NAIST-IS-MT9451054

修士論文

IP version 6 の実装による相互通信性の検証

島 慶一

1996 年 2 月 16 日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

島 慶一

指導教官 ： 荒木 啓二郎 教授
山本 平一 教授
平原 正樹 助教授

IP version 6 の実装による相互通信性の検証*

島 慶一

内容梗概

近年の高速回線の普及と計算機の性能向上、そして通信技術の進歩により、インターネットが飛躍的に発展している。その結果、インターネットに接続する計算機が指数関数的増加しており、2000 年初頭には現在用いられている IP version 4 のアドレスが足りなくなると考えられている。そこで、より大きなアドレス空間を持つ次世代インターネットプロトコルとして IP version 6 が提案され実験段階に入った。インターネットでは実証によってプロトコルの正当性が検証される。そこで本研究では BSD/OS 2.0 上に IP version 6 を実装することによって IP version 6 を検証する。また、完全に独立して実装された慶應義塾大学、株式会社日立製作所、株式会社ソニーによる IP version 6 の実装と相互通信実験をおこない、LAN 上で問題なく通信できることを確認した。さらに、実装によって明らかになった Path MTU Discovery や NDP の問題点に対する解決案も提案する。

キーワード

インターネット、次世代、IPng、IP version 6、IPv6、実証、相互通信

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 修士論文, NAIST-IS-MT9451054, 1996 年 2 月 16 日.

An impirical verification of interconnectivity of IP version 6*

Keiichi Shima

Abstract

Because of deployment of high speed communication lines, improvement of computing power and development of communication technology, the Internet has made remarkable progress recently. The number of computers connected to the Internet is consequently growing exponentially. Some analysts expect the disaster that IP version 4 will run out of its address space by the early 2000s. In such a situation, IP version 6 is proposed and under testing. IP version 6 has a large address space and will solve the address space problem. We implemented IP version 6 on BSD/OS 2.0 and verified interconnectivityby experiments of communicating with other independent implementations. In the experiment we find problems of Path MTU Discovery and NDP. We also propose solutions of these problems.

Keywords:

Internet, next generation, IPng, IP version 6, IPv6, imprical, interconnect

*Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT9451054, February 16, 1996.

目 次

1 はじめに **1**

1.1. 研究の背景 1

1.2. IP version 6 の概要 5

1.3. 本研究の意義 8

2 IP version 6 の実装 **10**

2.1. 設計の思想 10

2.2. インターフェース層 12

2.2.1 プロトコルキューの設計 12

2.3. ネットワーク層 13

2.3.1 IPv6 の実装 13

2.3.2 ICMPv6 の実装 17

2.3.3 NDP の実装 19

2.4. トランスポート層 19

2.5. ソケット層 20

3 IP version 6 の問題 **22**

3.1. Path MTU Discovery(ICMPv6) 22

3.1.1 Path MTU Discovery の概要 22

3.1.2 Path MTU Discovery の問題 24

3.2. アドレス解決のネットワーク層への抽象化 (NDP) 25

3.3. アドレス解決 (NDP) 26

4 相互通信実験 **29**
4.1. WIDE 相互通信実験 (第 1 回) 29
4.1.1 実験の目的と項目 30
4.1.2 実験の手順 30
4.1.3 実験の結果 31
4.1.4 考察 32
4.2. WIDE 相互通信実験 (第 2 回) 33
4.2.1 実験の目的と項目 33
4.2.2 実験の手順 33
4.2.3 実験の結果 34
4.2.4 考察 35
4.3. IOL 相互通信実験 35
4.3.1 実験の目的 36
4.3.2 実験の項目 36
4.3.3 実験の結果 37
4.3.4 考察 41

5 評価 **42**
5.1. 相互通信性の検証 42
5.2. 仕様上の不備の発見 43
5.3. IPv6 研究環境の整備 43

6 おわりに **45**
6.1. まとめ 45
6.2. 今後の課題 46

参考文献 **48**

謝辞 **52**

付録 **53**
A. 関連研究 53

B. 略語一覧 54

図 目 次

1.1 IPv4 アドレスのクラス分け 2
1.2 インターネットに接続された計算機数の推移 4
1.3 IPv4 のヘッダ形式 5
1.4 IPv6 のヘッダ形式 6
1.5 IPv6 オプションヘッダ形式 7

2.1 IPv6 を実装する 2 つの方法 11
2.2 Protocol フィールドと Next Header フィールド 15
2.3 プロトコル制御部の設計 16
2.4 ソケットとプロトコル制御部の関係 17
2.5 IPv6 ソケットアドレス 20
2.6 IPv4 ソケットアドレス 21

3.1 デフォルト経路 24
3.2 Neighbor Solicitation パケットの形式 27

4.1 NDP と ping の実験結果 31
4.2 NDP パケットのやりとり 32
4.3 TCP と UDP の実験結果 34

表 目 次

1.1 IPv4 アドレスのクラスの特徴 2
1.2 ヘッダフィールド名の対応 6

2.1 ソケット関連システムコール 21

4.1 NDP の状態遷移 38

第 1 章

はじめに

本章では本研究の意義を説明するとともに、次世代インターネットプロトコルとして注目されている IP version 6 の概要を説明する。

1.1. 研究の背景

近年、インターネットが急速に発展したため、Internet Protocol version 4(以下、IPv4 と略記する)[1] のアドレス空間の枯渇が避けられなくなった。IPv4 は、計算機同士が相互に通信するためのネットワーク層のプロトコルである。

IPv4 では通信相手を指定するために、IPv4 アドレスという 32 ビットの整数値を用いる。32 ビットという大きさは 0 から $2^{32} \simeq 4 \times 10^9$までの符号なし整数を表すことができ、一見広大な空間のように感じられる。しかしながら、過去の非効率的な IPv4 アドレスの割り当てと近年の急激な計算機数の増加が原因となり、IPv4 アドレスの枯渇が深刻な問題となってきた。

プロトコルの設計当初、IPv4 が持つ 32 ビットのアドレス空間は、図 1.1に示すように 5 つのクラスに分割されていた。5 つのクラスの中で、計算機の識別子として利用することができる部分はクラス A からクラス C までの値である。クラス D はマルチキャストアドレスとして利用され、またクラス E は予備として予約されている。表 1.1にそれぞれのクラスの特徴をまとめる。クラス A ネットワークは 128 個存在し、ひとつのクラス A ネットワークには約 1600 万台の計算

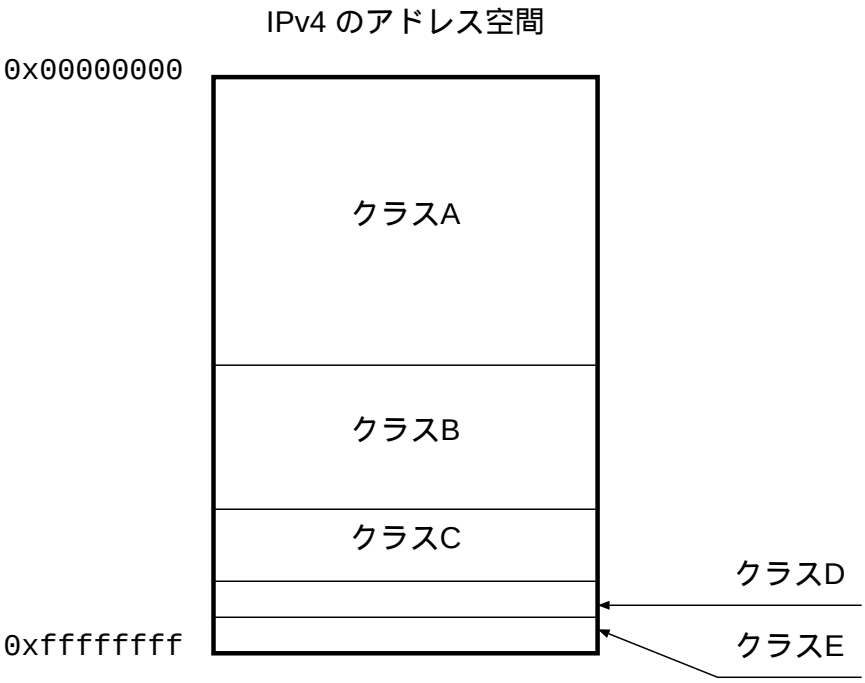


図 1.1 IPv4 アドレスのクラス分け

クラス	ネットワーク数	収容可能計算機数
クラス A	128 個	約 16000000 台
クラス B	16384 個	約 65000 台
クラス C	4194304 個	約 250 台
クラス D	(マルチキャストに利用)	
クラス E	(予約)	

表 1.1 IPv4 アドレスのクラスの特徴

機を収容することができる。ひとつのクラス A ネットワークをひとつの研究組織に割り当てると、例えその研究組織が 1000 台の計算機しか運用しない場合であっても 1600 万台分の IPv4 アドレスが割り当てられてしまう。また、128 個というクラス A ネットワークの数は、世界規模で考えると小さな値である。そのため、一般の研究組織にクラス A ネットワークを割り当てることは効率的ではない。そこで、ネットワークアドレスを割り当てる方法として次の 2 つの選択肢が考えられた。

1. ひとつの研究組織にひとつのクラス B ネットワークを割り当てる。

2. ひとつの研究組織に複数のクラス C ネットワークを割り当てる。

ひとつのクラス B ネットワークは約 65000 台の計算機を収容できる。しかしながら、現実的には 65000 台もの計算機を運用する組織は少ない。IPv4 アドレスを効率的に割り当てるには 2. の方法が望ましい。しかし、複数のクラス C ネットワークを割り当てることは、経路制御機構に問題を起こすことが予想された。当時の経路制御技術では、ひとつのクラス C ネットワークについてひとつの経路情報が必要だった。これは、複数のクラス C ネットワークを割り当てると、線形に経路情報が増加することを意味する。当時のルータの性能では経路情報の増加に耐えられなかったため、あえてひとつの組織にひとつのクラス B ネットワークを割り当ててきた。

また、インターネットの急速な発展も問題となった。図 1.2に 1981 年から 1995 年までの計算機数の推移を示す。図 1.2から分かるように、インターネットに接続された計算機は指数関数的に増加している。このまま増加していくと、2000 年初頭には IPv4 アドレスの上限に達してしまう。しかも、実際には前述のような非効率的な割り当てがおこなわれてきたため、割り当てることができる IPv4 アドレスがなくなってしまうのは図 1.2から得られる予想より早いと考えられる。

そこで、より大きなアドレス空間を持つ次世代インターネットプロトコルが必要となった。現在、IP version 6[2] と呼ばれるプロトコルが提案され、多くの研究機関で動作実験が開始されている。

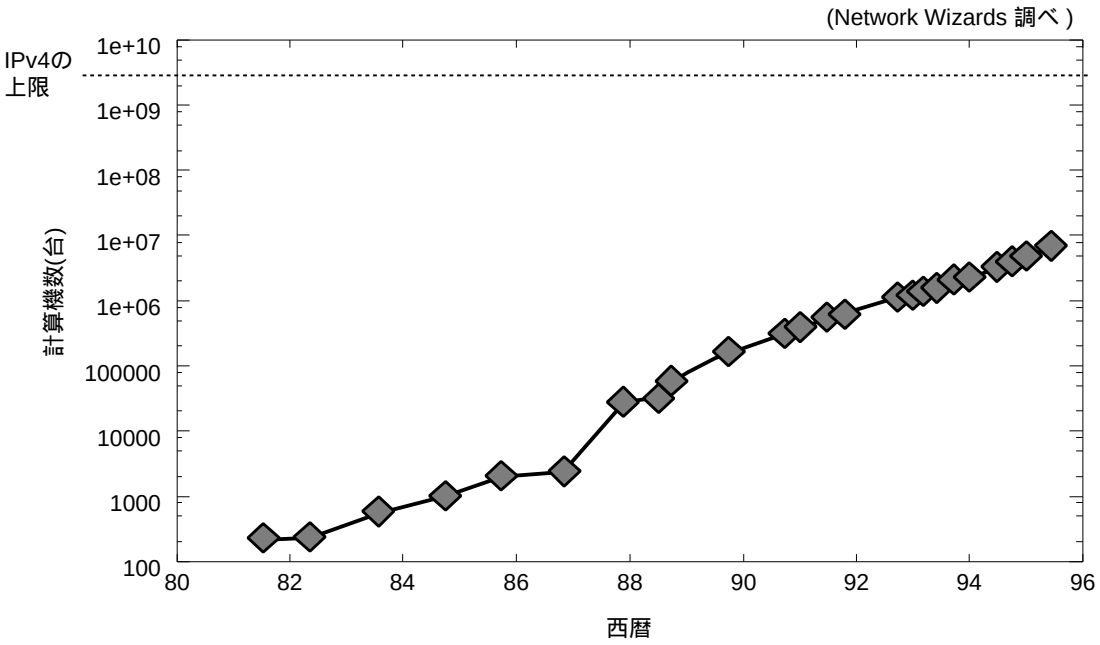


図 1.2 インターネットに接続された計算機数の推移

1.2. IP version 6 の概要

IP version 6 は Xerox Palo Alto Research Center の Stephen E. Deering 氏と Ipsilon Networks の Robert M. Hinden 氏によって提案されているネットワーク層の通信プロトコルである。以後、IP version 6 を IPv6 と略記し、IPv4 と IPv6 の両方を示す場合は、バージョン番号をつけずに単に IP と表記する。

IPv6 は以下に示す特徴を持っている。

大きなアドレス空間 IPv6 は 128 ビットの整数値で各計算機を識別する。IPv4 は 32 ビットの整数値を用いていたので、アドレス空間は $2^{96} \simeq 8 \times 10^{28}$倍に拡張されたことになる。

ヘッダの簡略化 20 年以上にわたる IPv4 の運用経験に基づき、有効に活用されることがなかった IPv4 のヘッダフィールドが削られた。図 1.3に IPv4 のヘッダ形式を、図 1.4に IPv6 のヘッダ形式を示す。図 1.3中、斜線で印を付けたヘッダフィールドが削除された。 なお、同じ目的であるものの、異な

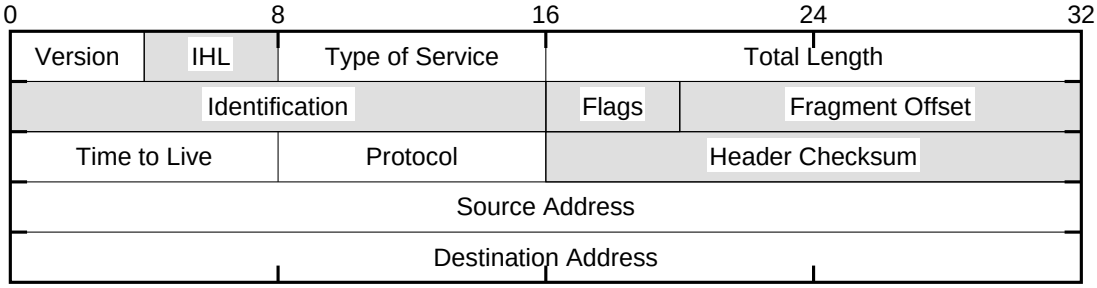


図 1.3 IPv4 のヘッダ形式

る名前に変更されたヘッダフィールドがあるため、参考として IPv4 ヘッダと IPv6 ヘッダのフィールド名の対応を表 1.2に示す。なお、それぞれの図の行幅は 32 ビットである。ヘッダを簡略化したことにより、IP アドレスの長さが 4 倍になっているにもかかわらず、ヘッダ全体の大きさは 2 倍に抑えられている。

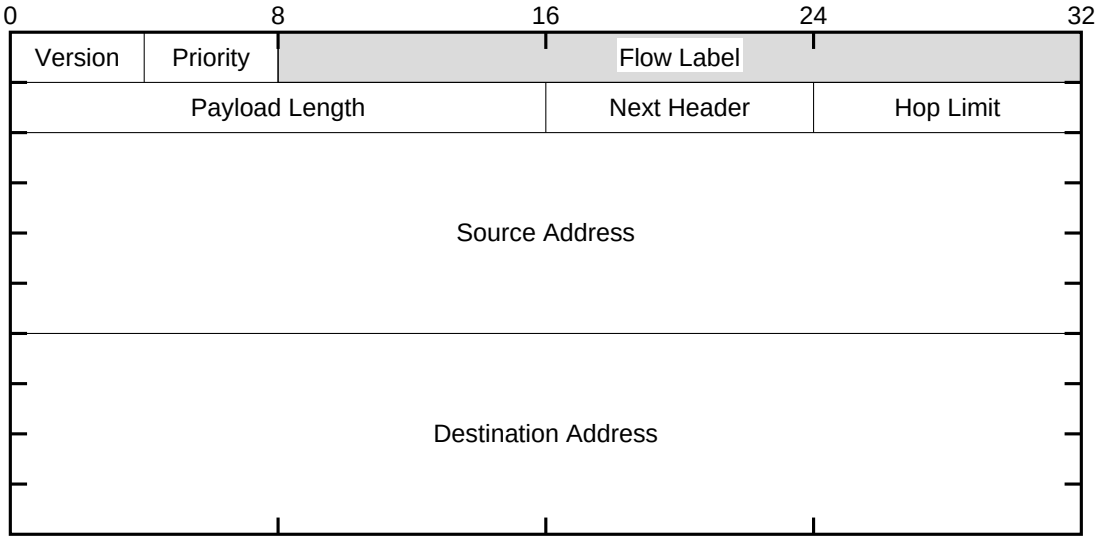


図 1.4 IPv6 のヘッダ形式

IPv4 のヘッダフィールド	IPv6 のヘッダフィールド
Type of Service	Priority
Total Length	Payload Length
Time to Live	Hop Limit

表 1.2 ヘッダフィールド名の対応

オプションヘッダの一般化 IPv4 のオプションヘッダはIPv4 ヘッダの一部となっている。IPv4 のオプションヘッダは可変であるため、ヘッダ解析に時間がかかり、転送時のオーバヘッドとなっていた。さらに、IPv4 はヘッダ長に制限があるため、おのずとオプションヘッダの長さに制限があった。そこで、IPv6 ではIPv6 ヘッダとオプションヘッダを完全に分離し、ほとんどのオプションヘッダが目的ごとに固定長のオプションヘッダを用意している。例外はルーティングヘッダである。ルーティングヘッダは途中経路上の計算機を指定しなければならないため、通過する計算機の数に比例してヘッダが長くなる。オプションヘッダをIPv6 ヘッダから追いだし、固定長にすることで、転送時のヘッダ解析によるオーバヘッドを減らしている。オプションヘッダはIPv6 ヘッダと上位層のプロトコルに渡すデータの間にいくつでも挿入することができる。つまり、目的毎に特定のオプションヘッダを用意し、IPv6 ヘッダと IPv6 データグラム本体の間にオプションヘッダを挿入することで長さの制限を解決している。図 1.5にオプションヘッダの一般的な形式を示す。

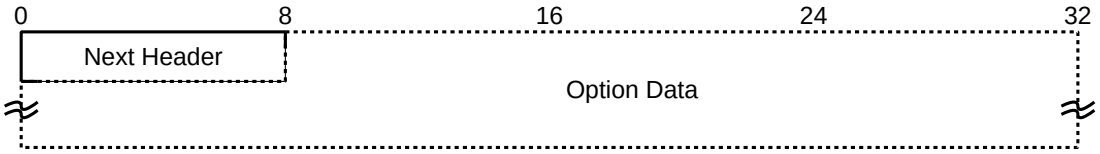


図 1.5 IPv6 オプションヘッダ形式

フローラベルの伝播 今後重要になると思われるマルチメディア通信に対する要求を満たすためには、リアルタイム通信が不可欠である。IPv6 は、データの流れを識別するためにフローラベルという識別子を設け、上位層がネットワーク層でのデータの流れに関与することができる枠組を提供している [3]。図 1.4の影を付けた部分、Flow Label がフローラベルを格納するヘッダフィールドである。

認証と機密性の枠組の提供 インターネットの利用環境が、研究から商用へ移行するに従い、通信相手の認証や通信内容の保護がますます重要になってき

た [4]。IPv6 はオプションヘッダの形式で認証 [5] や暗号化 [6] の枠組を提供する。

1.3. 本研究の意義

本研究は次の 3 つの目的を持っている。

1. 実装による IPv6 の相互通信性の検証。
2. 相互通信実験による、仕様上の不備の発見。
3. IPv6 研究環境の整備。

インターネットは、完全に独立した 3 つの実装が相互通信できることを示すことによってプロトコルの動作を検証するという、実証的な研究環境である。よって、仕様に従ってプロトコルを実装すること自体に大きな意義がある。さらに、実際にプロトコルを運用することで、紙の上での設計では見えなかった仕様の不備が発見されることもある。インターネットの世界は、完璧な仕様を考えだした後に初めて仕様に従った実装をするような環境ではなく、テスト運用をおこないながらプロトコルを洗練していくという実践的な研究環境であるので、仕様の不備を早期に発見することが重要である。IPv6 が今後のインターネットの主要プロトコルになることは、ほぼ間違いない。しかしながら、現在のところ研究者が安価で入手できる IPv6 のソースコードは少ない。無償で入手できるソースコードが、研究の発展上大きな役割を果たすことは、BSD のソースコードなどの前例から明らかである。本研究の結果得られる IPv6 のソースコードは、今後ますます活発になっていくと思われる IPv6 関連の研究基盤となる。

本論文は以下のように構成される。まず、第 2章で今回おこなった実装の設計の思想、従来のネットワーク部からの拡張部分などを解説する。次の第 3章では実装することによって明らかになった、仕様のあいまいな点を取りあげ、対策を提案する。第 4章は慶應義塾大学の徳田/村井研究室の南 政樹氏による実装、株式会社日立製作所の実装、株式会社ソニーの実装との相互通信実験、および 1996 年 2 月 5 日から 2 月 10 日にかけてアメリカの New Hampshire 大学で開催された

第 1 回 IPv6 相互通信実験の実験報告である。第 5章で本研究の評価をおこなった後、第 6章でまとめと今後の課題を述べる。

第 2 章

IP version 6 の実装

本章では、本研究でおこなった実装の設計の思想を述べるとともに、実装を次の 4 つにわけて詳しく見ていく。

- インターフェース層。
- ネットワーク層。
- トランスポート層。
- ソケット層。

実装は 4.4BSD Lite を基にして作られている BSDI 社の BSD/OS 2.0 上でおこなった。BSD/OS を採用したのは、BSD/OS 自体が商用であるので、カーネルの動作が安定していることと、最新に近いネットワークコードが利用されていることが理由である。

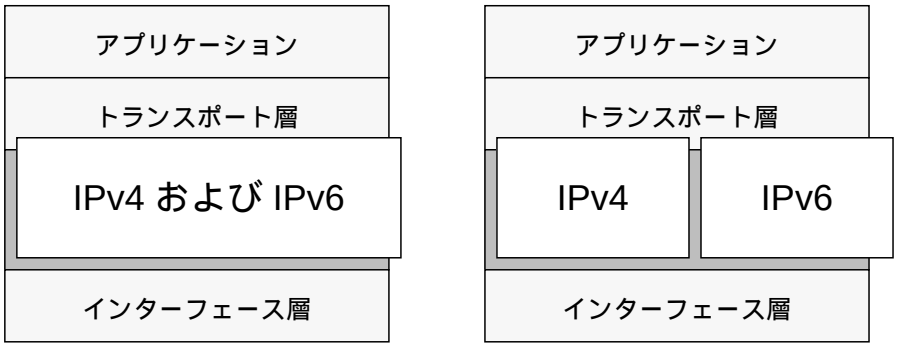
2.1. 設計の思想

IPv6 のパケットの処理方法は IPv4 に酷似している。それにもかかわらず、IPv6 は IPv4 と全く互換性がない。これは、図 1.3 と図 1.4 で示したように、ヘッダ形式が大きく変更されたことが原因である。IPv6 のようにヘッダ形式だけが異なり、他の部分は既存のプロトコル (今回の場合は IPv4) とほとんど違いがないようなプロトコルを実装する場合、次の 2 つの選択肢が考えられる。

1. IPv4 のパケット処理手続きを拡張し、IPv6 のパケット処理を追加する。

2. 新たに IPv6 専用のパケット処理手続きを追加する。

これを概念的に示した図が図 2.1である。今回は 2. の方法を採用した。理由は明



1. IPv4 の拡張　　2. IPv6 の追加

図 2.1 IPv6 を実装する 2 つの方法

晰さを追及するためである。近い将来、IPv6 は IPv4 と置き換わると思われる。そのとき、研究者が参照する IPv6 のソースコードに IPv4 の処理部が記述してあると理解しづらくなる。また、いずれ IPv4 の利用者が減っていくと考えられるので、現時点で IPv4 と IPv6 をわけて実装しておくのはよい考えである。2. の方法の欠点は、IPv4 と IPv6 が協調動作するときの処理が複雑になることである。例えば、現在 Internet Draft として公開されている BSD API[7] には、IPv4 を利用するソケットを、IPv6 での通信用に変更したり、またその逆をおこなう API の提供を示唆している。このようにソケットに結びつけられているプロトコルスタックを切り換えるような機能を実現するためには、プロトコルの実装レベルでの支援があるほうが効率的である。しかしながら、1.3節で述べたように、本研究は将来の研究基盤としての意味があるので、あえて明晰さの方を選ぶことにした。

2.2. インターフェース層

インターフェース層の役割は、ネットワークデバイス (例えば、イーサネット等) からデータリンク層のフレームを受け取り、フレームに含まれるデータを上位層であるネットワーク層に渡すこと、また、逆にネットワーク層からの入力をデータリンク層のフレームの形にして、ネットワークデバイスを通して出力することである。今回は、最も安価に利用できるネットワークデバイスであるイーサネットと、ソフトウェアのみで構成されているネットワークデバイスであるループバックデバイスを実装した。

2.2.1 プロトコルキューの設計

ネットワークデバイスからの入力はハードウェア割り込みを使って処理される。そのため、インターフェース層は入力を一時的に溜めておくキューを持たなければならない。このキューをプロトコルキューと呼ぶことにする。

4.4BSD Lite のインターフェース層のコードを拡張して IPv6 に対応させるときに考慮しなければならないことは、プロトコルキューの設計である。通常プロトコルキューはプロトコルの種類によって独立している。今回の実装にあたっては、次の 2 つの選択が可能であった。

1. IPv4 と IPv6 でプロトコルキューを共用する。
2. IPv6 専用に新たにプロトコルキューを作る。

IPv4 と IPv6 はどちらもインターネットプロトコルである。よって 1. の方法でキューを共用することは悪い考えではない。図 1.3と図 1.4を見ればわかるように、IP のヘッダにはバージョン番号を示すフィールドがある。プロトコルキューを共用したとしてもバージョン番号によって処理を振りわけることが可能であるので問題はない。実際、INRIA(Institut National de Recherche en Informatique et en Automatique) の Francis Dupont 氏によって NetBSD 上に実装された IPv6[8] は IPv4 と IPv6 でキューを共用している。しかしながら我々は、次に挙げる 2 つの理由より 2. の方法を採用した。

1. わかり易さの追及。
   IPv6 はIPv4 の後継プロトコルであるものの、IPv4 と全く互換性がないことは前節で述べた通りである。お互いに通信できないプロトコルを同じプロトコルキューで処理するとソースコードを読む者が混乱しかねない。

2. イーサネットのフレームタイプ。
   IPv4 を示すイーサネットのフレームタイプは0x0800 であり、IPv6 を示すイーサネットのフレームタイプは0x86dd である。フレームタイプは上位層のプロトコルを識別するためのものなので、フレームタイプ毎にプロトコルキューを用意する方が自然である。

2.3. ネットワーク層

IPv6 で通信をおこなうためにはネットワーク層で次のプロトコルを実装する必要がある。

- IPv6
- ICMPv6(Internet Control Message Protocol for IPv6)
- NDP(Neighbor Discovery Protocol)

ICMPv6[9] はICMP(Internet Control Message Protocol)[10] をIPv6 上で動作するように変更したものである。IPv4 のICMP にはなかった機能も拡張されている。ICMPv6 の概説と実装は2.3.2節で述べる。また、NDP[11] はARP(Address Resolution Protocol)[12] を一般化したプトロコルである。データリンク層のアドレスとネットワーク層のアドレスを対応づける。NDP は2.3.3節で述べる。

2.3.1 IPv6 の実装

IPv4 とIPv6 で大きく変更された点はアドレスの大きさとオプションヘッダの処理である。このことは、IPv4 とIPv6 ではヘッダ処理以外は大きな違いがないということを意味している。本実装でもオプションヘッダの処理を除くほとんど

の処理でIPv4のパケット処理手続きを参考にしている。本節ではまず、オプションヘッダの処理について述べる。また、ネットワーク層のプロトコルは通信のパラメータを保持するために、プロトコル毎にプロトコル制御部(Protocol Control Block、PCB)を持たなければならず、プロトコル制御部の設計が重要な意味を持つ。よって次に、本実装におけるIPv6プロトコル制御部の設計について述べる。

オプションヘッダの処理

IPv6のオプションヘッダがIPv4のオプションヘッダのようにIPヘッダの一部となるのではなく、独立したオプションヘッダになることは1.2節で述べた通りである。IPv6はオプションヘッダを識別するためにNext Headerフィールドを用いる。Next HeaderフィールドはIPv4のProtocolフィールドに対応するヘッダフィールドである。Protocolフィールドは IPv4パケットのデータ部に格納されている、上位層のプロトコルを識別するためのフィールドとして使われてきた。Protocolフィールドの名前がIPv6でNext Headerに変更された理由は、IPv6パケットのデータ部に上位層のプロトコルに渡すべきデータ以外のデータ、すなわちオプションヘッダが入るためである。IPv4のProtocolフィールドとIPv6のNext Headerフィールドの使い方を図2.2に示す。

IPv4では次の手順でパケットが処理される。

1. IPv4ヘッダにオプションヘッダが含まれているかどうかを調べる。
2. もし、オプションヘッダが含まれていた場合は、オプションヘッダを処理する。
3. IPv4データを上位層プロトコル(図2.2の例ではTCP)のパケット処理部へ渡す。

これに対してIPv6は、IPv6ヘッダ中にオプションヘッダが存在しないので、IPv6ヘッダ処理中にオプションヘッダのための例外処理をおこなう必要がない。図2.2からわかるように、IPv6ヘッダとIPv6オプションヘッダの関係は、IPv4ヘッダとIPv4での上位層プロトコルのデータの関係に似ている。そこで、IPv6オプションヘッダの処理は、順番にNext Headerフィールドの値を調べながらそ

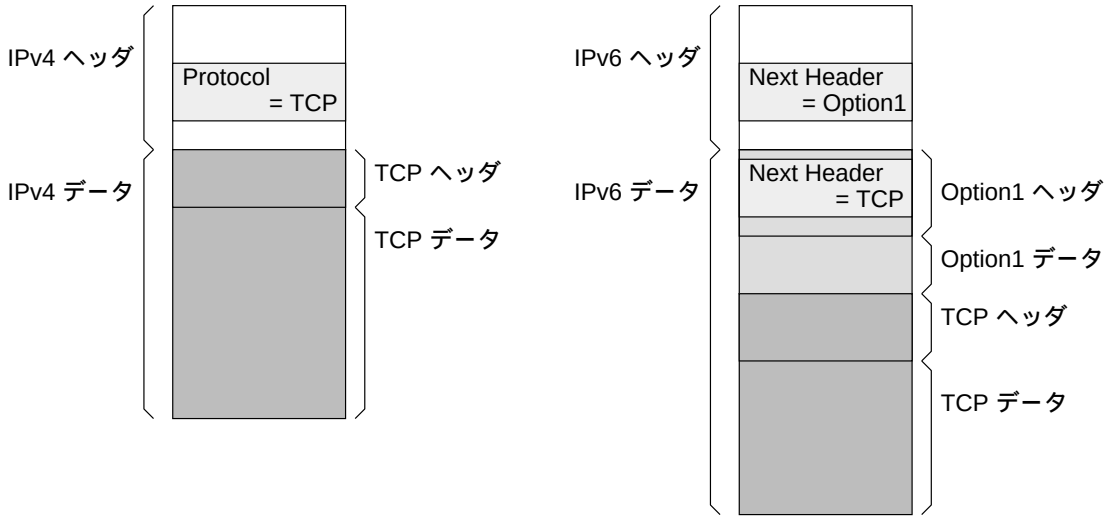


図 2.2 Protocol フィールドと Next Header フィールド

れぞれのオプションヘッダの処理部にパケットを渡していくとよいことがわかる。本実装ではオプションヘッダを次のように処理している。

1. Next Header フィールドで指定されている処理部にパケットを渡す。
2. 1. に戻る。

最終的に、最後のオプションヘッダの Next Header フィールドで指定された上位層プロトコルに処理がまわることになる。

プロトコル制御部の設計

プロトコル制御部はプロトコルの通信パラメータを保持しておくデータ群であり、通常各々のプロトコルに対して特定の制御部が定義されている。本論文中で何度も述べているように、IPv6 と IPv4 の大きな違いは IP アドレスの大きさだけである。よって IPv6 のプロトコル制御部と IPv4 のプロトコル制御部に大きな違いはない。IPv6 のプロトコル制御部は次の 2 つの方法のいずれかで実現できる。

1. IPv4 のプロトコル制御部を拡張して共用する。(図 2.3左)

2. IPv6 専用のプロトコル制御部を定義する。(図 2.3右)

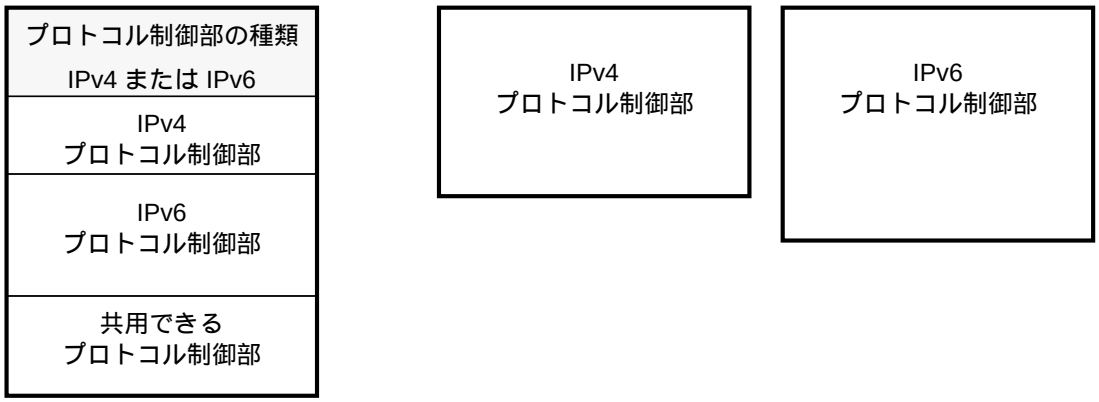


図 2.3 プロトコル制御部の設計

1. の方法には以下の利点がある。

a) プロトコル制御部を操作するコードを共用できる。

b) IPv4 のプロトコル制御部をオーバヘッドの少ない操作で IPv6 のプロトコル制御部に切り替えることができる。逆も真。

特に、b) 番目の利点は BSD API で提唱されている IPv4 ソケットと IPv6 ソケットの切り替えが簡単になることが予想される。4.4BSD Lite でのソケットとプロトコル制御部の関係を図 2.4に示す。図 2.4からわかるように、ソケットとプロトコル制御部は双方向に関連付けられており、ひとつのソケットがひとつの通信に対応している。このような実装を踏まえると、IPv4 ソケットとして作られたソケットを IPv6 の通信にも使うことができるようにするためには、図 2.3左で示した 1. の方法ようにプロトコル制御部が IPv4 と IPv6 の両方の情報を持つことができる方がよい。1. の方法を用いれば、プロトコルを変更するためにプロトコル制御部の種類を変更するだけですむからである。それに対して 2. の方法では、

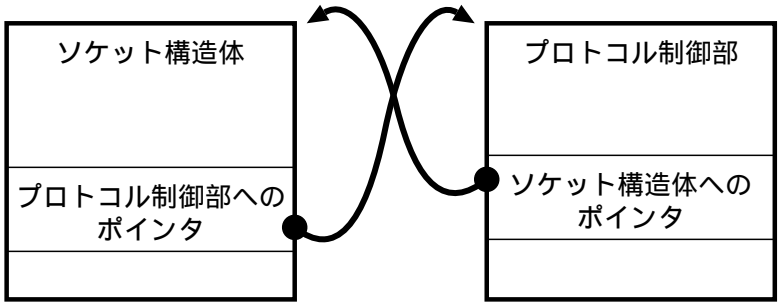


図 2.4 ソケットとプロトコル制御部の関係

IPv4 のソケットを IPv6 と結びつけるために、一旦 IPv4 のプロトコル制御部を廃棄し、新しい IPv6 のプロトコル制御部を作ったのちにソケットと再接続しなければならなくなる。

一方 2. の方法には次の利点がある。

c) 理解しやすい。

d) デバッグが容易である。

コードを読む研究者にとって、IPv6 のプロトコル制御部が IPv4 から切り離されていることは理解し易さにつながる。1.3節で述べたように、将来の研究基盤として利用することを考えた場合、コードの理解が容易であることは大きな利点である。また、経験的にいって、安定して動いているコードに手を加えると意図しない副作用が生じることがある。特に今回のように日常的に利用している IPv4 のコードに手を加えると、IPv6 の本質とは関係のない問題が生じる可能性がある。

これらの利点と欠点を考慮した結果、本実装では 2. の方法を採用した。

2.3.2 ICMPv6 の実装

ICMPv6 には以下の機能がある。

- 転送エラーの報告。

- マルチキャストグループのメンバ管理。
- NDP パケットの送受信。

それぞれ以下の節で順に説明する。なお、NDP はICMPv6 とは別のプロトコルである。それにもかかわらず上のリストに挙がっている理由は、IPv6 レベルではICMPv6 パケットの形態を取っているからである。本節では NDP に関して軽く触れるのみにとどめ、2.3.3節でより詳しく解説する。

転送エラーの報告

転送時に以下の 2 つ問題が発生すると、ICMPv6 はパケットの送信者に転送エラーを報告する。

- 目的地アドレスへの経路情報が得られない。
- IPv6 ヘッダが間違っている。
- Hop Limit が 0 になった。
- 分割されたパケットの再構築に失敗した。

これらの転送エラーの報告は、IPv4 で用いられている ICMP による転送エラーの報告とほとんど同じである。本実装でもICMP をIPv6 で利用できるように変更しただけなので、実装については触れないことにする。ICMPv6 のエラー報告にはひとつだけICMP になかった機能が導入された。RFC 1191 で定義されている、最小 MTU の発見 (Path MTU Discovery)[13] である。この機能は実装上問題がある可能性があるので第 3章で再度考察することにする。

マルチキャストグループのメンバ管理

IPv4 でのマルチキャストグループのメンバ管理は、IGMP(Internet Group Membership Protocol)[14] でおこなわれていた。IPv6 ではICMPv6 にその機能が繰り入れられている。なお、本実装はまだマルチキャストグループのメンバ管理を実装していない。

NDP パケットの送受信

NDP はデータリンク層のアドレスとネットワーク層のアドレスを対応づけるためのプロトコルである。NDP はICMPv6 の一部として仕様が定義されており、ネットワーク層からみるとICMPv6 のパケットと等価である。NDP の実装の詳細は次節で述べる。

2.3.3 NDP の実装

NDP は、データリンク層のアドレスとネットワーク層のアドレスを対応づけるためのプロトコルである。IPv4 ではARP がその役割を果たしていた。ARP はデータリンク層としてイーサネットのみを対象にしていたので、他のデータリンク層に対応するために何度か拡張がおこなわれている [15][16]。NDP はARP を発展させ、多様なデータリンク層に対応できるように改良したものである。さらに、サブネットワーク上に存在するルータを自動認識する機構 [17] と、IPv4 ではICMP に含まれていたリダイレクト機構も NDP に繰り入れられた。

今回はLAN 上で双方向通信をおこなうことを第 1 目標としていたので、ルータの自動認識機構とリダイレクト機構は未実装である。これらの機能は今後実装していく。

NDP はまだインターネットドラフトの段階で、仕様の洗練がおこなわれている途中である。実装にあたって問題になったことは次の 2 つの点である。

1. アドレス解決のネットワーク層への抽象化。
2. データリンク層アドレス解決のための仕様の不備。

以上の 2 点については、第 3章で詳しく述べ、解決案を提案する。

2.4. トランスポート層

実用的にIPv6 を利用するためにはトランスポート層のプロトコルが必要である。本実装は次の 2 つのトランスポート層プロトコルを提供する。

1. TCP(Transmision Control Protocol)

2. UDP(User Datagram Protocol)

TCP[18] と UDP[19] はどちらも IPv4 で広く利用されているトランスポート層のプロトコルである。TCP と UDP は、トランスポート層に位置しているにもかかわらず、IP に大きく依存している。そのため、ネットワーク層に IPv6 を用意しただけでは動作させることができない。本実装は IPv4 のソースコード中の IPv4 アドレス部分を IPv6 を取り扱えるように変更し、IPv6 専用の TCP と UDP を用意することで対処した。TCP および UDP の機能は IPv4 のときと同じである。

2.5. ソケット層

ソケットはアプリケーションプログラムとトランスポート層以下で実現されているプロトコルとの中継ぎとなる。それだけに、プログラマからみてわかり易くなければならず、従来のソケットと大きく外れるような使い方を定義してはならない。なお、ここでいうソケットとは BSD ソケットのことを指す。幸い、ソケットは様々なプロトコルのアドレスをソケットアドレスという抽象化された概念で取り扱うので、新しいプロトコルに柔軟に対応できる。

本実装では BSD API に従い、図 2.5に示す IPv6 用のソケットアドレスを新たに定義した。図 2.6の IPv4 用のソケットアドレスと比較すると、IPv6 特有のフロー情報が拡張されていることがわかる。 表 2.1のソケット関連システムコール

```
struct sockaddr_in6 {
    u_char  sin6_len;               /* struct sockaddr_in6 の大きさ */
    u_char  sin6_family;            /* プロトコル識別子 */
    u_short sin6_port;              /* トランスポートポート番号 */
    u_long  sin6_flowinfo;          /* IPv6 フロー情報 */
    struct  in6_addr sin6_addr;     /* 128 ビット IPv6 アドレス */
};
```

図 2.5 IPv6 ソケットアドレス

はソケットアドレスを引数にとる。プログラマは、IPv4 ソケットアドレスを IPv6 ソケットアドレスに変更するだけで、従来と同じように通信することができる。

```
struct sockaddr_in {
    u_char  sin_len;              /* struct sockaddr_in の大きさ */
    u_char  sin_family;           /* プロトコル識別子 */
    u_short sin_port;             /* トランスポートポート番号 */
    struct  in_addr sin_addr;     /* 32 ビット IPv4 アドレス */
};
```

図 2.6 IPv4 ソケットアドレス

システムコール名
int bind(*int*,*struct sockaddr **,*int*)
int connect(*int*,*struct sockaddr **,*int*)
int sendto(*int*,*char **,*int*,*int*, *struct sockaddr **,*int*)
int recvfrom(*int*,*char **,*int*,*int*, *struct sockaddr **,*int **)

表 2.1 ソケット関連システムコール

第 3 章

IP version 6 の問題

本章では、実装によって明らかになったIPv6 の仕様上の不備を示し、解決法を提案する。第2章で問題になったのは次の 3 つの点である。

- Path MTU Discovery(ICMPv6)。
- アドレス解決のネットワーク層への抽象化 (NDP)。
- データリンク層アドレス解決のための仕様の不備 (NDP)。

それぞれ、3.1節、3.2節、3.3節で考察する。

3.1. Path MTU Discovery(ICMPv6)

Path MTU Discovery は、Stephan E. Deering 氏が RFC 1191 [13] で仕様を提示した、通信ホスト間の最小 MTU を決定するアルゴリズムである。本節では Path MTU Discovery の概要を説明し、4.4BSD Lite 上に Path MTU Discovery を実装するときの問題点を考察する。

3.1.1 Path MTU Discovery の概要

現在、イーサネット、X.25、FDDI などの様々なデータリンク層の通信媒体が利用さている。これらの通信媒体は一度に送信できるフレームの大きさに上限が

ある。この上限を MTU(Maximun Transmission Unit) という。上の例では、イーサネットが 1500 バイト、X.25 が 576 バイト、FDDI が 4352 バイトとなっている。つまり、2000 バイトのデータをイーサネットと使って送信しようとすると、1500 バイト+500 バイトというように 2 つのパケットにわけて送信する必要がある。もし、2 台の計算機間で使われているデータリンク層の通信媒体がすべて同一のものであれば、ネットワーク層からデータリンク層へデータを渡すときに MTU を越えないように注意することで、ひとつのパケットを分割することなく通信することができる。しかしながら、実際には組織のバックボーンネットワークに FDDI を使い、末端のネットワークにはイーサネットを使うというように、異なるデータリンク層が相互接続されているのが実状である。

ここで、バックボーンに接続している計算機が末端のネットワークに接続している計算機と通信する場合を考える。バックボーンに接続している計算機は FDDI の MTU である 4352 バイトのパケットを送信できるが、イーサネットに転送する時点で 1500 バイト以下のパケットに分割しなければならない。ひとつのパケットが途中経路で複数のパケットに分割されることをパケットの分割 (Fragment) という。パケットの分割が通信に悪影響を与えることはよく知られているので [20]、なるべくパケットの分割を起こさないように通信することが望ましい。

パケットの分割を避けるためには、通信をおこなう計算機間の最小の MTU を知らなければならない。Path MTU Discovery はその最小 MTU を求めるアルゴリズムである。Path MTU Discovery のアルゴリズムを以下に示す。

- 送信計算機
  1. 最小と思われる MTU でパケットを送信する。
  2. 途中のルータからエラー報告があった場合は MTU を小さくして再送する。
  3. 最後の MTU を発見してから一定時間以上が経過した場合、MTU を出力インターフェースの最大 MTU に再設定する。
- 途中のルータ
  1. 転送しようとしたパケットがフラグメントを起こしたら、送信計算機にエラー報告をおこなう。

3.1.2 Path MTU Discovery の問題

通信をおこなうたびに Path MTU Discovery を用いて最小 MTU を調べることは効率的ではないので、Path MTU Discovery を適用して判明したある計算機に対する最小 MTU をキャッシュしておく必要がある。4.4BSD Lite では、計測された MTU を経路情報と一緒に経路制御表に含めている。この方法には次の利点と欠点がある。

利点 新たに MTU のための表を作る必要がなく、経路情報と同時に MTU が得られる。

欠点 デフォルト経路が経路制御表に登録されている場合、MTU の値が意味のないものになる可能性がある

デフォルト経路の先には図 3.1のように経路制御表に登録されていないあらゆるネットワークが接続されている。しかしながら、現在の 4.4BSD Lite を基にした

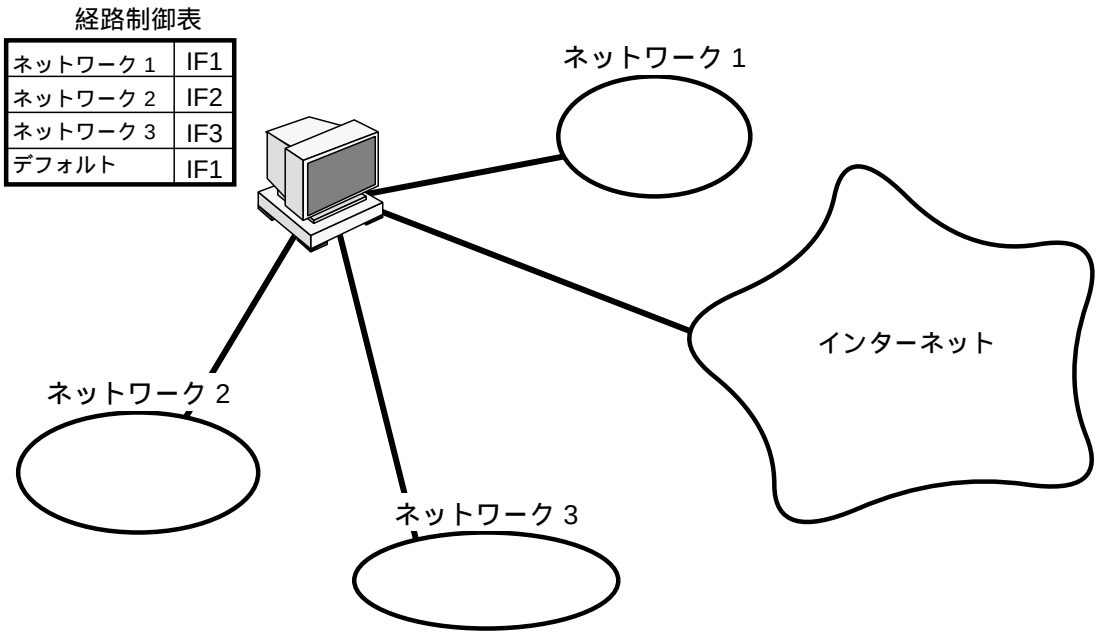


図 3.1 デフォルト経路

実装では、デフォルト経路にひとつの MTU しか設定できない。2 者間で通信をおこなう場合に最大のスループットを得るためには、パケットの分割を起こさない最大 MTU で通信をおこなうのが理想であるにもかかわらず、これでは必要以上に小さな MTU が採用されてしまう可能性がある。

そこで、MTU のキャッシュを経路制御表と別に持つ方法を提案する。経路制御情報の検索に加えて MTU の検索が必要になるという欠点があるものの、経路制御表中ではデフォルト経路になっている計算機と最適な MTU で通信できるようになると思われる。MTU のキャッシュ表の大きさに関しては、次の 2 つの理由から問題にならないと考える。

- MTU のキャッシュ表には、実際に通信している相手、あるいは通信したことのある相手しか登録されることはない。
- 一台の計算機が通信する相手は経路制御表のようには多くならない。

今後、一台の計算機が通信する相手の数を統計的に調べ、上で述べたことの妥当性を確かめるとともに、実装して実験していくことになるだろう。

3.2. アドレス解決のネットワーク層への抽象化 (NDP)

本節では、IPv6 でのアドレス解決がネットワーク層に抽象化されたことにより、IPv4 では問題にならなかった 4.4BSD Lite の IP パケットの出力ルーチンのアルゴリズムが、IPv6 では問題になることを述べる。

IPv6 では、ネットワーク層のプロトコルである ICMPv6 を用いてデータリンク層のアドレスを解決するようになった。ICMPv6 を用いることによって、様々なデータリンク層を統一的に取り扱うことができるようになっている。ICMPv6 は IPv6 の上位に位置するプロトコルであるため、ICMPv6 パケットは出力過程で IPv6 パケットの出力ルーチンを通る。IPv6 パケットの出力ルーチンは、上位層から入力された IPv6 パケットを次の手順で処理する。

1. 経路情報の検索
2. 1. で得られた経路情報から出力

ICMPv6 パケットも例外ではない。ここで問題が生じる。IPv6 ではアドレス解決要求がマルチキャストされる。そのため、IPv6 のパケット出力ルーチンには、宛先アドレスが IPv6 のマルチキャストアドレスであるような IPv6 パケットが渡されることになる。IPv6 パケット出力ルーチンは、マルチキャストアドレスに対して経路情報の検索を試みるが、マルチキャスト経路制御をしない計算機には通常そのような経路情報は存在しない。よって経路情報の検索に失敗し、ICMPv6 パケット、すなわちアドレス解決要求が出力できない。

なお、IPv4 ではこの問題が発生しない。なぜならば、IPv4 のアドレス解決プロトコルである ARP は、IPv4 パケットとして送信されるのではなく、別プロトコルである ARP として直接データリンク層に出力されるからである。

この問題は、データリンク層アドレスの解決という、本来データリンク層で閉じているべき問題を処理するために、複数のデータリンクを相互接続するために用いられる情報である経路情報を検索してしまうことが原因である。そこで、この問題を解決するために、経路情報の検索をせずに IPv6 パケットを出力する機構が必要になる。本実装では、まだ上述の要求を満たす IPv6 パケット出力ルーチンを実装していない。現時点では、マルチキャストアドレスに対する経路情報を静的に設定することで問題を回避している。しかしながら、本質的には経路情報に左右されずに処理されるべき問題である。

3.3. アドレス解決 (NDP)

本節で述べるのは、永久にアドレス解決が終了しない可能性があるという問題である。NDP はある IPv6 アドレスに対応するデータリンク層のアドレスを見付けるために、Neighbor Solicitation パケットと呼ばれる ICMPv6 パケットをマルチキャストする。Neighbor Solicitation パケットを受けとった計算機は、送信者に自分のデータリンク層のアドレスをユニキャストで返送する。返送される ICMPv6 パケットを Neighbor Advertisement パケットという。Neighbor Solicitation パケットの形式を図 3.2に示す。最初の 64 ビットは ICMPv6 ヘッダである。Target Address に対象となる IPv6 アドレスが格納して送信される。Options 以下は NDP のオプションヘッダで次のオプションが用意されている。

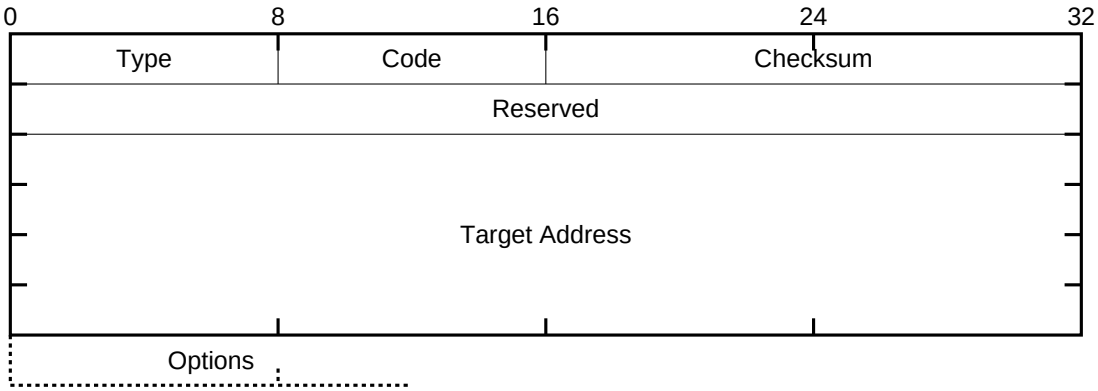


図 3.2 Neighbor Solicitation パケットの形式

Source link-layer address Neighbor Solicitation パケットを送信した計算機のデータリンク層のアドレス。

送信者のデータリンク層アドレスはオプションなので、必ずしも付ける必要はない。ここで問題が発生する。もし、Neighbor Solicitation パケットに送信者のデータリンク層のアドレスが含まれていないと、Neighbor Solicitation パケットの受信者は送信者に Neighbor Advertisement パケットを送信するので、送信者に対する Neighbor Solicitation パケットを発することになる。このときの Neighbor Solicitation パケットに、やはり送信者のデータリンク層アドレスを含めなかった場合、永久にアドレス解決がおこなわれないという事態が生じる。

IPv6 のアドレス体系 [21] には、リンク内アドレスという概念がある。これは、LAN 内でのみ有効な IPv6 アドレスで、通常データリンク層のアドレスを IPv6 アドレスの一部に組込む。リンク内アドレスはアドレスの先頭 8 ビットを調べることで判定できるので、Neighbor Solicitation パケットの送信ホストの IPv6 アドレスがリンク内アドレスである場合は、IPv6 アドレスからデータリンク層のアドレスが抽出できる。しかしながら、Neighbor Solicitation パケットの送信ホストの IPv6 アドレスにリンク内アドレスを用いなければならないという制限はない。NDP の仕様書の一部を引用する。

> If the source address of the packet prompting the solicitation is the same as one of the addresses assigned to the outgoing interface, that address SHOULD be placed in the IP Source Address of the outgoing solicitation. Otherwise, any one of the addresses assigned to the interface should be used.

つまり、Neighbor Solicitation パケットの送信ホストの IPv6 アドレスは出力インターフェースに付けられている IPv6 アドレスならどれでもよいということである。

この問題の原因は、Source link-layer address オプションの取り扱いにある。Source link-layer address について記述されている部分を引用する。

> The sender SHOULD include its link-layer address (if it has one) in the solicitation as a Source Link-Layer Address option.

仕様書では、Source link-layer address を含めるべきであるとしか規定していない。ここを Source link-layer address を含めなければならない、と規定すればこの問題は解決する。

なお、この問題を NDP の仕様設計者である Thomas Narten、Erik Nordmark 両氏に報告したところ、仕様の間違いであることが判明した。現在の仕様書では本論文で提案した通りに修正されている。

本実装は Neighbor Solicitation パケット送信時に、必ず Source link-layer address を含めている。

第 4 章

相互通信実験

本章では、1995 年 12 月 14 日 ~ 15 日および 1996 年 1 月 29 日 ~ 30 日に WIDE プロジェクトによって行なわれた慶応義塾大学での相互通信実験と、1996 年 2 月 5 日 ~ 10 日に New Hampshire 大学の Interoperability Laboratory で開催された第 1 回 IPv6 相互通信実験の概要を説明し、その結果を報告する。以後、それぞれの実験を WIDE 相互通信実験 (第 1 回)、WIDE 相互通信実験 (第 2 回)、IOL 相互通信実験と呼ぶことにする。

4.1. WIDE 相互通信実験 (第 1 回)

本節は 1995 年 12 月 14 日 ~ 15 日に慶応義塾大学でおこなわれた相互通信実験の結果を報告する。なお、参加組織は次の 4 つである。

- 奈良先端科学技術大学院大学 [22]
- 慶應大学 [23]
- 株式会社日立製作所
- 株式会社ソニー [24]

4.1.1 実験の目的と項目

IPv6 はネットワーク層の通信プロトコルである。実際に計算機同士が通信するためには、ネットワーク層の下に位置するデータリンク層の通信プロトコルを使う必要があり、そのためにはネットワーク層とデータリンク層のアドレスを結びつけなければならない。この動作をアドレス解決と呼び、IPv6 では NDP がこの役割を担っている。本実験の目的は、NDP によるアドレス解決の動作を確認することである。本実験では次の 2 つの項目を実験することによって NDP の動作を確認する。

1. 適切な NDP パケットが送受信されることを観測する。
2. Ping が動作することを確認する。

4.1.2 実験の手順

前節で提示した 2 つの項目に対し、それぞれ次の方法で実験する。

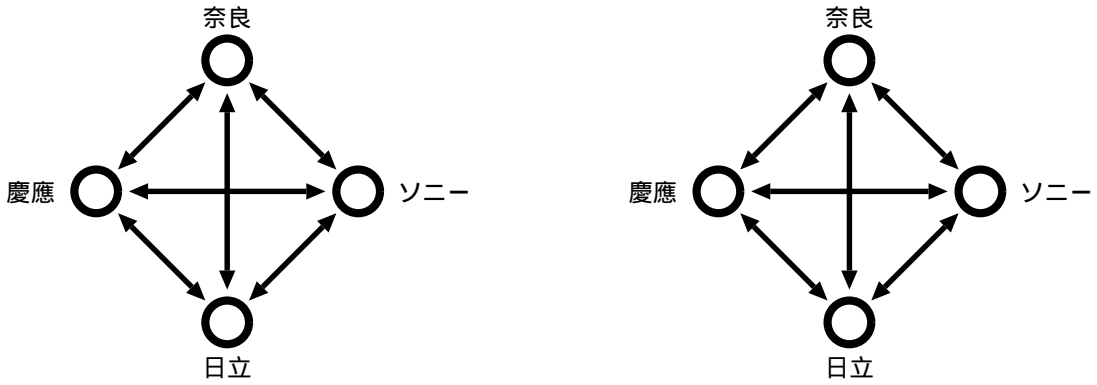
1. 適切な NDP パケットが送受信されることを観測する。
   IPv6 では以下の手順でデータリンク層のアドレスを獲得する。
   (a) パケットを送信する際に宛先ネットワーク層アドレスに対応するデータリンク層アドレスが未知である場合、Neighbor Solicitation パケットを送信する。
   (b) Neighbor Solicitation パケットを受信した計算機は、自分のデータリンク層アドレスを含めた Neighbor Advertisement パケットを返送する。
   (c) Neighbor Advertisement パケットを受信した時点でアドレス解決が終了する。

   LAN 上を流れるパケットを tcpdump コマンドを用いて観測し、NDP の仕様に従ったパケットが送受信されているかどうかを観測する。
2. Ping が動作することを確認する。
   1. の結果、適切に通信相手のデータリンク層アドレスが解決できていれば、

上位層から相手計算機に対してIPv6パケットを送信できるはずである。そこで、INRIAのIPv6配付パッケージに含まれているpingコマンドを用いてLAN上の他の計算機にICMPv6 Echoを送信し、ICMPv6 EchoReplyが返送されるかどうかを調べる。

4.1.3 実験の結果

図4.1に実験の結果を示す。まず、NDPのパケットのやりとりについての結果



1. NDPパケットの送受信　　2. Pingの送受信

図4.1 NDPとpingの実験結果

(図4.1左) を説明する。図中の矢印は、最初にパケットを送信した方向を示している。例えば、奈良の計算機にfe80::1b3d:9293というIPv6アドレスがついており、慶應の計算機にfe80::20:af75:6d49というIPv6アドレスがついているとする。奈良が慶應にパケットを送信しようとした場合、図4.2に示すようなNDPパケットのやりとりがなければならない。なお、IPv6のアドレスは16進数で表したIPv6アドレスを2バイトづつコロンで区切って表記する。また、コロンの連続は途中がすべて0であることを意味する[21]。

図4.2はtcpdumpコマンドの結果を一部引用したものである。最初のパケットがNeighbor Solicitationパケットであり、fe80::20:af75:6d49というIPv6アドレ

図 4.2 NDP パケットのやりとり

スを持つ計算機のデータリンク層アドレスを要求している。2 つ目のパケットは Solicitation に対する返答で、fe80::20:af75:6d49 に対応するデータリンク層アドレスが含まれている。この 2 つのパケットが観測されたとき、奈良から慶應への矢印がつけられる。実験の結果、図 4.1左に示したように、すべての実装の間で適切な NDP パケットのやりとりが観測された。

次に ping の送受信の結果である。矢印は NDP パケットのやりとりと同じ意味を持っている。例えば、奈良から慶應に対して ping を実行し、応答があれば奈良から慶應に矢印が引かれる。図 4.1右に示したとおり、すべての実装の間で ping の送受信が確認できた。

4.1.4 考察

WIDE 相互通信実験 (第 1 回) は、IPv6 を実装するにあたって必ず実現されなければならない基本的な機能であるアドレス解決を確認するためのものであった。NDP のパケットのやりとりを調べてみた結果、アドレス解決の手順に問題がないことがわかった。また、ping の送受信による実験から、アドレス解決の結果獲得されたデータリンク層アドレスが適切に利用されていることがわかった。よって、現状の実装の上に IPv6 の実装を作りあげていっても問題がないと予想できる。

なお、本実験とともに開かれたミーティングにおいて、NDP がアドレス解決に失敗する可能性が指摘された。この問題はすでに 3.3節で述べた。

4.2. WIDE 相互通信実験 (第 2 回)

本節は 1996 年 1 月 29 日 ~ 30 日に慶応義塾大学でおこなわれた相互通信実験の結果を報告する。なお、参加組織は次の 3 つである。

- 奈良先端科学技術大学院大学
- 株式会社日立製作所
- 株式会社ソニー

4.2.1 実験の目的と項目

WIDE 相互通信実験 (第 1 回) によって IPv6 の基本的な機能である NDP によるアドレス解決が動作することがわかった。そこで、本実験では IPv6 を利用するトランスポート層を実装し、お互いの実装の間で相互通信できるかどうかを確認する。なお、トランスポート層に実装するプロトコルは TCP と UDP である。実験では次の項目を確認する。

1. IPv6 を使う telnet コマンドで相手計算機に遠隔ログインし、TCP が動作することを確認する。
2. IPv6 を使う inetd を作り、inetd 内部で実装されている UDP のサービスを利用し、UDP が動作することを確認する。

4.2.2 実験の手順

前節で挙げた 2 つの項目に対し、それぞれ次の方法で実験する。

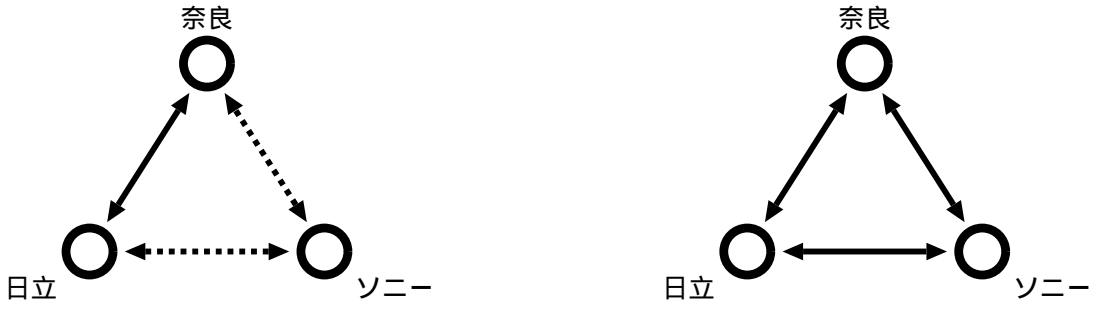
1. Telnet による TCP の確認。
   BSD/OS 2.0 に含まれる 4.4BSD Lite の telnet と telnetd のソースコード

を IPv6 を用いるように変更して再構築する。telnet コマンドで同一 LAN 上の他の計算機に遠隔ログインし、また telnetd を使って他の計算機からの telnet を受けつける。

2. Inetd による UDP の確認。
   UDP のポート 7 は echo サービスとして予約されており、ポート 7 宛てに UDP で送られたデータはそのまま UDP を用いて送信者に返される。Inetd は echo サービスを実装しているので、BSD/OS 2.0 に含まれる 4.4BSD Lite の inetd のソースコードを IPv6 を用いるように変更して再構築し、UDP echo を試すことによって UDP の動作を確認する。

4.2.3 実験の結果

図 4.3に実験の結果を図示する。まず、図 4.3左の TCP による通信の結果を説



1. TCP での通信結果　　2. UDP での通信結果

図 4.3 TCP と UDP の実験結果

明する。矢印の向きは、telnet コマンドを実行した方向を表している。奈良と株式会社日立製作所の実装の間では問題なく TCP 接続ができ、telnet、telnetd 共に動作した。株式会社ソニーの実装は TCP の緊急データ (Urgent data) が未実

装であったため、緊急データを用いて通信できなかったものの、通常データの送受信には問題がなかった。

続いて図 4.3右の UDP による通信の結果である。矢印の向きは UDP echo を要求した方向を示している。それぞれの実装で inetd を起動し、お互いに UDP echo を要求したところ、すべての実装の間で問題なく UDP データがやりとりされた。

4.2.4 考察

本実験は IPv6 をネットワーク層のプロトコルとして利用した場合に問題が生じないかを確認するためのものであった。そこで、IPv6 をネットワーク層として利用するトランスポート層プロトコルを実装し、相互通信実験をおこなった。実装したトランスポート層プロトコルは TCP と UDP である。TCP は株式会社ソニーの実装が不十分であったため、一部の TCP の機能を利用した通信ができなかったものの、独立した 3 つの実装の間での相互通信が確認できた。UDP での通信にはなんの問題もみられなかった。本実験の結果、IPv6 をネットワーク層の通信プロトコルとして利用できることが確認された。

4.3. IOL 相互通信実験

本節では、1996 年 2 月 5 日 ~ 10 日に New Hampshire 大学の Interoperability Laboratory で開催された第 1 回 IPv6 相互通信実験の結果を報告する。本実験の参加組織は以下のとおりである。

- WIDE Project
  - 奈良先端科学技術大学院大学
  - 慶応義塾大学
  - 株式会社日立製作所
- Bay Networks, Inc.
- Digital Equipment Corporation

- FTP Software, Inc.
- INRIA Rocquencourt
- Mentat, Inc.
- Process Software Co.
- SICS
- Sun Microsystems, Inc.
- Xerox/Palo Alto Research Center

4.3.1 実験の目的

本実験は仕様に従って独立して実装されたIPv6で相互通信できることを確認するためのもので、WIDE相互通信実験(第1回、第2回)と同じ目的である。本実験ではNDPの動作の確認と上位層プロトコル(TCP)の動作確認をする。

4.3.2 実験の項目

IOLで準備されていた実験項目は次のとおりである。

1. NDPの動作確認
   - Neighbor Solicitation/Advertisement
   - Router Solicitation/Advertisement
   - Redirect
2. 上位層の動作確認
   - ping
   - telnet

ただし、本実装はルータとしての機能を実装していないため、NDP のルータ関連項目は実験していない。結局、本実装に対して次の項目を実験した。

1. NDP が仕様どおりの状態遷移をするか
2. ping ができるか
3. telnet ができるか

なお、NDP の状態遷移を表 4.1に示す。

4.3.3 実験の結果

実験は IOL の実験手引書に従っておこなわれた。以下、その内容と結果を述べる。

1. 実験対象となっている計算機に ICMPv6 Echo を送信する。実験対象の計算機は、ICMPv6 EchoReply を返すために、Neighbor Solicitation パケットを送信する。その Neighbor Solicitation パケットに対して Neighbor Advertisement パケットを返送しなかった場合、実験対象の計算機は Neighbor Solicitation パケットを規定回数再送した後にエントリを消去する。この動作は表 4.1の 1、2、3 に相当する。このパケットの流れをパケットモニタで観測し、実験対象となっている計算機のコマンドを使ってエントリが消滅したことを確認する。

   結果 正しい動作が確認できた。この実験から、表 4.1の 1、2、3 の状態遷移が確認できた。

2. 実験対象の計算機が Neighbor Solicitation パケットを規定回数再送し終わる前に、Neighbor Advertisement パケットを送る。その際、Neighbor Advertisement パケットの Solicited フラグを 0 とし、Override フラグを 1 または 0 とする。正しく動作していれば表 4.1の 4 の状態遷移に従って STALE になるはずである。ここで ICMPv6 Echo を実験対象の計算機に送信する。STALE 状態の場合、実験対象の計算機は ICMPv6 EchoReply を返送した後、表 4.1の 12 に従って DELAY 状態に入る。DELAY 状態がタイムアウトすると、

	State	Event	Action	New state
1	-	Packet to send.	Create entry. Send multicast NS. Start retransmit timer	INCOMPLETE
2	INCOMPLETE	Retransmit timeout, less than N retransmissions	Retransmit NS Start retransmit timer	INCOMPLETE
3	INCOMPLETE	Retransmit timeout, N or more retransmissions.	Discard entry Send ICMP error	-
4	INCOMPLETE	NA, Solicited=0, Override=any	Record link-layer address.	STALE
5	INCOMPLETE	NA, Solicited=1, Override=any	Record link-layer address.	REACHABLE
6	!INCOMPLETE	NA, Solicited=1, Override=0	-	REACHABLE
7	!INCOMPLETE	NA, Solicited=1, Override=1	Record link-layer address.	REACHABLE
8	!INCOMPLETE	NA, Solicited=0, Override=0	-	STALE
9	!INCOMPLETE	NA, Solicited=0, Override=1	Record link-layer address.	STALE
10	!INCOMPLETE	upper-layer reachability confirmation	-	REACHABLE
11	REACHABLE	timeout, more than N seconds since reachability confirm	-	STALE
12	STALE	Sending packet	Start delay timer	DELAY
13	DELAY	Delay timeout	Send unicast NS Start retransmit timer	PROBE
14	PROBE	Retransmit timeout, less than N retransmissions.	Retransmit NS	PROBE
15	PROBE	Retransmit timeout, N or more retransmissions.	Discard entry	-

表 4.1 NDP の状態遷移

さらにPROBE状態に入る。PROBE状態ではNeighbor Solicitationパケットを規定回数送信するはずである。もしPROBE状態中に実験対象の計算機に対してNeighbor Advertisementパケットが返送されなかった場合はエントリが消去されるはずなので、故意にNeighbor Advertisementパケットを送信しないでおき、エントリが消滅することを確認する。これらの動作をパケットモニタで観測する。

結果 正しい動作が確認できた。この実験から、表4.1の4、12、13、14、15の状態遷移が確認できた。

3. 実験対象の計算機がNeighbor Solicitationパケットを規定回数再送し終わる前に、Neighbor Advertisementパケットを送る。その際、Neighbor Advertisementパケットの Solicitedフラグを1とし、Overrideフラグを1または0とする。正しく動作していれば実験対象の計算機は表4.1の5の状態遷移に従ってREACHABLEになるはずである。ここでICMPv6 Echoを実験対象の計算機に送信する。REACHABLE状態の場合、ICMPv6 EchoReolyが返送される以外実験対象の計算機にはなにも変化がおこらない。これらの動作をパケットモニタで観測する。

   結果 正しい動作が確認できた。この実験から、表4.1の5の状態遷移が確認できた。

4. まず、2.の手順を途中まで実行して実験対象の計算機の状態をSTALEにし、続けてSolicitedフラグが1でOverrideフラグが0であるようなNeighbor Advertisementパケットを送信する。表4.1の6に従って状態がREACHABLEになるはずなので、3.後半の手順でREACHABLEであることを確認する。

   次に、2.の手順を途中まで実行して状態をDELAYにし、同様のNeighbor Advertisementパケットを送信して同じ結果になることを確認する。

   同じく2.の手順を途中まで実行して状態をPROBEにし、同様のNeighbor Advertisementパケットを送信して同じ結果になることを確認する。

最後に 3. の手順を途中まで実行して状態を REACHABLE にし、同様の Neighbor Advertisement パケットを送信して同じ結果になることを確認する。

結果 正しい動作が確認できた。この実験から、表 4.1の 6 の状態遷移が確認できた。

5. ほぼ 4. と同じである。送信する Neighbor Advertisement パケットのフラグが Solicited=1、Override=1 となっている点が異なっている。4. と同じ結果になるはずなので確認する。

   結果 正しい動作が確認できた。この実験から、表 4.1の 7 の状態遷移が確認できた。

6. まず、2. の手順を途中まで実行して実験対象の計算機の状態を STALE にし、続けて Solicited フラグが 0 で Override フラグが 0 であるような Neighbor Advertisement パケットを送信する。表 4.1の 8 に従って状態が STALE になるはずなので、2. 後半の手順で STALE であることを確認する。

   次に、2. の手順を途中まで実行して状態を DELAY にし、同様の Neighbor Advertisement パケットを送信して同じ結果になることを確認する。

   同じく 2. の手順を途中まで実行して状態を PROBE にし、同様の Neighbor Advertisement パケットを送信して同じ結果になることを確認する。

   最後に 3. の手順を途中まで実行して状態を REACHABLE にし、同様の Neighbor Advertisement パケットを送信して同じ結果になることを確認する。

   結果 正しい動作が確認できた。この実験から、表 4.1の 8 の状態遷移が確認できた。

7. ほぼ 6. と同じである。送信する Neighbor Advertisement パケットのフラグが Solicited=0、Override=1 となっている点が異なっている。6. と同じ結果になるはずなので確認する。

結果 正しい動作が確認できた。この実験から、表 4.1の 9 の状態遷移が確認できた。

8. 同一 LAN 上の他の計算機に対して ping を実行してみる。

   結果 以下の参加組織から正しい応答が得られた。

   - Bay Networks, Inc.
   - Digital Equipment Corporation
   - INRIA Rocquencourt
   - Mentat, Inc.
   - Process Software Co.
   - Sun Microsystems, Inc.

9. 同一 LAN 上の他の計算機に telnet を実行してみる。

   結果 以下の参加組織に正しく遠隔ログインできた。

   - Digital Equipment Corporation
   - INRIA Rocquencourt
   - Process Software Co.
   - Sun Microsystems, Inc.

   ping の実験のときと結果が異なるのは次の理由である。

   - Bay Networks, Inc が telnet を実装していない。
   - Mentat, Inc の TCP 処理部に不具合がある。

4.3.4 考察

本実験は LAN 上での包括的な IPv6 の動作試験であった。本実装はルータ関係の処理がまだ作られていないため、ルータを越える実験には参加できなかったものの、同一 LAN 上の他の実装と問題なく相互通信できることが確認された。この結果は WIDE 相互通信実験 (第 1 回、第 2 回) での結果と一致する。本実験により、LAN 上における IPv6 の相互通信性が検証されたといえる。

第 5 章

評価

本章では、1.3節で述べた 3 つの目的に対する評価を、それぞれ 5.1節、5.2節、5.3節でおこなう。

5.1. 相互通信性の検証

本論文で評価した IPv6 のソースコードは BSD/OS 2.0 上で実装された。その際に参照した資料は、参考文献に示した RFC(Request for Comments) および、Internet Drafts のみである。また、参考にしたソースコードは BSD/OS 2.0 のネットワーク関係のコードのみである。つまり、公式に公開された情報のみを利用したことになる。

また、WIDE 相互通信実験での実験相手となった次の 3 つの実装も我々と同様に公式に公開された情報のみを利用して実装されている。

- 慶応義塾大学の徳田/村井研究室
- 株式会社日立製作所
- 株式会社ソニー

さらに、これらの実装の間では、あえてお互いのソースコードを参照しあうことなく作業が進められた。

インターネットでは、公開された仕様に従って独立に進められた 3 つ以上の実装が相互通信できたとき、そのプロトコルの動作が検証されたとみなされる。上で述べた条件を考えると、我々の相互通信実験によって IPv6 の動作が検証されたといえるだろう。ただし、第 4章の接続実験は LAN 上でしかおこなわれていないので、今後より広域なネットワーク上で接続実験していくべきである。

5.2. 仕様上の不備の発見

実装により次の 3 つの問題点を発見することができた。

1. Path MTU Discovery の欠点。
2. アドレス解決のネットワーク層への抽象化。
3. アドレス解決が失敗する可能性。

特に 2. 番目のアドレス解決のネットワーク層への抽象化の問題は、実装して NDP パケットを送信するまで気がつかなかった問題である。また、3. 番目の問題は相互通信実験をおこなっているときに発見された問題である。

Path MTU Discovery の弱点は MTU のキャッシュ表を持つという解決案を提示した。アドレス解決のネットワーク層への抽象化の問題は経路情報を検索せずに、直接 IPv6 パケットを指定されたインターフェースへ出力する機構を提供することによって解決できることを述べた。最後のアドレス解決に失敗する可能性は、Neighbor Solicitation パケットのオプションとして定義されている Source link-layer address オプションを選択可能とするのではなく、必ず付けるように仕様を変更することで対処できることを示し、実装によって確認した。また、この問題と解決案を NDP の仕様設計者に連絡したところ、最新の仕様書に反映された。

5.3. IPv6 研究環境の整備

本研究によって得られた IPv6 のソースコード [22] は無償で配付される予定である。また、我々も今後の IPv6 の研究において今回作成したソースコードを利

用していくつもりである。IPv6 は研究機関のみならず、企業でも実装されつつあるが、企業の多くはソースコードを公開しないか、あるいは高いライセンス料を支払わなければ入手できない。IPv6 の関連研究をおこなう研究者にとって、無償で入手可能なソースコードの存在意義は大きい。本ソースコードは今後の IPv6 の研究の基盤環境として十分に機能すると思われる。

以上のように、1.3節で提示した 3 つの目的はいずれも達成することができた。

第 6 章

おわりに

本章は本論文のまとめと今後の課題を述べる。

6.1. まとめ

近年の急速なインターネットの発展により、IPv4 アドレス空間の枯渇が問題になりつつある。IPv6 は 128 ビットのアドレス空間を持っており、32 ビットのアドレス空間しかなかった IPv4 のアドレス枯渇問題を解決した。現在 IPv6 は RFC として仕様が提示されており、多くの研究機関、および企業が実装と実験を繰り返している。

本研究の目的は以下の 3 つであった。

1. 実装による IPv6 の相互通信性の検証。
2. 相互通信実験による、仕様上の不備の発見。
3. IPv6 研究環境の整備。

イーサネットを使った LAN 上での相互通信は問題なくおこなうことができた。接続実験に用いた IPv6 の実装は完全に独立して開発されたものである。インターネットでは 3 つの独立した実装間で相互通信できた場合、プロトコルの動作が検証されたとみなされる。よって少なくとも LAN 上での通信に限っては IPv6 の

動作が証明されたといえる。今後、WAN 上で相互通信実験し、IPv6 がインターネットでの運用に耐えるかを実証していかなくてはならない。

また実装することによって、第 3章で述べたような問題点を発見することができた。4.4BSD Lite の実装では、デフォルト経路上の計算機に対して Path MTU Discovery を適用した場合、最適な値が得られるとは考えられない。そこで MTU キャッシュ表を作ることによってデフォルト経路上のホストとの効率的な通信を提案した。アドレス解決のネットワーク層への抽象化の問題に関しては、IPv6 パケット出力時に経路情報の検索をしない出力ルーチンを用意するという解決案を提示した。また、アドレス解決が永久に終了しないという可能性に対しては、NDP の仕様を変更することで対応できることを示し、実装によって確認されている。

本研究の結果、完全に無償の IPv6 ソースコードが完成しつつある。このソースコードは、これからの IPv6 関連研究に役にたつだろう。

6.2. 今後の課題

本研究に引き続き、次のような課題を解決していかなければならない。

- 本論文で提案した、MTU キャッシュの実装。
- パケット出力時に経路情報を検索しない IPv6 パケット出力ルーチンの実装。
- WAN を用いた相互通信実験。

さらに、IPv6 の仕様として提案されているものの、まだ実装していない機能として次のようなものが挙げられる。

- ICMPv6 によるマルチキャストグループのメンバ管理。
- NDP による LAN 内のルータ検索。
- 認証機構。
- 通信内容の暗号化機能。

今後、仕様に従い上述の機能を実装することによって検証し、問題点の発見と解決案の提案をおこなう予定である。

参考文献

[1] Jon Postel. Internet Protocol. Request for Comments RFC 791, Internet Engineering Task Force, September 1981. ftp://ds.internic.net/rfc/rfc791.txt.

[2] Stephan E. Deering and Robert M. Hinden. Internet Protocol, Version 6 (IPv6) Specification. Request for Comments RFC 1833, Internet Engineering Task Force, January 1996. ftp://ds.internic.net/rfc/rfc1883.txt.

[3] Craig Partridge. Using the Flow Label Field in IPv6. Request for Comments RFC 1809, Internet Engineering Task Force, June 1995. ftp://ds.internic.net/rfc/rfc1809.txt.

[4] Randall Atkinson. Security Architecture for the Internet Protocol. Request for Comments RFC 1825, Internet Engineering Task Force, August 1995. ftp://ds.internic.net/rfc/rfc1825.txt.

[5] Randall Atkinson. IP Authentication Header. Request for Comments RFC 1826, Internet Engineering Task Force, August 1995. ftp://ds.internic.net/rfc/rfc1826.txt.

[6] Randall Atkinson. IP Encapsulating Security Payload (ESP). Request for Comments RFC 1827, Internet Engineering Task Force, August 1995. ftp://ds.internic.net/rfc/rfc1827.txt.

[7] Jim Bound, Susan Thomson, and Robert E. Gilligan. IPv6 Program Interfaces for BSD Systems. Internet Draft draft-ietf-ipngwg-bsd-api-04, Inter-

net Engineering Task Force, January 1996. ftp://ds.internic.net/internet-drafts/draft-ietf-ipngwg-bsd-api-04.txt.

[8] Francis Dupont. An implementation of IP version 6 on NetBSD. Institut National de Recherche en Informatique et en Automatique, http://ftp.inria.fr/, ftp://ftp.inria.fr/network/ipv6/.

[9] Alex Conta and Stephan E. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6). Request for Comments RFC 1835, Internet Engineering Task Force, January 1996. ftp://ds.internic.net/rfc/rfc1885.txt.

[10] Jon Postel. Internet Control Message Protocol. Request for Comments RFC 792, Internet Engineering Task Force, September 1981. ftp://ds.internic.net/rfc/rfc792.txt.

[11] Erik Nordmark, Thomas Narten, and William Allen Simpson. Neighbor Discovery for IP Version 6 (IPv6). Internet Draft draft-ietf-ipngwg-discovery-05, Internet Engineering Task Force, February 1996. ftp://ds.internic.net/internet-drafts/draft-ietf-ipngwg-discovery-04.txt.

[12] D. Plummer. Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. Request for Comments RFC 826, Internet Engineering Task Force, November 1982. ftp://ds.internic.net/rfc/rfc826.txt.

[13] Jeffrey Mogul and Stephan E. Deering. Path MTU Discovery. Request for Comments RFC 1191, Internet Engineering Task Force, November 1990. ftp://ds.internic.net/rfc/rfc1191.txt.

[14] Stephan E. Deering. Host extensions for IP multicasting. Request for Comments RFC 1112, Internet Engineering Task Force, August 1989. ftp://ds.internic.net/rfc/rfc1112.txt.

[15] John K. Renwick and Andy Nicholson. IP and ARP on HIPPI. Request for Comments RFC 1374, Internet Engineering Task Force, October 1992. ftp://ds.internic.net/rfc/rfc1374.txt.

[16] Dave Katz. Transmission of IP and ARP over FDDI Networks. Request for Comments RFC 1390, Internet Engineering Task Force, January 1993. ftp://ds.internic.net/rfc/rfc1390.txt.

[17] Stephan E. Deering. ICMP Router Discovery Message. Request for Comments RFC 1256, Internet Engineering Task Force, September 1991. ftp://ds.internic.net/rfc/rfc1256.txt.

[18] Jon Postel. Transmission Control Protocol. Request for Comments RFC 793, Internet Engineering Task Force, September 1981. ftp://ds.internic.net/rfc/rfc793.txt.

[19] Jon Postel. User Datagram Protocol. Request for Comments RFC 768, Internet Engineering Task Force, August 1980. ftp://ds.internic.net/rfc/rfc768.txt.

[20] Christopher A. Kent and Jeffery C. Mogul. Fragmentation Considered Harmful. Technical Report 87.3, Western Research Laboratory, December 1987.

[21] Robert M. Hinden and Stephan E. Deering. IP Version 6 Addressing Architecture. Request for Comments RFC 1834, Internet Engineering Task Force, January 1996. ftp://ds.internic.net/rfc/rfc1884.txt.

[22] Keiichi Shima. An implementation of IP version 6 on BSD/OS. Nara Institute of Science and Technology, http://uranus.aist-nara.ac.jp/~keiiti-s/research.html, http://uranus.aist-nara.ac.jp/~keiiti-s/wide/IPv6/.

[23] Masaki Minami. An implementation of IP version 6 on BSD/OS. Keio University, http://www.sfc.wide.ad.jp/~minami/index-e.html.

[24] Atsushi Onoe. An implementation of IP version 6 Protocol Server. Sony Co., http://onoe2.sm.sony.co.jp/~onoe/, ftp://onoe2.sm.sony.co.jp/pub/v6/.

謝辞

まず最初に、本研究を進めるにあたり、指導と助言をいただいた奈良先端科学技術大学院大学の山本和彦先生と指導教官として研究の場を提供してくださった奈良先端科学技術大学院大学の荒木啓二郎教授に感謝します。

慶應大学の南 政樹さん、株式会社ソニーの尾上 淳さん、株式会社日立製作所の新 善文さん、渡部 謙さん、浜本新一さん、土屋一暁さんには相互接続実験で有用な議論をしていただきました。WIDE プロジェクト IPv6 分科会のメンバの方々にはメーリングリスト上で有用な議論をしていただきました。

また、2 度にわたる相互接続実験のために場所の手配をし、機材を貸していただいた慶應大学の徳田/村井研究室の方々にお礼を申し上げます。

インターネットは地球を包みこむほど巨大になっています。それを支える世界中の有志の方々がいなければ私の研究は成り立たないでしょう。最後に、インターネットの発展のために努力し続ける世界中の有志の方に感謝します。

付録

A. 関連研究

IPv6 に関する研究は Sun Microsystems Inc. の WorldWide Web サーバから参照することができる。URL は http://playground.sun.com/pub/ipng/html/ipng-main.html である。この Web サーバには、IPv6 に関連する RFC へのリンク、IPv6 の概要および IETF の IPng 分科会へのリンクが設けてある。

IPv6 の実装は http://playground.sun.com/pub/ipng/html/ipng-implementations.html にまとめてある。なお、この URL は前述の URL から辿ることができる。現在以下に示す IPv6 の実装が進められている模様である。

- Host implementation
  - US Naval Research Laboratory(NRL) on 4.4BSD Lite
  - Widely Integrated Distributed Environment(WIDE) on BSD/OS
  - Digital Equipment Corporation(DEC) on Digital UNIX
  - FTP Software on DOS/WINDOWS
  - SICS(Swedish Institute of Computer Science) on HP-UX
  - Linux
  - INRIA(Institut National de Recherche en Informatique et en Automatique) on NetBSD
  - BULL-SA on AIX-4.1
  - Siemens Nixdorf Informationssysteme AG on BS2000

  - Sun Microsystems Inc. on Solaris 2
  - Mentat on STREAMS

- Router implementation
  - Bay Networks
  - cisco Systems
  - Penril Datability Networks
  - Telebit Communications A/S
  - Ipsilon Networks, Inc.

国内の IPv6 に関する活動は WIDE プロジェクトの IPv6 ホームページから参照できる。URL は http://www.wide.ad.jp/wg/ipv6/である。WIDE プロジェクトの協力を得て、次の 4 つの実装が進められている。

- 奈良先端科学技術大学院大学
- 慶応義塾大学 徳田/村井研究室
- 株式会社日立製作所
- 株式会社ソニー

これらは上述の WIDE プロジェクト IPv6 ホームページから辿ることができる。

B. 略語一覧

API Application Program Interface。アプリケーションプログラムとカーネルのインターフェース。

ARP Address Resolution Protocol。データリンク層のアドレスとネットワーク層のアドレスを対応づけるためのプロトコル。IPv4 で用いられている。

ICMP Internet Control Message Protocol。IPv4 パケットの転送時に発生したエラーを通知したり、ローカルネットワーク上の経路の変更を通知したりするプロトコル。

ICMPv6 ICMP for IPv6。ICMP を IPv6 で動作するように変更したプロトコル。さらに、途中経路の最大 MTU 発見機構や、マルチキャストグループのメンバ管理などの機能が追加されている。

IGMP Internet Group Management Protocol。マルチキャストグループのメンバ管理をするプロトコル。IPv4 で用いられている。

IP Internet Protocol。異なるデータリンク層で相互接続された計算機ネットワークを相互接続するために用いられるネットワーク層の通信プロトコル。

IPv4 IP version 4。通常は IP と記述される。IPv6 と明示的に区別する場合に、そのバージョン番号を付けてこう呼ぶ。

IPv6 IP version 6。IPv4 の後継プロトコル。広大なアドレス空間、フロー伝播など、IPv4 にはなかった特徴を持つ。

LAN Local Area Network。イーサネットや FDDI などを用いて構成した、小規模ネットワーク。

MTU Maximum Transmition Unit。データリンク層が一度に転送できる最大のデータ長。データリンクの種類毎に異なる。

NDP Neighbor Discovery Protocol。同一データリンク上に存在する計算機のデータリンク層のアドレスを調べたり、同一データリンク上に存在するルータを検索するプロトコル。IPv6 で用いられている。

RFC Request for Comments。インターネット関連の情報を記載した技術報告書。

TCP Transmission Control Protocol。通信の信頼性を保証した一対一のトランスポート層通信プロトコル。

UDP User Datagram Protocol。データの到達を保証しないトランスポート層通信プロトコル。

WAN Wide Area Network。広域ネットワーク。専用線を用いて組織間を相互接続したネットワークのことを指す場合が多い。