次世代インターネットプロトコルの実現に関する研究

指導教官
徳田 英幸 ・村井 純

慶應義塾大学 環境情報学部
79254475
村井研究室
南 政樹

平成8年2月

概要

近年のインターネットに対する関心の高さを象徴するように，インターネットは急激に成長し新たに社会的な要求を受けるようになってきた．また，そのため，既存の技術では対応できない部分も多くなり，それを考慮した新しいインターネットの基盤が必要とされている．インターネットの最も基礎的な技術であるインターネットプロトコル (IP) は，今まさに次世代へ向けて進化しようとしている．

本稿では次世代インターネットプロトコルを既存のインターネットプロトコルと比較・考察し，どのようにそれらの問題が解決されようとしているのかを述べた後に，それの実現に向けた設計と実装について述べる．

Abstract

Since people's interest about Internet is rising in these days, Internet has grown up abruptly, and is newly required of social demand. Therefore, existing technology found it difficult to cope with that rapid growth ,then taking into consideration about everything, new Internet inflastructure is needed . The most basic technology in the field of Internet, IP is about to change for the next generation.

In this thesis, it is considered how IP for the next generation(IPng:IPv6) can replace present IP (IPv4),and how they cope with both technical and social problems caused by replacing. And then, we describe how the design and the implementation was going.

目 次

図目次

表目次

第 1 章

序論

1.1 はじめに

インターネットは次世代の情報社会基盤として注目されている．事実，ここ数年の間でインターネットは急激な成長を遂げている．インターネットに接続されているホスト数は，現在も指数関数的な増加傾向を示しており，今後は更にその増加率が大きくなることが予想されている．

ところが，このインターネットを支える技術的基盤の『IP(インターネットプロトコル)」は，開発当時の予想を大幅に上回る変化によって，様々な問題を抱えている．例えば，IPアドレス空間の枯渇問題や経路制御情報の氾濫などがそうである．

また，IP がこの 20 年間ほとんど変化しなかったのとは対照的に，インターネットを取り巻く計算機や通信媒体などの環境は大きく変化してきた．例えば，計算機の小型化，高性能化，廉価化，また通信媒体の高品質化，大容量化，高速化などである．これらの要因によって，人間は計算機の利用形態を変化させ，インターネットに対して新たな要求を行うようになってきた．それは，例えば人間は計算機を持ち歩き，移動先でネットワークと接続するといった移動端末のサポートであったり，VoD(Video on Demand) などのような，動画や音声の転送を行うための実時間通信のサポートなどである．

こうした問題や要求に対して，これまでにも我々はいくつかのアプローチで，その問題の解決をはかってきた．だが，これらの解決策は短期的なものにすぎず，長期的な視点から考えると根本的な解決方法とは言えないのである．

このような状況の中，1992 年に，インターネット全体の技術的な方針を決定する IAB (Internet Architecture Board) は新たな IP を設計しそれに移行するという解決策を提案した．これを受けて，インターネットにおける，技術的な問題点を解決するための組織である IETF(Internet Engineering Task Force) では，具体的に，この新しい IP に関する議論を行ってきた．その結果，1994 年 7 月，それまでに提案されたいくつかの候補から，現在の IP を簡素化しアドレス空間を 128bit に拡張した SIPP(Simple Internet Protocol Plus) を次世代 IP (IP for Next Generation :IPng) のベースとして決定した．

現在までに，この次世代 IP の基本的な仕様は決定しており，さまざまな組織で実装が試

みられている．

1.2　本論文の目的

本論文では，次世代 IP に関する以下に示す事柄について述べる．

- 現在のインターネットの背景を考察し，問題となっていることについて述べる．
- 現在のインターネットの問題点を解決すべく設計された，次世代 IP の仕様を考察し，現在の IP とどのように異なっているのか，また，どのように問題点が解決されているのかを明確にする．
- 次世代 IP の仕様に基づいた，BSD 系 UNIX のための設計を行い，実装を行う．
- 他の実装との比較を行い，今回実装したものの評価を行う．

1.3　本論文の構成

本論文では，第 2章でインターネットの現状と問題点を考察，次世代 IP が必要になった背景について説明する．第 3章では，現在のインターネットの問題点を挙げ，次世代インターネットではそれをどのようにして解決しているかを述べるとともに，次世代インターネットプロトコルについて様々な観点から行った考察を述べる．第 4章では，次世代 IP を BSD 系 UNIX へ実装することを前提とし，第 3章で考察した次世代 IP の機能を基に行った設計について述べる．第 5章では，実装について述べる．これは第 4章で行った設計を基にした実装である．ここでは，次世代 IP のいくつかの特徴を具体例として挙げ，それをどのように実現したのか，実際のコードを参照しながら述べる．第 6章では，これまでの内容をまとめ，今後の課題と研究の方針を述べる．

第 2 章

インターネットの現状と問題点

本章ではまず，インターネットのこれまでについて述べ，概略を説明する．次に，インターネットを支えている技術的な基盤であるインターネットプロトコルについて述べその設計仕様，実現について説明を行う．そして，これら二つを踏まえ，現在取り上げられている問題点について述べる．

2.1 インターネットのこれまで

2.1.1 歴史的背景

今日，インターネットは研究・開発・教育・商用の目的のために広く利用されている情報インフラストラクチャである．これはインターネット発祥の地であるアメリカで，NSF(Natinal Sceince Foundation) や DARPA(Defence Advanced Research Projects Agency) などの，多くの政府組織がインターネットの活動を支援して来たことからも裏付けられる．

インターネットの発祥は，1969 年に発足した ARPANET である．これを基に，1975 年頃までに，文字通りネットワークとネットワークを相互に結びつける「ネットワークのネットワーク」として発展して来た．そして，それが国を越え，文化を越え，全世界を覆う世界最大のコンピュータネットワークとして，現在の形に至るのである．

先に述べたように，DARPA はもともとは軍事目的で開発が行われたのであるが，その発展は，学術研究がその目的であった．

研究者たちが，それぞれの研究についての情報を交換したり，議論を行ったりしてきたのである．つまり，コンピュータネットワークとは，例えば，隣にいる人間と話しをするくらい簡単に，海を隔てた人間とのコミュニケーションができるような，知的作業を促進するべきものであり，そのために発展するべきである，と言われている．

インターネットに大きな変化が現れたのは，近年になってからである．それまで，インターネットを利用する人の大部分は，歴史的な経緯から，学術研究機関や教育機関に関係する人がほとんどで，しかも研究所や職場といった限られた場所からの利用であった．だが，最近では商用インターネットの普及とともに，利用者の中心は，以前のような特別な

人ではなく，一般の人へと移行をはじめている．

これはインターネットが新たな情報産業の市場になり得ることを示すとともに，情報化社会においてのインフラストラクチャとしての責任が大きくなりつつあることを意味している．

2.1.2 利用形態の多様化

IP によるインターネットが登場してからおよそ 20 年が経過した．IP やそれを取り巻く他のプロトコルが，この 20 年間ほとんど変化がなかったのに対し，インターネットを取り巻く計算機，通信媒体，利用者は大きく変化して来た．特に顕著な例として，計算機の小型化，高性能化，廉価化，通信媒体の多様化，大容量化などがあげられる．その結果，例えば，インターネットに携帯型計算機などの移動可能なホストが接続されたり，LAN だけではなく，電話回線でのダイアルアップ接続といった新たな利用形態が生まれて来た．

このように，現在ではインターネットを利用する形態が多様化し，それがインターネットに対する新たな要求となっている．

2.1.3 規模の拡大

現在の IP の仕様が決まり，事実上のインターネットが始まった 1982 年時点で，インターネットに接続していた組織数は，約 10 組織，接続ホスト数も約 250 台であったが，現時点での最新の統計 (1995 年 7 月) では，組織数が約 12 万，接続ホスト数，約 700 万台と，その驚異的な成長を果たして来たことが分かる．

特に近年の増加率が著しく大きいのは，これまで利用者が学術研究機関に関係している人がほとんどであったのに対し，商用のインターネットサービスプロバイダが激増したことによる，企業，個人のインターネット利用率の大幅な増加がその理由にあげられる．また，第 2.1.2節で述べた通り，インターネットを取り囲む環境の向上も，その理由である．

この変化は，アメリカ政府の NII 構想や GII 構想，さらにインターネットブームにより今後はさらにその傾向が強まる事が予想される．

2.2 インターネットプロトコル (IP)

インターネットは IP(インターネットプロトコル) に基づいて相互に接続されている，パケット交換方式によるコンピュータネットワークの総称である．ここで IP は，インターネットを支える最も基本的な技術のことを意味するが，正確にはそれを行うための送信者–受信者間での取り決め・手順[1]を意味している．これは決して複雑なものではなく，以下に示すように，非常に単純かつ明解であることが特徴である．

- データ転送のための基本単位とそのフォーマットの定義

---

[1] protocol: 外交儀礼，手順

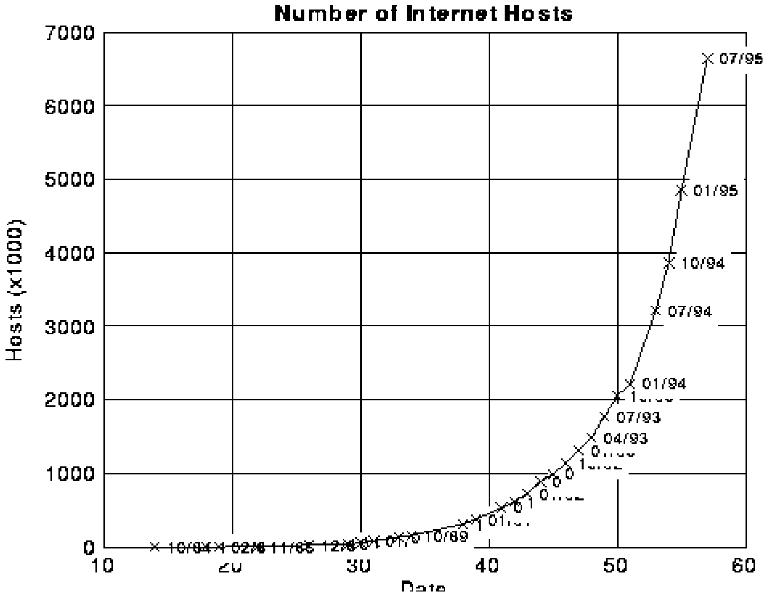

図 2.1: ホスト数の増加

- データを配送するためにそれを送る経路の決定
- データ配送の際のエラーの通知

この役割は図 2.2にあるように，OSI 参照モデルの第 3 層，ネットワーク層の二つの役割，個体の識別 (アドレッシング) とパケット配送のための経路の決定 (経路制御) を満足する．すなわち，IP はインターネットを構成するネットワーク層のプロトコルである．

以下では，より具体的にインターネットプロトコルの特徴について述べる．

2.2.1 コネクションレス型通信

コネクションレス型の通信とは，手紙のように一旦手元を離れたら送り主と手紙，そして受取主の間に何も関係がないのと同じで，一旦，送信者から離れたパケットは送信者，受信者のどちらからも独立のもの，すなわちデータグラムとして扱われる信頼性のない通信方式のことである．この通信方式は，送られる順序とは全く異なる順序で受取主にパケットが届いたり，途中で失われたりすることがある．

コネクションレス型の通信とは逆に，電話のように会話をはじめる前に，コミュニケーションの両端を論理的な回線で結び通信路を確保する方法を，コネクション指向型の通信と言う．データは必ずこの通信路上を流れ，その到着，順序が保証される．

IP は信頼性のないコネクションレス型で通信を提供している．すなわち，IP によって配送されるパケットは，失われたり，順序が入れ替わったり，重複したりすることもあり得るのである．

7	アプリケーション層
6	プレゼンテーション層
5	セッション層
4	トランスポート層
3	ネットーワーク層
2	データリンク層
1	物理層

図 2.2: OSI 参照モデル

このため，IP の特徴は最善努力式，すなわちパケットを受信者へ配送するというよりは，受信者へ配送しようと最大限の努力を行う方式であることである．つまり，信頼性は上位層で保証すれば良く，インターネットの仕組みでもっとも基本的な IP はそれを保証する必要がないのである．そのため，IP は最小限のエラー処理しか行わない．これは，例えば，インターネット上を配送されているパケットが，途中で破棄されるような場合やそのパケットの受信者が存在しない場合などに，送信者に対してエラー通知のみを行うことを指す．これは ICMP(Internet Contorol Message Protocol) と言い，信頼性のない通信でありながら，最善努力を行うための重要な仕組みの一つである．

もう一つの特徴は，信頼性がない反面，複雑な処理が必要ないため，IP はプロトコルとしての負荷が非常に低く，高速に処理が可能であり，規模の拡大には強靭であることである．

2.2.2 IP ヘッダフォーマット

IP はコネクションレス型の通信を行うと，第 2.2.1節で述べた．このため，一つ一つの IP パケットには，送信側と受信側を示す情報を持たせる必要がある．IP では，これらの通

信に必要な情報はすべてIPヘッダと呼ばれるプロトコルヘッダを利用している，例えば，小包を例にすると，自分と相手の住所を書いた荷受表にあたるものがIPヘッダである．

IPヘッダの内容は図2.3に示す通りである．

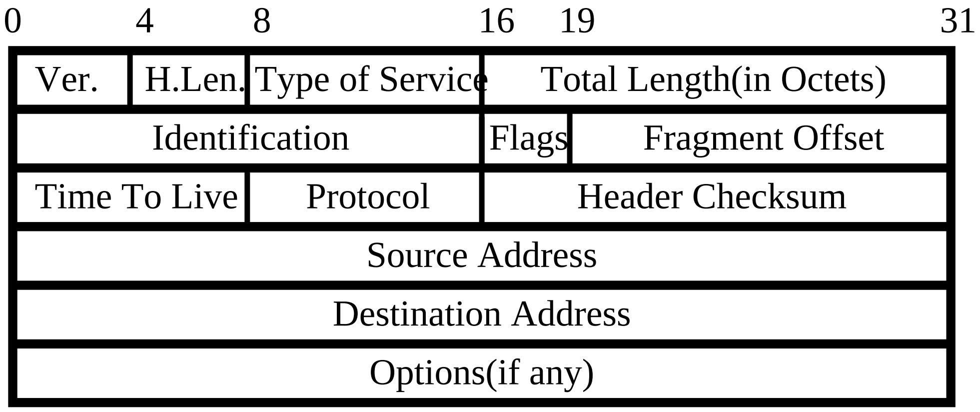

図2.3: IPヘッダフォーマット

表2.1: IPヘッダの構成

名称	役割
Ver.	IPのバージョンを番号．現在のIPはバージョン4
H.Len.	4オクテット(4バイト)単位で表したIPヘッダの長さ
Type of Service	そのパケットに望まれているサービスの品質の指定(第2.2.6節参照)
Total Length	オクテット(バイト)単位のIPヘッダを含んだパケット全体の大きさ
Identification	IPパケットひとつずつに振られる識別子
Flags	3bitで，そのパケットを分割しても良いか，また分割されたかどうかの目印(flag)
Fragment Offset	13bitで，分割された場合，元々のデータのどの部分か
Time To Live	このパケットがインターネット内に存在しても良い時間(秒) 通常，ルータなどで中継された場合1減らされる 0になったらそのパケットは破棄される
Protocol	上位層のプロトコル．“1” = ICMP，“6” = TCP，“17”= UDP
Header Checksum	IPヘッダのチェックサム
Source Address	送信ホストのIPアドレス(第2.2.3節参照)
Destination Address	受信ホストのIPアドレス(第2.2.3節参照)

2.2.3 IP アドレス

コミュニケーションを行うためには，必ず相手を識別するための識別子が必要である．IP では，IP アドレスと呼ばれる 32bit の識別子，すなわちアドレスによって，インターネット上の全てのホスト，ネットワーク，また任意のホストの集合を識別している．この 32bit のアドレス空間は 2^{32} ，およそ 43 億の識別子として利用可能である．

IP アドレスの特徴は，それが単なるホストの個体識別を行うだけではなく，そのホストが接続されているネットワークの識別子も，そのアドレス内に含まれていることである．言い替えれば，IP アドレスは，ネットワーク部とホスト部の二つの部分の組合せから定義されているのである．ネットワーク部とホスト部の関係を，図 2.4に示す．

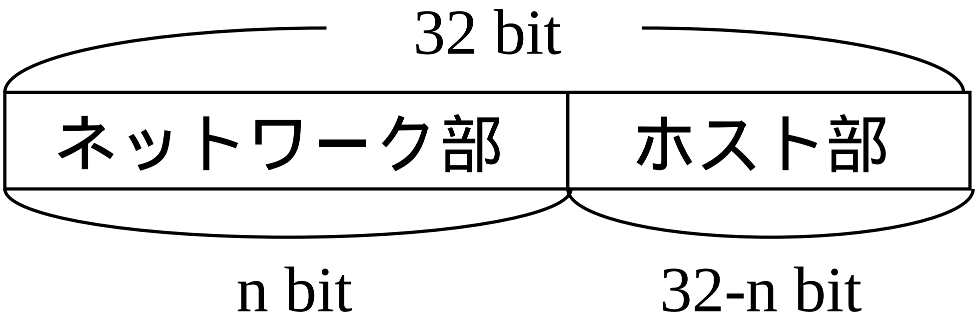

図 2.4: ネットワーク部とホスト部

ネットワーク部で，物理的に同一ネットワークにつながっているホストの集合を表すことができるので，一つのホストに一つの経路情報 (第 2.2.5節) を持たせる必要がなく，効率的である．その反面，ホストが移動した場合，接続しているネットワークのアドレスが変わってしまうため，アドレスを付け変える必要がある (図 2.5参照)．

2.2.4 クラスによるアドレス割り当て

IP アドレスは，そのネットワークを識別するネットワーク部と，そのネットワーク内でのホストを識別するホスト部に分けられることは第 2.2.3節で述べた通りである．だが，ネットワークの規模によって，ホスト数が異なるため，この 32bit の IP アドレス空間のうち，ネットワーク部とホスト部の配分が異なったアドレス割当を行わなければならない．これはクラスと言う概念で，その配分の仕組みが決められている．また，組織に割り当てられる IP アドレス空間は，そのネットワークに接続されるホスト数により決定される．具体的には，アドレスの先頭 4bit で，A，B，C，D のそれぞれクラス[2]を識別する．

クラス A は先頭 1bit が “0” で，接続ホスト数が $2^{16} = 65536$ 以上の大規模なネットワークに割り当てられるアドレスである．ネットワーク部が 7bit ($2^7 = 128$ の識別が可能)，ホ

---

[2] これら以外に予約されているが使われていないクラス E も存在する．

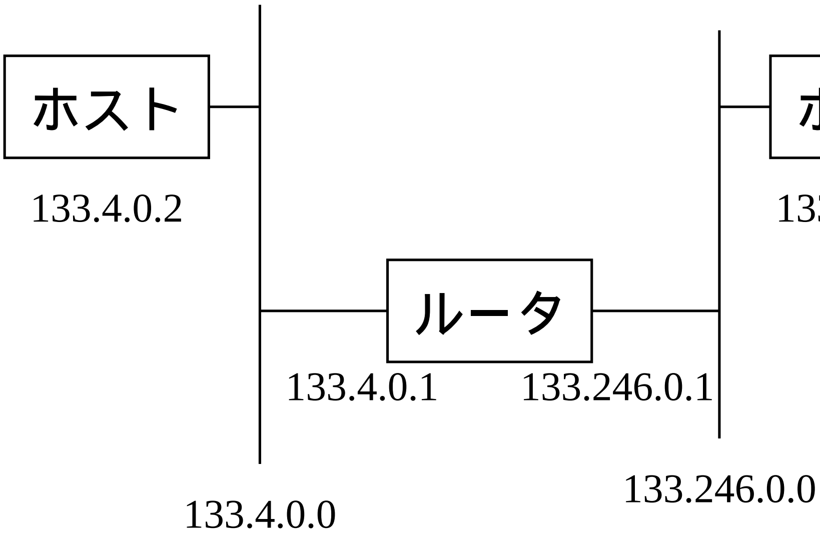

図 2.5: ネットワークと IP アドレスの関係

スト部が 24bit ($2^{24} = 16777216$ の識別が可能) である．

クラス B は先頭 2bit が “10” で，接続ホスト数が $2^8 = 256$ から $2^{16} = 65536$ までの中規模ネットワークに割り当てられる．ネットワーク部は 14bit($2^{14} = 16384$ の識別が可能)，ホスト部が 16bit($2^{16} = 65536$ の識別が可能) である．

クラス C は先頭 3bit が “110” で，接続ホスト数が $2^8 = 256$ 未満の小規模なネットワークに割り当てられるアドレスである．ネットワーク部が 21bit($2^{21} = 2097152$ の識別が可能)，ホスト部が 8bit($2^8 = 256$ の識別が可能) である．

クラス D は先頭の 4bit が “1110” で，現在は実験用アドレスとして利用されている．

このクラスという考え方によって，先頭の数 bit を評価することにより，そのアドレスのネットワーク部とホスト部を取り出すことができる．

だが，現実的に，同一ネットワークに 65536 台ものホストが接続されることはなく，この方法による割り当ては，非常に無駄が多いと言える．

そこで現在では，このクラスと言う考え方は，組織へのアドレス割当時にのみ使われており，実際にネットワーク内で，アドレスのネットワーク部とホスト部の区分けは，割り当てられたアドレス空間内でのサブネットと言う考え方を用いて行われている．

サブネットとは，ホスト部に当たる部分まで，ネットワーク部を拡張し，更に細かくネットワークを区切ることである．例えば，クラス B のアドレスでは，先頭から 16bit までがネットワークを表すが，ホスト部の先頭 4bit をネットワーク部として，残りの 12bit でホスト部を表すとする．このとき，ホスト部の先頭 4bit で表されたネットワークのことをサブネット部といい，先頭から 20bit がネットワーク部となる．この 20bit を表すには，先頭から 20bit が全て “1” で，残りが全て “0” のサブネットマスクという 32bit のマスクを利用する．

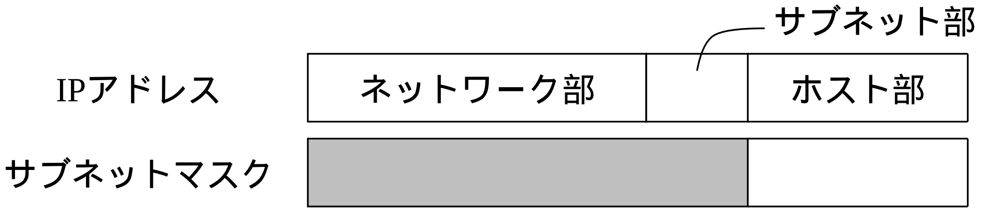

図 2.6: ホスト部，ネットワーク部，サブネット部

2.2.5　経路情報

OSI 参照モデルのネットワーク層にあたる IP の一つの大きな役割に，経路制御があげられる．経路制御とは，IP パケットが送信ホストから受信ホストまで到達する間に通過する，ネットワーク間の橋渡しをする中継ホストにおいて，そのパケットをどのネットワークに中継すれば良いか制御することである．この中継ホストのことをルータ (router) と言い，インターネット内では，基本的に一つの物理的なネットワークに一つのルータが存在することが必要である．

中継する際に，ルータは，自身が接続しているネットワークについてのネットワークアドレス，サブネットマスクなどの情報を元に，受け取ったパケットの受信ホストのアドレスと比較しながら，どのネットワークに中継するべきかを判断している．

このため，正しく，しかも効率良く経路制御を行うには，ルータが接続しているネットワークについてはもちろん，その先に接続されているネットワークについても知る必要がある．すなわち，ルータは，自身が接続しているネットワークの周りのネットワークの集合についても，経路制御のための情報を知る必要がある．このような情報を経路情報と言い，原則として，CIDR 等の特別な技術が使われていない限り一つのネットワークにつき一つの経路情報が存在する．このため，サブネットが一つ増えると，経路情報が増えることになる．このことは，経路情報とアドレスの利用効率には相関関係にあり，アドレスの利用効率を高めるため，サブネットを新たに設けると，経路情報が増えることを意味する．

2.2.6　ToS(Type Of Service)

コミュニケーションにはそれぞれ性質や要求があり，それをサービスとして提供することが望まれている．この性質や要求のことを QOS(Quality of Service) といい，IP はこれを実現する ToS という機能を持っている．

ToS は，その IP パケットの優先度と要求するサービスの内容を表し，ルータでパケットの中継が行われる時にそれが評価され，適切な経路へ中継される．

だが，実際に ToS をサポートしているルータはほとんどなく，アプリケーションレベルでこれを利用している例もほとんどない．

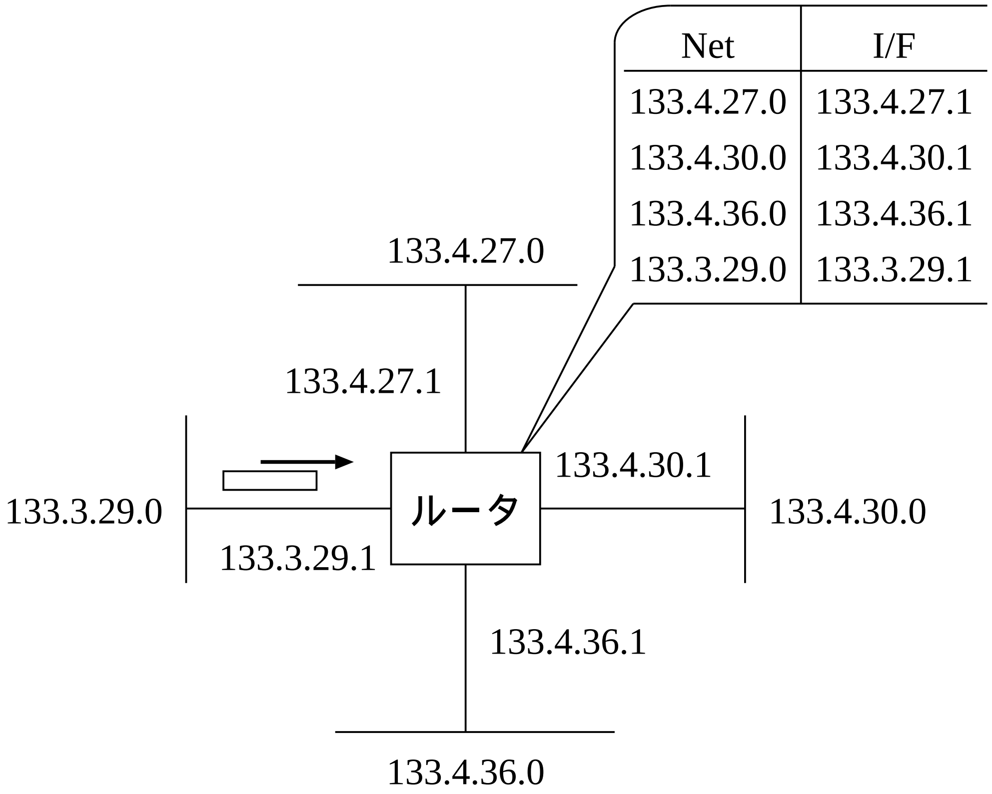

図 2.7: 経路情報と経路制御

2.3 インターネットの問題点

この節では，現在のインターネットが直面している問題点を挙げる．

2.3.1 アドレス空間枯渇

IPが識別子として32bit，すなわち約43億のアドレス空間を持つことは第2.2.3節で説明した．これからも分かるように，IPはもともと現在のような，大きなネットワーク環境を想定して設計されておらず，それが原因でアドレス空間の枯渇が発生している．

アドレスの割り当ては，第2.2.4節で述べた通り，クラスという概念にしたがって行われている．

現在は，新しくアドレスを取得しようとするとクラスCのアドレスが割り当てられるが，歴史的にクラスBのアドレス取得を勧めていた時期があった．これは，経路情報の増加が，クラスCのネットワークを増やした時より，クラスBのネットワークを増やした時の方が少なくて済むからである．しかし，第2.1.3節で述べた通り，接続ホストの数が増加すると，インターネット全体で14bit(16384)しかないクラスBのアドレス空間の枯渇が現実のもの

となり，クラスBのアドレス空間を割り当てる方針から，複数のクラスCを割り当てる方針へ転換したのである．

IETF(Internet Enginering Task Force)のALE(Address Lifetime Estimation BOF)によれば，現在のIPの持つ32bitのアドレス空間で，まだ利用されていない空間を100%に近い効率で利用できたとしても，西暦2008年± 10年の範囲で全てのアドレスを使い切ると予測されている．

インターネットを支えるIPの，その中核をなすIPアドレスの枯渇問題は，今，インターネット上で最も大きな問題であるといえる．

2.3.2 経路情報の氾濫

アドレス空間の枯渇と並んで，深刻なのが経路情報の氾濫である．第2.3.1節で述べた通り，現在IPアドレスはクラスCのアドレスを複数割り当てる方針を取っている．だが，第2.2.5節で述べた通り，IPアドレス的にネットワークが増加することは，経路情報が増加することを意味する．事実，このためにインターネットのバックボーンに接続されているルータでは，経路情報が爆発的に増えている．

2.3.3 計算機環境の変化

第2.1.2節で述べた通り，変化の著しい計算機環境から，新たな要求がインターネットに向けられている．具体的には，移動ホストのサポートであるとか，QOSに則った形でのデータの配送，また，データ通信の安全を保証するための仕組みなどである．特に最後の“データ通信の安全の保証”は，今後利用が予測されている，電子通貨，電子投票，電子署名などでその実現が期待されている．

2.4 IPの問題点の解決

インターネットの問題点は，第2.3節で述べた通りである．これらの問題点は，ほとんどがIPの仕様に原因がある．そのため，根本的にこれらの問題を解決するためには，IP自身を変更しなくてはならず，大がかりな作業が必要となる．これに対し，根本的な解決ではないものの，短期的に問題を対処する方法がいくつ考えられてきた．

この節では，その対処法について述べる．

2.4.1 規模に関する問題

CIDR(Classless Inter-Domain Routing)

CIDR(Classless Inter-Domain Routing)は，ネットワークアドレスとサブネットマスクの対によってアドレスを表現し，従来のクラスによるアドレス割り当て方法をより柔軟にした，アドレス割り当ての技術である．また，同時にCIDRによれば，いくつかのネット

ワークをインターネット上では一つの経路と見せかけることができるので，経路情報の増大を抑制するための技術でもあるといえる．

このCIDRを利用する場合には，次の原則に従わなければならない．

- クラスCのアドレスを利用する
- サブネットマスクは，連続してなければならない
- ネットワークアドレスの範囲は，必ず2の冪乗となる

具体的に，CIDRによるIPアドレス空間の割り当てについては，RFC1466で規定されている通り，クラスCのアドレス空間の上位部の `208.0.0.0 - 223.255.255.255` は当分割り当てない方針で，下位部 `192.0.0.0 - 207.255.255.255` を地域別に，表2.2のように割り当て，地域ごとでCIDRによる経路制御情報の集約を行なうことになっている．

表2.2: クラスC - CIDRによるアドレス割当

割り当て地域	アドレス空間
複数の地域	`192.0.0.0 - 193.255.255.255`
ヨーロッパ	`194.0.0.0 - 195.255.255.255`
その他の地域	`196.0.0.0 - 197.255.255.255`
北米	`198.0.0.0 - 199.255.255.255`
中南米	`200.0.0.0 - 201.255.255.255`
環太平洋地域	`202.0.0.0 - 203.255.255.255`
その他の地域	`204.0.0.0 - 205.255.255.255`
その他の地域	`206.0.0.0 - 207.255.255.255`

Private Address

歴史的に，インターネットに直接続されていないネットワークに対してまで，IPアドレスが割り当てられてきた．だが，これはIPアドレスが足りない昨今，非常に非効率的なことであり，そのアドレスを再利用するだけでも効果があると考えられている．そこで，インターネットに接続されていないネットワークに対しては，IPアドレス空間の一部を世界中で一意ではないアドレス，すなわち重複を許すアドレスとして割り当て，それを利用するよう促してきた．このアドレスのことをプライベートアドレスといい，RFC1597により，表2.3に示したアドレス空間は，プライベートアドレスとして一般のアドレス割り当てからは外されている．同様な理由から，このアドレスに対しては経路制御も行われない．

表 2.3: プライベートアドレス空間

10.0.0.0	-	10.255.255.255
172.16.0.0	-	172.32.255.255
192.168.0.0	-	192.168.255.255

2.4.2 移動ホストのサポート

DHCP

小型計算機が増え，計算機を持ち運びする利用者が増えている．このため，移動した先でのインターネットの利用という新たな利用形態が生まれてきた．だが，インターネットを利用するために必要となるIPアドレスは，接続しているネットワークのアドレスをホストのアドレスに含むため，移動先でIPアドレスを付け替えなければならず，非常に面倒である．

このようなホストのサポートのため，動的にインターネットとの接続に必要な設定を行うDHCP(Dynamic Host Configuration Protocol)が提案されてきた．これは，BOOTPの拡張として設計されており，既存のネットワークとの親和性が高いのが特徴である．また，動的にアドレスを割り当てるので，利用されていないアドレスの回収，再利用が行えるのも特徴である．

DHCPの実装例はすでにあり，商用のソフトウェアにも順次組み込まれてつつある．

これにより，移動ホストの接続しているネットワークが変わっても，利用者は何の操作もすることなく，インターネットの利用を継続できる．

第 3 章

次世代インターネットプロトコル

この章では，第 2章で述べたインターネットの問題点を根本的に解決するべく開発されている，次世代インターネットプロトコル (次世代 IP) について，その選定の条件と経緯，仕様を述べる．最後に，現在の IP やそれを支えるプロトコル群と次世代 IP のそれの間で，定性的な比較を行う．

3.1 次世代 IP への要求 – 次世代 IP に必要とされる機能 –

これまで述べてきた，現状と短期的な問題点の解決を踏まえ，IETF では、次世代 IP を選ぶための、あるいはそれを新たに考案するための指針となる技術的基準について多くの議論がなされてきた。本節ではこれらをまとめる．

3.1.1 基本原理

IETF によると，次世代 IP は次の 5 つを考慮すべきである．

1. 構造的に単純であること (Architectural Simplicity)
2. 複数の上位層プロトコルをサポートすること (One Protocol to Bind Them All)
3. 長期の運用が可能なこと (Live Long)
4. 発展的であること (Live Long AND Prosper)
5. 統制的ではないと同時に協調的であること (Co-operative Anarchy)

1) では上位層に位置するプロトコルが次世代 IP を利用して構築することが可能で，しかも，基盤となる多くの技術の相違を吸収する必要があること，2) はマルチプロトコルネットワークを実現するための基盤となるべきあることを述べている．3) は，次世代 IP が今後 20 年以上は利用されていかなければならないこと，また 4) で，移行を促すためには魅

力となる新しい機能を持っていなければならないと述べている．最後の 5）は，インターネットの今日の発展は，中心的な意思決定機関によって運営されず，多くの技術者が独自に，しかも互いに協調して作り上げてきたものであることを強調し，次世代の IP も同様でなければならないとしている．

3.1.2 技術基準

以下の項目が IPng を選択する際の技術的基準 (Criteria) である．それぞれの項目について，簡単な説明と，詳細な解説，さらにその機能が実現されるべき時間的制約が決められている．

以下はそれを簡潔にまとめたものである．

- ネットワークの規模
  次世代 IP は，最低限 10^{12}のホストと 10^9 のネットワークをサポートしなければならない．より安全を確保するためには，10^{12}のネットワークと 10^{15} のホストをサポートするべきである．

- 柔軟なトポロジーへの対応
  様々なネットワークトポロジーに対応することができなければならない．

- 性能
  次世代 IP のルータは、利用可能なもっとも高速なメディアの速度に追従できるだけの性能を実現しなければならない。また、ホストは 既存の IP と同等以上の性能を持たなければならない.

- 堅牢性
  次世代 IP は堅牢でなければならない．不正なデータや事故によって，ネットワークが重大な障害を受けるようなことがあってはならない．

- 移行
  既存の IP からの円滑な移行が可能であることが不可欠である．

- メディアからの独立性
  あらゆる種類のメディアに対応できなくてはならない．

- データグラムサービス
  UDP に相当する，信頼性のないデータグラムの通信が可能でなくてはならない．

- 構成, 管理, 運用
  ルータやホストに対する，容易で大規模に分散された構成管理が可能でなければならない．

- 安全性
  安全なネットワーク層を提供することができなくてはならない．

- 一意な名前付け
  全てのネットワーク上のオブジェクトに対して，一意の名前を与えることができなくてはならない．

- 利用
  次世代 IP やそれに関連するプロトコルや仕様は，誰でもが自由に入手することが可能で，それらを実装するのに費用がかかってはならない．

- マルチキャスト
  ユニキャストとマルチキャストの両方をサポートしなければならない．しかも，マルチキャストの動的な経路制御が可能で，マルチプロトコルを意識したものでなければならない．

- 拡張性
  拡張性が高く，将来要求されるであろうサービスを提供できるものでなければならない．また，拡張を必要としないノードに対して変更を強要せず，複数のバージョンのプロトコルが共存できるものでなければならない．拡張性の対象としては，アルゴリズム，ヘッダ，データ構造，パケットタイプなどが含まれる．

- ネットワークサービス
  複数のサービスクラスをサポートし，それぞれのクラス毎に異なるサービスを提供できなければならない．これには、政策的経路制御，流量制御，資源確保，サービスの型や質を制御することなどが含まれる．

- 移動性
  ホスト，ネットワーク，インターネットワークの移動性をサポートしなければならない．これには，接続を保ったままの物理的な移動と，一度接続を絶って別の地点で接続を再開する移動の両者が含まれる．

- 制御プロトコル
  ネットワークの検査やデバッグのための，基本的な制御プロトコルをサポートしなければならない．

- 独自ネットワークのサポート
  ユーザが基本的なインターネットワーク上に固有のネットワークを構築することが可能でなければならない．これは、主にトンネリング技術によって実現されると考えられる．

- 技術的基準に含まれない項目
  重要ではあるが，次世代 IP を選択する際に敢えて考慮する必要がない項目がいくつか存在する．例えば，フラグメンテーション，ヘッダチェックサム，ファイアウォール，ネットワーク管理，アカウンティングなどがそうである．しかし，これらの機能

を実装してはいけない，あるいはする必要がないということを意味するものではなく，技術的基準としては考慮する必要がないという意味を持つ．

3.2　次世代 IP — プロトコル仕様 —

本節では，IP の問題点を根本的に解決すべく開発が進められている次世代 IP について，その仕様について述べる．

3.2.1　階層モデル

次世代 IP は，IP の発展形として設計されているため，この二者には類似点が非常に多い．だが，問題点解決のために変更した部分やいくつかの新しい機能のために変更された部分があり，下位層，上位層との関係で変更された部分がある．

図 2.2で示した通り，IP はネットワーク層のプロトコルである．ネットワーク層は，トランスポート層から渡されたデータに，指定されたホストへ届けるために必要な情報を含むヘッダを付け加えたり，その逆にデータリンク層からそのホストに届けられたパケットを受け取り，それをトランスポート層へ渡すことを行っている．つまり，IP は二つの異なった動作を行っていることになる．図 3.1に既存の IP を，同様に図 3.2に，次世代 IP を OSI 参照モデルに当てはめた場合のプロトコルスタックを示す．

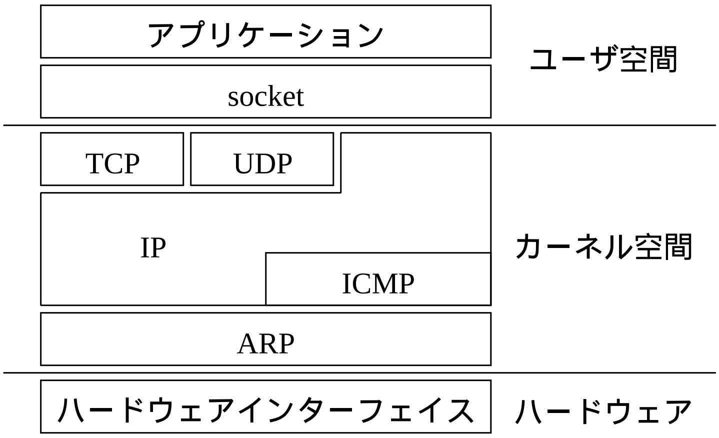

図 3.1: 既存の IP のプロトコルスタック

3.3　次世代 IP — IP と比較して —

本節では，次世代 IP と既存の IP の定性的な比較を行い，その特徴を述べる．

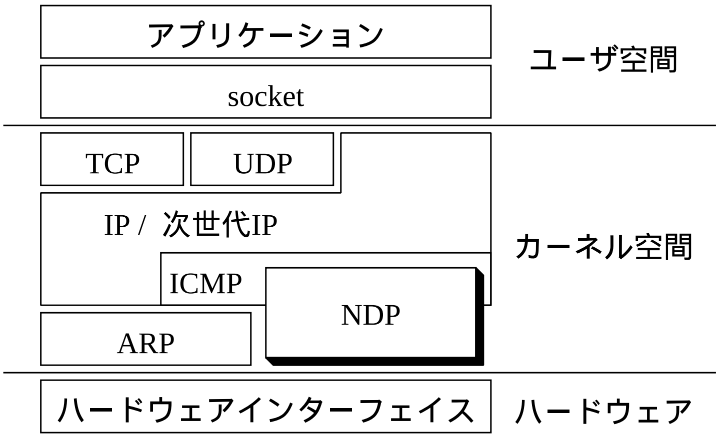

図 3.2: 次世代 IP のプロトコル体系

3.3.1 アドレス空間の拡張

次世代 IP は，インターネットの大きな課題であった規模の問題を解決するために，128bit のアドレス空間を設定した．第 2.2.3節で述べた通り，既存の IP では 32bit のアドレス空間，約 43 億のホストとネットワークの識別が可能であったのに対し，128bit になり約 3.4×10^{38} のアドレスを提供している．

これを世界中の人口一人あたりに換算すると，約 10^{28}/人のアドレスを，また，地球の表面積に換算すると $10^{20}/1cm^2$のアドレスを保持することが可能である．

これからも分かるように，非常に広大なアドレス空間であり，第 3.1.1節で述べた，基本原理の「長期運用が可能である」に一致する．

3.3.2 次世代 IP ヘッダ

次世代 IP では，IP ヘッダ (図 2.3) にあったのいくつかのフィールドを廃止，または拡張 (オプション) ヘッダ (第 3.3.3節) として付加している．これは，拡張機能をオプションヘッダとして指定することにより，標準のヘッダフォーマットを簡素化し，パケット処理のコストを低減することを目的としているためである．図 3.3に次世代 IP のヘッダフォーマットを示す．

既存の IP はヘッダが可変長のため “Header Length” でヘッダの長さをヘッダに持つ．これに対し，次世代 IP ではヘッダが固定長のため，このフィールドは必要なくなる．

同様の理由から，次世代 IP では，既存の IP ヘッダで “Total Length” とされていた，パケット全体の大きさを，データ部のみの大きさを示す “Payload Length” に変更している．これにより，純粋にデータ部の大きさを知ることが出来るため，IP で行われているように全体の大きさとヘッダ長からデータ部を求める必要がなくなる．

既存の IP，次世代 IP 共にヘッダの先頭にある “Version” フィールドは，IP のバージョ

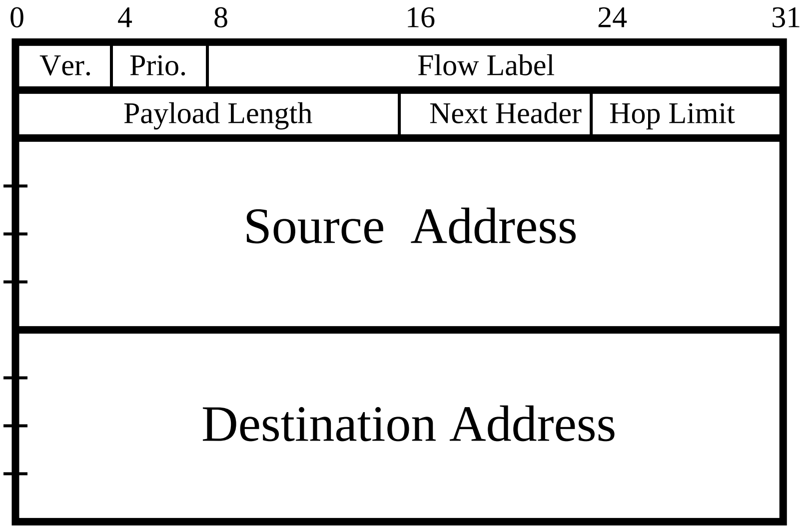

図 3.3: 次世代 IP ヘッダフォーマット

ンを示す 4bit のフィールドで IP では “4” が，次世代 IP では “6” が設定される．

“Hop Limit” は，次世代 IP のネットワーク内での寿命を示す．これは既存の IP の TTL フィールドの機能と類似しているが，TTL がルータがパケットを転送するために保持していた秒数分減らさなければならないのに対し，“Hop Limit” は明確に「ルータを通過するたびに 1 ずつ減らす」ことになっている．

“Source Address”，および “Destination Address” は，既存の IP と同じで IP パケットの送信元，受信先をそれぞれ表す．

“Priority” はそのパケットが配送される時に期待されている優先度が 4bit で表現される．これは既存の IP の ToS に似たているが，より細かく仕様が決められている．

“Priority” は，0–15 のうち，TCP のような輻輳制御下にある場合は 0–7 が，UDP のようなデータグラム型の転送の場合は 8–15 の数値が用いられる．前者の場合，表 3.1にあるように，それぞれのトラフィックの性質に応じた数値によって，後者の場合，数値の小さい方がパケット破棄の優先度の高さを表している．

“Flow Label” フィールドは，QOS の制御やリアルタイムサービスのために用いる，そのトラフィックを識別するためのフィールドである．このフィールドは 24bit である．次世代 IP ではこの “Flow Label” と “Priority” フィールドによって，タイプに応じた転送方法を定義できる．

“Next Header” フィールドは，次世代 IP ヘッダに続くオプションヘッダ，もしくは上位層のプロトコルのタイプを指定するために利用する 8bit のフィールドである．上位層のプロトコルタイプのを示す値は，ICMP を除いて，既存の IP と同じ値を使用する．

表 3.1: 輻輳制御下での優先順位

Number	Categories
0	uncharacterized traffic
1	“filler” traffic. (e.g. NetNews)
2	unattended data transfer (e.g. e-mail)
3	(reserved)
4	attended bulk transfer (e.g. FTP, NFS)
5	(reserved)
6	interactive traffic (e.g. telnet, X)
7	internet control traffic (e.g. routing protocols, SMTP)

3.3.3 拡張ヘッダによる機能の分割

次世代IPでは，ネットワーク層で必要な付加的な情報を，拡張ヘッダとして，ヘッダとは切り離してパケットに含めることができる．

IPでは，ネットワーク層で必要な付加情報をIPヘッダの内部に持っていた．すなわち，可変長のヘッダによりこれを実現していたのである．この仕組みは非常に複雑で，パケットを処理する度にIPヘッダとデータ部を分けるために，適切な計算を行わないといけないので非常に非効率的である．この点で比較すると，次世代IPの拡張ヘッダは非常に柔軟なオプションの設定を可能にしている．また，新たな機能を加える時も非常に容易に対応でき，拡張性にも優れている．

現在考案されている，拡張ヘッダの種類は次の通りである．

- Hop-by-Hop オプションヘッダ
- Routing ヘッダ
- Fragment ヘッダ
- Destination オプションヘッダ
- Authentication ヘッダ
- Encapsulating Security Payload ヘッダ

Hop-by-Hop option ヘッダ

Hop-by-Hop ヘッダは，パケットが通過するすべてのホストで処理されるヘッダであり，可変長のフォーマットを持ち複数のオプションを指定できる．オプションフィールドはTVL(Text Length Variable) というフォーマットで記述される．このオプションは次世代IP ヘッダの Payload よりも大きなデータを転送する時に “Jumbo Payload Length” が含まれて使用される．

Routing ヘッダ

routing ヘッダは，IP の Source Route オプションと類似した機能を持っている．このヘッダはパケットの通過経路のホストが付加すべきものではなく，送信元が経路制御に必要な情報をすべて付加する必要がある．

Fragment ヘッダ

Fragment ヘッダは，パケットの送信元が受信先の経路の MTU よりも大きなデータを送ろうとする場合に指定するヘッダである．同じパケットから分割されたパケットは，共通の Fragment ID を持っており，受信先ではこれを利用して分割されたデータを再構成する．次世代 IP では，IP とは異なり，中間ノードでのフラグメントは行なわれない．これは，Path MTU discovery という技術を用いることにより，MTU の検出しそれに見合ったデータ長でパケットを作るからである．

もし，途中で MTU がパケット長より小さかった場合，ICMP エラーメッセージを送信者に送り，再送を行う．

Authentication ヘッダ

Authentication ヘッダは，認証情報をパケットに付加することでセキュリティを提供する機能を持つ．Authentication ヘッダにより IP レベルでのセキュリティ機能が強化されたが，パケットモニタリングなどによるトラフィック解析に対するセキュリティは考慮されていない．この問題はかなり議論されており，結論としては「やるべきである」ということになっているが，現在のところ実装されている例はほとんどない．

Destination option ヘッダ

Hop-by-hop ヘッダと同様のフォーマットを持つが，Destination ヘッダで指定されたオプションはパケットの受信先でのみ有効となり，中間のノードでは無視される．

3.3.4 オプションヘッダと上位層プロトコル

通常，次世代 IP のパケットも IP のパケットと同じようなパケットのフォーマット (図3.4参照) でデータ転送が行われるが，拡張ヘッダが付けられた時は，次世代 IP ヘッダと上

位層のヘッダの間，つまり，次世代IPヘッダとデータ部の間にオプションヘッダを格納して(図3.5参照)データ転送を行う．オプションヘッダが複数付加された場合(図3.6参照)も同様である．

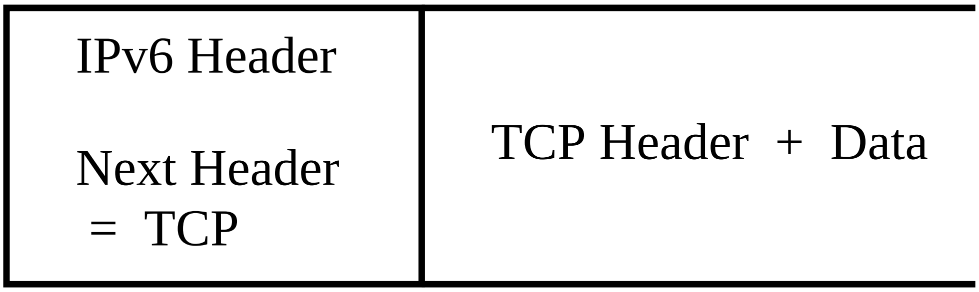

図 3.4: 拡張ヘッダのない次世代IPパケット

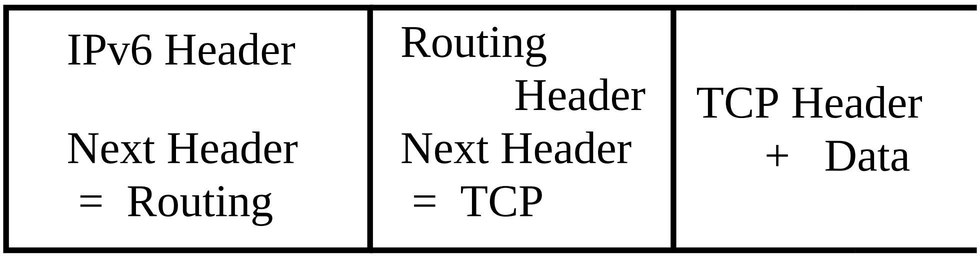

図 3.5: 次世代IP拡張ヘッダの例 (1)

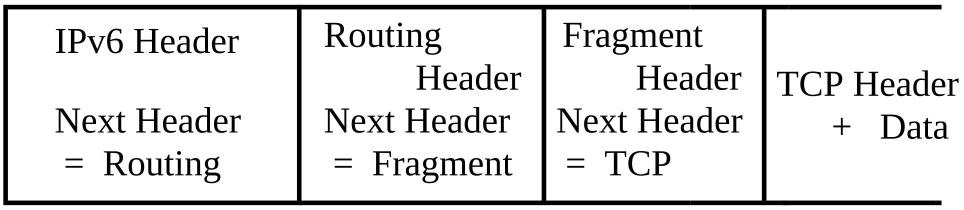

図 3.6: 次世代IP拡張ヘッダの例 (2)

複数のヘッダを指定する場合，他の機能との兼ね合いや，処理の容易さを考慮して，それぞれの拡張ヘッダの順番は次のようでなければならないとされている．

1. 次世代IP header

2. Hop-by-Hop Options header
3. Destination Options header
4. Routing header
5. Fragment header
6. Authentication header
7. Encapsulating Security header
8. Destination Options header
9. upper layer header

3.3.5 アドレスアーキテクチャ

次世代IPのアドレスには，既存のIPと同じ，ユニキャストアドレス (Unicast Address)，マルチキャストアドレス (Multicast Address) の他に，あるグループの適当な1台を示すエニイキャストアドレス (Anycast Address) の3種類が定義されている．ブロードキャストアドレスは，アドレスの種類としては定義されず，特別なマルチキャストアドレスと定義される．

また，IPのクラスと同じように，アドレスの上位数bitでアドレスのタイプを分けて表現できるように定義されている．つまり，アドレスの始めの数bitを解析することで，そのアドレスがどのタイプのアドレスなのか調べることができる．表3.2に，そのアドレスのタイプと意味をまとめる．

Provider-based Addressは，インターネットプロバイダが提供するアドレス空間を示している．NSAP，IPX用のアドレスは，次世代IPのアドレスとNSAP, IPXのアドレス体系との1対1のマッピングを実現するためのものである．

Local Use Addressは，インターネットに接続しないサイトが内部のノードの識別に使用するアドレスである．

3.3.6 ユニキャストアドレス

CIDRを利用しているIPの上でのユニキャストアドレスと構造は同じである．図3.7ではその一例として，インターフェイスアドレスにEthernetのMACアドレスを使う例を考える．

最初の“n” bitの“Subscriber Prefix”と“80-n”の“Subnet ID”はそのリンクに接続されているルータから得られる情報 (Router Advertisement) である．このような仕組みは，移動ホストに対するIPアドレスの自動設定機構などで有効であると考えられている．

表 3.2: アドレスタイプ

識別子	タイプ
0000 0000	Reserved
0000 0001	Unassigned
0000 001	Reserved for NSAP Allocation
0000 010	Reserved for IPX Allocation
0000 011	Unassigned
0000 1	Unassigned
0001	Unassigned
001	Unassigned
010	Provider-Based Unicast Address
011	Unassigned
100	Reserved for Geographical-Based Unicast Addresses
101	Unassigned
110	Unassigned
1110	Unassigned
1111 0	Unassigned
1111 10	Unassigned
1111 110	Unassigned
1111 1110 0	Unassigned
1111 1110 10	Link Local Use Addresses
1111 1110 11	Site Local Use Addresses
1111 1111	Multicast Addresses

3.3.7 エニイキャストアドレス

エニイキャストアドレスは，ユニキャストアドレスと同様に，1 対 1 の通信に使用するアドレスであるが，特定のノードを指定するものではない点が異なっている．エニイキャストアドレスは，同一識別子を持つアドレス群の中の一つのノードを指定する．エニイキャストアドレスは，例えばポリシールーティングで，この組織を通るように Routing Header に含めたり，ネットワークトポロジ的に最も近い，例えば DNS サーバを指定する利用方法が想定されている．

3.3.8 マルチキャストアドレス

次世代 IP のマルチキャストは IP のそれを拡張した形で提供される．マルチキャストアドレスは，図 3.8ようなフォーマットが定義されている．

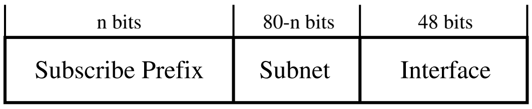

図 3.7: 次世代 IP のユニキャストアドレスの構造

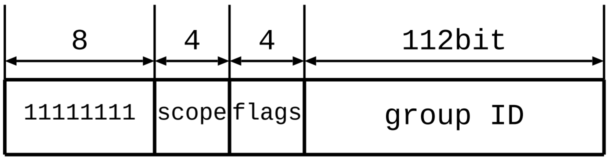

図 3.8: マルチキャストアドレスフォーマット

上位の 8 ビットは先に述べた識別子である．flags は 4 ビットのフィールドで上位 3 ビットは現在はまだ未定義で常に 0 にセットされる．最下位ビットが 0 の場合は，そのマルチキャストアドレスが恒久的に割り当てられたものであること (インターネット全体で ”well-known”であること) を示している．最下位ビットが 1 の場合は，マルチキャストアドレスが一時的なものであることを示している．

scope フィールドは 4 ビットのフィールドでマルチキャストの範囲を指定する．現在規定されている scope フィールドの値は，表 3.3のようなものである．

“Group ID” はその Multicast のグループを意味するもので，それぞれのグループに一つの ID が割り当てられる．

また，このアドレスは如何なる場合であっても，送信者のアドレスや Routing ヘッダの中に含まれてはならないことが決められている．

表 3.3: scope フィールド

0	reserved
1	node-local scope
2	link-local scope
3	(unassigned)
4	(unassigned)
5	site-local scope
6	(unassigned)
7	(unassigned)
8	organization-local scope
9	(unassigned)
A	(unassigned)
B	(unassigned)
C	(unassigned)
D	(unassigned)
E	global scope
F	reserved

第 4 章

次世代インターネットプロトコルの実現: 設計

本章では，BSD UNIX への次世代 IP の実現に向けた設計について述べる．

4.1 目標とする実装

本実装の第一の目標として，既に実装されている他のプロトコル体系，特に既存の IP に対して影響を与えないことがあげられる．これは，IP から次世代 IP への移行期間の配慮で，IP から次世代 IP への移行を緩やかに行うために必要である．なぜなら，移行期間中は IP と次世代 IP のどちらも利用可能であることが望まれているからである．

第二の目標として，既存の IP と同じユーザインターフェイスを提供することが挙げられる．これを実現すると，既存のアプリケーションが少しの変更のみで再利用可能になる．

第三の目標として，新しく加わった機能を既存の方法を利用して実現することである．これは新しい機能に対して，全く新しい手法を用いて実現することによる新たな問題の発生を防ぐためである．

これらの目標を達成するために必要な設計は次節以降述べる．

4.2 既存の IP との共存

まず第一の目標を実現するために，2 つの方法を考案した．一つは，現在の IP のコードには全く変更を加えず，IP とは完全に分けられた次世代 IP 独自のコードを用いる方法である．もう一つは，現在の IP のコードで，次世代 IP でも利用できる部分は利用し，IP と次世代 IP の異なる部分だけ新たなコードを加える方法である．前者は，部分によっては全く同じコードになるなど，無駄が多い半面，既存の IP の機能を損なうこと無く実装が出来る．後者は，既存の IP のコードに手を加える分，IP の機能を損なう危険性があるが，無駄になる部分がない．

今回は以下の理由により，基本的に既存の IP のコードに変更を加えないで，次世代 IP の実装を行う方法を選んだ．

- 構造が単純である
- IP のみと IP と次世代 IP を共存したカーネルの選択が容易
- 次世代 IP に関する新たな機能を実装しやすい
- カーネルの変更に慣れていない人でも容易に変更が可能
- IP の変更があった場合でも，次世代 IP は影響を受けない

つまり，既存の IP とはまったく別の形で実装を行うと，図 4.1のようなプロトコルスタックが，図 4.2のように，IP と次世代 IP が共存しているプロトコルスタックになる．つまり，IP や次世代 IP を直接利用する上位層，下位層からは全く別のプロトコルのように見える．

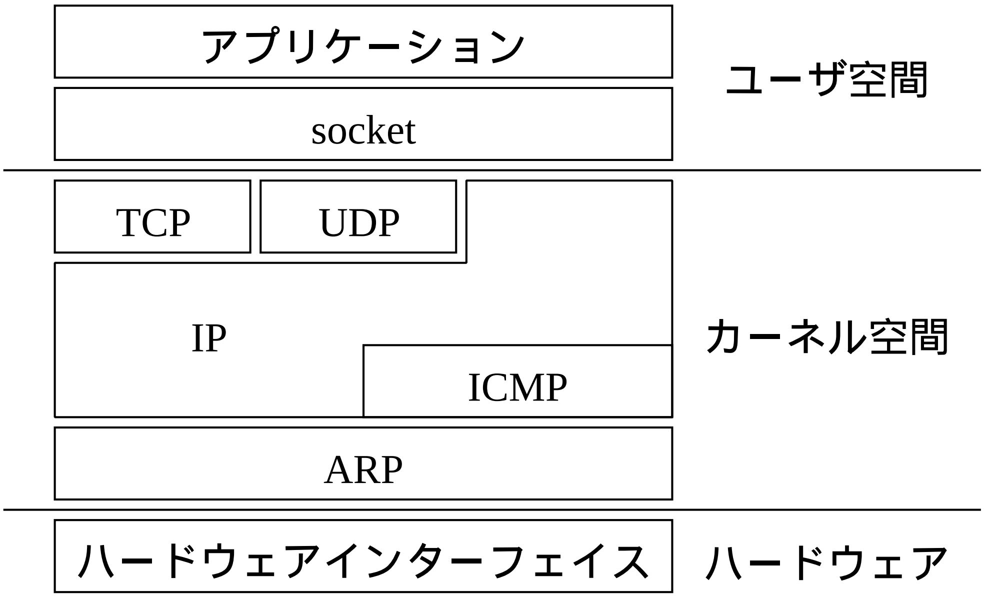

図 4.1: 既存の IP のプロトコル体系

以下では，この方針に基づいた具体的な設計案について述べる．

4.2.1　データの受渡し –ipintrq(割込み行列)–

BSD の IP の実装は，パケットを受け取った場合の処理，つまりデータリンク層からのデータの受渡しに，ipintrq という mbuf のリストの行列を用いている．ネットワーク層にあたる IP は，データリンク層のドライバからソフトウェア割込みを受けると，ipintrq か

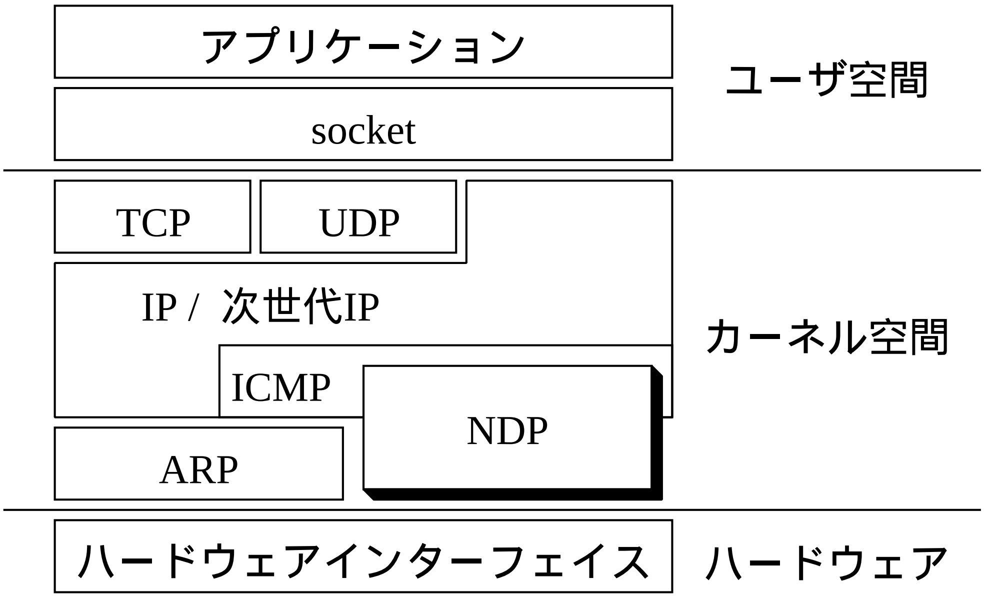

図 4.2: 次世代 IP のプロトコル体系

ら必要な mbuf を受け取る．イーサネットを例にとると，イーサフレームに組み込まれているデータグラムの種類を示す NH(Next Header) フィールドに，IP を示す 16 進数表記の 0x0800 という値が入っている．ソフトウェア割込みによって呼ばれた ether_input() という関数でこの部分を評価し，それぞれの適当な行列へ，そのデータリンク層から渡された mbuf を入れる．

表 4.1にパケットを受け取った時に動作する IP のコードを示す．

これに対して，パケットの出力はネットワーク層がデータリンク層の出力関数 (例えば，ether_output() など) を呼び出す時に，後述するアドレスファミリによって，ネットワーク層が何であるかの識別子を判断し適切なデータリンクフレームを作りそれを ipintrq に渡す，という流れになる．

表 4.2に，IP がアドレスファミリによって，それぞれの処理関数を呼び出す部分を示す．

次世代 IP では，イーサフレームの NH フィールドを 0x86dd に，また新しく次世代 IP を示すアドレスファミリを定義することが決められている．今回の実装ではこれらフィールドと，次世代 IP を処理するための行列を新たに定義することで，IP と次世代 IP のパケットを区別して処理する方法を提供する．

4.3 ユーザインターフェイス

第二の目標である，IP と同じユーザインターフェイスの提供を実現のために，BSD で広く利用されている socket インターフェイスを拡張する方法をとることにする．

4.3.1 socket インターフェイス

ユーザは，カーネルからシステムコールと言う形でネットワークの機能を提供されている．BSD UNIX では，socket() と呼ばれるインターフェイスを提供しており，多くのアプリケーションではこれを利用して実装が行われている．

UNIX は，全てがバイトストリーム，すなわちファイルであるという抽象化のもとで設計されている．例えば，ハードウェアは全て/dev 以下にあるデバイスファイルによって抽象化がなされており，このファイルに対してアクセスすることが，ハードウェアにアクセスすることになる．

全てをファイルにするという抽象化は，一方でカーネルの機能に対する単一的な方法でのユーザインターフェイスを提供している．例えば，open() システムコールによって作られるファイルへのアクセスポイント，つまりファイルディスクリプタを，ファイルにアクセスするのと同じようにネットワーク，特にインターネットを利用する場合，利用することができる．この時の open() にあたるシステムコールを socket() といい，これによって，ネットワークへのアクセスはファイルと同じように扱うことができる．

ネットワーク利用のための socket() に代表されるシステムコール群をまとめて socket インターフェイスと言う．次世代 IP でもこれと同じ方法を用いて，ユーザに対するインターフェイスを提供する．

4.3.2 アドレスファミリ

socket() では利用するネットワークの種類を指定することが必要で，これにより，カーネル内のどの部分を利用するかが決められる．このネットワークの種類を表すのがアドレスファミリである．通常，インターネットの場合は AF_INET を指定する．もし，新たにネットワーク機能の種類が増えると，それを示す新たなアドレスファミリを定義しなければならない．今回の実装では，次世代 IP のための新しいアドレスファミリを定義する．

4.3.3 プロトコル制御ブロック (PCB)

PCB とはユーザが利用する socket インターフェイスに関する情報を保持するための構造体である．例えば，接続している相手ホストの IP アドレス，上位層のプロトコル，ポート番号などが，この構造体に格納されている．表 4.3 に IP の PCB を示す．

BSD の実装では，この情報によって IP ヘッダを生成するため，ヘッダフォーマットが IP とは異なる次世代 IP では全く異なる PCB が必要になる．例えば，“Flow ID” や “Priority” などは，新たに定義しなくてはならない．

4.3.4 ライブラリ関数

IP を利用したアプリケーション作成のため，システムコールとは別に，ライブラリ関数で提供されている機能がある．例えば，gethostbyname() などに代表されるアドレスを扱う関数群である．

今回の実装では，第一の目標を考慮して，これらを新しく定義しIPのそれらとは区別する方法で行う．

4.4 新たな機能の実現

第三の目標を実現するために，IPと次世代IPの相違点から新たに実現しなければならない機能について述べる．

4.4.1 拡張ヘッダ

IPと次世代IPの最も大きな相違点はオプションの取り扱いである．IPではヘッダのすぐ後ろにオプションが位置し可変長の領域が割り当てられるが，次世代IPでは固定長のオプションヘッダが，いわゆる“数珠つなぎ”に次世代IPヘッダの後ろに位置する．

IPでは，一旦mbufに入ったパケットを，ヘッダ部，オプション部，データ部に分けて取り扱っている．この時，オプション部は，ヘッダ部の後ろから始まり，ヘッダ長と全体の長さを使って計算した大きさをオプション部としている．

次世代IPではオプションは全てオプションヘッダとして扱われるため，一つ前のヘッダのNHフィールドの値によってどのオプションヘッダを扱うのか判断しなくてはならない．

今回の実装では，この「ヘッダの後にヘッダが位置する」という考え方に着目し，IPの上位層の処理の方法であるプロトコルスイッチを利用する方法を考えた．IPでは，上位層であるTCPやUDPを処理する部分へデータを渡す時，IPのNHフィールドと，関数へのポインタの構造体配列であるプロトコルスイッチを利用して処理している．

次世代IPでもこの方法に踏襲し，新たに次世代IPのためのプロトコルスイッチを定義し，それぞれのヘッダの処理関数を呼び出すことにする．

これにより，第三の目標である，現在のコードを利用して新しい機能を実現することができる．

表4.4にプロトコルスイッチ構造体の定義を，また，例示として表4.5に，IPのプロトコルスイッチを示す．

4.4.2 ND(隣接ホスト探索) プロトコル

物理的に同一なネットワークでは，データリンク層で2点間の通信を行っている．このために，多くの場合データリンク層にもアドレスが必要がある．また，ネットワーク層がデータリンク層を利用して通信を行うためには，ネットワーク層のアドレスとそれによって，次の送るべき相手ホストのデータリンク層のアドレスを知る必要がある．

IPではARP(Adrees Resolution Protocol)による，データリンク層のアドレス取得を行っているが，次世代IPではARPをそのまま用いずに新たにNDP(Neighbor Discovery Protocol)というプロトコルを用いる．これは，128ビットの次世代IPのアドレスを，決まった先頭ビットパターン(0xfe80)とデータリンク層のアドレスで決定し，ネットワーク

層で隣接ホストのデータリンク層のアドレスを取得するものである．この NDP はネットワーク層の ICMP の機能の一部として定義されている．また，この時のアドレスをリンクローカルアドレスといい，このアドレスは，ネットワークインターフェイスに必ず持たなければならないアドレスとして決められている．この考え方は特に新しいものではなく，OSI の CLNP の ES-IS などがこれと良く似た方法で，隣接ホスト間の通信を提供している．

NDP を実現するために，二つの方法を考えた．

- 完全に新しいモジュールを設計する
- 既存の ARP や CLNP の ES-IS などのモジュールを派生させる

今回の実装では，この NDP に関しては完全に独自に設計することにした．その理由は以下の通りである．

- ARP は完全にデータリンク層での実装であるため，ネットワーク層へ移植するのは手間が大きい
- CLNP の実装が手近に存在しなかった
- 仕様が非常に細かく決められており，状態遷移もはっきり定義できるため，独自に実装を行ってもそれほど大きな手間にはならない

理由の最後にも述べたが，仕様がきめ細かく決められているため，NDP のモジュールを設計するにあたり，図 4.3のような状態遷移図を作り，モジュールを定義した．

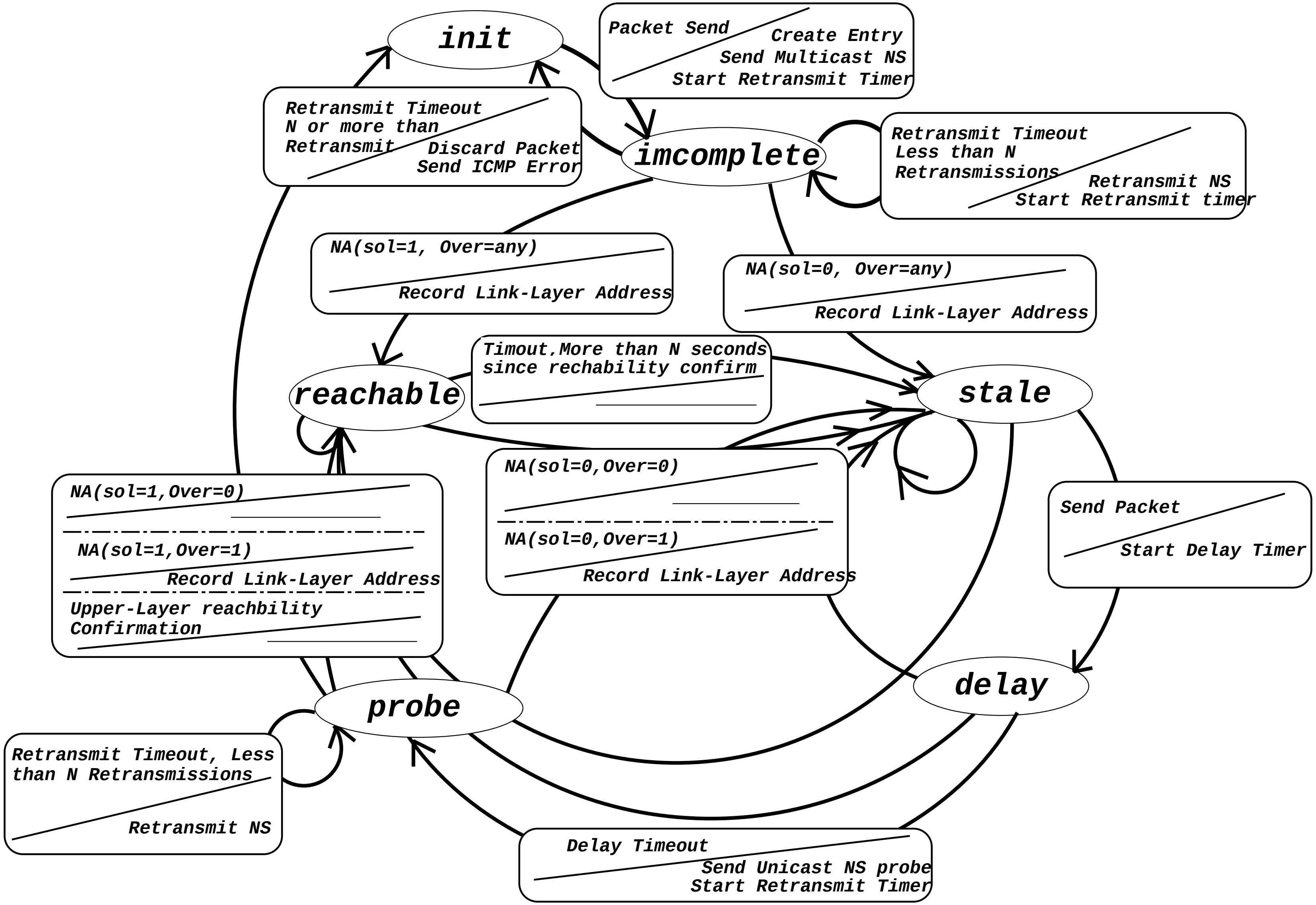

図 4.3: ND プロトコルの状態遷移図

表 4.1: それぞれのタイプと代入される行列

```
/* /sys/net/if_ethersubr.c: ether_input() */

 switch (eh->ether_type) {

 case ETHERTYPE_IP:
       schednetisr(NETISR_IP);
       inq = &ipintrq;
       break;

  case ETHERTYPE_ARP:
       schednetisr(NETISR_ARP);
       inq = &arpintrq;
       break;

  case ETHERTYPE_NS:
        schednetisr(NETISR_NS);
        inq = &nsintrq;
        break;

/* broadcast に対する処理など... */

  s = splimp();
  if (IF_QFULL(inq)) {
          IF_DROP(inq);
          m_freem(m);
  } else
          IF_ENQUEUE(inq, m);
  splx(s);
```

表 4.2: AF_INET の場合

```
   /* /sys/net/if_ethersubr.c: ether_output() */

    switch (dst->sa_family) {

    case AF_INET:
          if (!arpresolve(ac, rt, m, dst, edst))
                  return (0);  /* ARP でアドレス解決出来ない */
          if ((m->m_flags & M_BCAST) && (ifp->if_flags & IFF_SIMPLEX))
                   mcopy = m_copy(m, 0, (int)M_COPYALL);
          off = m->m_pkthdr.len - m->m_len;
          type = htons(ETHERTYPE_IP);
          break;
     /* 他のアドレスファミリに関する記述 */
     }
     eh = mtod(m, struct ether_header *); /* mbuf からポインタへ */
     bcopy((caddr_t)&type, (caddr_t)&eh->ether_type,
          sizeof(eh->ether_type));   /* type に入っている値を代入 */
     bcopy((caddr_t)edst, (caddr_t)eh->ether_dhost, sizeof (edst));
/* arpresolve() で検索した相手の MAC アドレスを代入 */
     bcopy((caddr_t)ac->ac_enaddr, (caddr_t)eh->ether_shost,
         sizeof(eh->ether_shost));

     s = splimp();
     /*
      * Queue message on interface, and start output if interface
      * not yet active.
      */
     if (IF_QFULL(&ifp->if_snd)) {
             IF_DROP(&ifp->if_snd);
             splx(s);
             senderr(ENOBUFS);
     }
     IF_ENQUEUE(&ifp->if_snd, m);
     if ((ifp->if_flags & IFF_OACTIVE) == 0)
             (*ifp->if_start)(ifp);
     splx(s);
     ifp->if_obytes += len + sizeof (struct ether_header);
     if (m->m_flags & M_MCAST)
             ifp->if_omcasts++;
     return (error);
```

表4.3: IPのプロトコル制御ブロック

```
/* /sys/netinet/in_pcb.h */
struct inpcb {
    struct  inpcb *inp_next,*inp_prev; /* PCBリスト */
    struct  inpcb *inp_head;        /* PCBの先頭 */
    struct  in_addr inp_faddr;      /* 接続先ホストのアドレス */
    u_short inp_fport;              /* 接続先ホストのポート */
    struct  in_addr inp_laddr;      /* 接続元ホストのアドレス */
    u_short inp_lport;              /* 接続元ホストのポート */
    struct  socket *inp_socket;     /* ソケットへのポインタ */
    caddr_t inp_ppcb;         /* 上位層プロトコルのPCBへのポインタ */
    struct  route inp_route;        /* 経路情報 */
    int     inp_flags;              /* IP フラグ */
    struct  ip inp_ip;              /* ヘッダのプロトタイプ */
    struct  mbuf *inp_options;      /* IP オプション */
    struct  ip_moptions *inp_moptions; /* IP マルチキャスト オプション */
};
```

表 4.4: protosw 構造体の定義

```
/* /sys/sys/ptotosw.h */
struct protosw {
        short   pr_type;   /* プロトコルファミリ（アドレスファミリ）*/
        struct  domain *pr_domain; /* プロトコルメンバーのドメイン */
        short   pr_protocol;
        short   pr_flags;
        void    (*pr_input)();  /* プロトコルの入力関数 */
        int     (*pr_output)(); /* プロトコルの出力関数 */
        void    (*pr_ctlinput)();   /* プロトコルの入力制御関数 */
        int     (*pr_ctloutput)();  /* プロトコルの出力制御関数 */
        int     (*pr_usrreq)(); /* ユーザからの ioctl() などの処理関数 */
        void    (*pr_init)();       /* 初期化の関数 */
        void    (*pr_fasttimo)();   /* タイムアウト（200ms）*/
        void    (*pr_slowtimo)();   /* タイムアウト（500ms）*/
        void    (*pr_drain)();    /* 全消去 */
        int     (*pr_sysctl)();   /* sysctl 処理 */
};
```

表 4.5: AF_INET のプロトコルスイッチ: inetsw[]

```
/* /sys/netinet/in_proto.c */

struct protosw inetsw[] = {
{ 0,            &inetdomain,    0,              0,
  0,            ip_output,      0,              0,
  0,
  ip_init,      0,              ip_slowtimo,    ip_drain, ip_sysctl
},
{ SOCK_DGRAM,   &inetdomain,    IPPROTO_UDP,    PR_ATOMIC|PR_ADDR,
  udp_input,    0,              udp_ctlinput,   ip_ctloutput,
  udp_usrreq,
  udp_init,     0,              0,              0,  udp_sysctl
},
{ SOCK_STREAM,  &inetdomain,    IPPROTO_TCP,
PR_CONNREQUIRED|PR_WANTRCVD,
  tcp_input,    0,              tcp_ctlinput,   tcp_ctloutput,
  tcp_usrreq,
  tcp_init,     tcp_fasttimo,   tcp_slowtimo,   tcp_drain,
},
 /* 以下省略 */
```

第 5 章

次世代インターネットプロトコルの実現: 実装

本章では，次世代インターネットプロトコルを実装について述べる．

5.1 実装

5.1.1 環境

実装は IBM 互換機 (Pentium100MHz/32MB メモリ) 上で，BSDI 社の BSD/OS Version 2.0.1 を用いて行った．この OS は 4.4BSD-Lite ベースの BSD UNIX で，ソースコードは有料だが誰でも入手できる．

5.1.2 目標

今回の実装では基本的な次世代 IP の機能を実現することを目標とした．具体的には，以下の通りである．

1. socket() によるエンドポイントの生成
2. インターフェイスにアドレス設定
3. リンクローカルアドレスの自動設定
4. 経路情報の付加，削除
5. 次世代 IP パケットの送信，受信
6. ICMP パケットの送信，受信
7. データリンク層のアドレスの探索

8. データリンク層のアドレスの広告
9. 同一物理ネットワークに接続されているホスト間通信
10. 経路情報を参照した同一物理ネットワーク外のホスト間通信
11. UDP/TCP による通信

基本的な機能の実現を第一に考え，拡張ヘッダの具体的な処理 (例えば，ルーティングヘッダの処理やデータの暗号化など) や，ルータとしての機能 (RA(Router Advertisement) や Flow ラベルによる経路制御) などは行わなかった．

5.2 ユーザインタフェイス

次世代 IP に対応した socket インターフェイスの実装を行った．基本的に socket インターフェイスの動作は，プロトコルスイッチで指示される pr_usrreq 関数 (例えば，udp_usrreq() など) が実際の処理を行うので，アドレスファミリを定義し，カーネル内にそのアドレスファミリによって指示される関数群，すなわちプロトコルスイッチに記述される関数群を定義すれば，他に変更するべき点はない．

次世代 IP を示すアドレスファミリは，“AF_INET6” とし表 5.1のように定義した．

表 5.1: 次世代 IP を表すアドレスファミリの定義

```
/* /sys/sys/socket.h */
#define AF_INET6        24              /* Internet Protocol version 6 */
```

プロトコルスイッチの中に記述する関数は，ネットワーク層，トランスポート層の実装によるが，今回の実装では表 5.2のような “inet6sw” を定義した．

5.2.1 PCB

socket インターフェイスを利用する時に，socket() 関数の第 2 引数にタイプを記述する．アドレスファミリが AF_INET の場合，つまり TCP/IP によるネットワークの利用の場合，このタイプには，“SOCK_STREAM”，“SOCK_DGRAM”，“SOCK_RAW” の 3 種類から選択することになる．それぞれ “SOCK_STREAM” は TCP に，“SOCK_DGRAM” は UDP，そして “SOCK_RAW” は IP に対応し，それぞれを操作するために使われる．このタイプに応じて，例えば “SOCK_STREAM” で TCP を利用する場合，カーネル内で tcp_attach()

が呼ばれ IP のプロトコルスタックの利用のための PCB である“ inpcb” 構造体が初期化される．

この PCB には，socket インターフェイスに必要な情報，例えば，通信相手のアドレス，経路情報などが含まれる．この PCB は通常新しいプロトコルが定義されると，新しく定義するのだが今回の実装では，IP と次世代 IP を共用している PCB を定義し，IP と次世代 IP で同じ PCB を共用することにした．これは，socket インターフェイスのオプションに，IP と次世代 IP のそれぞれの socket が，相反する socket に変化させることができるように定義されているためである．それを実現するために，それぞれの PCB を定義せず，一つにまとめて，フラグを利用して，その socket がどちらのものなのかを定義し，簡単に切り替えが出来るようにした．

表 5.3に，今回の実装で用いた，IP と次世代 IP が共用している PCB を示す．

5.3 NDP

NDP は第 4章の最後でも述べた通り，次世代 IP の最も斬新で重要な機能である．言い替えれば，これまでの IP とは全く異なる部分である．そのために今回は，図 4.3を基に全く独自に実装を行った．

NDP によって扱われるメッセージは，NS(Neighbor Solicitation)，NA(Neighbor Advertisement)，RS(Router Solicitation)，RA(Router Advertisement) の 4 つである．図 4.3にはそれぞれのメッセージがどの状態で送信されるかが記述されている．

実装ではまずこの 4 つのメッセージの入力，出力する関数を定義した．だが，ルータとして処理しなければならない関数，つまり RA の出力関数と RS の入力関数は，今回の実装の範囲ではないのですべてダミーの関数として定義してある．

5.3.1 関数の呼び出し

4 つの基本的なメッセージを扱う関数のうち，パケットを受け取る側，すなわち，関数名に“input” が含まれる関数は，ICMP パケットとしてネットワーク層の“ipv6_intr()”，“ipv6_input()” を経由して“ipv6_icmp_input()” で ICMP のヘッダタイプに合わせて，適切に呼び出される．

出力側の関数は，例えばネットワークに接続された直後にルータを探すために RS を送る場合や同一物理ネットワーク上のホストの MAC アドレスを探すために使われる．これからも分かるように，出力関数を統一的に扱うことは非常に難しい．そのため，今回の実装ではこれらの関数を，それぞれ必要なところから呼び出すように実装した．

表 5.6は，次世代 IP に対応した“ether_output()” を示す．この中で，“ipv6_discov_resolve()” が呼ばれているが，この関数から“ipv6_ns_output()” が呼ばれている．

5.4 ネットワーク層の処理

5.4.1 ipv6initrq(割り込み行列)

ネットワーク層，すなわち次世代IPに関する処理を行う部分の流れは基本的にIPのそれと同じである．表4.1のように，データリンク層からのデータの受渡しを行うために，IPはipintrqというmbufのリストの行列を使っている．次世代IPでは，新たにipv6intrqという行列を用い，データリンク層からのデータのやりとりを行うことにした．

IPではこの行列は“/sys/netinet/ip_input.c”の中で定義されている．

5.4.2 入力データの受け取り

IPでは，割込みによって呼び出される“ipintr()”で，mbufのリストから必要なデータを取り出し，IPヘッダの処理を行っている．

表5.7は，“ipintr()”の処理の流れを大まかにまとめたものである．

次世代IPでも同様に“ipv6intr()”で必要なデータを取り出す．だが，全ての処理をここで行うのではなく，mbufからの受渡しのみをこの関数で行う．基本的なパケットの処理は新たに“ipv6_input()”という新しい関数を定義し，この関数内で行うことにした．これによって，モジュール間の役割分担をはっきりさせることができただけでなく，カプセル化された次世代IPパケットを，個別に処理させるときに，そのまま流用できるという利点がある．

表5.8に，“ipv6intr()”を示す．

5.4.3 出力データの受渡し

IPでは，アドレスファミリとプロトコルスイッチによって，上位層からIPパケットの出力関数である“ip_output()”が呼び出される．この関数では，socketインターフェイスによって定義された相手ホストのアドレスなどの情報を基に，IPヘッダの生成を行い，データ部と共に下位層の出力関数(インターフェイスによる)に渡される．

今回の実装でもほぼ同じ方法で行った．次世代IPの出力関数として“ipv6_output()”という関数を定義し，ヘッダの生成と，オプションヘッダを生成する関数の呼び出しを行うことにした．

オプションヘッダは上位層ヘッダと次世代IPヘッダの間に位置するため，上位層の出力関数からオプションヘッダを呼び出す関数を呼び出し，必要なオプションの呼び出しが終った時点で次世代IPの出力関数を呼び出すのが素直な方法であると言える．だが，今回の実装ではあえてこの方法ではなく，一旦，上位層の出力関数から次世代IPの出力関数を呼び出し，必要なオプションのオプションヘッダを生成する関数群をプロトコルスイッチによって呼び出し，それによって生成されたオプションヘッダを次世代IPヘッダと上位層ヘッダの間に挿入して下位層の出力関数に渡す方法で実装を行った．なぜなら，前者の上

位層からオプションヘッダを生成する関数を呼び出す方法では，上位層に手を加えなければならず，複雑になると予想されるからである．

表5.9に次世代IPの出力関数を示す．

5.5 ライブラリ

IPでは，“gethostbyname()”や“inet_atod()”などと言った，アドレスを処理するためのライブラリ関数が提供されている．次世代IPでも，同様な関数を提供しなければ，ユーザに対してそのサービスの利用を困難にするばかりか，socketインターフェイスの利用手段がなくなるに等しいと言える．

今回の実装では，これらは既存のものとは別に実装する方針をとり，実装を行なった．だが，“gethosybyname()”などに見られる，いわゆる“Domain Name System”のリゾルバの実装は行なっていない．今回は，/etc/hostsに書かれたアドレスのみ解決できるように実装を行なった．

表 5.2: 次世代 IP のプロトコルスイッチ

```
/* /sys/netinet6/in6_proto.c */

struct protosw inet6sw[] = {
  { /* GENERIC/HBH */
    0, &inet6domain,  0, 0,
    ipv6_hop,  ipv6_output, 0, 0,
    0,
    ipv6_init, 0, ipv6_slowtimo, ipv6_drain, ipv6_sysctl
  },
  { /* ICMPv6 */
    SOCK_RAW,  &inet6domain,  NEXTHDR_ICMP,  PR_ATOMIC|PR_ADDR,
    ipv6_icmp_input, ipv6_icmp_output, 0, ripv6_ctloutput,
    ipv6_icmp_usrreq,
    0, 0, 0, 0, 0
  },
  { /* UDP */
    SOCK_DGRAM, &inet6domain,  NEXTHDR_UDP,  PR_ATOMIC|PR_ADDR,
    udp_input,  0, udp_ctlinput, ipv6_ctloutput,
    udp_usrreq,
    0, 0, 0, 0, 0
  },
  { /* TCP */
    SOCK_STREAM, &inet6domain, NEXTHDR_TCP,
PR_CONNREQUIRED|PR_WANTRCVD,
    tcp_input,  0, tcp_ctlinput, tcp_ctloutput,
    tcp_usrreq,
    0,  0,  0,  0,  0
  },
  {  /* RAW Socket */
    SOCK_RAW, &inet6domain, 0, PR_ATOMIC|PR_ADDR,
    ripv6_input, ripv6_output, 0, ripv6_ctloutput,
    ripv6_usrreq,
    ripv6_init,0,0,0,0
  },
};
```

表 5.3: IP と次世代 IP の PCB

```
/* /sys/netinet/in_pcb.h */

union inpaddru
{
  struct in_addr6 iau_addr6;
  struct
    {
      u_int8 pad[12];
      struct in_addr inaddr;
    } iau_a4u;
};

struct inpcb
{
  struct inpcb *inp_next, *inp_prev;
  struct inpcb *inp_head;
  struct socket *inp_socket;

  union inpaddru inp_faddru;
  union inpaddru inp_laddru;
  caddr_t inp_ppcb;
  union{
      struct route ru_route;
      struct route6 ru_route6;
  } inp_ru;
  union {
      struct ip hu_ip;
      struct ipv6 hu_ipv6;
  } inp_hu;
  struct mbuf *inp_options;
  union {
      struct ip_moptions *mou_mo;
      struct ipv6_moptions *mou_mo6;
  } inp_mou;
  int inp_flags;
  u_short inp_fport;
  u_short inp_lport;
};
```

表 5.4: メッセージを扱う関数群

NS	ipv6_ns_input(),ipv6_ns_output()
NA	ipv6_na_input(),ipv6_na_output()
RS	ipv6_rs_input(),ipv6_rs_output()
RA	ipv6_ra_input(),ipv6_ra_output()

表 5.5: ICMP メッセージとして処理される NDP

```
/* /sys/netinet6/ipv6_icmp.c: ipv6_icmp_input() */

    /*
     * IPv6 discovery messages.
     */
    case ICMPV6_ROUTERSOL:
      ipv6_rs_input(incoming_mbuf, ext, index, off);
      break;
    case ICMPV6_ROUTERADV:
      ipv6_ra_input(incoming_mbuf, ext, index, off);
      break;
    case ICMPV6_NEIGHBORSOL:
      ipv6_ns_input(incoming_mbuf, ext, index, off);
      break;
    case ICMPV6_NEIGHBORADV:
      ipv6_na_input(incoming_mbuf, ext, index, off);
      break;
```

表 5.6: 出力関数の例

```
switch (dst->sa_family) {

case AF_INET:
if (!arpresolve(ac, rt, m, dst, edst))
return (0);
if ((m->m_flags & M_BCAST) && (ifp->if_flags & IFF_SIMPLEX))
mcopy = m_copy(m, 0, (int)M_COPYALL);
off = m->m_pkthdr.len - m->m_len;
type = htons(ETHERTYPE_IP);
break;

        case AF_INET6:
                if (m->m_flags & M_MCAST)
                    ETHER_MAP_IN6_MULTICAST((
(struct sockaddr_in6 *)dst)->sin6_addr,edst);
                else {
                    if (!ipv6_discov_resolve(ifp, rt, m, dst, edst))
                      return 0;
                }
                type = htons(ETHERTYPE_IPV6);
                break;
```

表 5.7: IP パケットの処理 (ipintr())

```
void
ipintr()
{
/* 変数の定義 */
next:
s = splimp();
IF_DEQUEUE(&ipintrq, m);
splx(s);
if (m == 0)
return;
if (in_ifaddr == NULL)
goto bad;
if (m->m_len < sizeof (struct ip) &&
    (m = m_pullup(m, sizeof (struct ip))) == 0) {
goto next;
}

ip = mtod(m, struct ip *); /* mbuf に入ってるデータのポインタ */

/* 途中略  */
/* ヘッダの処理，アドレスの評価  */
/* マルチキャスト，ブロードキャスト，経路制御の処理  */

(*inetsw[ip_protox[ip->ip_p]].pr_input)(m, hlen);
goto next;
bad:
m_freem(m);
goto next;
}
```

表 5.8: 次世代 IP パケットの処理 (ipv6intr())

```
void
ipv6intr()
{
  struct mbuf *m;
  int s;

  while(1) {
      s = splimp();
      IF_DEQUEUE(&ipv6intrq, m);
      splx(s);

      if (m == NULL)
return;
      ipv6_input(m,0,NULL,0);
    }
}
```

表 5.9: ipv6_output() の動作の流れ

```
int
ipv6_output(outgoing,ro,flags,i6mo)
     struct mbuf *outgoing;
     struct route6 *ro;
     int flags;
     struct ipv6_moptions *i6mo;
{
  struct ipv6 *header;
  struct route6 ipv6route;
  struct sockaddr_in6 *dst, secsrc;
  struct in6_ifaddr *i6a;
  struct ifnet *ifp;
  int error=0;

  header = mtod(outgoing, struct ipv6 *);

  if ((flags & (IPV6_FORWARDING|IPV6_RAWOUTPUT)) == 0) {
      header->ipv6_versfl = htonl(0x60000000 |
  (header->ipv6_versfl & 0xf0000000));
  }

  /* 経路の決定 */

    if (ro->ro_rt == NULL){
      dst->sin6_family = AF_INET6;
      dst->sin6_len = sizeof(struct sockaddr_in6);
      dst->sin6_addr = header->ipv6_dst;
      dst->sin6_port = 0;
      dst->sin6_flowinfo = 0;
    }

   if (ro->ro_rt && (ro->ro_rt->rt_flags & RTF_TUNNEL))  {
          error = ipv6_tunnel_output(outgoing, dst, ro->ro_rt);
    }
    else error = (*ifp->if_output)(ifp, outgoing,
 (struct sockaddr *)dst, ro->ro_rt);
  /* 失敗した時の処理 */

}
```

第 6 章

まとめ

本論文では，次世代 IP を考察した後，それに基づいた設計を行ない実装を行なった．本章ではこれをまとめる．

6.1　考察 —まとめ—

次世代 IP は非常に単純化を強調したプロトコルである．その単純化は，実際には次世代 IP の実装の場面にまで通用するようなものであり，これまでの経験を非常にうまく活かせた仕様になっていると言える．例えば，これまでの ARP をやめ NDP という新しいプロトコルをデータリンク層のアドレス解決に用いることは第 4.4.2節で述べた．実際にこのプロトコルは，データリンク層のアドレス解決のみでなく，これまでの IP では実現が難しいと言われて来た，ホストやネットワークの自動設定を行なう機能や，IP では，実装によって解決が図られて来た，ホストやルータとの接続の確認などが行なわれる．これは，データリンク層とネットワーク層の関係が，それぞれの機能を行なうのに不可欠であるため，これまでの ARP と置き換えられたのである．言い替えるならば，この単純化によって得られた利益は，新たな機能や問題点の解決にあてられているのである．つまり，これによってアドレス空間が拡大しても既存の IP と同じくらいの負荷で，OS 自身には負荷にならない，新しい IP が実現できるのである．

6.2　設計 —まとめ—

次世代 IP の利点を実装に反映すべく，単純化されたプロトコルをどのようにして実現するか，設計を行なった．その結果，プロトコルスイッチを用いた，拡張ヘッダの処理の実現方法であるとか，PCB の共用化などの実現方法を提案することができた．

また，ユーザからの視点を取り入れ，ユーザインターフェイスの統一化や，新たなライブラリ関数の定義なども行ない，これまでのアプリケーションプログラムなどの資産を，十分に再利用できる環境を設計することができた．

6.3 実装 —まとめ—

本研究の最終目的である，次世代 IP の実装について，以下の点でその目標が達成された．

- 最も重要でかつ新しい機能である NDP の状態遷移に基づいた実装
- 既存の機能を再利用した，新しいプロトコルの実装
- ユーザインターフェイスを統一した実装
- 機能毎のモジュール性の高い実装

6.4 今後の課題

本研究で行なった実装は，次世代 IP の仕様のまだほんの一部分に過ぎず，完全な実装とはとても言えないものである．特に，インターネットで解決が叫ばれている，データグラムのセキュリティについて，次世代 IP はそれを行なおうとしている．また，次世代 IP のプロトコル自身も，実はまだ無駄が多いという指摘もある．

今後は，今回行なわなかった実装を仕上げ，新たな実装へ移るだけではなく，プロトコル仕様の設計や検証について行なわなければならない．また，今回行なった実装について，然るべき方法で公開を行ない，今後の研究の土台として維持を行なって行くつもりである．

謝辞

本研究を進めるにあたり，御理解と御指導を頂きました慶應義塾大学環境情報学部助教授の徳田英幸博士と同学部助教授の村井純博士に感謝します．また，日頃から絶えず多くの助言を頂きました環境情報学部講師の楠本博之博士，同学部助手の中村修博士，同学部環境情報研究所訪問研究員の植原啓介氏の諸氏に感謝します．

西田佳史氏，冨永明宏氏をはじめとする慶應義塾大学政策・メディア研究科の諸先輩方，慶應義塾大学環境情報学部徳田・村井研究室の諸氏には，日頃から様々な形での議論，温かい激励を頂きました．この場を借りてお礼を申し上げます．

また，メーリングリストや2回の合宿について，常に的確なアドバイスと励ましの御言葉を頂きましたWIDEプロジェクト，IPng-WG，ならびに徳田・村井研究室，IPng-WGの皆様に感謝します．

最後に，本稿を書き上げるに至り，根気良くアドバイスを頂きました慶應義塾大学政策・メディア研究科の新美誠氏，重近範行氏に感謝の念を表し，謝辞と致します．

本当にありがとうございました．