平成９年度

佐賀大学理工学部情報科学科卒業論文

次世代インターネット・プロトコル IPv6の実装と相互通信性の検証

論文提出者　大谷 誠 (94s410)

指導教官　近藤 弘樹 教授
田中 久治 技官
渡辺 健次 講師
(和歌山大学)

学科長　新井 康平 教授

論文提出日　平成10年2月6日

目 次

図 目 次

表 目 次

第 1 章

はじめに

本章では、本研究のテーマである、次世代インターネットプロトコルを研究するにいたった背景を説明するとともに、研究の目的について述べる。

1.1 研究の背景

現代のインターネットを支える基本規約は Internet Protocol version 4 (以下、IPv4 と略記する) と呼ばれ、1981 年に策定された。このプロトコルは計算機同士が相互通信するためのネットワーク層のプロトコルである。近年における高速回線の普及、計算機の性能向上、そして通信技術の進歩により、インターネットは急激に発展してきた。

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 $2^{32} \simeq 4 \times 10^9$ までの符号なしの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 IPv4 アドレスの割り当ては、ネットワークトポロジに対してランダムであり、またアドレス・クラスという概念による大雑把な割り当ても影響し、経路制御表の爆発的な増加、アドレス空間の枯渇などの問題が起きてきた。

ほかに近年の急激な計算機の増加も問題となった。図 1.1 に 1981 年〜 1997 年までの間にインターネットに接続された計算機数の推移を示す。図 1.1 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、2010 年 ± 5 年で 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行われてきたため、割り当てることのできる IPv4 アドレスがなくなってしまうのは図 1.1 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。これにともない、現在、次世代インターネットプロトコルとして Internet Protocol version 6(以下 IP version 6 と表記する) と呼ばれるものが提案され、各研究機関で実験が開始されている。

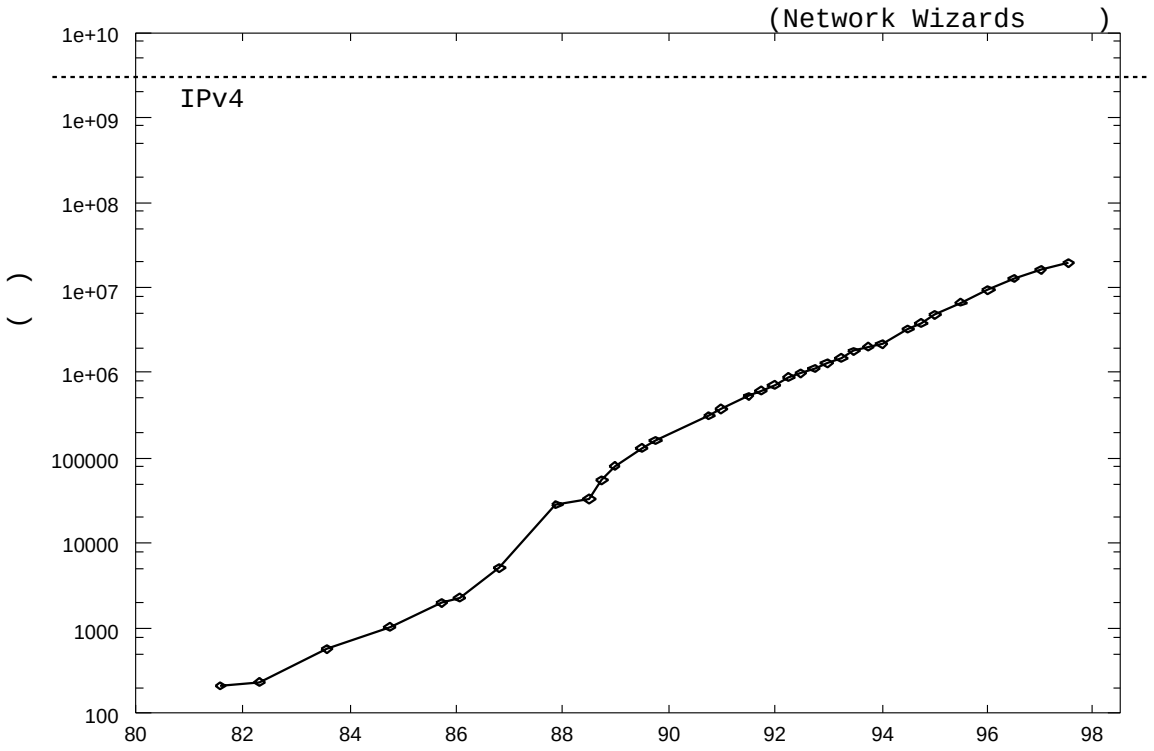

図 1.1: インターネットに接続された計算機数の推移

1.2 本研究の目的

現在、IP version 6 は仕様が正式に定まっておらず、各研究期間でテスト運用を行いながら、プロトコルの仕様を洗練する段階にある。よって IP version 6 を実装し、動作を検証すること自体に意義がある。IP version 6 は今後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。よって早い段階で IP version 6 を研究することにより、将来容易に移行できるといった利点もある。

よって本研究は、

- 実装による IP version 6 の相互通信性の検証
- IP version 6 の研究環境の構築

を目的とし、以下のように構成されている。

まず第 2 章で IP version 6 が登場するに至った背景を、次に第 3 章では IP version 6 の概要を述べる。第 4 章では IP version 6 の実装について、第 5 章では実際に行った相互通信の実験について述べる。第 6 章では、本研究で実装の評価と考察を、第 7 章でまとめと今後の課題について述べる。

第 2 章

IP version 6 の背景

2.1 現在のネットワークの問題点

1970 年代中ごろから開発が行なわれてきたインターネットは、1990 年代になってほぼ全世界をカバーする大規模なコンピュータ・ネットワークに成長した。この急激な成長のなかで、TCP/IP(Transmission Control Protocol/Internet Protocol) はさまざまな改善を施されながらインターネットで使い続けられてきた。しかし、1980 年代に TCP/IP がインターネットの基本的なプロトコルとして設計された当時には予測できなかった状況の変化が起こり、設計を変更せざるをえなくなった。このような状況の変化には以下のようなものが挙げられる。[2]

2.1.1 強力なネットワーク・ノード

TCP/IP が設計された当時、ゲートウェイで利用できるメモリ量は限られており、たかだか数百 KB 程度であった。そこで、プロトコルの処理効率を上げ、メモリの使用量をできるかぎり抑えるような実装が行われてきた。

ところが、現在ではゲートウェイで使われるプロセッサの性能は飛躍的に向上し、利用可能なメモリも数十 MB レベルに簡単に増設できる。それにともない、ゲートウェイで提供できるサービスも増やせるようになった。これはより多くの機能を IP に追加できることを意味する。このため、IP の機能拡張を積極的に主張する技術者も多い。IP の機能拡張は年を追うごとに増え続け、タイムスタンプ、経路登録、始点経路選択などと呼ばれる拡張機能が追加されていった。

2.1.2 回線の高速・広域化

当初、コンピュータ間の通信に使用できる回線は低速で、56Kbps、または 64Kbps の専用回線が主流であった。事実、TCP/IP によるコンピュータ間相互接続がおこなわれた 1980 年代始めの ARPANET(Advanced Research Projects Agency NETwork) で

は、56Kbps のデジタル専用回線が使われていた。1986 年からは、米国のスーパーコンピュータ・センター間を相互接続する NSFNET(National Science Foundation NETwork) がインターネットの発展の主舞台となり、45Mbps の T3 回線による基幹網の構築と、1.5Mbps の T1 回線による各組織との接続が進められてきた。さらに 1990 年代に入ると ATM/OC-3(155Mbps) 回線による接続が実現され、最近は、ATM/OC-12(622Mbps) 回線によるインターネット接続も行なわれるようになってきた。つまり、この 15 年あまりで、インターネットが利用している回線帯域は約 10000 倍になったわけである。

回線の高速・広帯域化は、プロトコル処理に大きな影響を与える。まず、高速広帯域回線を十分に活用するには、ヘッダの解析や経路制御といった各パケットに関する処理を十分高速におこなわなければならない。たとえば、64Kbps 程度の低速回線であれば、1 パケットあたりの処理時間は、そのパケットの送出にかかる時間にくらべてかなり短い。しかし、利用する回線が高速・広帯域になると、1 パケットあたりの処理には送出と同程度、場合によっては、より多くの時間が必要となる。したがって、回線の平均利用率を上げ、良好なスループットを得るには、パケット処理時間を高速化しなければならない。それには、ネットワークに接続される機器の処理能力を高める単純な解決方法もある。しかし、根本的にはプロトコル処理を単純化し、ヘッダなどの処理を短時間で終えるといった対策も必要である。インターネットの基本プロトコルである IP は、開発されてから長い年月のあいだに機能拡張が繰り返されてきた。しかしこれらの拡張機能を使用すると良好なスループットが得られなかったり、ホストやルータで機能の実装が完全に実装されていないといった問題があった。これには以下のような原因が考えられる。オプション付きパケットは特別な扱いを必要とする。また可変長なため、処理にも時間がかかる。よってアプリケーションの設計者は、オプションを使わない傾向にある。ゆえにルータの設計者は、オプション付きパケットを効率的に処理する利点がない。また IP に新たな機能を追加しようとする技術者から見ると, オプション・フィールドとして 20 バイトしか用意されていないために十分な機能を追加できない。

しかし指定経路制御やセキュリティなど特別な扱いを必要とするデータは存在する。そのため、IP の機能の最適化と単純化の必要性を主張する声が高まってきたのである。

2.1.3 スケーラビリティの考慮

TCP/IP が設計された時代には、インターネットに接続されているコンピュータの数は少なく、将来的にも数万台程度の接続しか予想しえなかった。1980 年代始めの ARPAnet 主としてホスト・コンピュータが接続されており、せいぜい 1000 台程度までしか増えないと思われていた。ところが、1980 年代後半になって Sun に代表される安価なワークステーションが開発され、インターネットへの接続台数が急増した。1990 年代になると、パーソナル・コンピュータがインターネットに接続されるようになり、しかも米国だけでなく世界各地のインターネットが相互接続されるようになった。これにより、インターネットに接続されているコンピュータは、飛躍的に増加した。す

なわち、TCP/IP に設計時点で考えたものよりもはるかに高いスケーラビリティが必要となったのである。

このために、さまざまなプロトコルでその不都合が表れ始めている。とくに、IP のアドレス空間の枯渇が大きな議論をまき起こすこととなった。

2.1.4 トラフィック特性の変化

TCP/IP の設計当時、インターネットで使用されるアプリケーションは、文字ベースの情報交換をおこなうものが大半であった。事実、1980 年代のインターネットは、もっぱらファイル転送や電子メールの交換のために利用されていた。これらの古典的なサービスを提供するためのプロトコル (FTP,SMTP,Telnet など) は、トランスポート層プロトコルとして TCP を使い、クライアントとサーバのあいだでのインタラクティブなデータ転送を前提として設計されることが多い。したがって、これらの古典的なサービスでは、少量のデータが比較的長い時間にわたって転送されるといったトラフィックが主流であった。ところが、1990 年代になると、UDP を用いたマルチメディア・データの転送が開始され、さらには、どちらかといえば大さなデータをいくつも連続して取得する WWW などをサービスも始められた。その結果、大量のデータが比較的短時間で転送されるトラフィックが主流になってきた。

2.1.5 クライアントの急激な増加

1990 年代のインターネットの急激な発展は、インターネットのサービスを利用するクライアントの急激な増加と捉えることができる。逆にいえば、１台のサーバの提供するサービスが、膨大な数のクライアントから利用される可能性があるということである。事実、サーバに対するトラフィックの集中が従来より頻繁にみられるようになってきた。その結果、これまでのネットワークのボトルネックであったトラフィック特性が場合によってはサーバのボトルネックになりかねない状況になっている。

2.1.6 移動ホストの出現

TCP/IP が設計されたころ、インターネットに接続されるシステムは汎用機やミニコンピュータ、ワークステーションといった固定的にインターネットに接続されるシステムばかりであった。しかし、1980 年代の後半から PC 関連の技術が急速に進化し、ラップトップ PC などの移動ホストが接続されるようになってきた。このようなシステムの登場により、インターネット環境においてもサービス、接続の確立といった面での Plug & Play 機能が求められるようになってきた。

2.2 次世代インターネットプロトコルの必要条件

このような変化に対応すべく、TCP/IP にはさまざまな改良が施され、新たな機能を追加しようと積極的に取り組む技術者も現れてきた。

2.2.1 セキュリティ

インターネットが発展していく過程で、セキュリティも重要な拡張機能として認識された。もともと、TCP/IP ではセキュリティ機能はあまり考慮されていない。セキュリティはアプリケーション層の機能であり、ネットワーク層プロトコルである IP、あるいはトランスポート層プロトコルである TCP や UDP が提供することはないと考えられていたからである。しかし、1990 年代始めには、さまざまなネットワーク・サービスにおいて IP レベルでの暗号化技術が必要であるとの認識が一般化した。これには次のような事情が大きく影響している。

- インターネットが普及するにつれ、不正アクセスに代表されるセキュリティ侵害が多く発生し、セキュリティ強化の必要性が認識されるようになった。
- セキュリティの実現に広く使われる暗号技術が普及するとともに、システムの性能向上によって、ソフトウェア的な暗号処理機構でも十分に実用的な機能が提供できるようになった。

このようななかで、IP レベルでの暗号化のフレームワーク (“IPSEC(IP Security)” と総称される) が IETF によって標準化された。IP 層のセキュリティ機能の標準では、ESP(Encapsulating Security Payload) と呼ばれる、IP データグラムのデータ領域でのフォーマットが定義されている。IPSEC を用いた具体的なサービスに、VPN(Virtual Private Network) がある。これらは、特定のゲートウェイ間でのパケットの暗号化を実施し、インターネット上で安全な通信を行う機構を提供する。

しかし、これらの標準化された機能も、既存の IP ではあくまでもオプション的な扱いである。これは既存の IP にセキュリティ機能をすべて付加するには、IP の実装を大幅に変更しなければならないからである。このため現状では、通常のホストで IPSEC が実装されていることはめったになくゲートウェイで VPN が実装されているにすぎない。しかし、IP 層におけるセキュリティ機能の必要性が認められ、十分に考慮した新たなネットワーク層プロトコルが必要であろうとの認識が一般化している。

2.2.2 Plug & Play

さきほど述べたように、インターネットには、固定的に接続されるホスト・コンピュータやワークステーションだけでなく、ラップトップ PC などの移動性をもつシステムも数多く接続されるようになった。インターネット・プロトコルにおいても、これらの

移動するシステムに対応した技術が求められている。同時に、インターネットに接続されるシステムが膨大になるにしたがって、ホストの設定を自動化したいという希望も強くなってきた。このようなことから、ホスト設定の自動化をサポートし、システムをネットワークに接続するとすぐに利用できるような、いわゆる Plug & Play 環境が求められるようになった。TCP/IP において Plug & Play 環境を実現する過程として DHCP(Dynamic Host Configuration Protocol) が開発され広く用いられている。DHCP は Windows95/NT4.0 でも標準で装備されており、UNIX の DHCP クライアントもフリー・ソフトウェアとして提供されているため、利用できるホストは着実に増えている。しかし、DHCP を利用するには、自動的に割り当てるアドレスを管理する DHCP サーバを用意しなければならない。その意味では、DHCP は完全な Plug & Play 環境とはいえない。よって IP 自体が単独でネットワーク設定をできるような機構が必要である。

2.2.3 アドレス空間

現在の IP(IPv4) では、32 ビットのアドレス空間が用意されている。IP が設計された当時、この空間を使いきるということは全く予想していなかったであろう。ところが 1990 年代半ばには、IP アドレスの深刻な問題となってきた。一方、インターネットの拡張のスピードも緩まる状況にはなかった。IETF ミーティングでは、インターネットのアドレスの割当を管理している NIC(Network Information Center) などでの割当状況の変化からは、2005 年あたりから 2015 年あたりで使いきってしまうとの予測が示された。

このため、現在のアドレス空間の消費を出来る限り抑えるために、NIC は、アドレスの割当に対して様々な制限を設け、可能な限りアドレスの有効利用を図る努力を始めた。

アドレスの有効利用を促進するための技術も開発された。IP アドレスのクラス構造は、アドレスの有効利用の阻む要因の 1 つであった。そこでサブネットという考え方を導入し、さらに最終的には従来のクラスの概念を取り払い、アドレスをつねにマスクとペアで評価する CIDR(Classless Inter Domain Routing) の利用を促進することでアドレスの有効利用ができるような環境を整えた。企業などの組織内ネットワークでは、ファイアウォールなどを介してインターネットを接続することは多くなったので、プライベート・アドレスと NAT(Network Address Translator) 技術の併用により、外部からの直接アクセスされないネットワークにおける IP アドレスの利用を抑える努力も広く行われるようになってきた。しかし、このような IP アドレスの利用を抑える対策による効果にも限度がある。結局のところ、いくらアドレスを節約してもインターネットは拡大の一途をたどるからである。このため、より広大なアドレス空間が利用できるように、新たなネットワーク層プロトコルを開発しようという試みが始まった。

2.3 IPng(IP next generation)

IPv4が抱える問題、とくに32ビットのアドレス空間の枯渇をきっかけとして、インターネットにおける新たなネットワーク層プロトコルを開発しようという動きが1993年ころから活発化した。これがIPng(IP next generation)である。

IPngの開発は、従来のインターネットが抱える問題点を明らかにし、それらの解決に必要な技術的要件を明確に定義することから始められた。これは、IPv4が抱えていたさまざまな問題を一気に解決使用と考えたからである。この技術的要件を満たすネットワーク・プロトコルの開発は、世界各地のいくつかの研究機関で進められた。これらのネットワークプロトコルは、IETFにおける活動のなかで1つのプロトコルにまとめられ、これが最終的にIP version 6として標準化された。

第 3 章

IP version 6の概要

IP version 6 は Xerox Palo Alto Reserah 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 と表記し、IPv4 と IPv6 の両方を示す場合には、バージョン番号をつけずに単に IP と表記する。

3.1 アドレス空間の拡大

まず IPv4 では 32 ビットであったアドレス空間が、IPv6 では 128 ビットに拡大された。よってアドレス空間は $2^{96} \simeq 8 \times 10^9$ 倍に拡張されたことになる。これにより巨大なアドレス空間を手にいれることができるため、理論的には地球の表面 1 平方メートルあたりに 6.65×10^{23} のアドレスを付加することができる。よってこのアドレス空間を近い将来枯渇することはないと考えられる。

3.1.1 アドレスの表記方法

IPv4 では 32 ビットのアドレスを 8 ビットずつ 4 つの組にピリオド (.) で区切っていたが、IPv6 では 128 ビットのアドレスを 16 ビットずつ 8 つの組にコロン (:) で区切る。区切られた各組は、16 進数で表記される。

例えば、

FEDC:BA98:7654:3210:FEDC:BA94:3859:1520
1080:0:0:0:8:700:200C:417A

といった表記方法となる。

また連続した 0 は 2 つのコロン”::” に置き換えることができるという規則があり、

FF01:0:0:0:0:0:0:43

というアドレスは、

FF01::43

と表記することもできるが、この規則は、1つのアドレスの中で1度しか使用することができない。

3.2 アドレスの初期割り当て

IPv6アドレスの先頭にはFP(Format Prefix)と呼ばれる可変長の領域があり、これによってアドレスの種類が識別できる。

割り当て	FP(2進数表記)
予約	0000 0000
NSAP 割り当て予約	0000 001
IPX 割り当て予約	0000 010
グローバル・ユニキャスト・アドレス	001
リンクローカル・ユニキャスト・アドレス	1111 1110 10
サイトローカル・ユニキャスト・アドレス	1111 1110 11
マルチキャスト・アドレス	1111 1111
未定義	[その他]

表 3.1: IPv6アドレスの初期割り当て

3.2.1 ユニキャスト・アドレス

IPv6のアドレス構造で、もっとも特徴的なのは、現在のインターネットの構造に合致したユニキャスト・アドレスの構造を定義したことにある。

3.2.1.1 グローバル・ユニキャスト・アドレス

1995年に標準化され、公開されたRFC1884では、プロバイダ型ユニキャスト・アドレスとして、プロバイダを単位としたアドレス構造を定義した。
トップレベルでのアドレス登録機関(registry)、アドレス登録機関からからアドレスを得るプロバイダ、そのプロバイダからサービスを購入するサイト(subscriber)といった仕組みを前提としたアドレス構造となっている。しかし、現在のインターネットは、このように図式化できるほど単純ではない。サービスの提出形態をアドレス構造に組み込むことに対する批判もあった。

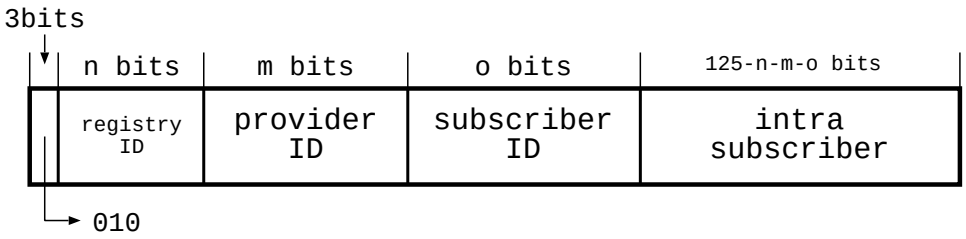

図 3.1: プロバイダ型ユニキャスト・アドレス

そこで新たな Internet Draft ではグローバル・ユニキャスト・アドレスとして新たな構造を定義した。このなかでは、従来はアドレス登録機関 (registry ID) や provider ID としていたフィールドを、それぞれ、

TLA ID: Top-Level Aggregation Identifier
NLA ID: Next-Level Aggregation Identifier
SLA ID: Site-Level Aggregation Identifier

に変更した。さらに各フィールド長についても明確に定義されている。もっとも大きな変更点は、FRC1884 ではアドレス構造に registry ID という経路制御にほとんど無関係情報が含まれていたのに対し、新たな定義では経路制御に関係するものだけから構成されている点である。詳細は [12][13]

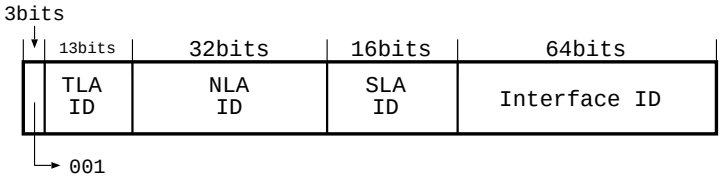

図 3.2: グローバル・ユニキャスト・アドレス

3.2.2 ローカルアドレス

IPvt では、ローカルだけで使用するアドレスが 2 種類存在する。これがリンクローカル・ユニキャスト・アドレスとサイトローカル・ユニキャスト・アドレスである。

リンクローカル・ユニキャスト・アドレスは、単一のデータリンクの中だけで使用されるユニキャスト・アドレスでそのリンク内だけでのデータ転送のみに使われる。具体的には、IPv6 のネットワークにマシンを接続した時点で、Plug&play で IPv6 アドレスを与える場合に使ったり、あるいは隣接ホスト探索 (neighbordiscovery) に利用したりする。

もう 1 つのサイトローカル・ユニキャスト・アドレスは、IPv4 のローカルアドレスのように、サイト内で有効なアドレスではあるが、サイト内で完全に閉じたか達で利用され、インターネット全体としては、経路を扱わないようなアドレスである。

3.3 ヘッダの簡略化

IPv4 の 20 年以上の運用で有効に活用されることがなかったヘッダフィールドが除かれ簡素化された。図 3.3 に IPv4 のヘッダ形式を、図 3.4 に IPv6 のヘッダ形式を示す (詳細は、付録 A)。図 3.3 の部分で、斜線で印をつけたフィールドが削除された。なお表 3.2 で示すように、同じ目的ではあるものの異なる名前に変更されたフィールドもある。

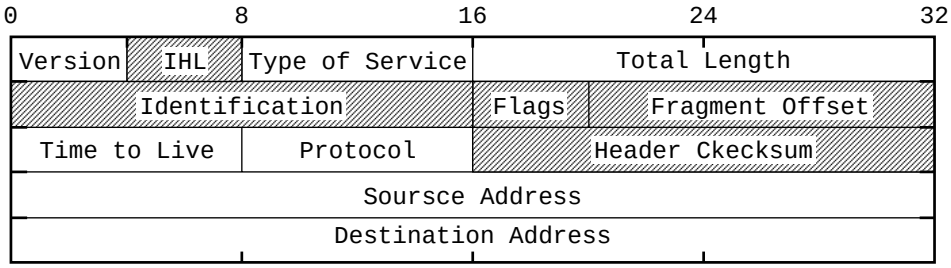

図 3.3: IPv4 のヘッダ形式

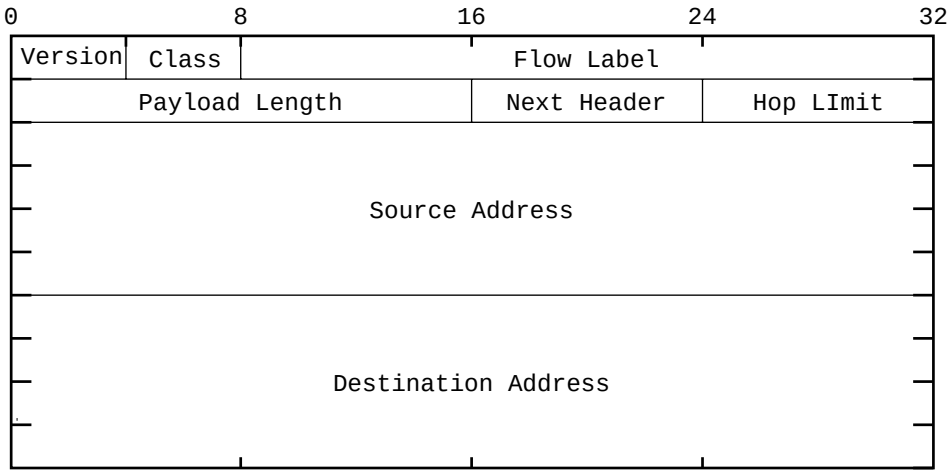

図 3.4: IPv6 のヘッダ形式

IPv4 ヘッダ	IPv6 ヘッダ
Total Length	Payload Length
Time to Live	Hop Limit
Protocol	NextHeader

表 3.2: IPv4 と IPv6 のヘッダの対応

簡素化の要点としては、ヘッダ長の固定長化、チェック・サムの廃止などがある。またIPv4のオプションフィールドに変わるものとして拡張ヘッダというものが採用され、これによって柔軟な機能の追加を実現している。

3.4 リアルタイム通信への対応

音声会議システムやビデオ会議システムなどリアルタイム性を要求するアプリケーションへの対応として、フロー・ラベルという識別子が設けられている。このフロー・ラベルによって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。

3.5 セキュリティー機能の導入

セキュリティに関して、認証ヘッダと暗号ペイロードという2つの識別子を持っている。認証ヘッダによって、受信者は、パケットの始点アドレスの確認と内容が途中で改ざんされていないことを確認することができる。また暗号ペイロードによってIPデータグラムの中に暗号化した情報を入れて送信することによって経路途中での盗聴を防ぎ、正当な受信者だけが内容を読むことができる。この機能は、送受信者の間でセキュリティ機能を利用する必要がないと判断した場合には、使用しなくてよい。

3.6 Plug & Play

IPv6には、マシンをネットワークに接続しただけで、自動的にネットワーク・インターフェイスの初期化などを行ない、ただちにネットワークが利用できる機能を提供している。これが”Plug & Play” と呼ばれるものである。

3.7 ルータでの分割処理の廃止

IPv4では、ルータでパケット分割処理を行うため、データの送信を行うときに送信者が、中継点のMTU(Maximum Transmission Unit)を考慮する必要はなかった。しかしこの方法では、ネットワークを効率的に扱えない場合ある。例えば、小さな断片しか伝送できないネットワークで大きなパケットを送信する場合、パケットの送信が成功するためには分割された全断片が届かなければならないからだ。そこでIPv6では、ルータでのパケット処理を廃止した。IPv6では”path MTU discovery(経路MTU探索)” という機能を用いて経路のMTUの検索を行い、そのMTUの値より小さなサイズに送信元が分割してパケットを送信する。

3.8 IP version 6 への移行

IPv6 に関する提案はその大部分が RFC(Request For Comment) として標準化され、IPv4 からの移行が徐々に始まりつつある。しかし現在のところ、インターネットに接続されている計算機の数は 1997 年 7 月現在約 1950 万台にも及ぶ。これらの計算機がいっせいに IPv6 に対応するということは、まず不可能に近い。実際は、かなり長い期間をかけて移行が進むことが予想される。これは長い期間 IPv4 と IPv6 の計算機が混在することを意味している。よって移行を円滑に行うためには、IPv4 と IPv6 の計算機が混在する環境下で相互通信ができることが望まれる。しかし第 2 章でも述べたように IPv4 と IPv6 は、ヘッダの形式が異なっており、互換性がない。実際のところ、下位層で IPv4 と IPv6 は区別されている。例えば Ethernet 上で IPv4 では、フレームタイプとして 0x8000 を用いるが、IPv6 では、0x86dd が用いられている。

よってこの 2 つをつなぐための相互接続技術が必要となってくる。

そこで IETF の次世代移行分科会は、

- デュアル・スタック
- トンネリング

と呼ばれる技術を用いた移行戦略を採択した。
この 2 つの移行技術について以下に述べる。

3.8.1 デュアル・スタック

一般的に表現すると、デュアル・スタックとは、ネットワーク層プロトコルを複数扱うことを意味している。例えば現在使用されている数多くのルータは IPv4 だけでなく、AppleTalk や Netware など複数をプロトコルを扱える。IPv6 でいうデュアル・スタックとは、IPv6 の機能を持つと同時に、IPv4 の機能を併せ持つ計算機のことである。このデュアル・スタック計算機は図 3.5 で示すように IPv6 計算機とは IPv6 で通信を行い、IPv4 計算機とは IPv4 で通信を行うことができる。

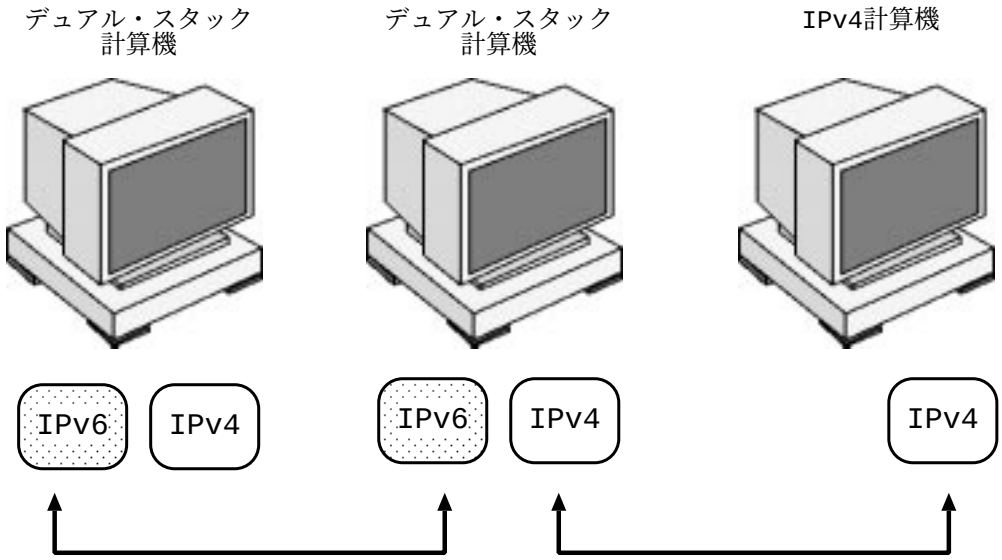

図 3.5: デュアル・スタック概念図

このデュアル・スタックによってローカルネットワーク内に混在する IPv4 計算機と IPv6 計算機が相互通信を行うことができる。しかしネットワーク的に切り離された IPv6 計算機同士は、経路上に IPv4 ルータしか存在しないため、相互通信を行うことができない。このような場合、次に示すトンネリングを使用して相互通信を行う。

3.8.2 トンネリング

トンネリングとは、ネットワーク層プロトコルを通すために、複数の異なるネットワーク層プロトコルを用いる技術のことを表す。トンネリングを用れば図3.6で示すようにIPv6パケットの前にIPv4のヘッダを付加することによりIPv4のネットワークを利用してIPv6パケットを転送することが可能となる。

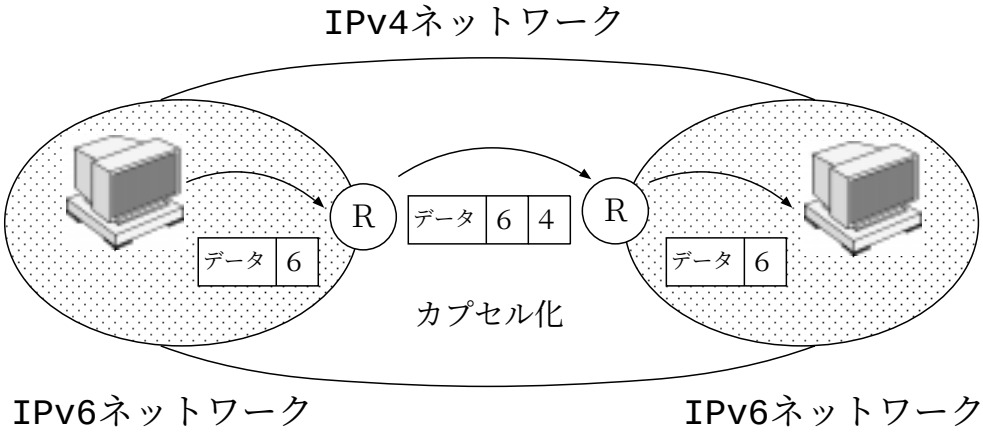

図 3.6: トンネリング概念図

第 4 章

IP version 6 の実装

本章では、本研究で行なった実装の設計概念と、以下にあげる各層での実装について説明する。

- ネットワーク層
- インターネット層
- トランスポート層
- ソケット層

実装は、4.4BSD UNIX(Berkeley Software Distribution UNIX version 4.4) の流れを組む FreeBSD2.2.2 上で行った。FreeBSD2.2.2 を採用したのは、

- “PC 互換機” コンピュータで使用可能
- 無料でソースコードが付属
- 最新に近いネットワークコードが使用可能
- FreeBSD 上で実装可能な IPv6 のソースコード の公開

などの理由からである。

IPv6 のソースコードとしては、INRIA(Institut National de Recherche en Informatique et en Automatique) から公開されている FreeBSD 用のソースコードを用い実装を行った。付録 B 参照。

4.1 設計理念

IPv6 のパケットの処理方法は IPv4 と似ている。にもかかわらずヘッダ形式が大きく変更されたことにより IPv6 と IPv4 では互換性が全くない。このようなプロトコルを実装する場合、大きく分けて 2 つの実装方法が考えられる。

- IPv4 のパケット処理を拡張し IPv6 パケットの処理を追加する
- IPv6 のパケット処理を別に追加する

これを概念的に示したものが図 4.1 である。

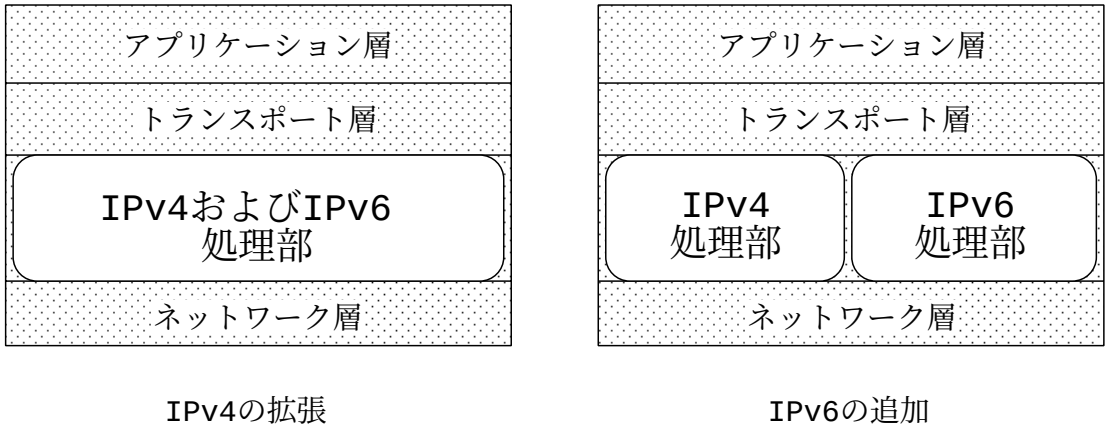

図 4.1: パケット処理部の概念図

INRIA のから公開されている IPv6 のソースコードをもとにしている本研究での実装は、IPv4 のパケット処理を拡張し IPv6 パケットの処理を追加するものを採用している。この実装の利点としては、IPv4 と IPv6 とが協調動作するときに、処理が簡単になることがあげられる。

4.2 ネットワーク層

ネットワーク層としての役割は、ネットワークデバイス (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となるインターネット層に渡すこと、また逆にインターネット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。本研究では、ネットワークデバイスとしてイーサネット、ソフトウェアのみで構成されるネットワークデバイスであるループバックデバイス、トンネリング時に使用されるデバイスとしてトンネリングデバイス、この 3 つを実装した。

4.3 インターネット層

IPv6 で通信をおこなうためには、インターネット層で IPv6 を実装する以外に、次のプロトコルを実装する必要がある。

- ICMPv6(Internet Control Message Protocol for IPv6)

- NDP(Neighbor Discovery Protocol)

ICMPv6 は ICMP を IPv6 で動作するように変更したものである。ICMPv6 では、ICMP で使われなかった機能は取り除かれ、IPv6 用に拡張された大きなフィールドを含んでいるため、従来の ICMP とは全く互換性がなく、新しく定義しなおされている。

NDP は ARP(Address Resolution Protocol) を一般化したプロトコルである。これはデータリンク層アドレスとネットワーク層のアドレスを対応づけをおこなうものである。以下に、ICMPv6 と NDP について以下に述べる。

4.3.1 ICMPv6

ICMPv6 にはおもに以下の機能がある。

- 転送エラーの報告
- エコー要求/応答
- NDP パケットの送信

なお NDP は ICMPv6 とは別のプロトコルである。にもかかわらず ICMPv6 の項目として存在するのは、IPv6 レベルで NDP は ICMPv6 のパケット形態をとっているからである。以下で転送エラーとエコーについては以下で述べる。NDP については、4.3.2 で説明を行なう。

4.3.1.1 転送エラーの報告

ICMPv6 には以下のような問題が発生した場合、パケット送信者に転送エラーを報告する。

- 宛先ホストに到着不可能
- IPv6 ヘッダのフォーマットエラー
- パケット送信時間超過
- パケット長過大

これらの転送エラーは、IPv4 で用いられている ICMP の報告とほぼ同じである。しかし、ICMPv6 のエラーには ICMP にはない機能がひとつ追加されている。それは、経路 MTU 探索 (Path MTU Discovery) である。IPv6 では、ルータでのパケットのフラグメントを行なわないためにこれが必要となってくる。

4.3.1.2 エコー要求/応答

エコー要求と応答は、宛先ホストが到着可能であるか調べるときに用いられる ping コマンドで使用されるメッセージである。これも IPv4 で用いられる ICMP と同様の働きをする。

4.3.2 NDP(Neighbor Discovery Protocol)

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の対応付けを行う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。本来 ARP はデータリンク層としてイーサネットのみ対象としていたので、他のデータリンク層に対応されるために何度か拡張が行われている。NDP は ARP を発展させて、多様なデータリンク層に対応できるように改良された。また、サブネット上に存在するルータやホストを自動認識する機構なども取り入れられた。

NDP はまだ仕様の洗練が行われている途中である。よって将来大きく仕様が変わる可能性もある。

4.4 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となる。本研究の実装では、2 つのトランスポート層プロトコルを実装する。

- TCP(Transmision Control Protocol)
- UDP(User Datagram protocol)

TCP と UDP はどちらも IPv4 で広く用いられているトランスポート層のプロトコルである。この 2 つのプロトコルはトランスポート層に位置しているにもかかわらず、チェックサムの計算に IP ヘッダを必要とするなど、IP に大きく依存している。そのためにネットワーク層に IPv6 を準備するだけではうまく動作させることができない。よって IPv6 用に TCP と UDP を準備する必要がある。しかし、IPv6 を実装したことによる、機能の変更などはなく、IPv4 のときと同様に扱うことができる。

4.5 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。そのため、プログラマからみてわかりやすくなければならず、従来のソケットの定義と大きくかけ離れてはいけない。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呼ばれる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。なお、ここでいうソケットとはBSDソケットのことを指す。

本実装では、IPv6用に定められたAPI(Application Programming Interface)を使って実装を行なっている。図4.2、4.3はそれぞれIPv6とIPv4のソケットアドレスの構造体を示す。それぞれの変数の型は違うものの、IPv6のソケットアドレスには、IPv6特有のフロー情報が追加されていることがわかる。ソケット関連システムコールはソケットアドレスを引数にとる。よってプログラマはIPv4ソケットアドレスをIPv6ソケットアドレスに変換するだけで従来と同じようにプログラミングすることができる。

```
struct sockaddr_in6 {
    u_int8_t  sin6_len;           /* 構造体の大きさ */
    u_int8_t  sin6_family;        /* アドレスファミリ */
    u_int16_t  sin6_port;         /* トランスポート層ポート番号 */
    u_int32_t  sin6_flowinfo;     /* IPv6 フロー情報 */
    struct  in6_addr sin6_addr;   /* IPv6 アドレス */
}
```

図 4.2: IPv6 ソケットアドレスの構造体

```
struct sockaddr_in {
    u_char  sin_len;              /* 構造体の大きさ */
    u_char  sin_family;           /* アドレスファミリ */
    u_short sin_port;             /* トランスポート層ポート番号 */
    struct  in_addr sin_addr;     /* IPv4 アドレス */
};
```

図 4.3: IPv4 ソケットアドレスの構造体

第 5 章

相互通信実験

この章では、本研究で行った相互通信性の検証実験について述べる。

5.1 実験内容

実験の内容として主に、

- ネットワークの自動設定
- IPv6 マシン間での相互通信
- IPv6 マシンと IPv4 マシン間の相互通信
- トンネリングによる IPv6 マシン間の相互通信
- プロトコルの違いによるパケット処理時間の計測

を行い、相互通信には、telnet と ping コマンドを用いた。

実験には、IPv6 マシンとして FreeBSD2.2.2 上で IPv6 を実装した”PC 互換機” を 2 台、IPv4 マシンとして SUN SPARC station 5 を使用した。

以下が実験に使用した計算機のドメインネームと IPv4 アドレスである。

計算機	ドメインネーム	IPv4 アドレス
IPv6 計算機 (FreeBSD2.2.2)	amano.ai.is.saga-u.ac.jp	133.49.31.55
IPv6 計算機 (FreeBSD2.2.2)	kibe.ai.is.saga-u.ac.jp	133.49.31.27
IPv4 計算機 (SUN 4.1.3-JL)	haruke.ai.is.saga-u.ac.jp	133.49.31.3

表 5.1: 実験に使用した計算機のネットワーク設定

5.2 ネットワークの自動設定

IPv6では2.2.2で示したように仕様として、ネットワークの自動設定機能 (Plug & play) を持っており、本実装の場合、autoconf6 と呼ばれるコマンドを実行することによって自動設定される。例として図 5.1 に autoconf6 コマンドを計算機”amano” で実行したときの表示画面を示す。

```
multi-homed level: none
got interface ed0
got interface sit0
got interface sit1
got interface lo0
default IEEE interface is ed0
anycast fe80:: on ed0
setup IEEE interface ed0
config ed0 fe80::4a54:45ff:fe00:5b4f/64
interface ed0 is booting
add route flags=801
        dest=ff02::
        gateway=fe80::4a54:45ff:fe00:5b4f
        mask=ffff::
add route flags=801
        dest=ff12::
        gateway=fe80::4a54:45ff:fe00:5b4f
        mask=ffff::
Sending ND sol for fe80::4a54:45ff:fe00:5b4f on ed0
got my ND sol
ed0 setup good
interface ed0 is booted
setup SIT interface sit0
config sit0 ::133.49.31.55/96
setup loopback interface lo0
config lo0 ::1/128
add route flags=801
        dest=ff01::
        gateway=::1
        mask=ffff::
add route flags=801
        dest=ff11::
        gateway=::1
        mask=ffff::
add route flags=901
        dest=::
        gateway=fe80::4a54:45ff:fe00:5b4f
        mask=::
```

図 5.1: autoconf6 コマンドによるネットワークの自動設定の画面表示

このコマンドによって、各デバイスにIPv6アドレスが割り当てられる。またルーティングなどの設定も同時に行われる。表5.2に割り当てられたIPv6アドレスを図5.2に例として計算機"amano" のルーティングテーブルの内容を示す。

IPv6 計算機	IPv6 アドレス	
amano	fe80::4a54:45ff:fe00:5b4f/64	(イーサネット)
	::1/128	(ループバック)
	::133.49.31.55/96	(トンネリング)
kibe	fe80::2c0:4fff:fed6:2e6e/64	(イーサネット)
	::1/128	(ループバック)
	133.49.31.27/96	(トンネリング)

表 5.2: IPv6 計算機に割り当てられたアドレス

```
IPv6:
Destination        Gateway            Flags    Refs     Use   Mtu  Netif Expire
::/96              0.0.0.0            UC          0       0     -  sit0   - =>
default            link#1             UCS         0       0  1500  ed0    -
::1                ::1                UH          0       0 16384  lo0
fe80::/64          link#1             UC          0       0  1500  ed0    -
ff01::/16          ::1                US          0       0 16384  lo0
ff02::/16          fe80::4a54:45ff:fe00:5b4f US        0       0  1500   ed0
ff11::/16          ::1                US          0       0 16384  lo0
ff12::/16          fe80::4a54:45ff:fe00:5b4f US        0       0  1500   ed0
```

図 5.2: ルーティングテーブルの内容

イーサネットアドレスの先頭部分のfe80は、リンクローカル・アドレスと呼ばれるアドレスに使われるプレフィックスで、このアドレスは、ローカルネットワーク内で一意であればよく、ルータによって中継されることはない。また最後から48ビットの部分はIEEE-802(イーサネットアドレス)と呼ばれるもので、自動設定した場合(イーサネットの場合)リンクローカル・アドレスの一部として用いられる。ループバックアドレスは、IPv6計算機が自分自身にパケットを送るときに使うアドレスである。トンネリングアドレスについては5.6節で述べる。

5.3 ドメインネームの設定

IPv6 の機能を有効的に使うためには、動的に DNS にアドレスを登録する必要があるが現在のところ仕様が決まっておらず、動的に DNS アドレスを登録することはできない。よって IPv6 計算機の /etc/hosts に以下の行を追加することによって代用している。

fe80::4a54:45ff:fe00:5b4f	amano
fe80::2c0:4fff:fed6:2e6e	kibe

表 5.3: IPv6 計算機のドメインネーム設定

5.4 相互通信検証 (IPv6-IPv6 間)

まず IPv6 計算機 (amano) から IPv6 計算機 (kibe) に telnet を行った。図 5.3 に実際の画面の表示を示す。

```
%telnet kibe
Trying fe80::2c0:4fff:fed6:2e6e...
Connected to kibe.
Escape character is '^]'.

FreeBSD (kibe.ai.is.saga-u.ac.jp) (ttyp0)

login:
```

図 5.3: IPv6-IPv6 間通信による telnet の画面表示

最初の Trying fe80::2c0:4fff:fed6:2e6e... の部分で、IPv6 のアドレスを使って通信を行っていることが分かる。

図 5.4 は、実際にやり取りしているパケットのダンプの一部である。

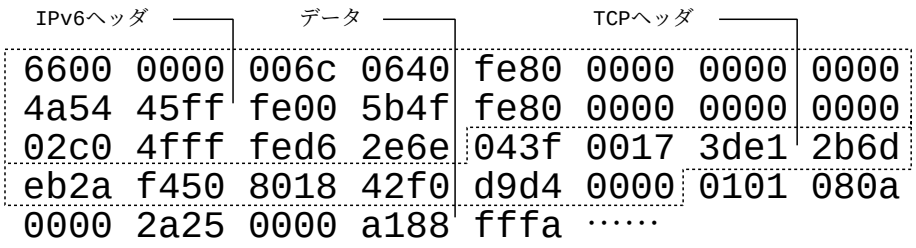

図 5.4: IPv6-IPv6 間通信のパケットダンプ

また telnet と同様に、IPv6 マシン (amano) から IPv6 マシン (kibe) に ping を行った。図 5.5 に実際の画面の表示を示す。

```
% ping6 kibe
trying to get source for kibe
source should be fe80::4a54:45ff:fe00:5b4f
PING kibe (fe80::2c0:4fff:fed6:2e6e): 56 data bytes
64 bytes from fe80::2c0:4fff:fed6:2e6e: icmp6_seq=0 ttl=255 time=1.128 ms
64 bytes from fe80::2c0:4fff:fed6:2e6e: icmp6_seq=1 ttl=255 time=0.991 ms
64 bytes from fe80::2c0:4fff:fed6:2e6e: icmp6_seq=2 ttl=255 time=0.955 ms
64 bytes from fe80::2c0:4fff:fed6:2e6e: icmp6_seq=3 ttl=255 time=0.956 ms
64 bytes from fe80::2c0:4fff:fed6:2e6e: icmp6_seq=4 ttl=255 time=0.951 ms
64 bytes from fe80::2c0:4fff:fed6:2e6e: icmp6_seq=5 ttl=255 time=0.951 ms
                                        :
                                        :
                                        :
```

図 5.5: IPv6-IPv6 間通信による ping の画面表示

5.5 相互通信検証 (IPv6-IPv4 間)

図 5.6 に、IPv6 計算機 (amano) から IPv4 計算機 (haruke) に telnet を行ったときの画面表示を以下に示す。

```
% telnet haruke
Trying ::ffff:133.49.31.3...
Connected to haruke.
Escape character is '^]'.

SunOS UNIX (haruke)

login:
```

図 5.6: IPv6-IPv4 間通信による telnet の画面表示

最初の Trying ::ffff:133.49.31.3... の部分のアドレスは、IPv4 射影アドレス (または IPv4 変換可能 IPv6 アドレス) と呼ばれ、一般に IPv4 しか扱えないホストに対して指定するものである。よって、IPv6 計算機から IPv4 計算機に IPv4 で通信できたことが確認できる。

図 5.7 は、実際にやり取りしているパケットのダンプの一部である。

telnet と同様に、IPv6 マシン (amano) から IPv4 マシン (haruke) に ping を行った。図 5.8 に実際の画面の表示を示す。

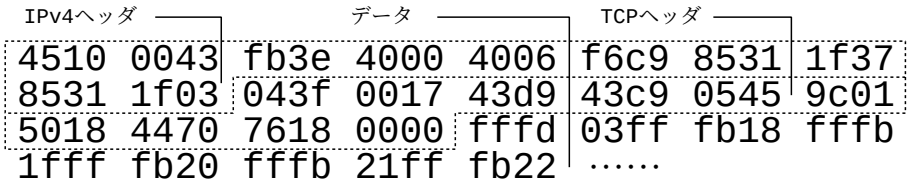

図 5.7: IPv6-IPv4 間通信のパケットのダンプ

```
% ping haruke
PING haruke (133.49.31.3): 56 data bytes
64 bytes from 133.49.31.3: icmp_seq=0 ttl=255 time=2.817 ms
64 bytes from 133.49.31.3: icmp_seq=1 ttl=255 time=0.903 ms
64 bytes from 133.49.31.3: icmp_seq=2 ttl=255 time=1.139 ms
64 bytes from 133.49.31.3: icmp_seq=3 ttl=255 time=1.015 ms
64 bytes from 133.49.31.3: icmp_seq=4 ttl=255 time=0.949 ms
64 bytes from 133.49.31.3: icmp_seq=5 ttl=255 time=0.892 ms
                                    :
                                    :
                                    :
```

図 5.8: IPv6-IPv4 間通信による ping の画面表示

5.6 相互通信 (トンネリング)

現在のところ、孤立した IPv6 マシンは、他のネットワークの IPv6 マシンとルータによるトンネリングによって通信することができる。しかし、トンネリングはホスト自身が行うことも可能である。

図 5.9 に、amano から kibe に telnet を行ったときの画面表示を示す。

```
% telnet ::133.49.31.27
Trying ::133.49.31.27...
Connected to ::133.49.31.27.
Escape character is '^]'.

FreeBSD (kibe.ai.is.saga-u.ac.jp) (ttyp0)

login:
```

図 5.9: トンネリングによる IPv6-IPv6 間通信の telnet の画面表示

図 5.6 に、実際にやり取りしているパケットのダンプの一部である。

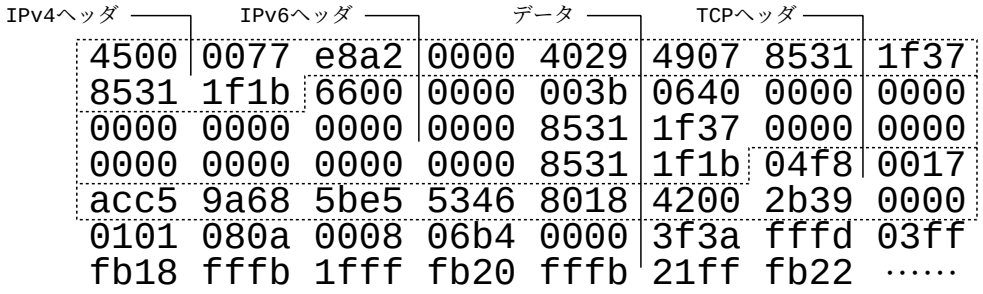

図 5.10: トンネリングによる IPv6-IPv6 間通信のパケットのダンプ

同様に、IPv6 マシン (amano) から IPv6 マシン (kibe) にトンネリングによる ping を行った。図 5.11 に実際の画面の表示を示す。

```
% ping6 ::133.49.31.27
trying to get source for ::133.49.31.27
source should be ::133.49.31.55
PING ::133.49.31.27 (::133.49.31.27): 56 data bytes
64 bytes from ::133.49.31.27: icmp6_seq=0 ttl=255 time=1.592 ms
64 bytes from ::133.49.31.27: icmp6_seq=1 ttl=255 time=1.049 ms
64 bytes from ::133.49.31.27: icmp6_seq=2 ttl=255 time=1.077 ms
64 bytes from ::133.49.31.27: icmp6_seq=3 ttl=255 time=1.138 ms
64 bytes from ::133.49.31.27: icmp6_seq=4 ttl=255 time=1.053 ms
64 bytes from ::133.49.31.27: icmp6_seq=5 ttl=255 time=1.042 ms
                                    :
                                    :
                                    :
```

図 5.11: トンネリングによる IPv6-IPv6 間通信の ping の画面表示

5.7 パケット処理時間の計測

以上の実験で各ホスト間において、IPv4、IPv6 およびトンネリングでの通信が行えることが証明された。しかし、各プロトコルはヘッダの大きさが全て異なるためプロトコルによりパケットの処理時間が異なると推測される。

そこでそれぞれのプロトコルにおいて ping を IPv6 マシンから IPv6 マシンへ同時に 300 回を 3 回繰り返して実行しプロトコルによる Round-Trip-Time(以下 rtt と訳す) の違いを調べた。結果を表 5.4 に示す。

プロトコル	rtt(1)	rtt(2)	rtt(3)
IPv4	0.8133	2.0985	0.7940
IPv6	1.0465	2.1093	0.9405
IPv6 トンネリング	1.1158	2.1975	1.0368

表 5.4: 各プロトコルによる ping の平均 Round-Trip-Time

また図 5.7 に ping の rtt(1) グラフを示す。

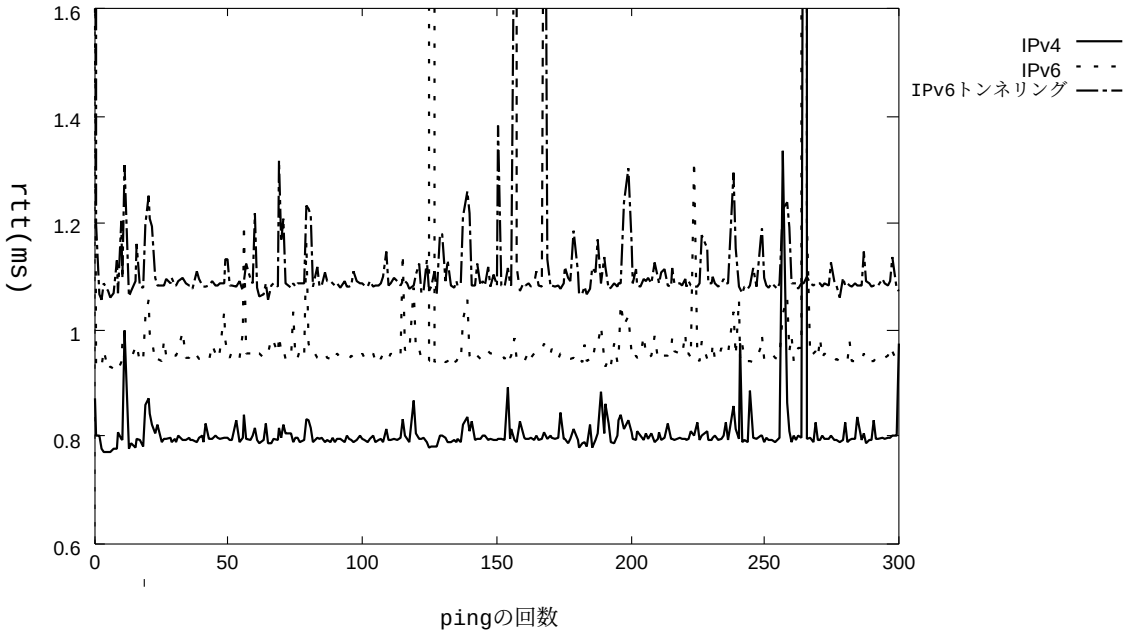

図 5.12: 各プロトコルによる ping の rtt のグラフ

以上の結果から、本研究の実装ではプロトコルのヘッダの長さに比例して、パケットの処理時間が増加していることが分かる。

第 6 章

おわりに

本章では本論文のまとめと今後の課題について述べる。

6.1 まとめ

近年の急激なインターネットの普及により、IPv4 のアドレス空間が問題になりつつある。次世代インターネットプロトコルである IPv6 は、128 ビットのアドレス空間を持ち、32 ビットのアドレス空間しかもたなかった IPv4 のアドレス空間の枯渇の問題を解決した。また IPv6 はアドレス空間の枯渇の問題を解決するだけでなく様々な新しい機能を提供している。現在 IPv6 は RFC として仕様が提示されいろいろな研究期間や企業が実装や実験を行い仕様の検証などを行っている。本研究では、まず IPv6 の調査研究を行い、

- 実装による IPv6 の相互通信性の検証
- IPv6 研究環境の構築

を目的とした。

本研究の実装では、イーサネット上での IPv6 マシン間での通信、IPv4 と IPv6 マシン間での通信、ローカルネットワーク内およびルータを経由したトンネリングについて何ら問題なく相互通信が行うことができた。このことにより IPv6 の研究環境は少なからず整ったとみても良いだろう。

6.2 今後の課題

IPv6 は未だ開発段階にあり、仕様が定まっていない部分や、仕様として提案されてはいるものの実装していない機能がある。よって今後、実装されていない部分の仕様に基づく実装や、現在の仕様の問題点の発見と解決案の提示を行っていく予定である。また現在 IPv6 の実験に参加している組織を接続した 6bone(http://www.6bone.net) が

IPv4 ネットワーク上に構築され、日本でも WIDE プロジェクトが中心となって運用が行われている。このネットワークへの接続を行い、広域ネットワークでの IPv6 の動作などの確認を確認することなどもあげられる。

謝辞

本研究を行うにあたって、御指導下さいました近藤弘樹教授、田中久治技官、和歌山大学の渡辺健次講師を始めとする本研究室の皆様に心から御礼申し上げます。

参考文献

[1] Christian Huitema, 村井 純 監修,WIDEプロジェクト IPv6 分科会 監訳、松島 英樹 訳,"IPv6 次世代インターネットプロトコル, プレンティスホール出版",(Jan. 1997)

[2] 山口 英," カーネルを読もう (14):新たな TCP/IP に向けて", UNIX Communication Notes pp38-42,UNIX MAGAZINE, アスキー出版,(Oct. 1997)

[3] 山口 英," カーネルを読もう (15):IPv6(1)", UNIX Communication Notes pp14-24,UNIX MAGAZINE, アスキー出版,(Nov. 1997)

[4] 山口 英," カーネルを読もう (16):IPv6(2)", UNIX Communication Notes pp13-20,UNIX MAGAZINE, アスキー出版,(Dec. 1997)

[5] 山口 英," カーネルを読もう (17):IPv6(3)", UNIX Communication Notes pp22-28,UNIX MAGAZINE, アスキー出版,(Jan. 1998)

[6] 井口信和," 絵ときイントラネット TCP/IP バイブル", オーム社,(May. 1997)

[7] Douglas Comer, 村井 純・楠本博之訳," 第 2 版 TCP/IP によるネットワーク構築 Vol.1 -原理・プロトコル・アーキテクチャ-", 共立出版, (Dec. 1991)

[8] Craig Hunt, 村井 純監訳,"TCP/IP ネットワーク管理", インターナショナル・トムソン・パブリッシング・ジャパン (Nov. 1994)

[9] 島 慶一,"IP version 6 の実装による相互通信の検証", 奈良先端科学技術大学院大学情報科学研究科修士論文,(Feb. 1996)

[10] Robert M. Hinden and Stephen E. Deering,Internet Protocol Version 6 (IPv6) Specification,Request for Comments 1883,Internet Engineering Task Force,(Dec. 1995)

[11] Robert M. Hinden and Stephen E. Deering,IP Version 6 Addressing Architecture,Request for Comments1884,Internet Engineering Task Force,(Dec. 1995)

[12] Robert M,Mike O'Dell and Steohen Deering, An IPv6Aggregatable Global Unicast Address Format,Internet Draft: draft-ietf-ipngwg-unicastaggr-02.txt,(Jul. 1997)

[13] Robert M. Hinden and Mike O'Dell,TLA and NLA A ssignment Rules,Internet Draft: draft-ietf-ipngeg-tla-assignment-00.txt,(Jul. 1997)

付録 A

IPv6ヘッダ・フォーマットの詳細

A.1　ヘッダフォーマット

IPv6 のヘッダは標準ヘッダと拡張ヘッダと呼ばれるものから構成され、IPv6 の標準ヘッダは、IPv4 の 20 年以上の運用で有効に活用されることがなかったフィールドが除かれ、簡素化されている。簡素化の要点としては、ヘッダ長の固定長化、チェック・サムの廃止などがある。また、IPv4 のオプションフィールドに変わるものとして拡張ヘッダが採用され、これによって柔軟な機能の追加を実現している。

A.1.1　標準ヘッダ

図 A.1 に IPv6 に標準ヘッダのフォーマットを示す。

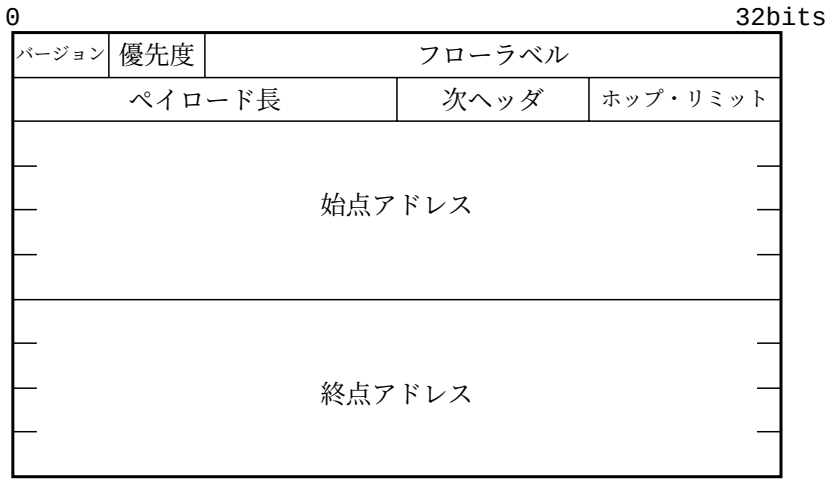

図 A.1: IPv6 ヘッダフォーマット (標準ヘッダ)

A.1.1.1 バージョン (Version)[4ビット]

プロトコルのバージョンを示し、IPv6は6が設定される。

A.1.1.2 クラス (Class)[4ビット]

トラフィックの種別を表すためのフィールドフィールドで最初の1ビットは、スループットよりも遅延を優先して欲しいトラフィックを表すビットとして使用することが推奨されている。現段階では、このフィールドは標準が行なわれていない。

A.1.1.3 フローラベル (Flow Label)[24ビット]

サービスの品質 (スループット、優先度、遅延など) やリアルタイム性など、データ転送において特別な処理を必要とするものに利用される。

A.1.1.4 ペイロード長 (Payload Length)[16ビット]

標準ヘッダに続くデータグラムの長さをバイト単位で示す。16ビット長あるので、指定できるデータグラムの長さは6万5535バイト以下となる。これ以上の大きさでデータグラムを送るには、拡張ヘッダで指定することで、より多くのデータグラムを送ることができる。

A.1.1.5 次ヘッダ (Next Header)[8ビット]

IPヘッダに続くヘッダを識別するため識別子が指定されます。IPv4のプロトコル・フィールドに相当する。拡張ヘッダや上位層のヘッダ (TCP,UDP) が示される。

A.1.1.6 ホップリミット (Hop Limit)[8ビット]

IPv4のTTLに相当するもので、IPv6では通過可能なルータの数を示している。この値によってパケットをどこまで転送できるかが決まる。パケットが作られるときに、ホップ・リミットに初期値が入れられ、パケットがルータを通過するたびに1つ減らされ、0になるとそのパケットは破棄される。これにより無駄なパケットがいつまでもネットワークに浮遊するのを防ぐことができる。

A.1.1.7 始点アドレス (Source Address)[128ビット]

送信元のIPアドレスを128ビットで表している。

A.1.1.8　終点アドレス (Destination Address)[128ビット]

受信先の IP アドレスを 128 ビットで表しいる。

A.1.2　拡張ヘッダ

IPv6 では、IP ヘッダとトランスポート層プロトコルの間に拡張ヘッダを挿入ことができる。拡張ヘッダによって、多くの機能を柔軟に追加することができる。拡張ヘッダを挿入する場合、IP ヘッダの次ヘッダフィールドには、その拡張ヘッダの識別子が設定される。拡張ヘッダにも次ヘッダフィールドが用意されていて、これをりようすることにより、次ヘッダフィールドに次の拡張ヘッダを指示することで、いくつかの拡張ヘッダを続けて指定することができる。最後の拡張ヘッダの次ヘッダ値には、トランスポート層プロトコルを示す識別子が指定される。

おもな拡張ヘッダには以下のようなものがある。

A.1.2.1　ホップ毎オプションヘッダ (Hop-by-Hop Option Header)

経路上にあるすべてにシステムに関係するオプションを指定することができる。

A.1.2.2　ルーティングヘッダ (Routing Header)

送信者がデータグラムの経路を制御したい場合に使用する。ルーティング・ヘッダは、あて先アドレスと組み合わせて、データグラムが通過する経路を指定する。

A.1.2.3　認証ヘッダ (Authentication Header)

認証ヘッダによって、受信者は、パケットの始点アドレスの確認と、内容が途中で改ざんされていないことを確認することができる。

A.1.2.4　暗号ペイロード (Encapsulating Security Payload)

暗号ペイロードは、IP データグラムの中に暗号化した情報を入れて送信する。これによって、経路途中での盗聴を防ぎ、正当な受信者だけが内容を読むことができる。

付録 B

IPv6の実装について

B.1 IPng Implementations

以下が今回 IPv6 を実装するにあたって参考にしたホームページ (http://playground.sun.com/ipng/ipng-implementations.2.html) の抜粋である。対応する FreeBSD のバージョンやアプリケーションなどが記載されている。ソースコードもここから入手できる。

```
---------------------------------------------------------------------
INRIA Rocquencourt

                      INRIA Rocquencourt is building a full (4.4BSD)
                      UNIX implementation based on NetBSD 1.2.1 and
                      FreeBSD 2.2.2

They are targeting a Sun SPARC sun4c (but can be easily ported to any
platform supported by
NetBSD, ie x86 PC, some MC 68k, MIPS, NS 32x32 workstations, etc) and
PC's capable of
running FreeBSD.

Applications currently supported include:

    Any Platform (SunOS, Digital Unix, etc.)
       tcpdump with IPv6 packet decoding
    NetBSD or FreeBSD w/ IPv6
       ftp/ftpd
       telnet/telnetd
       tftp/tftpd
       ttcp
       ping/traceroute
       multicast
       tcpdump
       netstat
       neighbor discovery utilities
       route
       ifconfig
       finger/fingerd
```

```
       inetd
       sendmail
       HTTP client (MMM) and server (Apache)
       X11 and RPC ONC tools

Work in underway or planned includes:

    Security (legally restricted in France)
    Multi-homing
    DHCPv6
    gated and MRT w/ RIPng and BGP4+
    video-conferencing
    NBMA (including IPv6 over ATM)
    BIND w/ dynamic address support, security, and transition tools
    NFS

The IPv6 implementation is available at
ftp://ftp.inria.fr/network/ipv6/ or
ftp.imag.fr/archive/networking/ipv6/INRIA/ . Information on MMM can be
found at
http://pauillac.inria.fr/~rouaix/mmm/ .

The mailing list for core developers is ipv6-bsd-core@imag.fr.

For more information contact Francis Dupont at
Francis.Dupont@inria.fr.
------------------------------------------------------------------------
```

B.2 実装の手順

```
How to install IPv6:
 1 - install FreeBSD 2.2.2 (optionaly install altq 0.3.1 too)
 2 - install IPv6 files:
  * gunziped and untar New.tar.gz (use the z option of GNU tar)
  (I use ``gzip -d < New.tar.gz | (cd /; tar xvBpf -)'' command)
  * NOTE: the script inst-new is provided for the next step.
  You use the list of new files (src-new):
   + if an entry is a directory the Makefile in the parent is up to
date
     then you have nothing to do.
   + if an entry is a regular file copy it in the ordinary version
     (I use ``cp -p XXX-new XXX'')
 3 - rebuild everything (``make world'' in /usr/src for all the
     utilities (it takes time)), ``config XXX'' and ``make depend;
make''
     for the kernel.
 4 - read KERNEL and HOWTO-USE
```