しているからである。

ネットワークの全ルータが一貫した経路情報を持つことを、一般的に「経路が収束する」と言う。経路が収束されていないネットワークでは、ネットワークのパフォーマンスと安定性に大きな影響を及ぼし、通信が全く行なえない等の問題が起こる。安定した通信環境の実現には、ネットワークにおける経路の収束が絶対条件である。

ネットワークが大規模になるほど、経路を収束させるのが困難になる。このため、これまでのインターネットでは「できるだけ長い期間経路を変化させないこと」を目標として、以下に述べるような様々な経路制御技術が開発されてきた。

### 2.1.1　静的経路制御、動的経路制御

一度設定した経路表がネットワークや中継ノードの状態によって変更されないものを「静的経路制御」と呼び、経路表を経路制御プロトコルなどによって収集したネットワークや中継ノードの状態から構築するものを「動的経路制御」と呼ぶ。

今日のように巨大になったインターネットでは、動的経路制御を用いてその大部分のネットワークを運用している。インターネットの全部のネットワークに対

する経路を、人間が計算し設定することは不可能である。さらに、静的経路制御では、ネットワークトポロジの変更やネットワークトラブルに対応した経路表を保持することが現実的に不可能である。

動的経路制御では、ルータが経路の変化を検知し、新しい経路を計算し設定するため、ネットワークの接続状態の変化に自動的に対応することができる。あるネットワークへの通信経路上に物理的な障害が起こった場合、動的経路制御機構によりそのネットワークへの代替経路が計算され、通信環境が自動的に回復する、という利点を持つ。

しかし、現在の動的経路制御を用いたネットワークには、経路計算の頻度が過度に高くなるという問題が起こり得る。あるリンクでの物理的な問題等から、一瞬のネットワーク切断が周期的に起こった場合、ネットワーク上の全ルータが周期的に経路表を変化させる場合がある。これを経路フラップと呼ぶ。経路フラップが起こると、転送するための通信データの一部分が欠落する。このため、経路フラップが起こったネットワークでは通信の性能が著しく損なわれる。このため、これまでの動的経路制御機構では、経路が迅速に計算されることよりも、経路変化の頻度を抑えることが重要視されてきた。

またこれに関連して、代替経路が物理的に存在しないネットワークなど、動的経路制御によって代替経路を検知する必要が無い一部分のネットワークでは、現在でも静的経路制御が用いられる。

### 2.1.2 AS によるインターネットの階層化

インターネット全体をたった一つの経路制御機構で動作させるのは、現実的ではない。ある一点での地域的な経路変化が世界中のルータに到達する間に、別地点で新しい経路変化が起きる場合があるからである。この場合、経路変化が絶えず発生し、前述の経路変化の頻度の問題から、好ましくない。また、経路制御はネットワークの運用ポリシに大きく依存するので、インターネット全体として統一することは不可能である。

このため、インターネット全体での経路制御は二つのレベルに分けられた。レベルを分けることによって経路制御ドメイン内のノード数を縮小させ、経路変化が影響する範囲を区画することができる。これにより、経路変化の頻度を抑制し、動的経路制御を安定させることができる。

レベル分けは、AS (Autonomous System:自律システム) と呼ばれる、「単一の経路制御ポリシを持つネットワークの集合」を定義することによって実現される。図 2.1 に、AS を利用した現在のインターネット経路制御の構造を示す。AS 内での経路制御と AS 間での経路制御を分離し、それぞれの経路制御ドメイン内で経路情報を一貫させている。

現在のインターネットでは、AS 間動的経路制御には BGP [3] を、AS 内動的経路制御には OSPF[4] または IS-IS[5] を利用するのが一般的である。

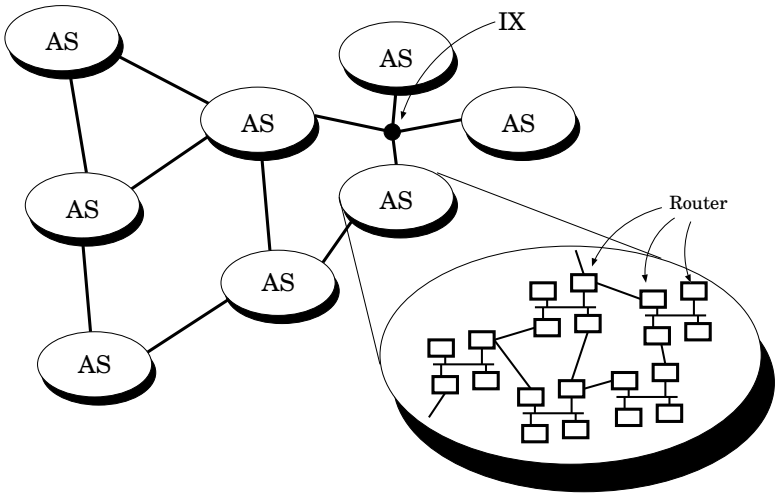


図 2.1: インターネット経路制御の構造

APNIC[6] によると、2001 年 6 月現在の AS 数は 11,000 程、AS 間経路 (BGP full route) は約 100,000 から 110,000 となっている。

### 2.1.3 IX 型接続形態

インターネットにおける通信遅延は、主にルータを経由することによって生まれる。インターネットが成長するにつれて、通信を行なう際に経由するルータの数が増加し、通信遅延の大きさが問題となってきた。

これを解決するために、IX (Internet eXchange) と呼ばれる新しい AS 間接続形態が生まれた (図 2.1 右)。

IX の出現以前では、個々の AS が独立に 接続を確立していた (図 2.1 左)。IX では、接続する 各 AS がルータを一つの場所に集め、Full mesh に近い形で 経路を交換する (図 2.2)。

IX に接続された AS は、IX に接続されない AS より多くの隣接 AS を持つ。このような AS を始点または終点とする通信は、経由する AS の数が減少するため、経由するルータの数も減少し、通信遅延が減少する。これまでの AS の接続を 図 2.3 に示し、IX 接続によって経由 AS が減少する様子を 図 2.4 に示す。

AS 間接続は大別して、他 AS から他 AS への通信トラフィックを転送するかしないかに分けることができる。通信トラフィックを転送するようなものを Transit AS と呼び、転送しないものを Non Transit AS として 図 2.3、2.4 に示した。APNIC によると、現在の Transit AS 数は約 1,500 となっている。接続形態の違いによって、図中左下の AS から 右上の AS への通信トラフィック (Sample Flow) が経由する AS 数が、図 2.3 では 4 となっているのに対し、図 2.4 では 1 に減少しているのがわかる。

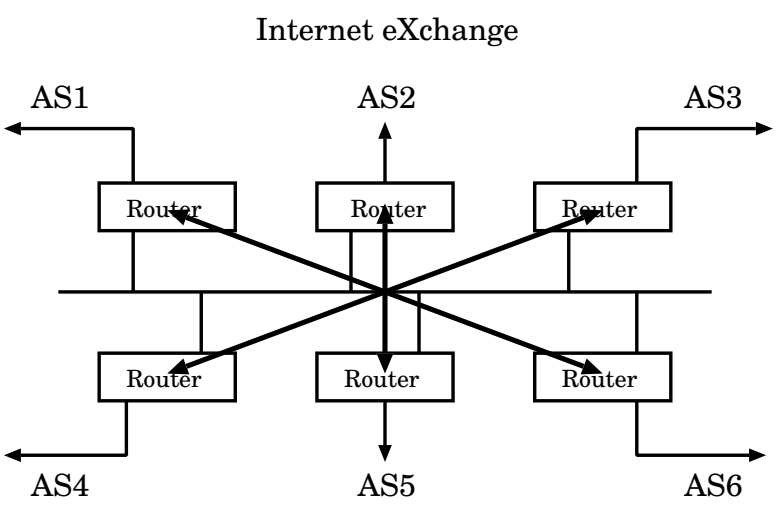


図 2.2: IX 型接続形態

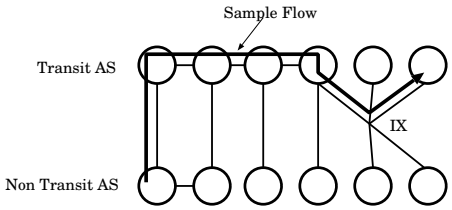


図 2.3: 従来の AS 間接続

現在では、AS 間の接続は IX を利用したものが主流となりつつある。先行研究 [7] によると、これが原因となり、インターネットの成長は、AS 数の増加にも関わらず、インターネット全体の AS を単位とした直径は変わらない傾向にあることがわかっている。

本論文では、最も離れた AS の最短経路において、いくつの AS を経由しなくてはならないかということを、「AS を単位とした直径」と呼ぶ。AS レベルでのインターネット成長傾向の概念図を図 2.5 に示す。Model A、Model B と書かれた AS がつながる以前は、ある AS から別の AS へ到達するためには、最大でも 1 つの AS を経由すれば到達できた。このため、このときの AS 単位でのネットワークの直径は 3 と言える。Model A と書かれた AS が接続されたとすると、Model A の AS から別の AS まで到達するためには、最大 2 の AS を経由しなくてはならない。このため、AS の直径が 3 から 4 に増えたと言える。直径が増えない新しい AS のつながり方というのは、Model B のような接続の仕方であり、現在のインターネットはこの成長傾向を見せている。

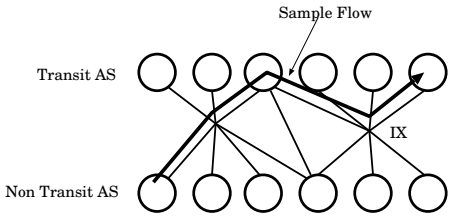


図 2.4: IX を用いた AS 間接続

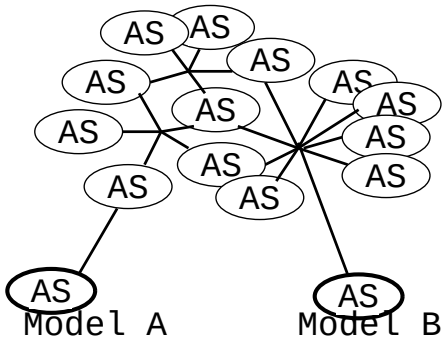


図 2.5: AS レベルでのインターネット成長傾向

## 2.2 現在の経路制御機構の問題と制限

前節で分析した現在の経路制御の状況を元に、その問題点と制限事項について考察する。

これまでの中継ルータは、ユーザの通信データである IP パケットの転送と、経路制御の処理を並行して行なっていたため、CPU 時間を多く占有するような経路制御機構は好まれなかった。経路に大きな変化や連続した複数の変化があると、変化した経路に直接関係の無い通信の IP パケットが落とされる、ルータが短期間 IP パケット転送を止めてしまうなどの問題があった。このため、経路制御機構を設計する際に、CPU 時間の消費を抑えるような制限が課せられていた。

また、前述の経路フラッピングについて、二つの重要な原因が存在する。これらはすべて経路制御機構設計の際の制限となる。

一つ目は、物理的なリンクの故障による経路フラッピングの発生である。ハードウェアの故障を無くすことはできないため、どのようなネットワークにおいてもこの問題は存在する。

二つ目は、経路制御機構の接続の監視方法に関する問題である。

ネットワーク上のリンクにおいて、利用可能な帯域幅を越えるトラフィックによる混雑が起きた状態を「輻輳」と呼ぶ。輻輳は、大量のパケットがリンクに集中することにより発生する。

インターネットでは、短期間にパケットが集中するバーストトラフィックが一般的に発生する。通常さほどトラフィック量がないリンクにおいても、このバーストトラフィックと呼ばれるトラフィックパターンが、輻輳を発生させる場合がある。輻輳が発生しているリンク上では IP パケットの欠落が発生する。このため、輻輳しているリンクでは、経路制御パケットも欠落する恐れがある。経路制御パケットの欠落により、経路制御機構は誤ってリンク切断を検出する。誤って切断を検知した場合には、経路制御機構が経路情報を更新するため、そのリンクへのトラフィック転送が停止され、結果として輻輳が解消される。輻輳が解消されると、経路制御機構が接続を検知し、そのリンクの経路が再設定され、リンクへのトラフィック転送が開始されるため、再び輻輳が発生する。

このように、経路制御機構が誤った判断をすることにより、ハードウェアの故障の無い状態でも経路フラッピングを引き起こす危険性がこれまで存在していた。[8] では、リンク輻輳時の経路制御機構の不具合と、キューイング技術の必要性が述べられている。

これら 2 つの経路フラッピングの原因から、現在の経路制御機構の状態検知間隔は余分に長く設定するのが一般的となっている。この制限から、経路変化の迅速な収束は実現されていない。

## 2.3 経路制御処理の向上

本節では、経路制御処理性能を向上する関連技術について分析を行なう。

CPU やメモリ等、ハードウェアの性能が向上するにしたがって、経路計算やネットワークトポロジ計算を行なうためのルータの能力が飛躍的に向上している。現在では、PC ルータを用いて、現状の BGP full route である 100,000 経路を計算し、保持することも可能である。

さらに現在では、第 2.2 節に述べた問題を解決する様々な技術が開発されている。

まず、out-of-band 処理と呼ばれる、経路制御処理と IP パケット転送処理の分離が進んでいる。これには、古くからルートサーバ [9] などが開発されてきた。さらに近年のルータは、経路制御処理と IP パケット転送処理を別々のハードウェアで行なうものが多い。これらは、Routing Module、Switching Module 等と呼ばれ、全く別の処理を並行して行なえる。どれほど経路計算に CPU を消費しても、並行して IP パケットの転送を行なうことが可能になった。

このため、経路制御機構は第 2.2 節で述べた、ルータの CPU 占有率などを気にせずに経路計算やネットワークトポロジ計算に専念できるようになった。

また、DiffServ[10] 等のキューイング技術の発展により、現在では経路フラッピングの原因の一つを解決することができる。ALTQ [11] 等の実装により、実際に輻輳が起きたリンクでの経路制御パケットの欠落を抑制することができるようになってきた。このため、第 2.2 節で述べた経路フラッピングの二つ目の原因を解決

することができる。キューイング技術を経路制御パケットに対して用いることによって、経路制御パケットの欠落が防げるため、輻輳による経路フラッピングの危険性が無くなる。このため、経路制御機構の状態検知間隔を短くすることが可能になり、迅速な経路の収束が実現できる。

これらのことは、これまで様々な外部要因によって、経路制御機構が充分に性能を発揮できていなかったということを示している。現在では、経路制御が性能を発揮するために必要な、これらの技術が発展している。

# 第3章

# 次世代インターネットモデルと経路制御

本章では、次世代インターネットのモデルを定義する。そして、定義した次世代インターネットモデルにおける経路制御について考察を行い、次世代インターネットに求められる経路制御の要素を定義する。

## 3.1 次世代インターネットモデルの定義

次世代インターネットにおいても経路制御機構が有効に機能するかどうかを検証するためには、現在のインターネットと次世代インターネットでの相違点を明確にする必要がある。そこで本節では、第 2 章にて行なった現状分析をもとに、次世代インターネットモデルを定義する。

### 3.1.1 次世代インターネットの要素

本研究では、次に挙げる事項を、次世代インターネットを特徴づける要素として定義する。

#### 接続組織の増加

従来は大学、企業、ISP(インターネットサービスプロバイダ) といった大規模な組織において常時接続が行なわれていた。しかし、現在 ADSL[12] や CATV といった、小規模な組織や家庭を常時接続可能にする技術が、急速に普及している。次世代インターネットにおいては、各家庭が常時接続になるだけでなく、車や携帯電話、様々なセンサーといった機器もインターネットに接続されると考えられる。これはインターネットへの接続組織ならびに接続ノードの増加を意味する。

#### リンク構成要素の多様化

車や携帯電話といった、有線による接続が困難なものは、無線によってインターネットに接続される。無線にも様々な技術が用いられ、方式や用途によって、異なる帯域や遅延特性を持ったリンクが存在する。さらに、衛星を利用した同報配送技術も実用化されると考えられる。すなわち、次世代インターネットにおいては、通信路中に様々な帯域や遅延の特性を有したリンクが混在することになる。

#### ネットワーク構成の複雑化

次世代インターネットは、計算機同士が通信するためのインフラとしてだけではなく、様々な機器が通信するための、ライフラインとなる。すなわち、今以上に信頼性が求められるネットワークとなる。信頼性を確保するためには、第 2.1.3 節にて述べた IX 型接続形態が有効である。次世代インターネットでは、信頼性と冗長性を確保するため、この接続形態が多用されると考えられる。この接続形態によって信頼性が向上する一方で、ネットワーク同士の接続点は増加し、ネットワーク構成が複雑化すると考えられる。

#### バックボーンの広帯域化

接続組織の増加に伴い、通信量も増加すると考えられる。すると必然的にバックボーンは現状より広帯域化する。複数のリンクを束ねて、広帯域のリンクとして利用する技術や、多重化のための技術が利用されると考えられる。

### 3.1.2 社会基盤としてのインターネット

次に、インターネットが果たす役割の観点から、次世代インターネットを分析する。

#### 規模の拡大

情報通信技術 (IT) 戦略本部は、「e-Japan 重点計画」 [13] において、2005 年までに 4,000 万世帯が高速インターネットに常時接続できる環境を目標としている。また、総務省の調べでは、2000 年末の日本のインターネット利用者は、携帯電話での利用も含めておよそ 4,708 万人、前年比 78 % 増加 となっている。近年の小中高校での教育や、企業のインターネットへの取り組みから、これまで以上のスピードでインターネットが広まって行くことがわかる。

**社会基盤としての信頼性**

前述の「e-Japan 重点計画」にも見られるように、IPv6 の発展とマイクロノードの概念によって、 これまでコンピュータにしか考慮されなかったインターネット接続が、すべてのものについて考慮されるようになってきた。無限とも言える広大な IP アドレス空間を持つ IPv6 が開発されたため、現在では自動車、携帯電話、冷蔵庫などの家庭内電気製品、椅子などのインターネットへの接続が考えられている。

自動車が現在どの位置を走行しているか、および走行中の自動車のワイパースイッチが入っているかということがインターネットを利用して収集できると、現在の降雨量を示す地図の精度が飛躍的に向上すると言われている。このように、日常生活の中にある、様々な細かい情報を収集し蓄積することによって、新しい有益な情報が生成できることがわかってきた。

さらに、ネットワークの広帯域化から、インターネット電話、インターネット TV などが実現されると予想される。これらのことはインターネットの急速な規模拡大とともに、社会基盤としてのインターネットの信頼性の重要さを示している。

### 3.1.3 次世代インターネットモデル

第 3.1.1 節ならびに第 3.1.2 節にて行なった分析により、本研究では、次の要素を持ったネットワークを、次世代インターネットモデルと定義する。

- ライフラインとしての信頼性
- 様々な機器が接続される
- 常時接続組織の増加
- リンク構成要素の多様化
- 耐故障性向上
- 広帯域なバックボーン

ライフラインとして利用されるであろう次世代インターネットは、安定した信頼性を持たなくてはならない。そのためには、通信路の多重化が必要である。耐故障性を有した通信路の多重化を実現するためには、経路制御機構の性能が重要な要件となる。すなわち、次世代インターネットにおいては、経路制御機構に期待される性能がこれまで以上に増加する。通信媒体となる一部のハードウェアが故障しても、経路制御機構によって迅速に代替経路が検知され、通信環境が自動的に回復されることが要求される。

## 3.2 次世代インターネット経路制御機構の考察

本節では、第 3.1 節にて定義した次世代インターネットモデルをもとに、次世代インターネットにおける経路制御機構について考察を行なう。

次世代インターネットにおける経路制御機構の設計では、安定した通信環境や、規模性という概念が変わる。これまでの規模性は規模に対する経路変化の頻度の低さであったが、次世代インターネットにおける規模性は、ある規模のネットワークにおいてどれだけ迅速に経路を変化できるか、である。また、これまでの通信環境の安定度合は通信経路が変化しない度合であったが、次世代インターネットにおける通信環境の安定度合は、冗長経路の数と、ネットワークの状態の変化の検知間隔の短さによって決まる。

### 3.2.1 経路制御ドメインの分類

インターネットは、3 段階の経路制御ドメインに分けられる。AS 間経路制御、AS 内経路制御、AS 内で分割されたエリア内経路制御である。

そこで、それぞれの経路制御ドメインを、次のようにモデルを定義する。AS 間接続は IX を利用したものが一般的となる。そのため、AS を単位としたインターネットの直径は現在と同じ 10 ホップ程度に保たれる。そして、AS 内のルータは激増し、最大で 2000 台程度となる。また、NOC 間の接続は full mesh 構成に近いものとなり、AS 内のネットワーク直径は 10 ホップ程度に抑えられる。AS 内を分割するエリアは、最小でも 200 台程度のルータを含む。

表 3.1: ネットワークトポロジのシナリオ

| | 経路制御エリア規模 | ネットワーク直径 | 備考 |
|---|---|---|---|
| シナリオ 1 | 200 | 5 | 最少最短モデル |
| シナリオ 2 | 200 | 10 | 最少最長モデル |
| シナリオ 3 | 2,000 | 5 | 最多最短モデル |
| シナリオ 4 | 2,000 | 10 | 最多最長モデル |

### 3.2.2 経路制御ドメインの規模

経路制御ドメインが階層化されると、経路制御ドメイン内のノード数や経路数をまとめることができるため、スケーラビリティが増す。しかし、一つあたりの経路制御ドメイン内の経由ノード数 (ホップ数) がある程度一定だとすると、階層化すればするほど経由ノード数が倍増されていく。多段階に階層化された経路制御を行なうネットワークでは、通信が利用する経由ノード数が増加するため、良

好な通信環境を提供できない恐れがある。そのため、次世代インターネットにおける経路制御ドメインの階層化は、現在と同じ構造を保たなくてはならない。

よって、AS による経路制御ドメインの分割は次世代インターネットでも利用される。AS レベルでの経路制御は、「頻繁に接続や切断が起きにくいネットワークでの経路制御」と捉えることができる。ここでは、経路収束の速度より、スケーラビリティが重視される。

インターネットがどんなに巨大になったとしても、「頻繁に接続や切断が起きないインターネット全体の経路制御」と「例えば日本国内など、決められたドメイン内での経路の収束を保証する経路制御」というレベル分けは有効であり、必要である。

次世代インターネットでは、第 2.1.2 節で説明した AS による階層化の他に、AS 内で一段階の階層化を行なう必要がある。これは、OSPF や IS-IS が持つエリア分割の機能を利用して実現される。このため、次世代のインターネットでは 大きく分けて 3 つのレベルで経路制御が行なわれる (図 3.1 中 Inter AS、Backbone、Area)。

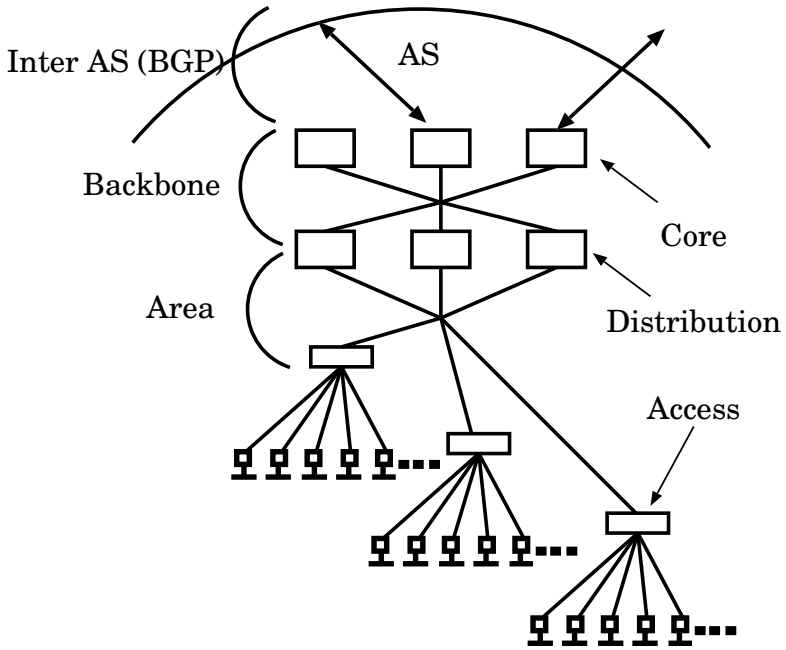


図 3.1: 次世代インターネットにおける AS 内経路制御ドメインの分割

図 3.1 のモデルは、現在と同じであり、大規模なネットワークを持つ組織では既にこのモデルで運用を行なっている。次世代インターネットではこのモデルは変化せず、規模が変化する。

現在の世界最大の ISP では、数百のルータを含むネットワークを運用している。次世代インターネットでは、最大でおよそ 2000 ほどのルータを持つ AS が存在す

ると考えられる。これをエリア分けして 400 程度のルータ を含むエリアに分けることが考えられる。

しかし、冗長性を含む ネットワークトポロジでは エリア分けをすることが困難であることを考慮し、本論文では、1000 から 2000 程度のルータを含むネットワークを 一つのエリアとして 経路制御することを想定する。

### 3.2.3 ネットワークトポロジの変化

ネットワークの接続の形態は、第 2.1.3 節で示したように、full mesh に近付くべきである。full mesh に近付けば近付くほど、通信に利用する経由ノードが減るため、ネットワーク通信のパフォーマンスが向上する。また、full mesh は冗長な経路を複数持つために、強靭で頑健なネットワーク構築のために望ましい。

現実的には、あるドメイン内のノードを全て full mesh でつなぐことはできない。これには、OSI 7 階層 モデルにおける、データリンク層プロトコルの距離制限やノード制限が関わって来る。地理的に広範な範囲を一つのドメイン内に納めるために、ネットワークは一部数珠つなぎにされる。ここから、ネットワークのホップ数が決まる。

しかし、近年のギガビットイーサネット (IEEE 802.3z[14]) 技術の開発や、DWDM(Dense Wave Division Multiplexing) の開発により、数百キロの範囲に渡る full mesh 接続の構築も可能となってきた。

現在一般的なネットワーク設置方法は、地域ごとに NOC(Network Operation Center) と呼ばれるコアネットワークを設置することである。人間がネットワークを運用することを考えると、これは変えることができない。ある程度の単位でネットワーク装置が固まっていなければ、運用コストが高くなってしまうからである。現在の NOC 間接続は、ATM[15][16] 等を利用した数珠つなぎに近い構成となっているが、次世代インターネットでは広域ギガビットイーサネット等を利用した full mesh 接続に近い構成となる。図 3.2 はこの様子を示している。また、Area 分割が行なわれるとすれば、 NOC 毎または複数の NOC をまとめた単位で行なわれることが予測される。

ISP などでは、下流となる顧客のネットワークは、バックボーンルータとは別のルータで収容される。ポートデンシティ (ポート密度) などの違いから、ベンダから「収容用のルータ」と「バックボーンルータ」は種類の違う装置として販売される。

このため、NOC 間が full mesh で接続されたとしても、ホップ数が激減することは有り得ない。現在の典型的な大規模ネットワークでのホップ数は 5, 6 程度であり、規模が拡大する次世代インターネットでも 10 ホップ弱程度で保たれると考えられる。

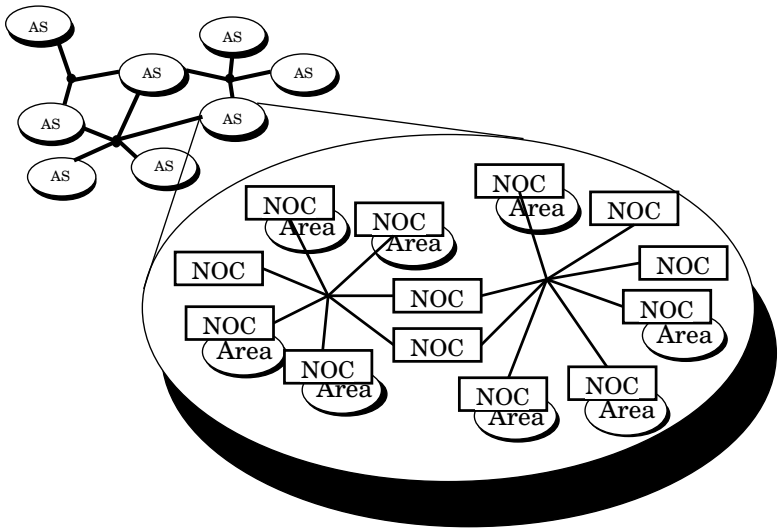


図 3.2: NOC 間の full mesh 接続

## 3.3 次世代インターネット経路制御機構に求められる要件

第 2 章で行なった分析ならびに第 3.1 節にて定義した次世代インターネットモデルから、これまでの経路制御機構で目標にしてきたものと、これからの経路制御機構に期待される要件が違うことがわかる。

経路制御機構はこれまで「できるだけ長い間経路が収束した状態を保つ」ことを第一の目標に構築されてきた。この場合の「経路の収束」が意味するものは、経路制御ドメイン内のルータが持つ経路情報が一貫している、ということのみである。これはつまり、経路情報の正確さをある程度無視し、通信環境の安定性を重視した、ということである。これは当然のことであり、安定して通信が実現できる環境を提供することが、経路制御の第一目的である。

しかし、次世代インターネットにおいて、生活の基盤としてネットワークを利用するためには、経路制御機構にさらなる要件が課せられる。

通信環境の強靭さ、頑健さを向上させる必要があると、第 3.1 節で述べた。これを実現するためには、ネットワーク接続に冗長性を持たせ、かつ、経路制御機構が経路の障害を迅速に検知しなくてはならない。

求められる要件は、大きくわけて 2 つある。

1. 経路変化の速度
   通信環境の安定度合は、通信トラフィックの喪失度合に影響される。これまでの経路制御機構では、これを実現するために、経路変化の頻度を抑える設計となっている。次世代インターネットでは、経路変化の頻度によって通信ト

ラフィックの喪失が起きないため、経路変化を迅速に起こすことが通信環境の安定につながる。

2. スケーラビリティ
   通信環境の安定度合は、通信遅延によっても影響される。通信遅延が大きい環境は良好な通信が行なえず、安定していると言えない。
   通信遅延を少なく保つため、また高い冗長性を提供するために、経路制御ドメインの階層化は可能な限り抑制されなくてはならない。
   ここから、経路制御機構が持つ規模と構成の制限を少なくし、スケーラビリティを向上しなくてはならない。

これらを実現するためには、現在の経路制御機構の設計における、時間の粒度の粗さを解決しなくてはならない。経路変化の速度を向上させるには、経路制御機構がネットワーク状態を検知する間隔を短くしなければならない。また、スケーラビリティを向上させるためには、大規模なネットワークにおける経路の収束速度が重要となる。

## 3.4 次世代インターネット経路制御機構における問題点

### 3.4.1 経路変化と TCP 通信パフォーマンスの関係

これまで経路変化の頻度が通信環境の安定性に影響を与えている理由は、実は第一に TCP[17] にある。WWW、メールなど、今日広く利用されているアプリケーションのほとんどは TCP を利用している。現在の TCP の実装の多くは、TCP フラグメントの順番が変わったり、パケット落ちを検知すると、通信のパフォーマンスが激減するようになっている。これが、マルチパスや経路変化の場合に良好な通信環境が得られない理由である。TCP フラグメントの順序変化や、パケット落ちの検知の際にも通信パフォーマンスが低下しない TCP のアルゴリズムや技術が開発されることが理想である。次世代のインターネットではこれが実現されていると期待し、本論文の研究範囲には含めない。TCP のパフォーマンスを調整するアルゴリズムが改善されたとき、経路制御の機能であるマルチパスや、迅速な収束が自由に利用できる。

### 3.4.2 物理的なリンク発振による経路フラップ

経路制御機構がネットワークの状態を検知する間隔の問題として、物理的なリンクの発振による経路フラップの問題がある。

経路の計算時に優先される経路上のリンクが物理的に発振し、かつその経路に安定した代替経路が存在したとする。この場合、安定した通信環境の実現のためには、代替経路を利用し続けるべきである。

しかし、状態検知の間隔が短ければ短いほど、通信環境が一時的なリンクの回復に影響を受ける。また、不必要な経路制御トラフィックをネットワーク全体に伝播することとなる。

この問題は、状態検知の間隔が短いことが原因の本質では無い。BGP の Route flap damping[18] のように、リンクに対する安定度を計算し、より安定した経路を利用し続けるような新しい技術が開発されない限り、「代替経路を利用すれば通信可能であるのに、通信ができない」といった不便な状態は解消されない。この新しい技術については課題とし、本論文では解決しない。

### 3.4.3 次世代インターネットへの適応時の問題点

AS 間の経路制御には、BGP が利用される。BGP はパスベクター型アルゴリズムを用いており、AS を単位としたインターネットの直径が変化しない限り、スケーラビリティや収束速度は変化しにくい。このとき、前提としてルータメモリの増加および経路集約の技術が有効に働き、経路数に関するスケーラビリティなどの問題は起きないと予測している。

第 2.1.3 節に述べたように、インターネットの成長は、AS レベルでの直径が変化しない傾向にあるため、次世代インターネットにおいても BGP は現在と同じ性能を持つと想定できる。

AS 間経路制御機構は、世界全体で一つの経路制御システムであるため、一地点の観測から、他の地点での動作を推測することができる。このことから、AS 間経路制御に関するこれまで研究が有効であることが言える。これに対して、AS 内経路制御プロトコルの有効な評価はいまだに行なわれていない。このため、次世代インターネットに現在の AS 内経路制御機構を適応した際に、どのような問題が起こるのかわかっていない。

### 3.4.4 経路収束時間に関する先行研究

Route flap damping 等、BGP の技術は研究が進んでいる [7] [19] [20] [21] 。[20] によれば、当時のインターネットでの BGP 経路の収束はおよそ 1 分から 3 分となっている。

これに対して、AS 内経路制御については、スケーラビリティや経路収束速度に関する有効な指標が示されていない。

[22] による実運用時の統計は、Router の数が 13 から 15 程度である場合の OSPF の動作概要を示している。これは、ネットワークの規模が小さいため、次世代インターネットの指標とはならない。

[23] では、200 程度のルータ数を持つ OSPF のトポロジ計算にかかる時間の例が示されている。しかし、規模性や収束速度にはネットワークのトポロジ、プロト

コルパラメータの設定など様々なものが関わって来るため、トポロジ計算にかかる時間のみでは、OSPF の規模性や収束速度を予測することはできない。

[24] では、OSPF の収束時間に関するシミュレーションが行なわれた。ここでは、ルータ数が 20、50、80 であるような 3 つのトポロジ例について、リンクスピードと OSPF の再送間隔の設定と、経路の収束時間についてのシミュレーションが行なわれた。しかし、このシミュレーションはルータの起動から経路収束が起こるまでの時間に関するものに限定されている。ネットワークの接続のみが変化した場合のシミュレーションは行なわれていない。さらに、OSPF のネットワーク変化への対応速度 等のプロトコルパラメータに関する考察が行なわれておらず、実験結果の経路収束も 何十秒といったレベルであり、次世代インターネットに適さない。

## 3.5 次世代インターネットを想定した AS 内経路制御機構の開発

第 3.4 節で述べた問題点を解決するためには、AS 内経路制御機構の改善が必要である。そこで本研究では、理想の次世代インターネット通信環境を実現するために、現在の経路制御機構について考察し、問題を改善した AS 内経路制御機構の設計および実装を行なった。

まず、ネットワークの実測値を用いた収束時間に関する考察を行ない、AS 内経路制御機構のどのような点が問題となるのか、また、それに対してどのような改善が可能かを導き出す。

次にこれらの問題点、改善点を考慮し、次世代インターネットに適応する AS 内経路制御機構を設計、実装する。AS 内経路制御機構には、IPv6 をサポートした新しい OSPF プロトコル、OSPFv3[25] を採用した。

本研究で開発した AS 内経路制御機構が本当に有効なのかの検証は、次世代インターネットを想定したネットワークにおいて実運用することで評価する。

本研究が想定する前提を以下にまとめる。

1. 経路制御機構は経路制御に専念できる
   (a) 輻輳による経路制御パケットの欠落がない
   キューイング技術の発達により、経路制御パケットの欠落が無い。これまで憂慮されてきた輻輳による経路変化の問題が解決された。
   (b) 経路計算が通信トラフィック転送に影響を与えない
   経路制御ハードウェアと通信トラフィック転送ハードウェアの分離により、経路計算やトポロジ計算などが通信トラフィック転送のパフォーマンスに影響を与えない。
2. リンク発振を検知する新しい機能が開発されている

リンクの物理的な発振を検知する新しい機能が、経路制御機構の一部として開発されている。これにより、経路の安定度を認識することができ、経路フラップの危険性を解消された。

# 第4章

# OSPF: Open Shortest Path Firstプロトコル

本章では、AS 内経路制御プロトコルの一つ、OSPF について説明する。IPv4 用の OSPF を OSPFv2 と呼び、IPv6 用の OSPF を OSPFv3 と呼ぶが、特に記述しない限り、ここでの説明は両者にあてはまる。

## 4.1 概要

OSPF はリンクステート型経路制御プロトコルである。

OSPF では、ルータおよびネットワーク回線を、接続点という意味を持つ「リンク」として表現する。各ルータやネットワークは、他とどのように接続 (リンク) されているかの情報を持つ。この情報を、LSA (Link State Advertisement、リンク状態広告) と呼ぶ。

OSPF ドメインに参加するルータは、お互いに全ての LSA を交換する。つまり、全てのルータが全ての LSA を持つことになる。全てのリンクの情報を持っているルータは、そのネットワークのトポロジを計算することができる。OSPF では、各ルータが djikstra アルゴリズム [26] を用いて、LSA の集合からネットワークトポロジを独自に計算する。これを SPF 計算 (Shortest Path First Calculation) と呼ぶ。

Ethernet のような マルチアクセスのネットワークにおいて、ネットワーク回線の接続状態を広告するためには、そのネットワーク回線上のルータの内一つが接続状態を調べ、LSA として広告する必要がある。これを行なうルータを、そのネットワーク回線の DR (Designated Router、代表ルータ) と呼ぶ。あるネットワーク回線において、どのルータが DR となるかは、Hello サブプロトコルにおいて決定される。

各ルータが持つ LSA の集合を、LSDB(LSA DataBase) と呼ぶ。各ルータが持つ経路表が一貫することを保証するためには、全てのルータが全ての LSA を取得できることを保証しなくてはならない。つまり、全 OSPF ルータの LSDB が同期することを保証しなくてはならない。これを実現するために、OSPF では、相手ルー

タが LSA を受信したことを確認し、信頼性のある LSA 交換を行なう。このため、OSPF ルータにおける LSA 交換の処理はとても複雑なものになっている。

隣接する OSPF ルータを Neighbor と呼ぶ。LSDB の同期を行なう二つの OSPF ルータの間の接続を、Adjacency と呼ぶ。

## 4.2 動作概要

OSPF ルータは、経路計算を行なうために、OSPF ドメイン上の他のルータと LSDB を同期させなければならない。OSPF では、Neighbor を発見するために Hello サブプロトコルを利用する。発見された Neighbor は、Adjacency の確立を行なう必要があるかが確認される。必要であれば、OSPF ルータはその Neighbor と LSDB の同期を始める。

LSDB の同期は、Adjacency が Full(完全) であるかどうかによって、二つの方法にわかれる。LSDB の同期を始めた初期の段階では、Database Exchange(データベース交換) と呼ばれる方法で LSDB を同期させる。これは、基本的に LSA 要求と応答により実現される。

Database Exchange が完了することによって、Adjacency が Full になったと認識される。これ以降の LSDB の同期は、Flooding と呼ばれる方法で行なわれる。

LSDB が同期すると、OSPF ルータは SPF 計算を行ない、経路表を計算する。

ネットワークの接続は、Hello サブプロトコルを利用して監視される。切断や接続など、ネットワークトポロジに変化があった場合、Hello サブプロトコルにより検知され、ルータは新しい LSA を生成する。新しい LSA は Flooding され、それを用いてトポロジ変化後の経路が全ルータにより再計算される。

## 4.3 各概念および動作の詳細

### 4.3.1 Hello サブプロトコル

OSPF では、切断を含むネットワーク接続の変化の検知と、DR の選出をするために、Hello プロトコルを内包している。Hello プロトコルは、OSPF ルータの各 Interface に一つ起動される。

各 OSPF Interface には、Hello プロトコルの設定として、HelloInterval、RouterDeadInterval、RouterPriority という設定可能な値が存在する。

Hello プロトコルは、HelloInterval 秒毎に、Interface に Hello パケットを送信する。Hello パケットを受信すると、OSPF は受信 Interface 上に、Hello パケット送信元を Neighbor として認識する。各 OSPF ルータには Router ID という一意な識別子が割り振られ、これによって 複数の Neighbor を識別する。

Neighbor から RouterDeadInterval 秒の間 Hello パケットが届かなくなると、

ルータは Neighbor との接続が切断されたと認識する。RouterDeadInterval 秒の間に、一つでも Neighbor からの Hello パケットが到着すると、その Neighbor の状態は維持される。

認識されている Neighbor の Router ID は、次回に送信する Hello パケットにリストされる。Hello パケットは、その時点で送信元に認識されている Neighbor Router ID のリストを含む。この機能を利用して、Hello プロトコルは Neighbor との双方向の通信を絶えず監視する。Neighbor からの Hello パケットが到着し、かつ自分の Router ID が Hello パケットに記載されていれば、その Neighbor との双方向の通信が可能であるとわかる。

このため、ネットワークの接続は HelloInterval 秒毎、ネットワークの切断は RouterDeadInterval 秒毎に 確認されている、と言える。

Hello プロトコルのもう一つの仕事は、マルチアクセスネットワークにおける DR の選出である。DR は、そのマルチアクセスネットワークの状態を広告する LSA を作るルータである。

DR が変化すると、OSPF はネットワークトポロジが変化したと認識する。このため、DR が変化するのは極力避けたほうが良い。そのために、あらかじめ DR のほかに、Backup DR(BDR) を選出しておく。

Hello パケット内に、送信元の DR 選出結果を記述する DR、BDR のフィールド、および送信元のルータの RouterPriority を記述するフィールドがある。それらのフィールドと、Router ID の大小を用いて、DR を選出する。

### 4.3.2 Database Exchange

Neighbor が新たに検知され、Adjacency を張る必要があることが確認されると、LSDB の同期が始まる。Adjacency 確立初期の LSDB 同期は、Database Exchange と呼ばれる 要求応答型の方法である。

まず、Database Description パケットを用いて、その時点で双方のルータが持つ LSA を互いに教え合う。その後、LS Request パケットで LSA を要求し、 LS Update パケットで実際の LSA を交換する。

Database Description パケットは、送信側が相手がパケットを受け取ったことを確認できる信頼性のある方法でなくてはならない。このため、Database Exchange を行なう二つのルータは、ネゴシエーションをして Master/Slave を決定する。Master が Slave に Database Description パケットを再送し、Slave は Master からの Database Description パケットに Database Description パケットで応答すると決めておく。Slave が応答の Database Description パケットを送信し、Master がそれを受信したということをもって、Master の直前の Database Description パケットが Slave に受信されたことを確認する。このようにして、信頼性のある方法で、双方の持つ LSA をすべて教え合う。Master/Slave の関係は、 Database Description パケットの交換にだけ関係する。

Database Description パケットによって教えられた LSA のなかに、自分が持っていないものがあった場合、ルータは LS Request で該当する LSA を要求する。LS Request を受け取ったルータは、該当する LSA を LS Update に含めて応答する。LS Request は、要求した LSA が LS Update の受信によって得られるまで再送を続けるので、この LS Update に対する確認応答は必要がない。

実際の LSA は LS Update パケットだけに含まれる。Database Description パケット、LS Request パケットには LSA を識別するための情報しか含まれていない。

Master の Database Description パケット、および LS Request の再送は、Interface の設定である RxmtInterval 秒後に 行なわれる。このため、Database Exchange にかかる時間は、パケットが喪失される度に RxmtInterval 秒 長くなる。

### 4.3.3 LSA Flooding

LSA Flooding は、Adjacency が Full になった後に利用されると前述した。厳密には、Database Exchange が開始された後に変更された LSA の伝播は、全て LSA Flooding 機構を用いる。

LSA Flooding では、基本的に 全ての Adjacent Neighbor に対して行なわれる。具体的には、単に LSA Update パケットに該当する LSA を含め、相手に送信する。送信は、LS Acknowledgement パケットによって相手がその LSA を受信したことが確認できるまで、再送される。再送の間隔は、前述の Database Description パケットや LS Request パケットと同様、Interface の設定である RxmtInterval を用いる。

LSA Flooding にかかる時間も、パケット喪失が発生する度に RxmtInterval 秒長くなる。しかし、冗長的なネットワーク接続を行なっている場合には、LSA が迂回経路の方向にも Flooding されて行くので、この限りではない。

### 4.3.4 SPF 計算

SPF 計算は、トポロジの計算を行なう。トポロジの記述に IP アドレスを利用する OSPFv2 では、エリア内経路に関して、SPF 計算は 経路計算を兼ねる。

SPF 計算は、自分の LSA を起点として、LSA 内に記述されている接続をたぐって行き、発見したノードを SPF Tree に追加する。追加されたノードの LSA を LSDB から検索し、再度それに記述されている接続をたぐる、というのが基本的な処理である。ループを取り扱うため、SPF Tree に追加しようとする度に、そのノードが既に SPF Tree 内に存在しないことを確認する。既に追加されていた場合には、新たに追加はしない。

設定の変更によって計算結果となる経路を制御するために、Interface の設定である Cost を利用する。

LSA を走査している間に発見されたノードは、その場で SPF Tree には加えられず、発見されたノードまでの Cost とともに、候補リストに記憶される。たぐることができる接続が全て候補リストに載った時点で、候補リスト中の最短 Cost のノードは、SPF Tree 内に存在しない限りそのノードへの最短パスであることが、アルゴリズムから保証される。

このようにして、候補リストが空になるまで走査を繰り返す。候補リストが空になったとき、すべての終点への Shortest Path(最短経路) が計算されている。

# 第5章
# 収束時間に関する考察

OSPF の収束速度の考察では、リンクやルータの転送遅延、および SPF 計算にかかる時間が重要なパラメータである。このパラメータを決定するために、実際に動作しているルータの処理速度を測定した。次に、計算量を用いて、SPF ノード数が増大した場合の SPF 計算時間を推測した。これらの値を用いて、ルータが 2000 ほどまでのネットワークの規模と、収束時間の関連性を推測した。

## 5.1 リンク転送遅延の測定

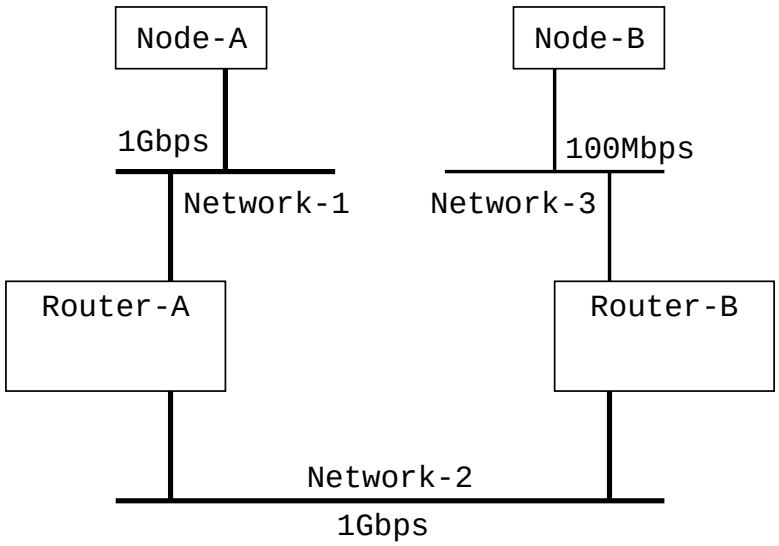


図 5.1: 測定環境

図 5.1 の環境において、Node-A から Router-A に向け、1 秒置きに ping を 1000 回試行した。ping の結果として示された 応答時間を 図 5.2 に示す。

これを全て平均 すると、0.3780 (msec) となる。この値は RTT (Round Trip Time、往復時間) なので、片道 0.1890 (msec) となる。 ここから、本章で行なう

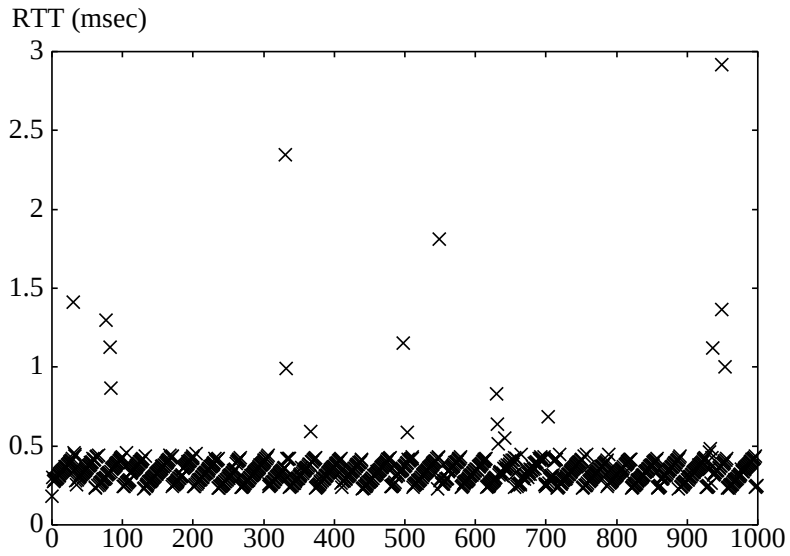


図 5.2: ping 応答時間

考察において、リンクの転送遅延は 0.1890 とする。実験に利用した リンクは 1 Gbps であるので、次世代インターネットの環境に近いと思われる。

## 5.2 ルータにおける LSA 転送遅延の測定

次に、ルータが LSA を転送する際に発生する遅延を調べる。

この実験は、図 5.1 の環境で行なった。Node-A、 Node-B において、ネットワーク Network-1、 Network-3 をそれぞれタップし、同時期に観測される LSA を記録した。Node-A と Node-B は NTP [27] によって時刻を同期した。このため、本節での時間誤差は、 NTP に準ずる。

Node-A、 Node-B ではそれぞれ NetBSD [28] が動作している。各ノードで、BPF[29] を用いて OSPF LS Update メッセージを収集した。Node-A では Network-1 の LSA が収集され、Node-B では Network-3 の LSA が収集される。その結果 を表 5.1 に示す。

表 5.1: LSA 記録実験の統計

| | Node-A | Node-B |
|---|---|---|
| LSA 記録時間 | 5091 秒 | 5091 秒 |
| LSA 数 | 21535 | 20423 |
| Unique LSA 数 | 16588 | 16588 |
| Duplicate LSA 数 | 4947 | 3835 |

表 5.1 において、 LSA 記録時間 は 最初に記録した LSA から最後に記録した LSA の間の時間を示す。LSA 数は記録された LSA の総数を、Unique LSA 数は記録された LSA のなかで、一意に識別される LSA の数を示す。Duplicate LSA 数は、マルチアクセスネットワークにおける、 Flooding の特徴によってネットワークに再送された LSA の総数を示している。

LSA 記録時間と Unique LSA 数が、 Node-A、 Node-B それぞれにおいて変わらないことから、OSPF ドメイン内の同時期の LSDB のスナップショットが、二つの地点で記録できたことがわかる。

LSA Flooding の機能によって、 Network-1 に出現した LSA は、Network-3 にも 出現する。このため、二地点に出現した LSA の出現時刻の差が、ルータやネットワークの転送遅延を表している。

このトポロジでは、様々な Flooding 経路が存在するため、ある LSA が、どのような経路でいくつのルータとネットワークを通過したのかを調べなければ、一つのルータが LSA を転送するのにかかる時間を計算できない。しかし、LSA がどのような経路を通って Flooding されてきたのかを知るのはほとんど不可能である。

そこで、LSA の Age の差に注目した。本実験はすべてのルータの TransmitDelay を あらかじめ 1 に設定して行なった。LSA の Age はルータを通過するたびに TransmitDelay 秒 増加されるため、Network-1、 Network-3 に出現した同一の LSA の Age の差が 1 であるものは、かならず収集されたネットワークに出現するまでにルータ一つとネットワーク一つ分の差がある経路を利用して Flooding されているとわかる。

二地点に出現した際に、Age の 差が 1 であった LSA を抜きだし、その出現時刻の差を表したものを、 図 5.3 に示す。

この平均をとると、

$$19.9392184447548 \doteqdot 19.93(msec)$$

となった。これは、ルータの LSA 転送遅延とネットワークの LSA 転送遅延の合計なので、ネットワークの LSA 転送遅延を引くと、

$$19.93 - 0.1890 = 19.741 \doteqdot 19.74(\mathrm{msec})$$

となる。これを、ルータの LSA 転送遅延とする。

## 5.3 SPF 計算時間の測定

次に、WIDE IPv4 Backbone 上の OSPF ルータで、SPF 計算時間の測定を行なった。WIDE IPv4 Backbone の規模を表 5.2 に示す。

WIDE IPv4 Backbone の OSPF ドメインに属する OSPF ルータにおいて、SPF 計算にかかる時間を測定した。この OSPF ルータは、 NetBSD の PC Unix ルー

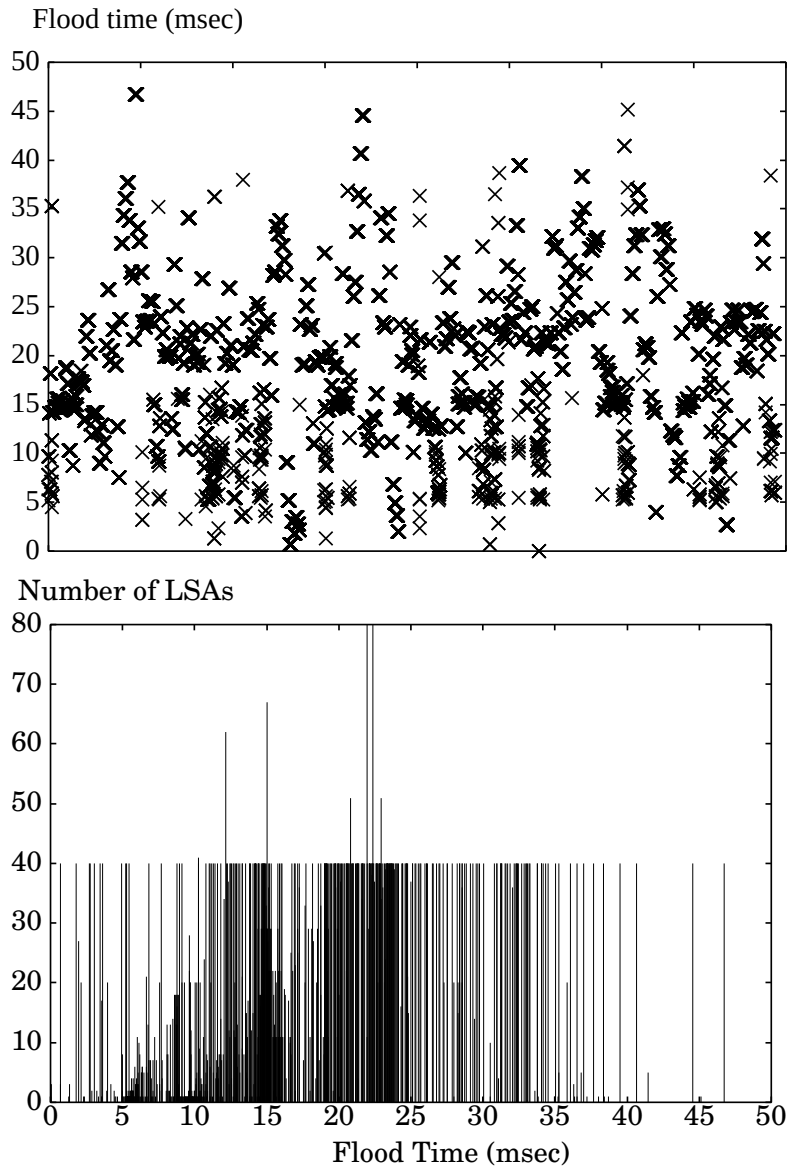


図 5.3: Age 差が 1 の LSA 出現時刻の差

表 5.2: WIDE IPv4 Backbone の規模

| | |
|---|---|
| ルータ数 | 46 |
| ネットワーク数 | 9 |

タであり、 Zebra[30] ospfd を動作させている。この Zebra ospfd に改変を行ない、SPF 計算時間を取得した。

収集時間を 表 5.3 に示す。この結果を平均し、本章では ルータ数 46、ネットワーク数 9 の規模のネットワークにおいての SPF 計算時間を

$$77.808/10 = 7.7807 \doteqdot 7.781(\text{msec})$$

とする。

表 5.3: SPF 時間の測定

| 測定時刻 | SPF 計算時間 (msec) |
|---|---|
| 2001/06/29 11:05:54 | 7.812 |
| 2001/06/29 11:06:04 | 7.773 |
| 2001/06/29 13:49:18 | 7.786 |
| 2001/06/29 13:49:28 | 7.871 |
| 2001/06/29 14:23:50 | 7.776 |
| 2001/06/29 14:24:00 | 7.729 |
| 2001/06/29 14:27:00 | 7.747 |
| 2001/06/29 14:27:10 | 7.752 |
| 2001/06/29 14:27:20 | 7.840 |
| 2001/06/29 14:27:34 | 7.721 |

## 5.4 収束速度の推定

### 5.4.1 SPF 計算の時間

SPF の計算では、候補を候補リストに挿入する際に、終点へのコストでソートしつつ挿入する。このとき、挿入場所をどのサーチアルゴリズムを用いて検索するかで SPF 計算の計算量が決まる。一般的に、バイナリサーチを用いると SPF 計算の計算量は $On(\log n)(n = \text{SPF ノード数})$ まで縮小できると言われている。リニアサーチの場合の計算量は $O(n^2)$ となる。

第 5.3 節で測定した SPF の計算時間は 7.781 であったが、測定に用いた Zebra ospfd は、 SPF 計算に リニアサーチ (線形探索) を用いている。SPF ノード数は WIDE IPv4 Backbone のルータ数とネットワーク数の合計なので、 $46 + 9 = 55$ となる (表 5.2)。

ここで、計算量と計算時間が比例していると仮定すると、計算時間と計算量から、SPF 計算量の係数を 導き出せる。

SPF 計算量 の係数 を $X$ とすると、リニアサーチを利用した場合の SPF 計算量 $O(n^2)$ と、実測された SPF 計算時間 7.781 から、

$$X = 0.002572$$

となる。

$$\begin{aligned} 7.781 &= X \times On(n^2) \qquad (n = 55) \\ 7.781 &= X \times 3025 \\ X &= 7.781/3025 \\ &= 0.0025722314 \\ &\div 0.002572 \end{aligned}$$

これを用いて、SPF 計算時間を推定する。

### 5.4.2 収束時間の推定

収束時間は、ネットワークの LSA 転送時間、ルータの LSA の転送時間、最後に LSA を受け取ったルータの SPF 計算時間の合計となる。

ネットワークの直径を 8 、ネットワークの LSA 転送時間を 0.1890(第 5.1 節参照)、ルータの LSA 転送時間は 19.74 とすると (第 5.2 節参照)、収束時間は、以下の式から求められる。

$$\begin{aligned} \text{収束時間} &= \text{ネットワークの } LSA \text{ 転送時間} \times (\text{ネットワークの直径} - 1) \\ &\quad + \text{ルータの } LSA \text{ 転送時間} \times \text{ネットワークの直径} \\ &\quad + SPF \text{ 計算時間} \\ &= 0.1890 \times (8 - 1) + 19.74 \times 8 \\ &\quad + SPF \text{ 計算量} \times X \end{aligned}$$

この式を用いて、ノード数が 1 から 2000 のネットワーク規模である場合の計算を行なった。SPF 計算にリニアサーチとバイナリサーチを使った場合のそれぞれの収束時間を、図 5.4 に示す。

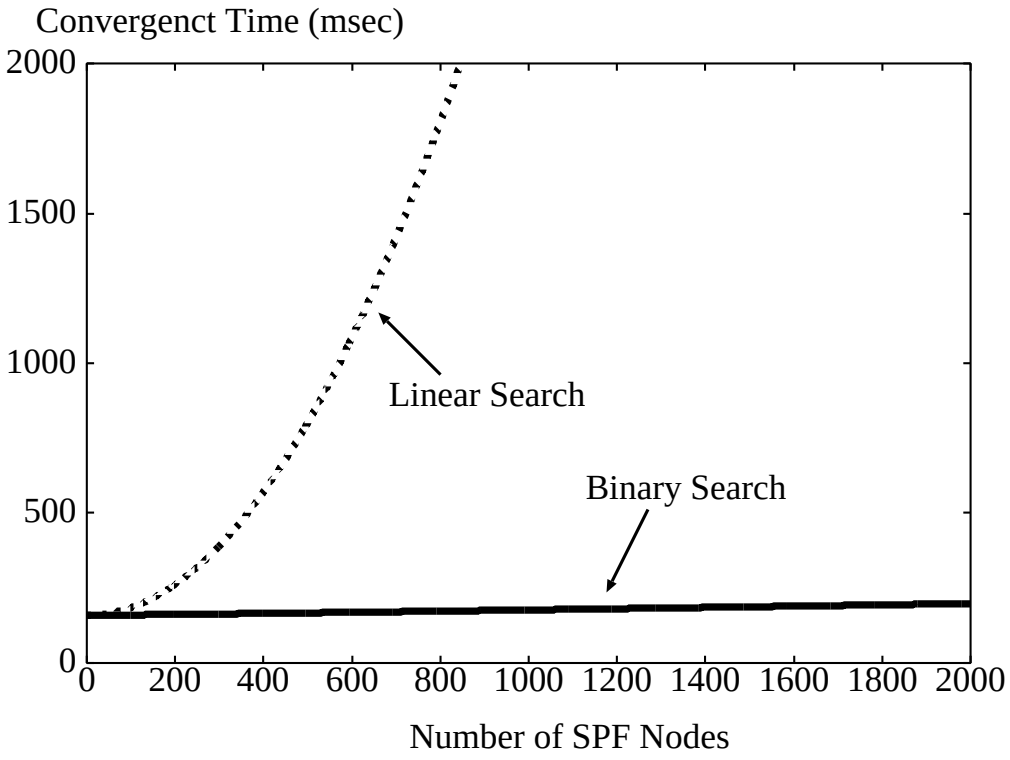


図 5.4: ネットワーク規模と収束時間の関係

表 5.4 に、この結果を抜き出したものを示す。表 5.4 から、リニアサーチを用いた SPF 計算では、ノード数 2000 のネットワークトポロジを計算するのに 11 秒近くかかってしまう。しかし、バイナリサーチを用いたSPF 計算では、同じ規模のネットワークのSPF 計算でも 200 ミリ秒の時間しか必要としない。

表 5.4: ネットワーク規模と収束時間の関係

| ノード数 | 収束時間 (バイナリサーチ)(msec) | 収束時間 (リニアサーチ)(msec) |
|---|---|---|
| 20 | 159.4 | 160.3 |
| 50 | 159.7 | 165.7 |
| 100 | 160.4 | 185.0 |
| 200 | 162.0 | 262.1 |
| 500 | 167.2 | 802.3 |
| 1000 | 177.0 | 2,731 |
| 1500 | 187.5 | 5,947 |
| 2000 | 198.3 | 10450 |

## 5.5 推定に関する考察

本章で行なった推定は、実測値を用いているため、信頼できる指標である。これが経路制御機構に関わる制限を示すため、より良い経路制御機構の設計を行なえる。

本章で行なった推定では、経路計算については省かれた。このため、経路が膨大となる次世代経路制御に関して、正しい指標とならない可能性がある。しかし、ルータの LSA 転送処理能力の向上、ネットワークの伝播遅延の減少などから、次世代インターネットでの経路収束は、より早いものとなる。そのため、経路制御機構にかかる制限は、本章の指標よりも緩いものとなるので、本章の指標を経路制御機構の制限として用いることは有効である。

# 第6章

# OSPFv3の実装

OSPFv3 プロトコルの実装を C 言語にて行なった。以下この実装を ospf6d と呼ぶ。 ospf6d は、経路制御ソフトウェアパッケージ Zebra 上に実装した。ospf6d では、Zebra の機能を利用し、設定の動的な変更、確認が行なえる。

## 6.1 次世代インターネットへの適応

第 5 章における考察の結果から、SPF 計算を行なう際の、候補リストの検索のアルゴリズムが スケーラビリティに大きな影響を与えることがわかった。このため、ospf6d の SPF 計算には、バイナリサーチアルゴリズムを採用した。

また、ネットワーク接続の検知間隔が大きく通信の安定性に関わることから、OSPF プロトコルにおいて時間を表すパラメータの default 値を、一般の実装から大きく変更した。その様子を 表 6.1 に示す。単位は秒となっている。

表 6.1: OSPF の時間パラメータの default 値

| | ospf6d | 一般の実装 |
|---|---|---|
| HelloInterval | 1 | 10 |
| RouterDeadInterval | 4 | 40 |
| RxmtInterval | 5 | 5 |

HelloInterval は、接続の検知間隔を示している。これを 1(秒) に設定することにより、体感的には接続とほぼ同時に経路が収束する環境を構築できる。RouterDeadInterval は、切断の検知間隔を示している。一般的に HelloInterval の 4 倍に設定される。これは、Hello パケットが連続して 3 つ欠落した場合は、切断として検知しても良い、ということを表している。これは経験的に決定され、現在のネットワークでは充分に有効な考え方なので、ospf6d でも採用した。次に、一般の実装では リンクが切断されてからその検知に最大 40 秒かかるが、ospf6d では最大 4 秒で切断を検知する。これは、 5 秒間隔で 発振するリンクがある場合など

は問題になるが、本研究の課題とする経路の安定度を監視する機能によって解決される。

RxmtInterval は、全ての OSPF メッセージの再送間隔を指定する。パケットの損失が現在のネットワークではレアケースだと考えられること、実装の相互接続テストを行なっていることなどから、一般の実装と同じ 5 秒に決定した。将来的には 最小値である 1 秒に変更する。

これらの値は設定により変更可能であるので、管理者の意向を反映する自由度を持っているが、考察された default 値を実装に含めて配布することは、実際のネットワークの通信環境を改善するためにとても有効である。

## 6.2 ospf6d の構造

ospf6d の構造を、図 6.1 に示す。

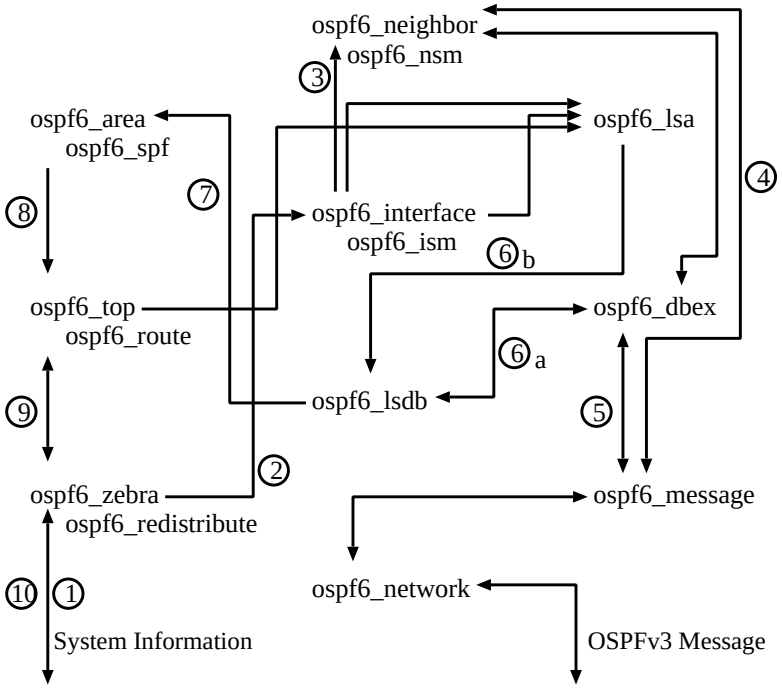


図 6.1: ospf6d の構造

ospf6d 内の各モジュールについて、以下に述べる。

### 6.2.1 ospf6_top モジュール

ospf6_top モジュールは、 OSPFv3 経路制御プロトコル全体のデータを格納する。このモジュールはルータを識別するためのルータ ID、経路の計算結果を保持す

るための経路表、OSPFv3 ドメイン全体に伝播される LSA を保持する LSDB、AS 外経路として広告する経路と LSA のマッピングテーブル、動作している OSPFv3 Area へのリストを持つ。

ospf6_top モジュールは、ospf6_area モジュールから各エリア内の経路を受け取り、経路表を作成する。また、AS 外部の経路の計算は ospf6_top モジュールが行なう。この経路表が変化した場合、ospf6_top モジュールは ospf6_zebra モジュールを介して、システムの経路表を更新する。

ospf6_zebra モジュールによって広告する AS 外経路が更新された場合、経路と LSA のマッピングテーブルを検索し、該当する LSA を特定する。そして、経路に対応する LSA の生成、更新、削除を ospf6_lsa モジュールを通して行なう。

図 6.2 に ospf6_top モジュールの構造を示す。

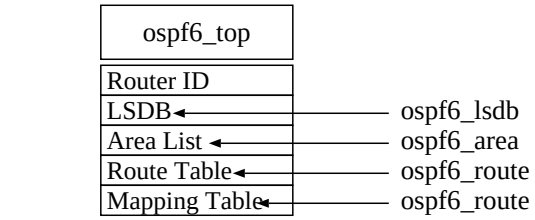


図 6.2: ospf6_top モジュールの構造

### 6.2.2 ospf6_area モジュール

ospf6_area モジュールは、Area に関するデータを格納し、SPF 計算を用いて Area 内の ネットワークトポロジを計算する。このモジュールは、Area を識別するための Area ID、Area 内のネットワークトポロジの計算結果を保持する SPF Tree、Area 全体に伝播される LSA を保持する LSDB、Area に含まれる Interface のリストを持つ。

SPF 計算および経路計算によって得られた Area 内の経路は、ospf6_top モジュールの経路表に格納される。

図 6.3 に ospf6_area モジュールの構造を示す。

### 6.2.3 ospf6_interface モジュール

ospf6_interface モジュールは、ルータの Interface に関するデータを格納し、Hello サブプロトコルを動作させる。このモジュールは、Interface を識別するための Interface ID、リンクのみに伝播される LSA を保持する LSDB、ルータ の Interface

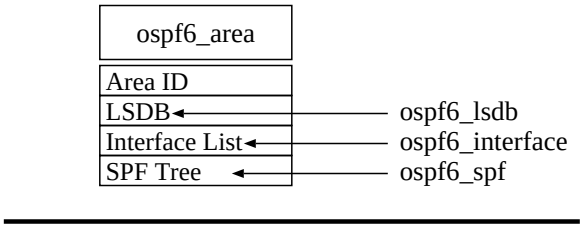


図 6.3: ospf6_area モジュールの構造

設定および状態、Hello プロトコルの設定および状態、Hello プロトコルによって自動的に検出される Neighbor のリストを持つ。

IP アドレスやリンクコストなどの Interface 状態が変化した場合、ospf6_interface モジュールは ospf6_lsa モジュールを呼び出しルータ自身の LSA を更新する。

図 6.4 に ospf6_interface モジュールの構造を示す。

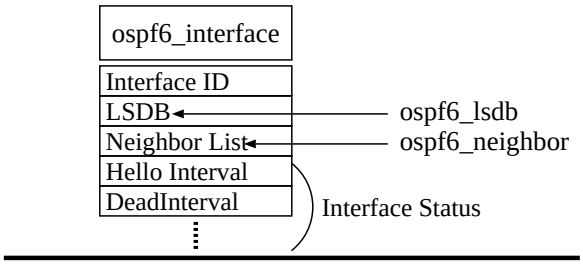


図 6.4: ospf6_interface モジュールの構造

### 6.2.4 ospf6_neighbor モジュール

ospf6_neighbor モジュールは、Hello プロトコルによって検出された Neighbor に関するデータを格納し、ospf6_dbex モジュールによって利用される Neighbor との LSDB 同期の状態を保持する。また、 OSPF メッセージ Hello を取り扱う。このモジュールは、Neighbor を識別するための Neighbor Router ID、ospf6_dbex モジュールに利用される 特殊な LSDB、Hello プロトコルに利用される Neighbor 状態を持つ。

図 6.5 に ospf6_neighbor モジュールの構造を示す。

### 6.2.5 ospf6_lsdb モジュール

ospf6_lsdb モジュールは、ospf6_top モジュール、ospf6_area モジュール、ospf6_interface モジュール各モジュールが持つ LSDB を操作するためのモジュールである。LSDB へ

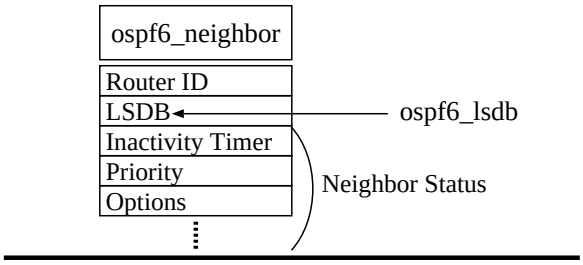


図 6.5: ospf6_neighbor モジュールの構造

の LSA の追加、削除はこのモジュールを利用して行なわれ、必要ならば ospf6_dbex モジュールを利用して新しい LSA をネットワークに伝播させる。

図 6.6 に ospf6_lsdb モジュールの概念を示す。

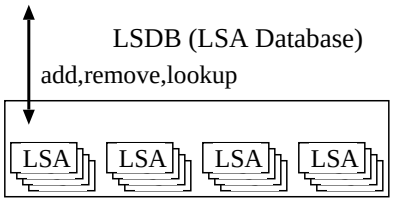


図 6.6: ospf6_lsdb モジュールの概念

### 6.2.6 ospf6_zebra モジュール

ospf6_zebra モジュールは、Zebra 独自のプロトコルを用いて、 zebra デーモンとシステム情報を交換する。交換されるシステム情報は、ルータインターフェースの状態と、経路である。

zebra デーモンから経路が渡された場合、ospf6_top モジュール内の AS 外経路の経路表を更新する。ospf6_top モジュールから経路が渡された場合、zebra デーモンに経路を渡し、システム経路表を更新する。

zebra デーモンからルータインターフェースの状態を渡された場合、適切な ospf6_interface モジュールの状態を更新させる。ルータインターフェースの状態は、ospf6_zebra モジュールから zebra デーモンへ渡されることは無い。

図 6.7 に ospf6_zebra モジュールの概念を示す。

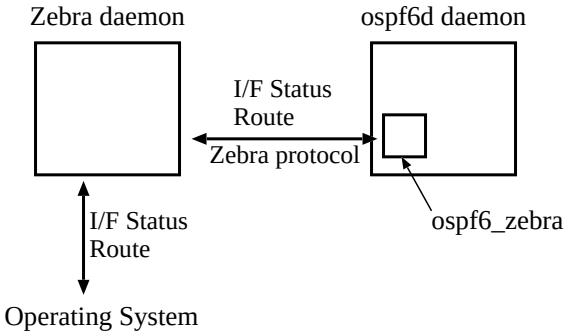


図 6.7: ospf6_zebra モジュールの構造

### 6.2.7 ospf6_lsa モジュール

ospf6_lsa モジュールは、ospf6_top モジュール、ospf6_area モジュール、ospf6_interface モジュール各モジュールから呼ばれ、ルータ自身の LSA を生成する。生成された LSA は、適切なモジュールの LSDB へと格納される。

### 6.2.8 ospf6_dbex モジュール

ospf6_dbex モジュールは、 ospf6_neighbor モジュール内の Neighbor の状態を利用して、Neighbor と信頼性のある LSDB の同期を行なう関数群である。

ospf6_dbex モジュールは、Database Description、 LS Request、 LS Update、LS Acknowledgement の四つの OSPF メッセージを取り扱う。LSDB の同期によって得られた LSA は、適切なモジュール内の LSDB に格納される。

## 6.3 ospf6d の動作概要

図 6.1 を用いて、 ospf6d の動作概要を説明する。

まず、ospf6d は ospf6_zebra モジュールによりシステムのインターフェース情報を取得する (①)。これにより OSPF インターフェースが設定され (②)、Hello プロトコルが動作を開始する。Hello プロトコルにより Neighbor が検知され (③)、Neighbor との LSDB の同期が始まる (④)。

LSDB の同期中に得た LSA は、ospf6_dbex モジュールを経て ospf6_lsdb モジュールにインストールされる (⑤、 ⑥a)。またこれと並行して、Neighbor 状態によってルータ自身の LSA が生成される (⑥b)。

LSA が変更される度に、ospf6_area モジュールの SPF 計算が行なわれる (⑦)。計算された Area 内の経路は ospf6_top モジュールの経路表に格納される (⑧)。ospf6_top モジュールでは Area 間の経路計算が行なわれ、ospf6_zebra モジュール

を通じてシステムに計算された OSPF 経路が設定される (⑨、⑩)。

## 6.4 ospf6d の使用方法

以下に、ospf6d の使用方法を示す。[] で囲まれた記述は、省略することが可能になっていることを示す。

### 6.4.1 起動

ospf6d は、設定ファイルを利用して動作する。default の設定ファイルは、/usr/local/etc/ospf6d.conf であるが、-f コマンドラインオプションによって変更可能である。

また、-d コマンドラインオプションによって、daemon モードで動作する。default では foreground のプロセスとなる。

-P コマンドラインオプションは、後述の動的設定ターミナルの待ち受け TCP ポート番号を指定するためのものである。default は 2606 である。

```
# ospf6d [-d] [-f 設定ファイル ] [-P ポート番号]
```

ospf6d は zebra daemon から インターフェース情報と経路情報を取得する。また、計算された経路は zebra を通してシステムに設定する。このため、 zebra daemon をあらかじめ起動しておく必要がある。

また、ospf6d は空の設定ファイルを最低限必要とする。設定ファイルが存在しないとき、ospf6d は起動されない。

### 6.4.2 基本設定

ここで説明する ospf6d の設定は、後述する動的設定ターミナルを利用しても行なえる。

ospf6d の設定ファイルは、コマンドの羅列となっている。また、NODE と呼ばれる設定モードを持ち、ある事柄に関する設定はこの NODE 内のみで行なわれる。

ospf6d を動作させるのに最低限必要なコマンドは、以下になっている。

```
router ospf6
  router-id Router-ID
  interface IfName area Area-ID
  interface IfName area Area-ID
        :
```

ここで、router ospf6 コマンドは、OSPF6 NODE と呼ばれる OSPFv3 全体の設定を行なうためのモードに入るコマンドである。OSPF6 NODE の中に、router-id コマンドと interface コマンドが存在する。router-id で OSPF Router-ID を設定し、OSPF で経路交換を行なうインターフェースを interface コマンドによって記述する。

router-id コマンドで設定する *Router-ID* は、OSPF ドメイン内で一意の識別子でなくてはならない。*Router-ID* は、0.0.0.1 など IPv4 アドレス表記を用いる。

interface コマンドでは、インタフェース名を *IfName* に指定するとともに、必ずインターフェースが属する OSPF Area の *Area-ID* を指定する。*Area-ID* も、*Router-ID* と同じく IPv4 アドレス表記を用いる。ここで指定した Area は自動的に作成される。interface コマンドは、OSPF を動作させたいインターフェースの分だけ記述する。

ここで作成される OSPF Interface の設定は、default の値が自動的に埋められるが、後述する INTERFACE NODE 内のコマンドで自由に設定できる。

### 6.4.3 動的設定ターミナル

動的設定ターミナルは、ユーザに設定や状態の確認、動的な設定の変更を提供する TELNET[31] ユーザインターフェースである。

```
% telnet ::1 2606
Trying ::1...
Connected to localhost.
Escape character is '^]'.

Hello, this is zebra (version 0.91).
Copyright 1996-2001 Kunihiro Ishiguro.


User Access Verification

Password:
```

動的設定ターミナルは、認証、アクセス制限をサポートしている。また、動的ターミナルの内部で動作するシェルは、コマンドラインの編集、コマンドヒストリ、コマンドの補完、ヘルプの表示など様々な機能を有している。

設定ファイルに記述できるコマンドは全て動的設定ターミナルを通じて可能である。また、動的設定ターミナルを通じて設定できるコマンドはほぼすべて設定ファイルにおいても可能である。例外として、動的設定ターミナルにしか存在し

ないコマンドは、show コマンド、enable コマンド、configure コマンドである。これらは、動的設定ターミナルのみに利用される目的を持つコマンドである。

ユーザは、動的ターミナルに接続し、認証された直後、VIEW NODE に位置する。VIEW NODE は、状態の確認のみを行なえる制限された NODE である。現在自分がどの NODE にいるかは、ターミナルのシェルプロンプトによって識別することができる。VIEW NODE のプロンプトの最後は > となっている。

```
Hostname>
```

enable コマンドは、VIEW NODE から ENABLE NODE に移行するためのコマンドである。ENABLE NODE に移行することは、管理権限が許可されたことを意味する。VIEW NODE と同じ show コマンドを持ち、状態の確認が行なえる。また、VIEW NODE では許されていない設定ファイルの確認など、管理権限を持つ者に対してのみ許される show command も存在する。ENABLE NODE のプロンプトの最後は # となっている。

```
Hostname> enable
Password:
Hostname#
```

configure コマンドは、ENABLE NODE から CONFIG NODE に移行するためのコマンドである。CONFIG NODE 内では、ユーザは様々な設定を行なえる。configure terminal コマンドは、動的設定ターミナルから設定を行なう、という指定を意味する。CONFIG NODE では、プロンプトに (config) が挿入される。

```
Hostname# configure terminal
Hostname(config)#
```

第 6.4.2 節で説明した OSPF6 NODE に移行するためには、router ospf6 コマンドを使用する。OSPF6 NODE では、プロンプトの一部が (config-ospf6) となる。

```
Hostname(config)# router ospf6
Hostname(config-ospf6)#
```

現在の NODE を抜け、直前の NODE に戻る場合は exit コマンドを使用する。

```
Hostname(config-ospf6)# exit
Hostname(config)# exit
Hostname#
```

### 6.4.4 インターフェースの設定

Interface の各種の設定を行なうためには、INTERFACE NODE を利用する。INTERFACE NODE への移行は、CONFIG NODE の interface コマンドによって行なえる。INTERFACE NODE では、プロンプトに (config-if) が挿入される。

```
Hostname(config)# interface IfName
Hostname(config-if)#
```

設定可能な OSPF Interface の値を、コマンド、default 値とともに以下に説明する。

- ipv6 ospf6 cost

  ```
  ipv6 ospf6 cost <0-65,535>
  ```

  ipv6 ospf6 cost コマンドによって、この OSPF Interface の Cost を設定する。管理者がこの値を調節することによって、OSPF が計算する経路を制御することができる。default 値は 1 である。

- ipv6 ospf6 hello-interval

  ```
  ipv6 ospf6 hello-interval <0-65,535>
  ```

  ipv6 ospf6 hello-interval コマンドによって、この OSPF Interface の HelloInterval を設定する。これは秒単位であり、OSPF が新しい接続を検知する間隔を意味している。前述の収束速度の推定により、これからのネットワークで経路の収束に 1 秒以上かかることは考えられない。このため、default 値を 1 に設定した。

- ipv6 ospf6 dead-interval

  ```
  ipv6 ospf6 dead-interval <0-65,535>
  ```

  ipv6 ospf6 dead-interval コマンドによって、この OSPF Interface の RouterDeadInterval を設定する。これは、OSPF がネットワークの切断を検知する間隔の意味を持つ。値の単位は秒となっている。

  この Interface での経路を有効にするためには、RouterDeadInterval は HelloInterval より大きい値に設定しなければならない。また、経路変化が通信トラフィック転送のパフォーマンスに影響を

与える現在のネットワークでは、経路変化の頻度を抑えるために、RouterDeadInterval は、HelloInterval の倍数として設定される。現在、一般的に RouterDeadInterval は HelloInterval の 4 倍として設定される。このため、default 値を 4 に設定した。

- ipv6 ospf6 instance-id

```
ipv6 ospf6 instance-id <0-255>
```

ipv6 ospf6 instance-id は、この OSPF Interface の Instance-ID を設定する。この値は、この Interface で動作する OSPF システムが複数存在したときに、どの OSPF システムであるかを識別するために用いられる。default 値は 0 である。

- ipv6 ospf6 priority

```
ipv6 ospf6 priority <0-255>
```

ipv6 ospf6 priority は、この OSPF Interface の Priority を設定する。これは、この Interface が接続するリンクにおいて、このルータが DR となる 優先の度合を意味する。この値は マルチアクセスネットワークのみにおいて意味を持つ。default 値は 1 である。

- ipv6 ospf6 retransmit-interval

```
ipv6 ospf6 retransmit-interval <0-65,535>
```

ipv6 ospf6 retransmit-interval は、この OSPF Interface におけるすべての OSPF メッセージの再送間隔を秒単位で設定する。これは、OSPF が消費する帯域幅と関係する。相互接続性の問題等から再送を繰り返す状況が考えられるので、経路制御トラフィックとユーザの通信トラフィックが競合する現在のネットワークでは、retransmit-interval はある程度長い値を設定するのが良い。ここから、default 値を 5 とした。

- ipv6 ospf6 transmit-delay

```
ipv6 ospf6 transmit-delay <0-65,535>
```

transmit-delay は、この Interface に LSA を送出する際に、LSA がリンク上を転送されるのに どれくらいの時間がかかるのかを秒で設定する。

OSPF では、情報が新しいかを調べるため、また古い情報を削除するために、すべての LSA を Aging する。LSA の Aging を正しく動作させ、正しい情報元が発信した新しい情報を経路制御に採用するためには、ある同時刻で見た OSPF ドメイン全体の 同一 LSA に関して、正しい情報元が持つ LSA の Age が一番 若くなくてはならない。

このため、LSA が実際にネットワークを流れる時間より 大きな時間をあらかじめ Aging してから、LSA を送出する必要がある。transmit-delay はこの 送出の際に、実際の LSA Age に足される秒数を示す。transmit-delay が実際のネットワーク転送時間より短い時間を示すと、正しい LSA を用いた計算が行なえない場合が考えられ、経路制御が正しく行なわれない可能性がある。

第5.1 節による考察から、これからのネットワークのリンク転送にかかる時間が 1 秒以上かかることは考えにくい。ここから、transmit-delay の default 値は 1 としている。

### 6.4.5 外部経路広告の設定

ospf6d では、Zebra プロトコルを用いた zebra daemon との相互動作により、外部経路の動的な広告をサポートする。このため、 RIP[32] や BGP により管理される IPv6 経路の変化は、動的に OSPF ドメインに反映される。

外部経路の広告は、OSPF6 NODE の redistribute コマンドを利用して行なう。Zebra の route-map 機能を利用することにより、経路のフィルタリングをサポートしているが、 route-map の機能については本論文の範囲外とし、ここでは触れない。ここでは、外部経路の種類と、redistribute コマンドの書式についてのみ触れる。

- redistribute kernel

  redistribute kernel [route-map *RouteMapName*]

  Zebra では、基本的に経路の設定はすべて zebra daemon を通して行なうことが想定されているが、実際には 人間が route(8) コマンド等を利用して経路設定を行なう場合、また Zebra とは別の経路制御ソフトウェアが Zebra と同時に動いている場合がある。このような経路はすべて kernel 経路として扱われる。

Zebra は、Zebra 起動前に設定されていた経路等を検知し、ospf6d に通知する。このような経路を OSPF 経路として 再広告するには、 redistribute kernel コマンドを OSPF6 NODE 内で実行しておく。

- redistribute static

```
redistribute static [route-map RouteMapName]
```

zebra daemon によって設定された経路は、static 経路として ospf6d に通知される。zebra daemon の設定が動的であることから、静的経路を動的に変更することが可能になっている。

- redistribute connected

```
redistribute connected [route-map RouteMapName]
```

ospf6d では、OSPFv3 によって経路交換する Interface に設定された経路は、内部経路として取り扱う。しかし、ルータの他の Interface に設定された経路は、内部経路として扱われない。ルータの他の Interface に設定された経路は、connected な外部経路として扱われる。直接接続する全てのネットワーク経路を OSPF で制御し、かつ全ての Interface で OSPF を動作させるわけではない、という OSPF ルータでは、redistribute connected コマンドによって connected 経路を OSPF に広告する必要がある。

- redistribute ripng

```
redistribute ripng [route-map RouteMapName]
```

Zebra の ripngd によって制御される IPv6 RIP 経路は、このコマンドを利用することによって OSPF 経路制御ドメインに再広告できる。

- redistribute bgp

```
redistribute bgp [route-map RouteMapName]
```

Zebra の bgpd によって制御される IPv6 BGP 経路は、このコマンドを利用することによって OSPF 経路制御ドメインに再広告できる。

これらを設定することによって、それぞれの経路に対する変更は自動的かつ迅速に ospf6d に通知され、広告される。広告された経路は、外部経路として OSPF により計算され、制御される。そのため、経路変化時にも即座に新しい経路が計算され、通信環境に与える影響が少ない。

# 第7章

# 評価

実装の評価は、WIDE 6Bone において実運用することによって行なった。WIDE 6Bone の構造を図 7.1 に示す。このネットワークは、本研究で実装した ospf6d 以外にも複数のルータベンダによる OSPFv3 の実装されたルータが運用されており、他の実装との間で相互接続性に問題がないことも確認されている。

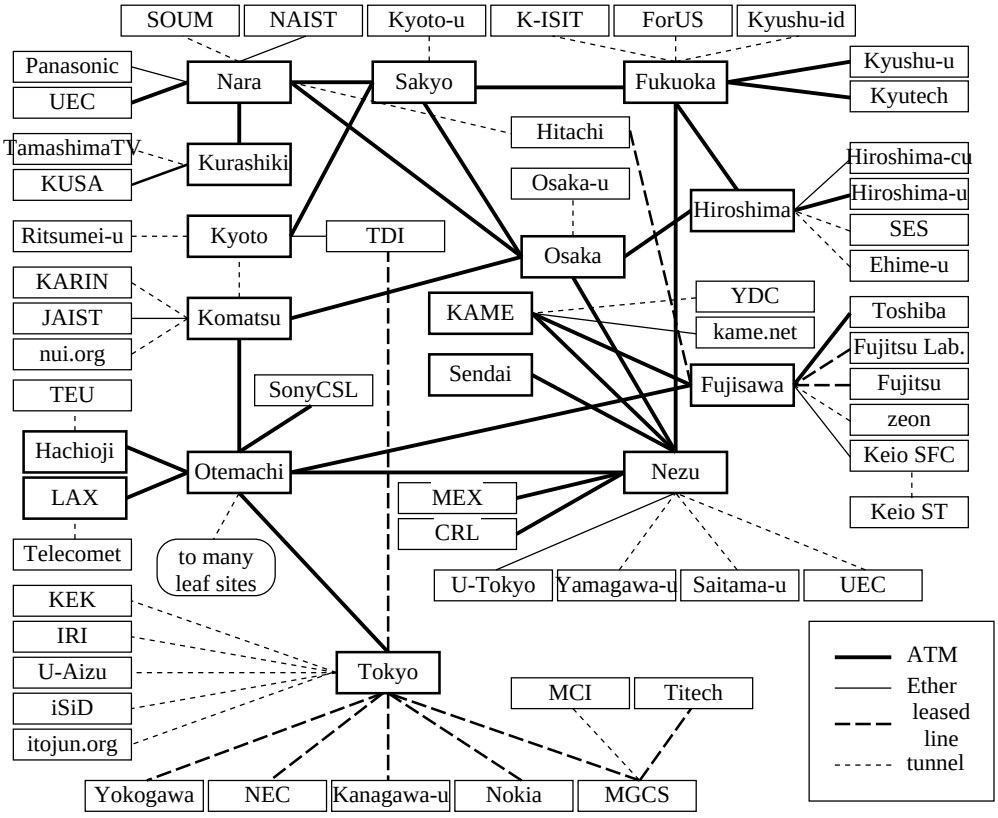


図 7.1: WIDE 6Bone の構造

表 7.1 の規模を持つ WIDE 6Bone は、現在のネットワークとしては中規模のものである。ここで、安定した経路制御を実現し、良好な通信環境を提供している。

表 7.1: WIDE 6Bone の規模

| | |
|---|---|
| ルータ数 | 28 |
| ネットワーク数 | 6 |
| ネットワーク直径 | 8 |

表 7.2 に、ospf6d 運用時の統計を示す。

表 7.2: ospf6d 運用統計

| | rg-gate.sfc | pc3.nezu |
|---|---|---|
| 動作時間 | 7 日 17 時間 11 分 58 秒 | 4 時間 33 分 6 秒 |
| 経路計算回数 | 1359 回 | 82 回 |
| SPF 計算回数 | 996 回 | 65 回 |
| 平均 SPF 計算時間 | 0.016438 秒 | 0.006438 秒 |
| 最大 SPF 計算時間 | 0.098326 秒 | 0.023430 秒 |
| 平均 SPF 計算間隔 | 669.063246 秒 | 245.576855 秒 |
| 経路数 | 145 | 146 |
| AS 内経路 | 37 | 36 |
| AS 外経路 | 108 | 110 |
| Equal Cost MultiPath | 0 | 41 |
| 最大 Hop 数 | 7 | 5 |

## 7.1 実装の動作時間

表 7.2 の動作時間から、実装が安定して動作していることがわかる。

WIDE 6Bone は実験ネットワークで、実験のためにルータを再起動することが多い。また、ospf6d も開発を続行しているので、新機能を利用するために、不具合が無くてもしばしば ospf6d を再起動する。pc3.nezu の動作時間が 4 時間となっているのはこのためである。

表 7.2 の rg-gate.sfc の動作時間から、少なくとも 1 週間は安定して実装が動作することがわかる。

## 7.2 SPF 計算時間

表 7.2 の 最大 SPF 計算時間、平均 SPF 計算時間から、経路の収束が十分に迅速であることがわかる。

ここで、Zebra ospfd の収束時間と ospf6d の収束時間を見比べると、ospf6d は SPF 計算に若干長い時間を必要としているのがわかる。ネットワークの規模がそれほど変わらず、厳密には WIDE 6bone の方が規模が小さいのにもかかわらずこの結果が出ている原因は、二つ考えられる。

まず一つ目に、ospf6d はログをファイルに記録する設定になっているが、ospfd はそうではないことである。一般的に、ログのファイルへの記録には、システムの I/O 処理を待つために、長い処理がかかる。ospf6d はシステムの解析のために多くのログがとられていたので、この差が出ていると考えられる。

二つ目に、WIDE IPv4 Backbone と WIDE 6Bone は違うネットワークである、ということである。WIDE IPv4 Backbone と WIDE 6Bone では重複する部分もあるが、基本的には別のネットワーク構成を持っており、ネットワークトポロジが変わる。SPF 計算の速度は厳密には計算量のみで測定できず、ネットワークトポロジが大きくかかわる。

このように、構成の違うネットワークの収束速度は一概には比べられないが、ospf6d は、ログ等の障害にもかかわらず迅速な収束を実現していると言える。

## 7.3 通信環境の安定度

通信環境の安定度は、定量的な評価が難しい。経路がいかに多く変化していようとも、ユーザが行なう通信に影響を与えなければ、通信環境は安定していると言える。

表 7.2 の平均 SPF 計算間隔を見ると、11 分と 4 分のように、高い頻度でネットワークトポロジが変化しているのがわかる。これは、WIDE 6Bone が実験ネットワークであることに起因しているが、実際にこの IPv6 ネットワークを利用していても、このようなトポロジ変化は体感できない。ここから、迅速な収束による、安定した通信環境の提供が実現できていると言える。

## 7.4 冗長性と耐故障性

表 7.2 の Equal Cost MultiPath は、OSPF が計算した利用できる冗長経路を示している。ここで、冗長な経路とは、一つの経路に対して複数のネクストホップルータが計算できたものを指す。

rg-gate.sfc はネットワークの中心に位置しないので、終点に対する冗長なネクストホップルータを持たないが、ネットワークの中心に位置する pc3.nezu では、41 もの経路に対して、冗長な経路が利用できる状態になっていることがわかる。

IP パケット転送機構側の問題により、これらの複数経路を同時に利用することはできないが、どちらか一方の経路が切断されたとしても、通信環境に影響を与えないような強靭なネットワークが構築されているのがわかる。

また、このような冗長的なネットワーク構成では、ループが多く存在するため、経路制御機構に負担がかかる。これからの経路制御機構は、ループを多く持つようなネットワーク構成においても安定した経路を計算する必要がある。これが達成されることによってネットワークの耐故障性が向上する。

ospf6d はネットワークのループ構成による負担にも影響されない、安定した実装であることがわかる。

## 7.5 実装の配布

ospf6d は、Zebra に含まれて世界中に配布されている。表 7.3 に、Zebra の FTP サーバにおける zebra-0.91a.tar.gz のダウンロード件数を示す。一日に約 100 件ほどのダウンロード件数がある。また、 Zebra のメーリングリストには 約 1200 人ほどが参加している。

表 7.3: Zebra FTP サーバの統計

| 日付 | ダウンロード件数 |
|---|---|
| 2001/06/28 | 103 |
| 2001/06/27 | 119 |
| 2001/06/26 | 82 |

## 7.6 相互接続性

OSPFv3 の異なる実装との間の相互接続性については実証試験により確認する以外の方法がないが、テストベッドでの運用以外に検証の機会が持てる場合には積極的に参加している。

2001 年 6 月に行われた Networld+Interop 2001 Tokyo の中のプロジェクトである IPv6 Show Case において行われた相互接続試験には、OSPFv3 の実装のひとつとして参加し、open source であるため実装のリファレンスとしての役割を果たした。

# 第8章

# 結論

次世代インターネットでは、規模が爆発的に増大する。社会基盤として利用されるようになるインターネット通信環境は、頑健かつ強靭でなくてはならない。

これを実現するためには、次世代インターネットにおける経路制御機構に、経路変化速度およびスケーラビリティの向上が重要となる。この点について問題になるのは、AS 内経路制御機構であることが第 3 章で示された。

本論文では、第 5 章において、OSPF の収束速度がどれほどなのか、さらにネットワーク規模 (スケーラビリティ) との関連性が考察された。

次に、IPv6 AS 内経路制御プロトコルである OSPFv3 を実装し、考察結果から次世代インターネットに適応するように改善した。これを 第 6 章にまとめた。

実装した経路制御機構を、次世代インターネットを想定した実験ネットワークで運用し、これを評価した。実際に運用を行なった上で問題が無いことから、本論文で述べた経路制御機構の改善は有効であることが示された。また、早くから zebra のパッケージに含めて配布することで、世界中の OSPFv3 に関わる実装をすすめる上でのリファレンスとしての役割を果たしており、意義は大きい。

本論文の成果により、社会基盤となる次世代インターネットを強靭にし、より安定した通信環境を提供できることがわかった。このため、電話、TV、あらゆるコミュニケーションをインターネットで行なうことができる。通信経路上にハードウェアの故障があろうとも、ユーザに気付かれること無くサービスが継続される。このような環境であれば、社会基盤としてもこれまでに無い利便性を持つ。

本論文によって、インターネットの規模に関しても問題が無いことが確認された。このため、我々は今まで躊躇してきた巨大なネットワークの構築を行なうことができる。これによってはじめて、「いつでもどこでもインターネットに接続できる」という環境が実現できる。

# 謝辞

本研究を進めるにあたり、御指導をいただきました、慶應義塾大学環境情報学部教授村井純氏、同じく慶應義塾大学環境情報学部教授の徳田英幸氏に深く感謝いたします。また、副査として御助言と御指導をいただいた、慶應義塾大学環境情報学部助教授の中村修氏と東京大学大学院情報理工学系研究科助教授江崎浩氏に感謝いたします。また、あたたかく著者を見守ってくださった慶應義塾大学環境情報学部助教授の楠本博之氏に深い感謝の意を表します。

日頃より研究活動のご指導をいただきました、慶應義塾大学環境情報学部専任講師南政樹氏に深く感謝します。論文についてご指導をいただいた慶應義塾大学環境情報学部講師重近範行氏に感謝の意を表します。また、公私に渡って御指導いただいた博士課程関谷勇司氏に感謝いたします。

慶應義塾大学環境情報学部・総合政策学部の徳田村井楠本中村研究室の諸氏に、感謝の念を表します。特に、SING グループの皆さんに深い感謝の意を表します。

学外では、WIDE プロジェクト IPv6 ワーキンググループにおいてご指導を頂きました東京大学情報基盤センター助手 加藤朗氏に感謝いたします。KAME プロジェクトの山本和彦氏、萩野純一郎氏、神明達哉氏にも深く感謝の意を表します。zebra の開発者である石黒邦宏氏にも多くの助言を頂きました。ここに感謝の意を表します。また、関係グループのメンバー諸氏に暖かいご支援を頂きました。ここに感謝の意を表します。

最後に OSPF の創始者、John Moy と、僕の最愛の女性のひとりである奥菜恵に深い感謝と敬愛の念を示します。

# 参考文献


[1] J. Postel. Internet protocol. Request For Comments 791, IETF, September 1981.

[2] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. Request For Comments 2460, IETF, December 1998.

[3] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). Request For Comments 1771, IETF, March 1995.

[4] J. Moy. OSPF version 2. Request For Comments 2328, IETF, April 1998.

[5] D. Oran. OSI IS-IS intra-domain routing protocol. Request For Comments 1142, IETF, February 1990.

[6] APNIC. BGP statistics. WWW. http://www.apnic.net/stats/bgp/.

[7] R. Govindan and A. Reddy. An analysis of internet inter-domain topology and route stability. In *Proceedings of the IEEE INFOCOM '97*, 1997.

[8] A. Varma A. Shaikh, L. Kalampoukas and R. Dube. Routing stability in congested networks: Experimentation and analysis. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 163–174, aug 2000.

[9] D. Haskin. A BGP/IDRP route server alternative to a full mesh routing. Request for Comments 1863, IETF, October 1995.

[10] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. Request for Comments 2475, IETF, December 1998.

[11] K. Cho. Managing traffic with ALTQ. In *Proceedings of the USENIX 1999 Annual Technical Conference: FREENIX Track*, pp. 121–128, jun 1999.

[12] A. Standard and f Telecom. Network and customer installation interfaces – asymmetric digital subscriber line (adsl) metallic interface, 1995.

[13] 情報通信技術 (IT) 戦略本部. e-Japan 重点計画. WWW. http://www.kantei.go.jp/jp/it/network/dai3/3siryou40.html.

[14] IEEE 802.3 Working Group. IEEE 802.3 CSMA/CD (ETHERNET). WWW. http://grouper.ieee.org/groups/802/3/index.html.

[15] International Telecommunication Union. The itu telecommunication standardization sector (itu-t). WWW. http://www.itu.int/ITU-T/.

[16] ATM TECHNOLOGY: the FOUNDATION for BROADBAND NETWORKS. The atm forum. WWW. http://www.atmforum.com/.

[17] J. Postel. Transmission control protocol. Request for Comments 793, IETF, September 1981.

[18] R. Chandra C. Villamizar and R. Govindan. BGP route flap damping. Request for Comments 2439, IETF, November 1998.

[19] R. Malan C. Labovitz and F. Jahanian. Origins of internet routing instability. In *Proceedings of the IEEE INFOCOM '99*, 1999.

[20] A. Bose C. Labovitz, A. Ahuja and F. Jahanian. Delayed internet routing convergence. In *Proceedings of the SIGCOMM '00*, pp. 175–187, 2000.

[21] T. Griffin and G. Wilfong. An analysis of BGP convergence properties. In *Proceedings of the SIGCOMM '99*, pp. 277–288, 1999.

[22] J. Moy. Experience with the OSPF protocol. Request For Comments 1246, IETF, July 1991.

[23] J. Moy. OSPF protocol analysis. Request For Comments 1245, IETF, July 1991.

[24] S. Abdallah D. Sidhu, T. Fu and Raj Nair. Open shortest path first (OSPF) routing protocol simulation. In *Proceedings of the SIGCOMM '93*, pp. 53–62, sep 1993.

[25] D. Ferguson R. Coltun and J. Moy. OSPF for IPv6. Request For Comments 2740, IETF, December 1999.

[26] Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, Vol. 1, pp. 269–271, 1959.

[27] D. Mills. Network time protocol (version 3) specification, implementation and analysis. Request For Comments 1305, IETF, March 1992.

[28] The NetBSD Project. The NetBSD project. WWW. http://www.netbsd.org.

[29] S. MacCanne and V. Jacobson. The BSD packet filter: A new architecture for user-level packet capture. In *Proceedings of the Winter 1993 USENIX Conference*, pp. 259–270, dec 1993.

[30] GNU Zebra. Gnu zebra. WWW. http://www.zebra.org/.

[31] J. Postel and J.K. Reynolds. Telnet protocol specification. Request For Comments 854, IETF, May 1983.

[32] C. L. Dedrick. Routing information protocol. Request For Comments 1058, IETF, June 1988.