NAIST-IS-MT9551055

修士論文

ネームサーバとヘッダ変換ゲートウェイを用いた IPv4 と IPv6 の相互通信

角川 宗近

1997 年 2 月 14 日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

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。

角川 宗近

指導教官： 福田 晃 教授
山本 平一 教授
最所 圭三 助教授

ネームサーバとヘッダ変換ゲートウェイを用いた IPv4 と IPv6 の相互通信[*]

角川 宗近

内容梗概

インターネットの急速な成長に伴い，アドレス空間の枯渇が予測されている．この問題を解決するために，従来のインターネット・プロトコルである IPv4 に代わり新しく IPv6 が策定されたが，IPv6 はヘッダ形式が異なるため，従来の IPv4 とは互換性がない．そこで，IPv4 から IPv6 へとインターネット全体が緩やかに移行できるように，デュアル・スタックとカプセル化という二つの技術を用いた移行戦略が採択された．しかし，この戦略だけでは移行期後半での相互接続性が提供できない．本研究では移行をより円滑に進めるために，ネームサーバとヘッダ変換ゲートウェイを用いたトランスレータを提案し実装した．また，同様の相互接続機能を提供する他のシステムと比較し，提唱するトランスレータが高速性，汎用性に優れていることを示す．

キーワード

インターネット，IPv6，移行，ヘッダ変換，ネームサーバ

---

[*] 奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 修士論文, NAIST-IS-MT9551055, 1997 年 2 月 14 日.

Communication between IPv4 and IPv6 through header translate gateway with name server[*]

Munechika Sumikawa

Abstract

As the Internet grows rapidly, the exhaustion of the IPv4 address space became a significant problem. IPv6 was proposed to resolve the problem, however, it does not maintain backward compatibility with IPv4. The Internet community thus adopted dual-stacks and tunneling strategies to migrate from IPv4 to IPv6. But it is expected that IPv4-only hosts cannot communicate IPv6-only hosts in the end of transition phase. This paper proposes a header translator helped with a modified name server to carry out the migration smoothly. It achieves high performance and portability comparing other translators.

Keywords:

Internet, IPv6, transition, header translation, name server

---

[*] Master's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT9551055, February 14, 1997.

目次

図 目 次

表 目 次

1. はじめに

本章ではまず研究の背景として，IP アドレスの枯渇が問題になっており，これを解決するために提案された次世代インターネット・プロトコルの概要を説明する．そして，最後に本研究の目的が，IPv4 から IPv6 への移行をより円滑に進めることであることを述べる．

1.1 研究の背景

現在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) と呼ばれ，1981 年に策定された [1]．これ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．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，インターネットに接続している計算機の数は指数関数的に増大し続けている．その結果 IP アドレスの枯渇が深刻化している．IP アドレスとは，インターネット上の計算機を識別するための整数値で，IPv4 では $2^{32} \simeq 4 \times 10^9$ の空間を持つ．過去の非効率なアドレスの割り当ても伴って，2000 年初頭には IP アドレスは足りなくなると予想されている [2]．この時点で計算機に割り当てるべきアドレスは無くなり，インターネットに新しい計算機を接続するのは不可能になる．

そこで，IPv4 に代わる新しい通信プロトコルとして，Internet Protocol Version 6(以下，IPv6 と略記) が提案された．

1.2 IPv6 の概要

IPv6 は Cisco Systems の Stephen E. Deering 氏と，Ipsilon Networks の Robert M. Hinden 氏によって提案されたネットワーク層の通信プロトコルである．IPv6 の大まかな特徴を以下に述べる．

アドレス空間の拡張

IPv6 ではアドレス空間は，従来の 32 ビットから 128 ビットに拡張された．すなわち，アドレス空間は $2^{96} \simeq 8 \times 10^{28}$ 倍に拡大した．

ヘッダフォーマットの簡略化

IPv6 は IPv4 の優れた設計を踏襲する一方で，10 数年間の運用経験からあまり使われなかった機能を簡略化した．IPv4 のヘッダ形式を図 1に示す．斜線のフィールドが IPv6 では削除された部分，網のフィールドが拡張ヘッダとなった部分である．その結果，ヘッダ長が固定となり，ヘッダ解析による計算機およびルータの負荷が軽減した．また，アドレス長が 4 倍になったにも関わらず，ヘッダ自体の大きさは 2 倍に留まっている．IPv6 のヘッダ形式を図 2に示す．斜線の部分は IPv6 で新しく導入されたフィールドである．

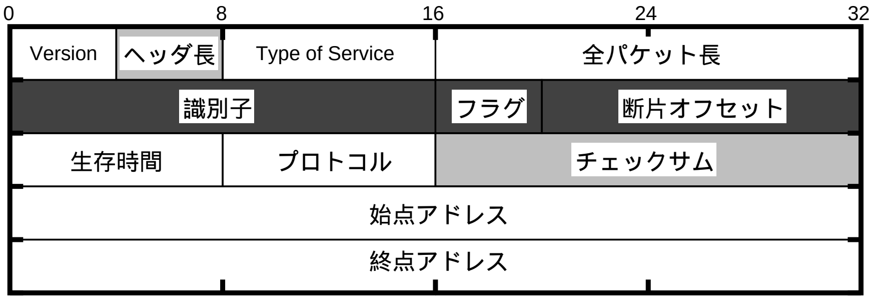


図 1　IPv4 のヘッダ形式

数珠つなぎヘッダ

IPv6 では，経路制御ヘッダや断片ヘッダなどを機能ごとに分割して，それぞれを一つの拡張ヘッダとした．これらの拡張ヘッダを，数珠つなぎに並べることによって，任意のヘッダを挿入できる．この結果，IPv6 では，柔軟なオプションの設定を可能にしている．また，新たな拡張ヘッダを簡単に付加でき，拡張性にも優れている．拡張ヘッダの詳細については，付録Aで述べる．

フローラベルによるパケット転送

IPv6 ではフローラベルという 24 ビットの識別子を設けている (図 2の斜線のフィールド)．これは，リアルタイム通信などの回線品質を保証するサー

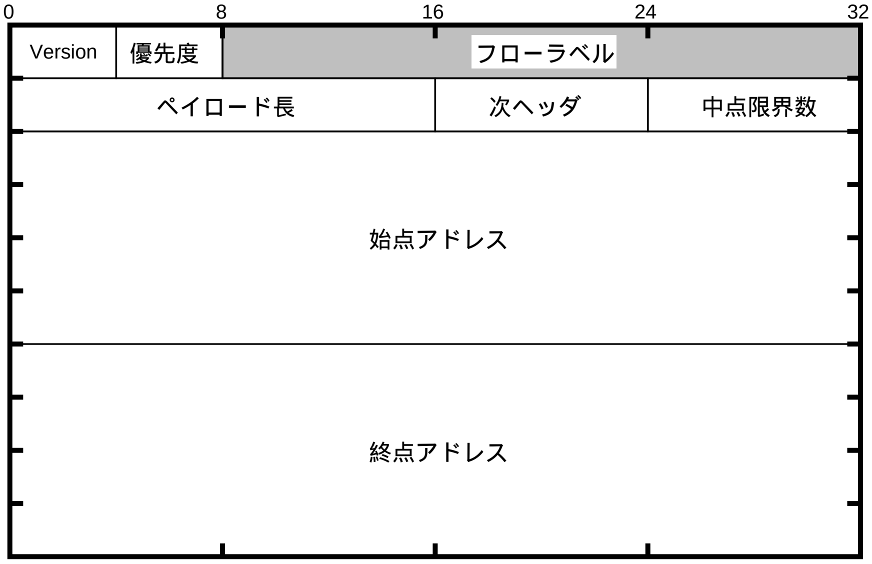


図 2 IPv6 のヘッダ形式

ビスに利用される．

認証と秘匿性の枠組

インターネットの利用環境が研究から商用へ比重が移るにしたがって，通信相手の認証やその内容の保護が重要になってきている．IPv6 では拡張ヘッダの形式で認証や暗号化の枠組を提供する [3][4][5] ．

プラグ&プレイ

インターネットをより簡単に使用するために，ネットワークに対する知識がなくても計算機をインターネットに接続できる必要がある．IPv6 ではアドレスなどのインターネットに接続するための情報が，自動的に設定されるプラグ&プレイの機能を持つ．

1.3 本研究の目的

IPv6に関する提案はその大部分がRFCとして標準化され，IPv4からの移行が始まりつつある．しかし，インターネットに接続している計算機の数は1997年1月現在で1600万台[1]にも及ぶ．これらの計算機が一斉にIPv6に対応することはできない．実際には，かなり長い期間をもって移行が進むことになろだろう．これは長期間，IPv4とIPv6の計算機が混在することを意味する．そこで，移行を円滑に進めるためにはIPv4とIPv6の混在環境下で両者の計算機が相互に通信できることが望まれている．しかし，第1.2節で述べたように，IPv4とIPv6はヘッダ形式が異なり互換性がないため，なんらかの相互接続技術が必要となる．

IETF(Internet Engineering Task Force)の次世代移行分科会はデュアル・スタックとトンネリングを用いた移行戦略を採択した[6]．しかし，この方法では，特に移行期後半におけるIPv4とIPv6の相互接続性が確保できないと予想される．

そこで本研究は，IPv4からIPv6への移行をより滑らかに進めることを目的とし，ヘッダ変換ゲートウェイとネームサーバを用いたIPv4とIPv6の相互接続技術TOWNS (TranslatOr With Name Server)を提案する．TOWNSは現在の移行戦略を補完する形で動作し，移行をより滑らかに進めることができる．

本論文は以下のような構成になっている．まず，第2章で異種プロトコルの相互接続技術について総合的に解説するとともに，現在の移行戦略の問題点を明らかにする．第3章ではTOWNSの設計目標とその機能について，第4章ではTOWNSの実装について述べる．次に第5章では設計目標が達成されているかを評価し，他のトランスレータとの比較検討を行う．最後に，第6章でまとめと今後の課題を述べる．

なお，本論文では表1の用語を定義し，使用する．

---

[1] Network Wizards調べ．`http://www.nw.com/zone/WWW/report.html`

表 1　用語定義

IPv4 計算機	少なくとも IPv4 の機能をもつ計算機
IPv6 計算機	少なくとも IPv6 の機能をもつ計算機
IPv4 のみの計算機	IPv4 の機能しかもたない計算機
IPv6 のみの計算機	IPv6 の機能しかもたない計算機
デュアル・スタック計算機	IPv4 と IPv6 の両方の機能をもつ計算機

2. 異種プロトコルの相互接続技術

この章では，IPv4 と IPv6 のように異なるネットワーク層のプロトコルを相互接続する方法を整理する．相互接続技術には以下の種類がある．

- ネットワーク層での相互接続技術
  - デュアル・スタック
  - トンネリング
  - ヘッダ変換
- 上位層での相互接続技術

次節以降では，これらの技術について概説し，さらに現在の移行戦略の問題点を明らかにする．

2.1 デュアル・スタック

より一般的な表現では，複数のネットワーク層のプロトコルを扱うことである．例えば，市販されている多くのルータは IPv4 だけではなく，AppleTalk や Netware 等の複数のプロトコルを扱える．

本論文でいうデュアル・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．デュアル・スタック計算機は，IPv4 計算機と

通信する時には IPv4 を用い，IPv6 計算機と通信する際は IPv6 を使用する (図 3)．

デュアル・スタックを用いることで，ローカル・ネットワーク上に IPv4 と IPv6 の計算機が混在していても，それぞれが相互に通信できる．しかし，ネットワーク的に隔離された IPv6 計算機同士，すなわち途中経路に IPv4 ルータしか存在しない IPv6 計算機同士は通信できない．このような状況下では次節のトンネリングを併用して，相互接続性を確保する．

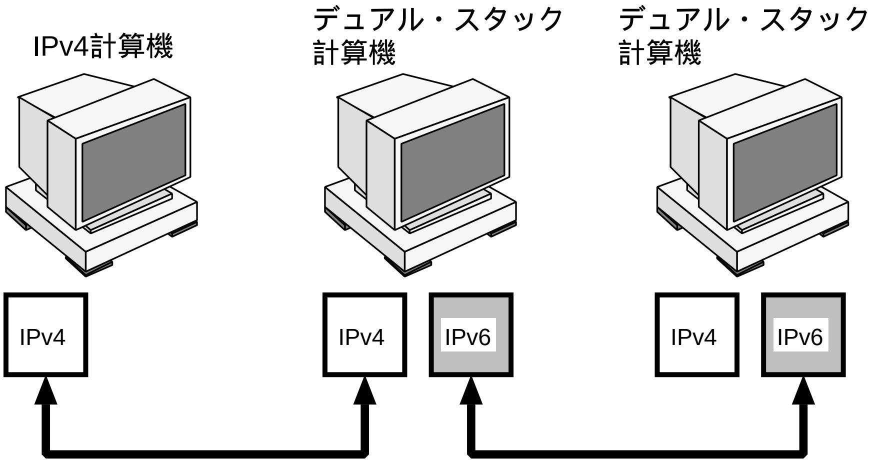


図 3 デュアル・スタックの概念図

2.2 トンネリング

あるネットワーク層のプロトコルを通すために，異なるネットワーク層を利用する技術をトンネリングという．トンネリングを用いれば，IPv6 パケットの前に IPv4 ヘッダを付加することにより，IPv4 ネットワークを利用して IPv6 パケットを転送できる．その結果，ネットワーク的に隔離された IPv6 の計算機同士でも通信が可能になる．(図 4)．

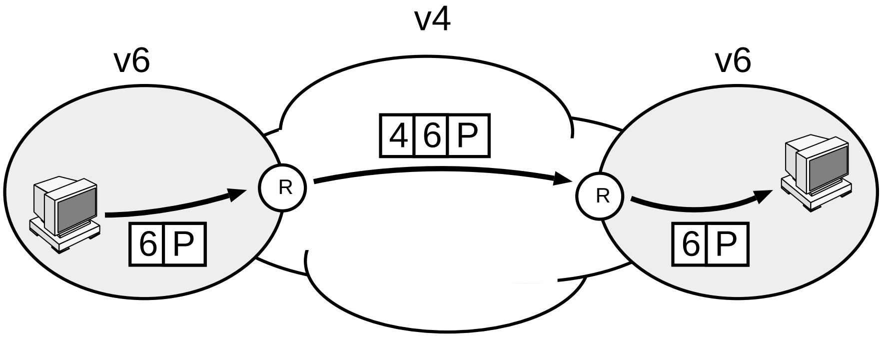


図 4 トンネリングの概念図

2.3 ヘッダ変換

ヘッダ変換は途中経路でパケットそのものを変換してしまう技術である．その例としては，プライベート・アドレスをグローバル・アドレスに変換する NAT[7] がある．これは IPv4 から IPv4 へのヘッダ変換である．

ヘッダ変換を用いると，IPv4 のみの計算機と IPv6 のみの計算機が通信できる (図 5)．しかし，異なるプロトコル同士の完全な 1 対 1 の変換は不可能であり，異なるアドレス空間を持つ IPv4 と IPv6 のアドレスをどう対応づけるかが問題となる．

また，FTP のような上位層でネットワーク層のアドレスを利用するプロトコル[2]はヘッダだけではなく，ペイロード部も変換する必要がある．

2.4 上位層での相互接続技術

途中経路のゲートウェイで，ネットワーク層よりも上位の階層で通信を一旦終端してやり，改めて終点へコネクションを確立することでネットワーク層プロトコルの差異を吸収できる．

終端する層によって以下の 2 つのゲートウェイがある．

- トランポート層ゲートウェイ

---

[2] 本論文ではこのようなプロトコルを “階層侵害のプロトコル” と定義する．

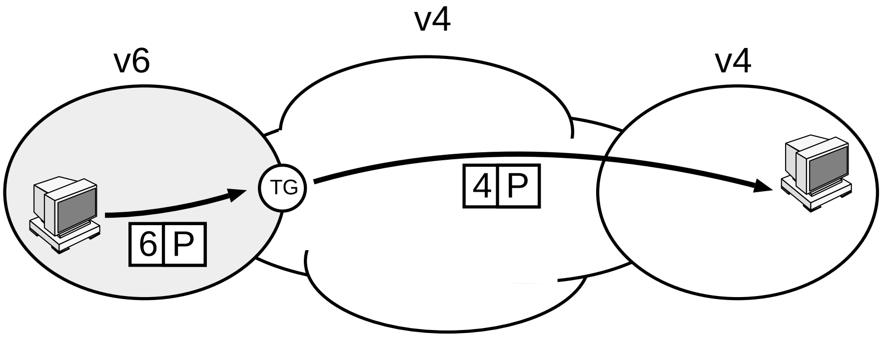


図 5 ヘッダ変換の概念図

- アプリケーション層ゲートウェイ

従来は，相互接続技術としてではなく，SOCKS[8] などの防火壁 (Firewall) を構築するための技術として利用されてきた．

途中経路でコネクションを再接続するため，ヘッダ変換と同様に IPv4 と IPv6 のアドレスをどう対応づけるかが問題である．

2.5 現在の移行戦略とその問題点

IETF(Internet Engineering Task Force) の次世代移行分科会はデュアル・スタックとトンネリングを用いて移行を進めることに決定した [6]．しかし，この 2 つの技術だけで混在環境下であらゆる計算機が相互に通信できるわけではない．

IPv4 と IPv6 の混在環境化では，次の 3 種類の計算機が存在する．

- IPv4 のみの計算機
- デュアル・スタック計算機
- IPv6 のみの計算機

それぞれの計算機が相互に通信すると表 2の組合せが考えられる．デュアル・スタックとトンネリングを併用するだけでは，IPv4 のみの計算機と IPv6 のみの

計算機が通信できないのが分かる．移行期の前半では，IPv6 計算機はデュアル・スタックであることが義務づけてられているので，IPv4 のみの計算機と IPv6 のみの計算機が通信する状況はほとんどありえない．

表 2　相互接続表

	IPv4 のみ	デュアル・スタック	IPv6 のみ
IPv4 のみ	IPv4 で通信	IPv4 で通信	通信不可
デュアル・スタック	IPv4 で通信	IPv6 で通信	IPv6 で通信
IPv6 のみ	通信不可	IPv6 で通信	IPv6 で通信

しかし，移行期の後半では，IPv4 アドレスは既に枯渇してしまっているので，たとえデュアル・スタック計算機であっても IPv4 アドレスを割り当てることはできない．この計算機は IPv6 のみの計算機と同義であり，IPv4 のみの計算機と通信できない．

3. TOWNS の設計

前節で述べたように，現在提案されている方法では，特に移行期後半で円滑な移行ができないと考えられる．そこで，インターネットの移行をより円滑にすすめるため，ヘッダ変換ゲートウェイとネームサーバを用いた IPv4 と IPv6 の相互接続技術 TOWNS (TranslatOr With Name Server) を提案する．これは，第 2.3 節のヘッダ変換を基にした技術である．TOWNS は従来のデュアル・スタックとトンネリングによる移行機構を補完する形で動作する．すなわち，表 2の “通信不可” の部分を通信可能にする技術である．ヘッダ変換の際，変換する IPv4 と IPv6 の始点，終点アドレスをどう対応させるかが問題になるが，この問題を改良を加えたネームサーバを用いることで解決する．

この章では，まず TOWNS が対象とするインターネットのモデルと，TOWNS を設計するにあたっての目標を示す．次に，IPv4 と IPv6 のアドレスを対応づけ

る方法を示し，具体的なTOWNSを利用した通信手順を解説する．最後にヘッダ変換の詳細について述べる．

3.1 TOWNSの対象モデル

TOWNSは移行期の後半にIPv4とIPv6の相互接続性を確保することを狙いとしている．この時期にはバックボーンを始めとするインターネットの大部分はIPv6に対応し，一部のサイト[3]のみがまだIPv4のまま残っている．IPv4アドレスは枯渇し，IPv6計算機の大部分はIPv6の機能しか有してない．

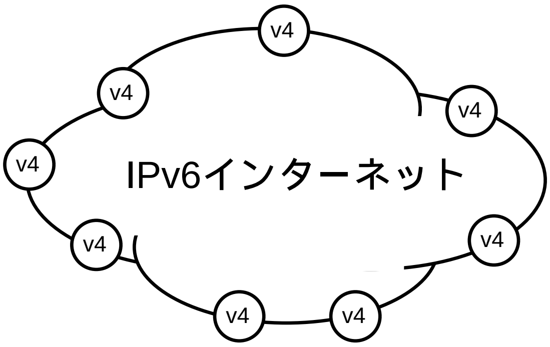


図6 TOWNSの対象モデル

この対象モデル内の一つのIPv4サイトがTOWNSを導入することによって，IPv6インターネット全体との相互接続性を確保できる．つまり，このIPv4サイト内の全計算機は，透過的にサイト外のIPv6計算機と通信できるようになる．

3.2 TOWNSの設計目標

TOWNSの設計にあたって次の目標をたてた．実際に，それぞれの目標を達成できているかについては第5章で検討する．

高速に動作すること

---

[3] 本論文では「サイト」とは1大学や1企業などある程度の規模のネットワークを指す

ヘッダはサイトと外部インターネットの境界に位置するルータで変換されるので，負荷が集中する．変換と転送のための処理はできる限り高速であることが望ましい．

IPv4 と IPv6 は異なるプロトコルであるので，完璧な 1 対 1 の変換は不可能である．そこで，完全性よりも高速性を追求する．

導入コストが小さいこと

TOWNS を導入するのに，サイト全体を IPv6 化するのと同じ手間がかかっては意味がない．導入コストはできるだけ低く抑えることが望ましい．

3.3 アドレスの対応づけ

IPv4 と IPv6 のヘッダを変換するためには，IPv4 と IPv6 の始点アドレスと終点アドレスが一意に変換できなければならない．しかし，IPv4 に比べて，IPv6 のアドレス空間は著しく大きいため，IPv6 アドレスを一意に IPv4 アドレスに割り当てることは無理である．

そこで，通信したい IPv6 計算機のアドレスを一時的に IPv4 アドレスに割り当てる方法を採用する．TOWNS では設定によって任意の IPv4 アドレスへの割り当てが可能だが，プライベート・アドレス `10.0.0.0/8` の利用を推奨する．これはサイト内で自由に使えるアドレス空間である ([9])．この IPv4 アドレスの割り当てはネームサーバを利用して自動的に行う．クライアントからネームサーバへの問い合わせに応じて，IPv4 アドレスを割り当てていくのである．具体的な通信手順は次節以降で説明する．なお，以降の例では IPv4 アドレスは全てこのプライベート・アドレスを利用するものとする．

逆に IPv4 から IPv6 へ対応づけは後者の方がアドレス空間が広大なので，静的な割り当てが可能である．TOWNS ではサイトが取得した IPv6 アドレスの空間の一部を任意の IPv4 アドレスに割り当てることができる．以降の例では，IPv4 アドレス `1.1.1.1` を IPv6 アドレス `1234::1.1.1.1` に割り当てている．

アドレス以外のヘッダ・フィールドの変換については第 3.5節で述べる．

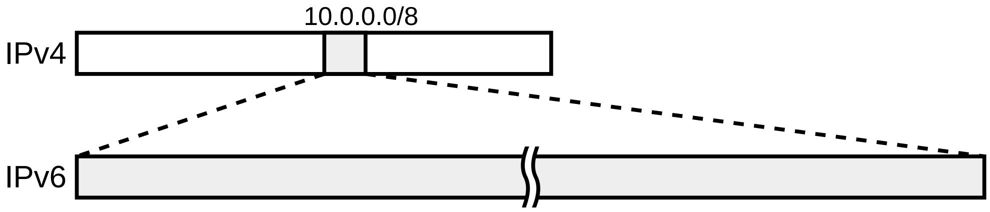


図 7 IPv4 アドレスと IPv6 アドレスのマッピング

3.4 TOWNS の通信手順

本節では最も簡単なモデルとして，境界ゲートウェイ 1 つ，ネームサーバ 1 つのモデルを対象として，TOWNS を利用した通信手順を解説する．

3.4.1 通常の通信

まず，IPv4 計算機同士が通常の環境下で通信するときの流れについて説明する．

図 8のサイト `v4.com` にある IPv4 計算機 A が，インターネットを通じてサイト `v4.ac.jp` にある計算機 B と通信したいとする．A，B のホスト名，IPv4 アドレスはそれぞれ以下の通りである．

	ホスト名	IPv4 アドレス
計算機 A	`a.v4.com`	`1.1.1.1`
計算機 B	`b.v4.ac.jp`	`2.2.2.2`

インターネットの通信において，A は直接 B の IPv4 アドレスを用いて通信することは稀で，実際には B のホスト名を用いて通信する．

A はまず最寄りのネームサーバ NS に `b.v4.ac.jp` の IPv4 アドレスを問い合わせる．問い合わせには，IPv4 アドレスを意味する “A レコード” を用いる．NS は必要ならば外部のネームサーバに問い合わせて[4]，`b.v4.ac.jp` に対応する IPv4 アドレス `2.2.2.2` を A に返す．

---

[4] 外部ネームサーバに問い合わせる手順は本論文とは関係がないので，省略する

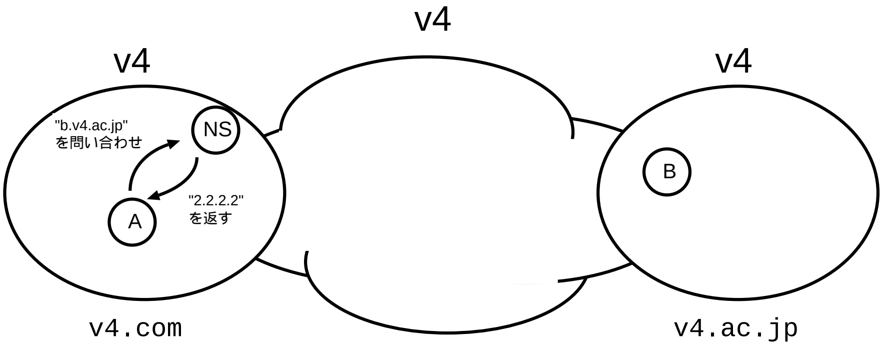


図 8　通常の通信手順 その 1

以上の手順により，A は B の IPv4 アドレスが 2.2.2.2 であることが分かる．そこで，A は以下の IPv4 パケットを送信する (図 9)．

始点アドレス　`1.1.1.1`
終点アドレス　`2.2.2.2`

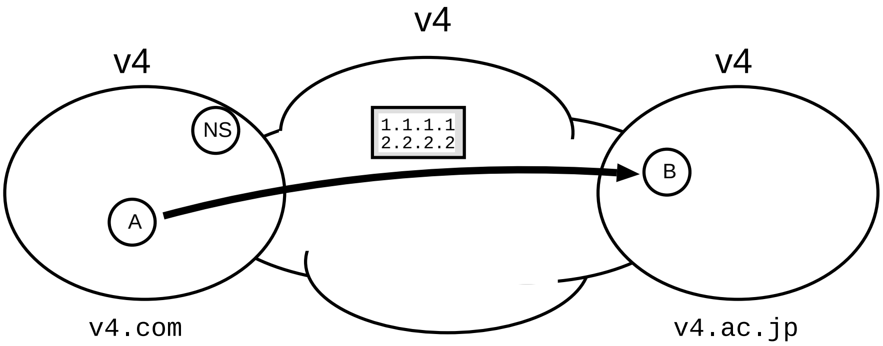


図 9　通常の通信手順 その 2

IPv6 の場合も基本的な通信方法は変わらない．ただし，IPv6 ではアドレスの問い合わせには，“A レコード” の代わりに IPv6 アドレスを示す “AAAA レコード” を用いる．

3.4.2 対応表があるときのヘッダ変換

次に，TOWNSを利用したときの通信手順を解説する．このときの通信手順は繁雑になるので，分かりやすくするために2段階に分けて説明する．

対象モデルにしたがって，図10のようなネットワークを考える．これは，モデル中の一つのIPv4サイトに注目したものである．IPv4サイト`v4.com`にある計算機Aは前節と同様にIPv4アドレスを持つ．AはIPv4の機能しか持たないが，IPv6アドレスが静的に割り当てられている．このAがIPv6のみの計算機Bと通信したいとする．Bの所属するサイト`b.v6.ac.jp`はIPv6のサイトである．サイト`v4.com`と外部インターネットの境界にはヘッダ変換ゲートウェイTGがある．AとBのホスト名，IPv4アドレス，IPv6アドレスの関係は以下のようになっている．

	ホスト名	IPv4アドレス	IPv6アドレス
計算機A	`a.v4.com`	`1.1.1.1`	`1234::1.1.1.1`(静的割り当て)
計算機B	`b.v6.ac.jp`	なし	`abcd::1`

この節では，まずアドレス対応表がAとTGにとって既知という前提を置く．すなわち，Bに一時的にIPv4アドレス`10.0.0.1`が割り当てられていることをAとTGが知っているとする．Aがどうやって割り当てられたアドレスを知るかについては次節で説明する．

AはBと通信しようとして，以下のIPv4パケットを送信する．

始点アドレス　`1.1.1.1`
終点アドレス　`10.0.0.1`

このパケットはTGによってIPv6パケットに変換される．この時，始点アドレスは`1.1.1.1`に静的に対応している`1234::1.1.1.1`，終点アドレスは対応表により，`abcd::1`に変換される．

始点アドレス　`1234::1.1.1.1`
終点アドレス　`abcd::1`

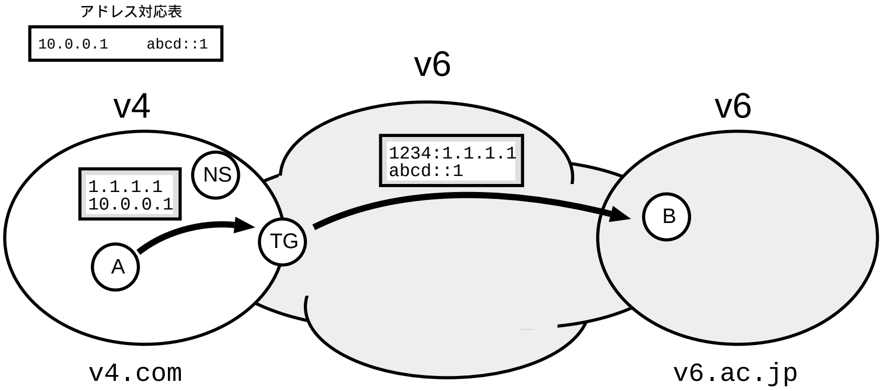


図 10 対応表があるときのヘッダ変換 その 1

一般的に通信は双方向的である．逆に B は A と通信しようとして以下の IPv6 パケットを送信する (図 10)．

始点アドレス　`abcd::1`
終点アドレス　`1234::1.1.1.1`

このパケットは同様に TG で変換される．

始点アドレス　`10.0.0.1`
終点アドレス　`1.1.1.1`

以上の手順でパケットを変換して，IPv4 計算機 A と IPv6 計算機 B が通信できる．

3.4.3 ネームサーバを利用したヘッダ変換

最後に TOWNS を利用した完全な通信手順を説明する．

図 12では，さらにアドレス対応を管理するためのマップ・サーバ MS が追加されている．計算機 A が計算機 B に一時的に割り当てられた IPv4 アドレスを知る手順を説明する．まず，A は NS に対して B の名前を知るために “A レコード” で問い合わせる．

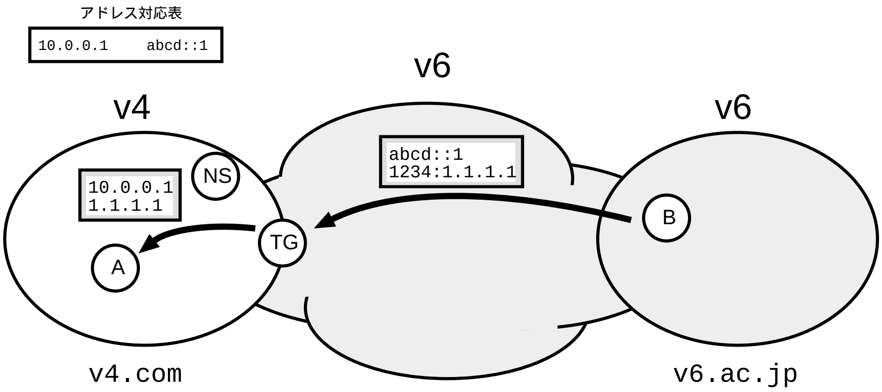


図 11 対応表があるときのヘッダ変換 その 2

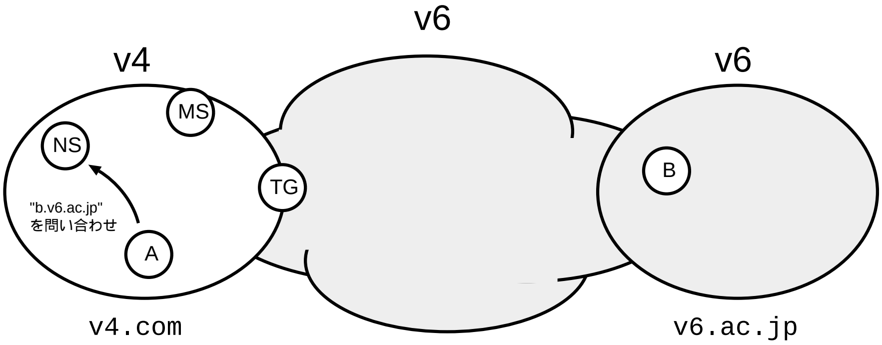


図 12 ネームサーバを利用したヘッダ変換 その 1

NS は外部ネームサーバに対して B の IP アドレスを問い合わせると，IPv4 アドレスは存在せず，IPv6 アドレス `abcd::1` だけが返ってくる．NS はこの IPv6 アドレスをそのまま IPv4 のみの計算機 A に返すわけにはいかない．A は IPv6 アドレスを理解できないからである．

そこで，NS は MS に問い合わせて `abcd::1` に対応する IPv4 アドレスを要求する (図 13)．MS は `abcd::1` に一時的に IPv4 アドレス `10.0.0.1` を割り当てて NS に返す．NS はこの `10.0.0.1` を A に返す．

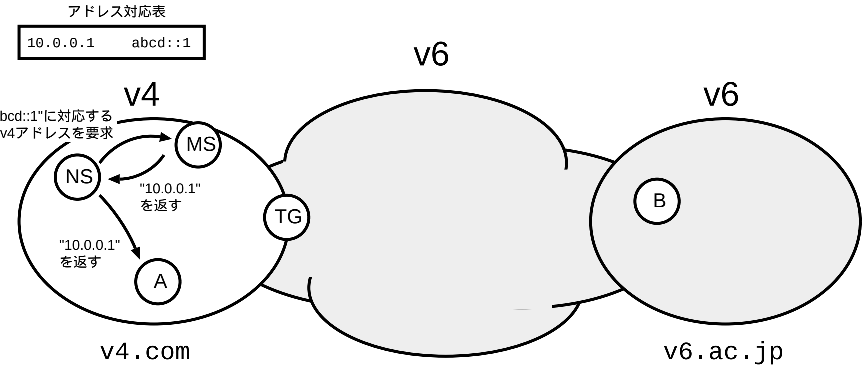


図 13 ネームサーバを利用したヘッダ変換 その 2

以上の手順により，A は B の IPv4 アドレスが `10.0.0.1` であることが分かるので，前節と同様のパケットを生成して通信する．すなわち，以上の手順により，A は B と通信しようとして，以下のパケットを送信する (図 14)．

始点アドレス	`1.1.1.1`
終点アドレス	`10.0.0.1`

TG もパケットを変換しようとするが，`10.0.0.1` に対する IPv6 アドレスが分からないので，MS に問い合わせる．すると，`abcd::1` という返答を受け取るので，IPv6 パケットに変換できる (図 15)．

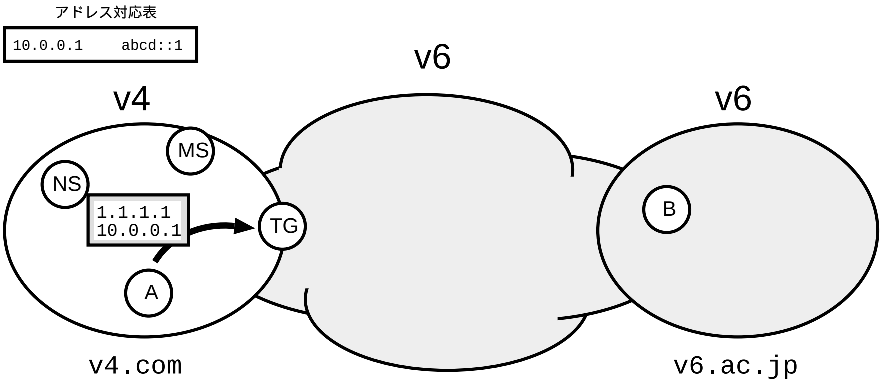


図 14 ネームサーバを利用したヘッダ変換 その 3

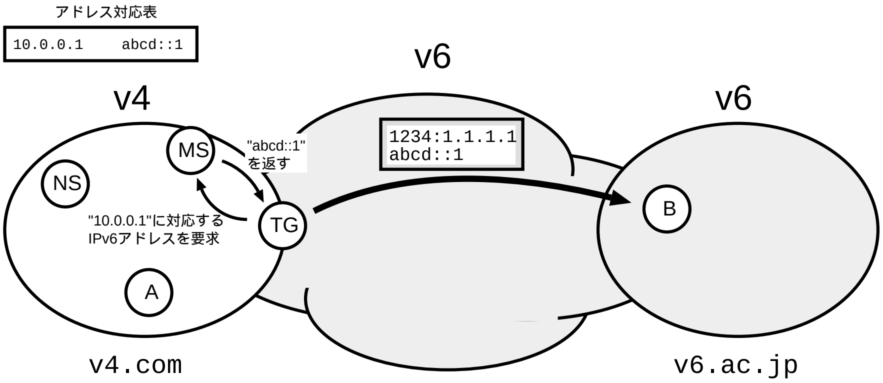


図 15 ネームサーバを利用したヘッダ変換 その 4

3.5 ヘッダ変換の詳細

この節では具体的なヘッダ変換の詳細について述べる．第3.2節で述べたように，IPv4とIPv6の完全な変換は不可能である．TOWNSでは高速性を重視して，何を変換して何を変換しないかというポリシを決定した．

この節では，以下の順に変換のポリシを述べる．

- IPv6ヘッダからIPv4ヘッダへの変換
- IPv4ヘッダからIPv6ヘッダへの変換
- IPv6拡張ヘッダからIPv4オプション・ヘッダへの変換
- IPv4オプション・ヘッダからIPv6拡張ヘッダへの変換
- 上位層(TCP，UDP)の相互変換
- 上位層(ICMP)の相互変換

3.5.1 IPv6ヘッダからIPv4ヘッダへの変換

まず，IPv6ヘッダからIPv4ヘッダへ変換する時の各フィールドの算出方法を示す．始点アドレスと終点アドレスの変換については，第3.3節で述べたので省略する．

Version

バージョン番号なので「4」である．

Type of Service

IPv4ではほとんど使われていないフィールドである．「0」を代入する．

ヘッダ長

変換後のオプションを含めたIPv4ヘッダの長さを2ビット右にシフトする．

全パケット長

IPv6のペイロード長に上記のヘッダ長(右シフトする前の値)を加えたものである．

識別子，フラグ，断片オフセット

以上の3つのフィールドは，IPv6拡張ヘッダに断片ヘッダが存在していれば，1対1に変換できる．存在しなければ0で埋める．

生存時間

IPv6の中継限界数を代入．

プロトコル

IPv6ヘッダまたは，数珠つなぎになったIPv6拡張ヘッダのうち，最後に現れるものの「次ヘッダ」を代入．

チェックサム

IPv6ではヘッダでチェックサムを計算しない．よって，変換後にIPv4ヘッダのチェックサムを計算して代入する，

3.5.2 IPv4ヘッダからIPv6ヘッダへの変換

次に，IPv4ヘッダからIPv6ヘッダへ変換する時の各フィールドの算出方法を示す．同様に始点アドレスと終点アドレスの変換については省略する．

Version

バージョン番号なので「6」である．

優先度

デフォルト値である「0」を代入する．

フローラベル

IPv4では存在しない概念であるので，「0」を代入．

ペイロード長

IPv4 の全パケット長から，ヘッダ長を左に 2 ビットシフトしたものを引いた値を代入する．変換後に IPv6 拡張ヘッダが現れるならば，その長さも加える．

次ヘッダ

IPv4 の「プロトコル」を代入する．ただし，変換後に拡張ヘッダが現れるのであれば，その番号を代入する．

中点限界数

IPv4 の生存時間を代入．

3.5.3 IPv6 拡張ヘッダから IPv4 オプション・ヘッダへの変換

まず，IPv6 拡張ヘッダから IPv4 オプション・ヘッダへの変換の詳細について述べる．表 3は IPv6 の拡張ヘッダの一覧表である．それぞれの拡張ヘッダについて IPv4 オプションに変換可能か否かを理由を併せて説明する．

表 3 IPv6 の拡張ヘッダ

	IPv4 では
中継点オプション・ヘッダ	存在しない
終点オプション・ヘッダ (1)	存在しない
経路制御ヘッダ	部分的に可能
断片ヘッダ	可能
認証ヘッダ	セキュリティ的に不可能
暗号ペイロード・ヘッダ	セキュリティ的に不可能
終点オプション・ヘッダ (2)	存在しない

中継点オプション・ヘッダ

終点オプション・ヘッダ (1)

終点オプション・ヘッダ (2)

現在 [10] で規定されているオプションのうち，同等の機能を持つ IPv4 オプションは存在しない．よって，これらの拡張ヘッダは変換時に無視する．

認証ヘッダ

暗号ペイロード・ヘッダ

これらの拡張ヘッダはセキュリティのためにパケットが途中経路で盗聴，変更されないのを保証するためのヘッダである．よって，途中経路であるヘッダ変換ゲートウェイでパケットを変更したところで，終点において破棄されてしまうので変換する意味がない．これらのヘッダが含まれている場合，パケット全体を破棄する．

断片ヘッダ

IPv4 ヘッダ中の「識別子」,「フラグ」,「断片オフセット」の各フィールドと 1 対 1 に変換できる．

経路制御ヘッダ

経路制御ヘッダは，IPv4 オプションの「厳密な経路制御オプション」,「緩やかな経路制御オプション」の上位互換であるので，完全な 1 対 1 の変換はできない．具体的には，IPv6 ではヘッダ内に指定している途中経路に対して，「厳緩ビットマップ」フィールドを用いて厳密か緩やかかを個々に指定できる．しかし，IPv4 では経路全体に対してしか，厳密か緩やかかを指定できない．よって，「厳緩ビットマップ」が全て 1，すなわち全ての途中経路を厳密に指定してあるときには，IPv4 の「厳密な経路制御オプション」に変換する．それ以外は，「緩やかな経路制御オプション」に変換する．

3.5.4 IPv4 オプション・ヘッダから IPv6 拡張ヘッダへの変換

次に，IPv4 オプション・ヘッダから IPv6 拡張ヘッダへの変換の詳細について述べる．表 4は IPv4 の IP オプションの一覧表である．それぞれのオプションについて IPv6 拡張ヘッダに変換可能か否かを理由を併せて説明する．

表 4 IPv4 のオプション

	IPv6 では
経路記録オプション	存在しない
厳密な経路制御オプション	可能
緩やかな経路制御オプション	可能
タイムスタンプ・オプション	存在しない

経路記録オプション

タイムスタンプ・オプション

同等の機能を果たす拡張ヘッダは，IPv6 では存在しない．これらのオプションは変換時に破棄される．

厳密な経路制御オプション

緩やかな経路制御オプション

IPv6 の経路制御ヘッダはこの 2 つのオプションの上位互換である．よって，IPv4 から IPv6 へは完全に変換できる．

3.5.5 上位層データの変換 (TCP,UDP)

TCP，UDP はトランスポート層のプロトコルであり，IPv4 と IPv6 でその違いはない．しかし，一部にネットワーク層の情報を利用しているそのフィールドがあるので，そのフィールドに対しては変換しなければならない．具体的にはチェッ

クサム計算に疑似ヘッダを利用しているのでチェックサム・フィールドは再計算する必要がある．例えば，IPv4 から IPv6 へヘッダを変換するときは，チェックサムから IPv4 疑似ヘッダの値を引いて，IPv6 疑似ヘッダの分を付け加える．IPv6 から IPv4 へはその逆の手順を踏む．

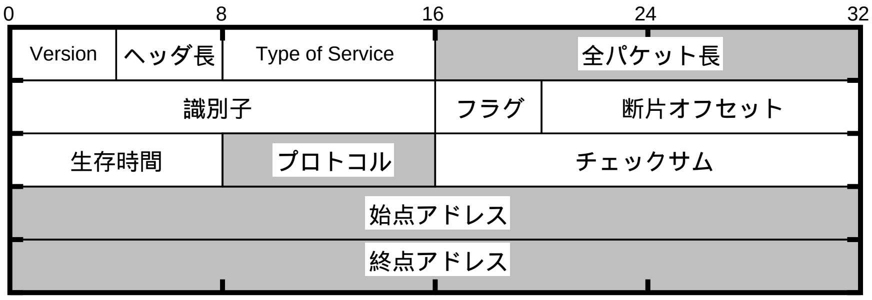


図 16 チェックサムの計算に使われるフィールド (IPv4)

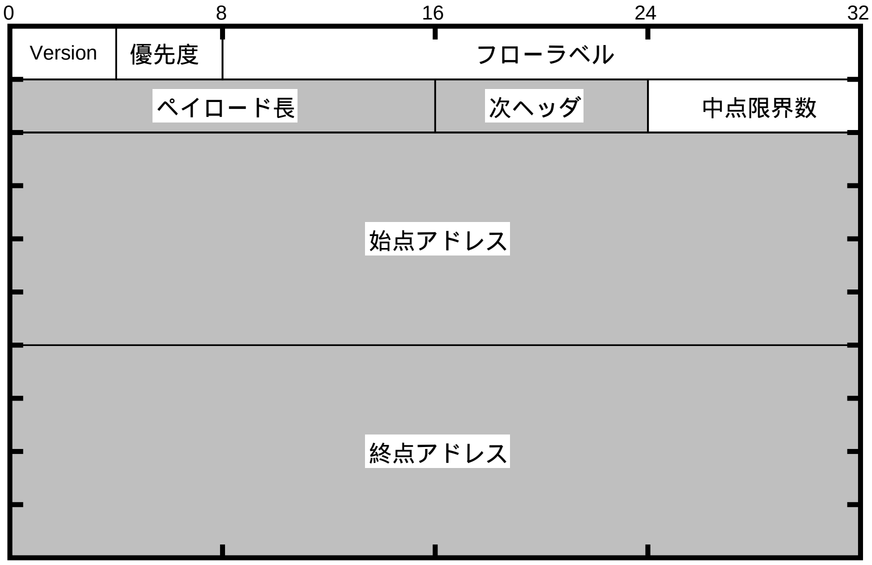


図 17 チェックサムの計算に使われるフィールド (IPv6)

FTP 等の階層侵害プロトコルについては，上位層パケットを解析してアドレ

スを書き換えないと相互接続できないが，TOWNS ではこれを行わないことにした．理由は上位層パケットを厳密に解析するとなると，処理が繁雑になり変換速度が落ちるからである．これは，当初の目標である高速性を損なうことになる．これらのプロトコルに対しては代理サーバを用意するなどして，別の枠組で相互接続性を提供する．

3.5.6 上位層データの変換 (ICMP)

ICMP はネットワークの状態を伝えるプロトコルである．IPv4 と IPv6 では構造は似ているが，割り当てられたタイプとコードはかなり異なる．しかし，ネットワークの状態を伝えるためにできる限り変換する必要がある．

[11][12] で定義されている IPv4 の ICMP(以下，ICMPv4 と略記) のうち，ヘッダ変換ゲートウェイで変換しなければならないタイプのみを表 5に挙げる．すなわち，ルータ通知などのローカル・ネットワークでしか利用されない ICMP メッセージは省き，ネットワークを越えていくタイプのみを挙げた．

右 2 つの項目が ICMPv6 に変換後のタイプとコードである．“–” は ICMPv6 では存在しない概念なので，変換不可能を示す．このパケットは変換ゲートウェイで破棄される．

同様に，IPv6 でで定義されている ICMP[13](以下，ICMPv6 と略記) の一覧表を表 6に挙げる．右 2 つの項目が ICMPv4 に変換後のタイプとコードである．

4. TOWNS の実装

TOWNS のシステムは以下の 3 種類のソフトウェアから構成される (図 18)．

変換ゲートウェイ

IPv4 と IPv6 パケットを相互に変換する．

マップサーバ

変換のために一時的に対応させた IPv4 と IPv6 のアドレスの組を管理する．

表 5 ICMPv4 の種類

			ICMPv6 では	
タイプ	コード	内容	タイプ	コード
0	0	エコー要求	128	0
3		終点到達不能		
	0	ネットワーク到達不能	1	0
	1	ホスト到達不能	1	0
	2	プロトコル到達不能	4	1
	3	ポート到達不能	1	4
	4	DF がセットされているが，分割が必要	2	0
	7	終点ホスト不明	1	7
	11	TOS のためネットワーク到達不能	1	0
	12	TOS のためホスト到達不能	1	0
	13	アクセス拒否	1	1
	14	優先順位違反	1	0
4		始点抑制	-	-
5		向け直し		
	0	ネットワークの向け直し	137	0
	1	ホストの向け直し	137	0
	2	ネットワークと TOS の向け直し	137	0
	3	ホストと TOS の向け直し	137	0
8		エコー返答	129	0
11		時間超過		
	0	転送中に生存時間が 0 になった	3	0
	1	再構成中に生存時間が 0 になった	3	1
12		パラメータ問題		
	0	IP ヘッダが異常	4	0
	1	オプションが異常	4	2

表 6 ICMPv6 の種類

			ICMPv4 では	
タイプ	コード	内容	タイプ	コード
1		終点到達不能		
	0	終点経路なし	3	1
	1	アクセス拒否	3	13
	2	非近隣	3	7
	3	アドレス到達不能	3	1
	4	ポート到達不能	3	3
2		パケット過大	3	4
3		時間超過		
	0	転送中の中継限界数超過	11	0
	1	再構成許容時間超過	11	1
4		パラメータ問題		
	0	ヘッダ・フィールドでのエラー	12	0
	1	次ヘッダ番号が認識できない	3	2
	2	IPv6 オプションが認識できない	12	1
128		エコー要求	0	0
129		エコー通知	8	0
137		向け直し	5	1

改良ネームサーバ

IPv6 計算機に対して IPv4 アドレスを一時的に割り当て，それをクライアントに返す．

これらのサーバ，ゲートウェイは “変換アドレス要求”，“変換アドレス返答” という二つのプロトコルで互いに情報を交換しあう．このプロトコルは IPv4 の ICMP で実現されている．

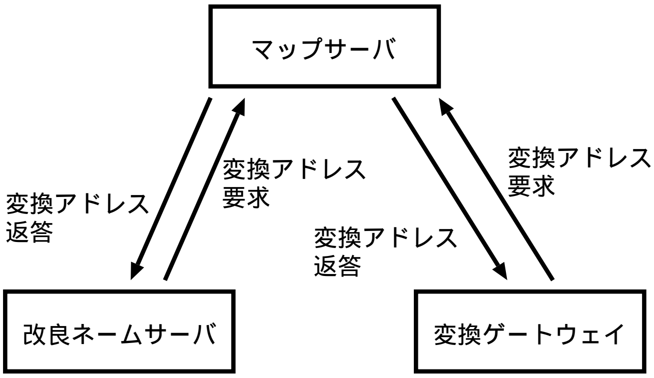


図 18　TOWNS のシステム構成

これらを順に説明していく．

4.1　変換ゲートウェイ

変換ゲートウェイは，Hydrangea[5] を拡張し，ヘッダ変換機能を持たせることによって実現した．このヘッダ変換機能はカーネル・レベルで動作している．Hydrangea とは我々のグループが現在開発している IPv6 のネットワーク・コードで，BSDI 社の BSD/OS 2.1 をベースに実装している．

拡張部は大きく分けて表 7に示す 5 つの部分からなる．その構成を図 19に示す．

それぞれの機能は以下のようになっている．

---

[5] `ftp://ftp.aist-nara.ac.jp/pub/IPv6/hydrangea/`

表 7 相互接続表

機能	関数名
ヘッダ変換受付部	trans_4to6(), trans_6to4
ヘッダ変換，出力部	trans_4to6_output(), trans_6to4_output()
対応アドレス検索部	trans_lookup4(), trans_lookup6()
“変換アドレス要求” 送信部	ta_output4(), ta_output6()
“変換アドレス返答” 送信部	ta_input()

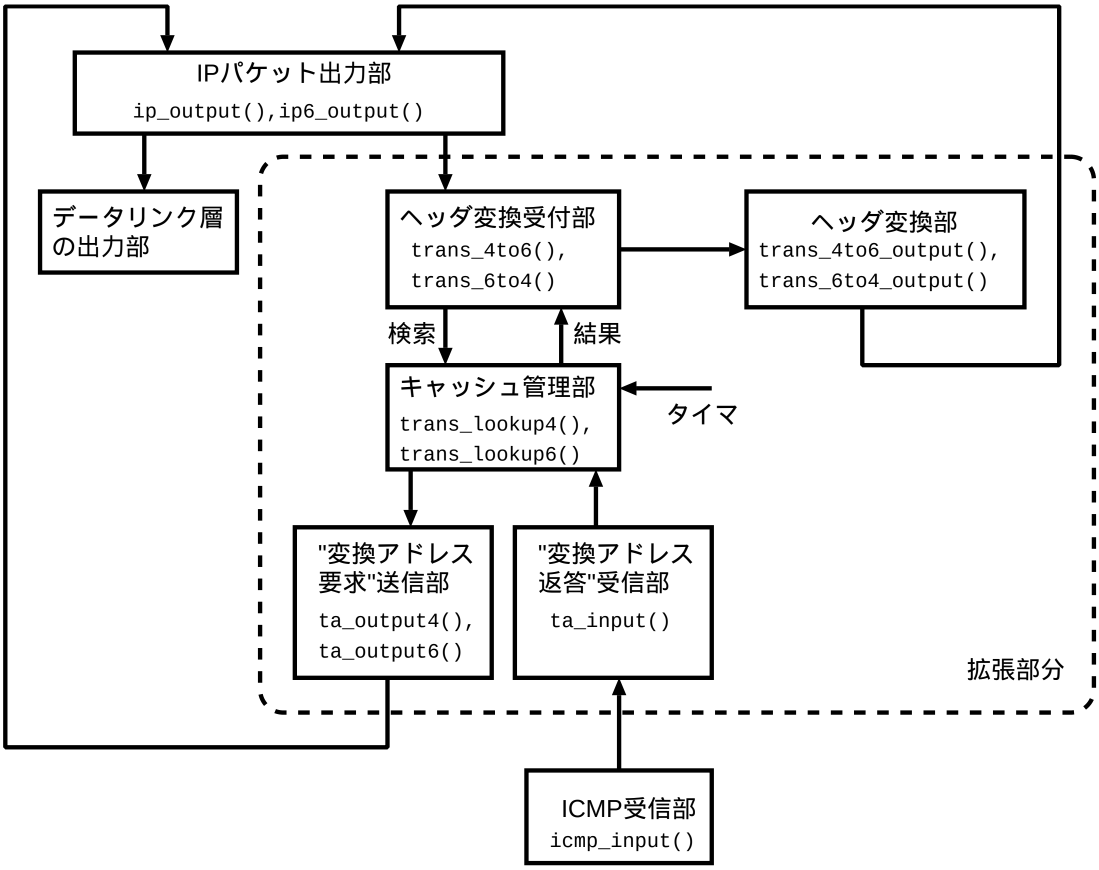


図 19 ヘッダ変換ゲートウェイの構成

ヘッダ変換受付部

IPv4 および IPv6 パケットの出力部から呼び出される．ヘッダ変換の対応アドレスを「キャッシュ管理部」に問い合わせる．もし対応アドレスが未解決であれば，ヘッダ変換できないのでそのパケットを保留する．対応アドレスが分かれば，ヘッダ変換部を呼び出す．

ヘッダ変換部

IPv4 ヘッダを IPv6 に，もしくは IPv6 ヘッダを IPv4 に変換する．ヘッダ変換の詳細は 3.5で述べた．

キャッシュ管理部

IPv4 アドレスと IPv6 アドレスの対応表をキャッシュしている．「ヘッダ変換受付部」より検索要求があると，キャッシュを調べる．対応アドレスが存在するとそのアドレスを返し，存在しなければ，「変換アドレス問い合わせ」を送信し，未解決を返す．

“変換アドレス要求” 送信部

“変換アドレス要求” パケットをマップサーバに向けて送信する．

“変換アドレス返答” 受信部

“変換アドレス返答” を受信すればキャッシュを更新する．もし，保持されているパケットがあればヘッダ変換部を呼び出す．

4.2 マップサーバ

マップサーバは IPv4 アドレスと IPv6 アドレスの対応をデータベースとして一元管理している．要求に応じて，対応するアドレスを返す．

IPv4 アドレスの要求を受け付ける時は以下のアルゴリズムで動作する．

- IPv4 アドレスの要求があると，引数の IPv6 アドレスに対応する IPv4 アドレスを検索する．

- 対応する IPv4 アドレスがあれば，それを返す．
- 対応する IPv4 アドレスがなければ，未割り当ての IPv4 アドレスを割り当てて，それを返す

IPv6 アドレスの要求を受け付けるときは以下のアルゴリズムで動作する．

- IPv6 アドレスの要求があると，引数の IPv4 アドレスに対応する IPv6 アドレスを検索する．
- 対応する IPv6 アドレスがあれば，それを返す．
- 対応する IPv6 アドレスがなければ，未解決を示す “::” を返す．

現在の実装では 10.0.0.0/8 の空間を単なるリング・バッファとみなして，順番にアドレスを割り当てている．アドレスが一周すると，アドレスの使用期限が切れたことになる．

4.3 改良ネームサーバ

改良ネームサーバは慶應義塾大学の土井裕介氏が開発した yans を基に，アドレスを変換する機能を拡張した．

本来の yans に対して A レコードで問い合わせると，以下のアルゴリズムで動作する．

- キャッシュにエントリが存在するか検索する．
  - 存在すれば，その IPv4 アドレスを返す．
  - 存在しなければ，外部ネームサーバに対して問い合わせる．
- キャッシュの更新．
- IPv4 アドレスを返す．

AAAA レコードの場合も同様の動作をし，IPv6 アドレスを返す．

これを以下のように拡張した．

- キャッシュにエントリが存在するか検索する．
  - 存在すれば，その IPv4 アドレスを返す．
  - 存在しなければ，外部ネームサーバに対して A レコードで問い合わせる．
- もし，外部ネームサーバから，
  - IPv4 アドレスが返ってくれば，キャッシュを更新し，そのアドレスを返す．
  - IPv4 アドレスが返ってこなければ，改めて外部ネームサーバに対して AAAA レコードで問い合わせる．
  - IPv6 アドレスが返ってくれば，マップサーバに問い合わせて対応した IPv4 アドレスを取得する．
  - IPv6 アドレスが返ってこなければ，エラーを返す．
- 対応した IPv4 アドレスを用いてキャッシュを更新し，そのアドレスを返す．

AAAA レコードで問い合わせがあったときは，拡張前と同じ動作をする．

4.4 変換アドレス要求・回答プロトコルの定義

マップサーバが，変換ゲートウェイ，改良ネームサーバと情報を交換するためのプロトコルは，IPv4 上の ICMP に新しいタイプを拡張する形で実現した．IPv4 で通信する理由はサイト内が IPv4 しか通さないからである．このプロトコルのヘッダ形式を図 20に，新しく割り当てたタイプとコードを表 8に示す．

例として，改良ネームサーバがある IPv6 アドレス `abcd::1` に，一時的に割り当てる IPv4 アドレスをマップサーバに要求し，マップサーバは `10.0.0.1` を割り当てる場合を考える (図 21)．ネームサーバは図 20のパケットをマップサーバに送信する．この時，タイプは 19，コードは 0，IPv6 アドレスは `abcd::1`，そして IPv4 アドレスは未解決なので `0.0.0.0` が入っている．マップサーバは，この IPv6 アドレスに `10.0.0.1` を割り当て，返答パケットを返す．この時，タイプは

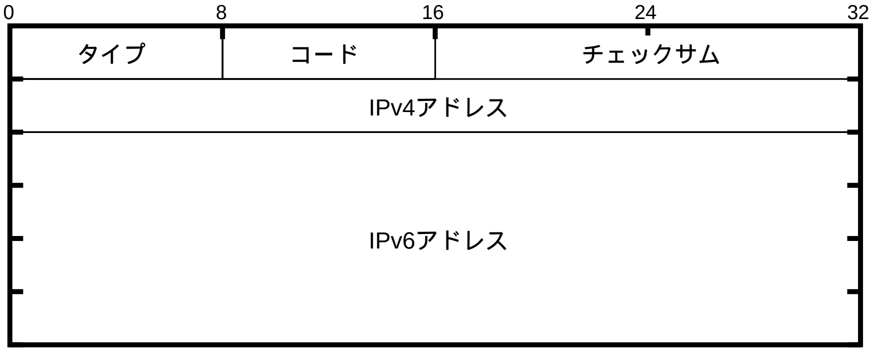


図 20　変換アドレス要求・回答プロトコルのヘッダ形式

表 8　新しく定義した ICMPv4 のタイプとコード

タイプ	コード	内容
19		変換アドレス要求
	0	IPv4 アドレス要求
	1	IPv6 アドレス要求
20		変換アドレス返答
	0	IPv4 アドレス返答
	1	IPv6 アドレス返答

20，コードは 0，IPv6 アドレスは `abcd::1`，そして IPv4 アドレスは `10.0.0.1` が入る．

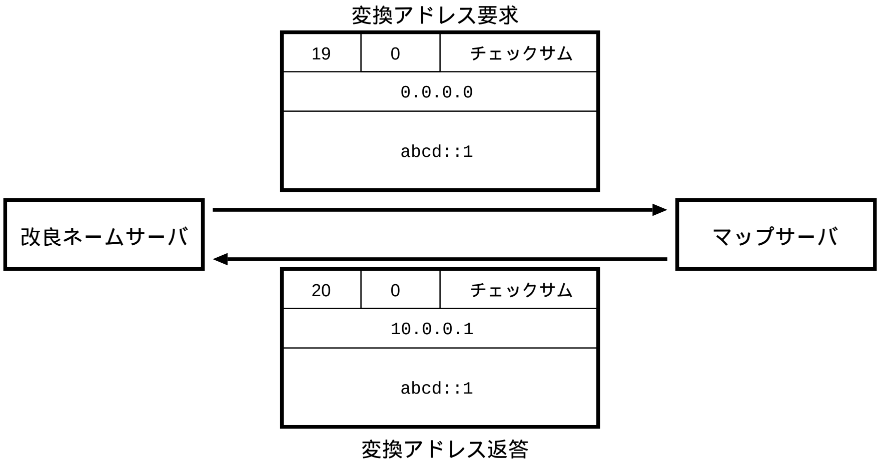


図 21 変換アドレス要求・回答プロトコルの動作例

5. 評価

本章ではまず，実際に TOWNS がトランスレータとして，適切に動作していることを確認する．次に，TOWNS 以外のトランスレータを挙げ，その特徴を比較する．最後に TOWNS が当初の設計目標を達成できているかを評価し，TOWNS が汎用性，高速性という面で優れていることを示す．

5.1 TOWNS の動作確認

まず，TOWNS がトランスレータとして正常に動作していることを確認する．

5.1.1 TCP の動作確認

IPv4 のみの計算機上で netscape を用いて，IPv6 の実験ネットワークである 6bone 上の WWW サーバ `http://altavista.ipv6.digital.com/`のホームペー

ジを表示した．画面図を図22に示す．netscapeはバイナリのみで提供されているWWWブラウザで，現在のものはIPv4の機能しか持たないが，TOWNSを利用することでIPv6計算機と接続できている．

以上より，TOWNSがTCPパケットの変換に成功していると分かる．

5.1.2 ICMPの動作確認

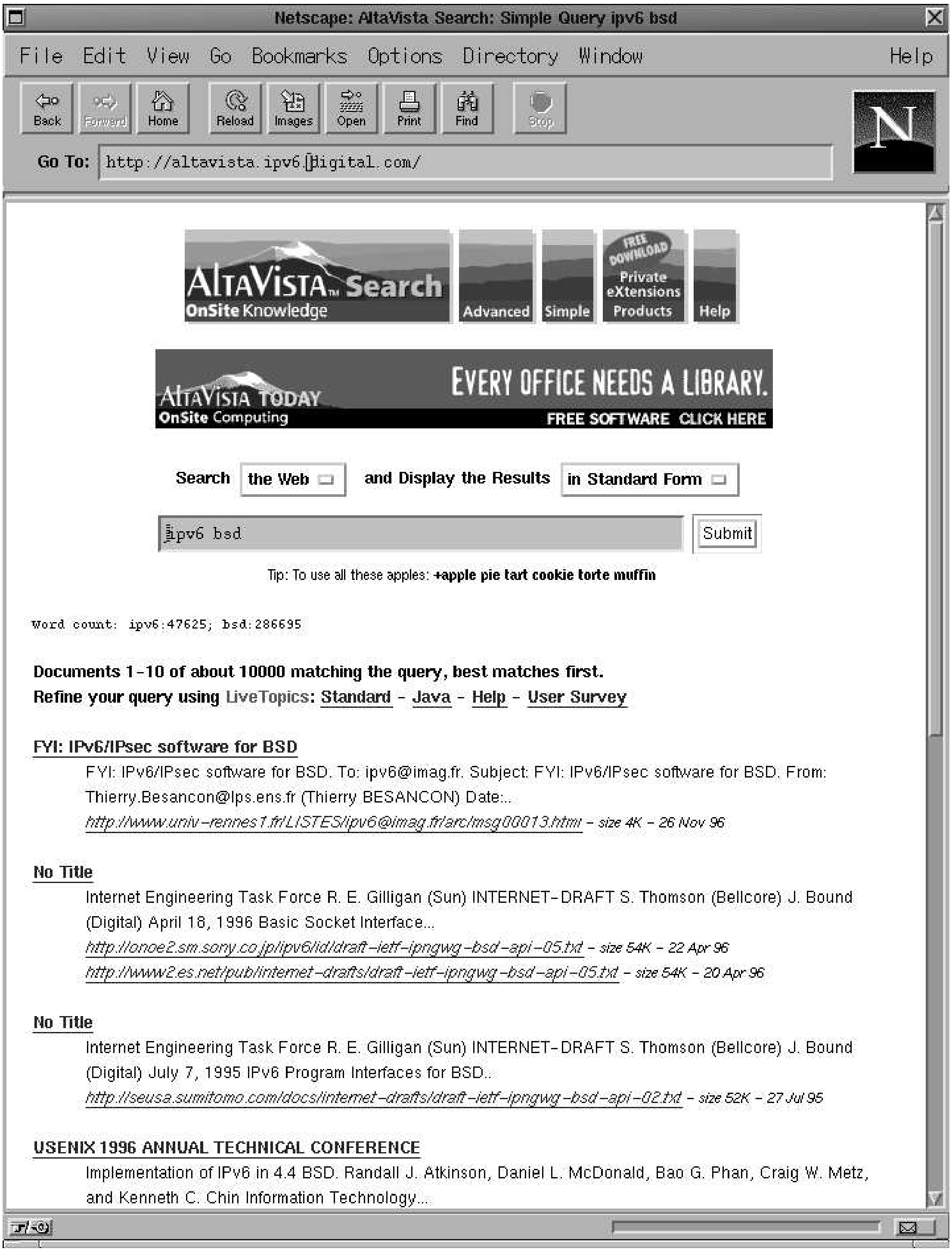
次にICMPの動作を確認するために，6bone上のIPv6計算機`tokyo.v6.wide.ad.jp`に対してpingコマンドを実行した．

```
lobster:~% ping tokyo.v6.wide.ad.jp
PING toyko.v6.wide.ad.jp (10.0.0.1): 56 data bytes
64 bytes from 10.0.0.1: icmp_seq=0 ttl=253 time=48.258 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=253 time=42.785 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=253 time=42.711 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=253 time=42.769 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=253 time=42.735 ms
^C
--- tokyo.v6.wide.ad.jp ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 42.711/43.851/48.258 ms
```

以上より，TOWNSがICMPパケットを意図した通りに変換していることが分かる．

5.2 TOWNS以外のトランスレータ

TOWNS以外のトランスレータは現在のところ，FAITHとSOCKS64の二つが提案されている．次節以降でこれらのトランスレータとの比較検討を行うため，本節ではそれぞれの概要を説明する．



Go To: http://altavista.ipv6.digital.com/

ipv6 bsd

Tip: To use all these apples: **+apple pie tart cookie torte muffin**

`Word count: ipv6:47625; bsd:286695`

Documents 1-10 of about 10000 matching the query, best matches first.
Refine your query using LiveTopics: Standard - Java - Help - User Survey

FYI: IPv6/IPsec software for BSD

FYI: IPv6/IPsec software for BSD. To: ipv6@imag.fr. Subject: FYI: IPv6/IPsec software for BSD. From: Thierry.Besancon@lps.ens.fr (Thierry BESANCON) Date:..
http://www.univ-rennes1.fr/LISTES/ipv6@imag.fr/arc/msg00013.html - size 4K - 26 Nov 96

No Title

Internet Engineering Task Force R. E. Gilligan (Sun) INTERNET-DRAFT S. Thomson (Bellcore) J. Bound (Digital) April 18, 1996 Basic Socket Interface...
http://onoe2.sm.sony.co.jp/ipv6/id/draft-ietf-ipngwg-bsd-api-05.txt - size 54K - 22 Apr 96
http://www2.es.net/pub/internet-drafts/draft-ietf-ipngwg-bsd-api-05.txt - size 54K - 20 Apr 96

No Title

Internet Engineering Task Force R. E. Gilligan (Sun) INTERNET-DRAFT S. Thomson (Bellcore) J. Bound (Digital) July 7, 1995 IPv6 Program Interfaces for BSD..
http://seusa.sumitomo.com/docs/internet-drafts/draft-ietf-ipngwg-bsd-api-02.txt - size 52K - 27 Jul 95

USENIX 1996 ANNUAL TECHNICAL CONFERENCE

Implementation of IPv6 in 4.4 BSD. Randall J. Atkinson, Daniel L. McDonald, Bao G. Phan, Craig W. Metz, and Kenneth C. Chin Information Technology...

図 22 TOWNS の動作確認 (TCP)

FAITH

FAITH は奈良先端科学技術大学院大学の山本和彦氏と (株) シャープの島慶一氏の開発によるアプリケーション層のゲートウェイである．ユーザは外部の計算機と透過的に通信している感覚を受けるが，実際には途中経路の FAITH ゲートウェイで一旦コネクションを終端する．そして，ゲートウェイ上の代理サーバを通して外部の計算機と通信する．ステルス型の防火壁として動作しながら，トランスレータとしても動作する．FAITH はアプリケーション空間で実現されている．

第 2.4節で問題になった IPv4 と IPv6 のアドレスの対応づけは，TOWNS で提案した機構をそのまま使用している．つまり，改良ネームサーバを用いて一時アドレスを割り当てることで解決している．

SOCKS64

SOCKS64 は米国 NEC が開発した SOCKS[8] を (株) 富士通研究所の陣崎明氏と小林 伸治氏が IPv4-IPv6 ゲートウェイとして拡張したトランスレータである [14]．本来，SOCKS は防火壁を構築するためトランスポート層のゲートウェイであるが，SOCKS64 はアプリケーション層ゲートウェイとして動作するように拡張している．クライアントは SOCKS 独自のライブラリを用いて，SOCKS64 ゲートウェイに FQDN で接続要求を出す．その結果，SOCKS64 は一種の代理サーバとして動作し，第 2.4節で問題になった IPv4 と IPv6 のアドレスの対応づけを解決している．

5.3 設計目標との比較

第 3.1節で，2 つの設計目標を掲げた．これらの目標が達成できているかを検討する．

高速に動作しているか?

TOWNS の変換速度を測定するために，図 23の実験環境を構築した．IPv4 計算機 A から TCP を用いて 1000 バイトの IPv4 パケットを 10000 個送出

し，ヘッダ変換ゲートウェイを用いて IPv6 計算機 B と通信するのに要した時間を計算した．

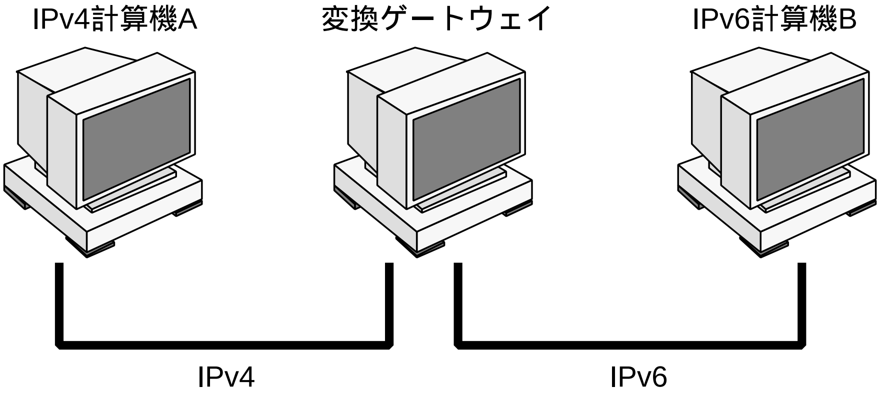


図 23　速度比較の実験環境

また，高負荷時の変換遅延を調べるため，ゲートウェイ上にプロセスを 1 つ，プロセスを 2 つ，それぞれ実行した時の時間を調べた．

他のトランスレータとの比較を行うため，同様に FAITH についても計測した．また，参考のために単純に IPv4 の転送，IPv6 の転送に要した時間も計測した．この 2 つの時間の平均値がヘッダ変換の速度の上限だと考えられる．SOCKS64 はまだ開発中のため，変換速度を評価できなかった．しかし，その動作原理を考えると，FAITH と同程度の転送速度であると考えられる．

この速度表を表 9に挙げる

TOWNS は負荷の有無に関わらず，単純転送にかなり近い速度で動作することが分かる．FAITH については負荷がない状態では TOWNS と変わりない速度で動作するが，負荷が上がるにしたがって，変換時間が増大することがわかる．

実際の運用を考えると，TOWNS は代理サーバを必要としないが，FAITH はゲートウェイ上でコネクションの数だけ代理サーバを起動するので負荷

表 9 各トランスレータの速度比較

	負荷なし (秒)	負荷 1(秒)	負荷 2(秒)
IPv4	15.71	16.21	16.26
IPv6	16.28	16.97	17.19
TOWNS	17.30	17.73	17.88
FAITH	17.73	32.82	50.03

が増大する．よって，TOWNS と FAITH の速度比は大きくなるものと予想される．

以上の結果より，TOWNS は他のトランスレータと比して，その転送速度は十分に速いと言える．

導入コストが小さいか?

TOWNS は，変換ゲートウェイ，マップサーバ，改良ネームサーバの 3 つを導入するだけで，IPv4 サイト全体が IPv6 インターネットと通信できるようになる．よって，導入コストは小さいといえる．

5.4 TOWNS 以外のトランスレータとの比較

本節では TOWNS 以外のトランスレータとの比較を行う．TOWNS も含めたそれぞれの特徴を以下に示し，比較表を表 10に示す．

TOWNS

- パケットの変換が速い．
- ゲートウェイ上でサーバを立ち上げる必要はない．
- プロトコルに依存しないでヘッダ変換できるので，未知のプロトコルにも基本的には対応できる．しかし，階層違反のプロトコルには対応が困難である．

- アドレス対応表を管理する必要がある．
- 導入しなければならないシステムは，改良ネームサーバ，マップサーバ，ヘッダ変換ゲートウェイである．
- クライアントには一切手を加えなくてよい．
- 外部ホストから内部ホストへのネットワーク透過性を実現できている．

FAITH

- パケットの転送が遅い．
- ゲートウェイ上にコネクションの数だけサーバをに立ち上げるため負荷が集中する．
- プロトコルごとに専用のサーバが必要となるが，階層違反のプロトコルにも対応できる．
- アドレス対応表を管理する必要がある．
- 導入しなければならないシステムは，改良ネームサーバ，マップサーバ，FAITH ゲートウェイとその上で動作するプロトコルごとの転送サーバである．
- クライアントには一切手を加えなくてよい．
- コネクションを (ユーザには感じさせないが) 終端するので，きめの細かいアクセス制御ができる．
- 外部ホストから内部ホストへのネットワーク透過性を実現できている．

SOCKS64

- パケットの転送が遅い．
- ゲートウェイ上にコネクションの数だけサーバをに立ち上げるため負荷が集中する．
- プロトコルごとに専用のサーバが必要となるが，階層違反のプロトコルにも対応できる．

- アドレス対応表は不要．
- コネクションを (ユーザには感じさせないが) 終端するので，きめの細かいアクセス制御ができる．
- 導入しなければならないシステムは，SOCK64 ゲートウェイとその上で動作するプロトコルごとの転送サーバである．
- クライアントで使用したいアプリケーションを SOCKS 対応にする必要がある．
- 外部ホストから内部ホストへのネットワーク透過性を実現できている．しかし，外部ホストのアプリケーションも SOCKS 対応にしなければならない．
- 既に SOCKS5 を導入しているサイトではクライアントの変更は不要である．

表 10　各種トランスレータの比較

	導入コスト	階層違反プロトコル	転送
TOWNS	低い	困難	速い
FAITH	低い	可能	遅い
SOCKS64	高い	可能	遅い

	アクセス制御	アドレス対応表	プロトコルごとのサーバ
TOWNS	弱い	必要	不要
FAITH	強い	必要	必要
SOCKS64	強い	不要	必要

以上の比較検討より，高速性と汎用性においては TOWNS が優れてるといえる．アプリケーション空間でヘッダを変換する FAITH や SOCKS64 に比べると，カーネル空間でヘッダ変換を実現する TOWNS は高速に動作する．また，プロ

トコルごとに代理サーバを用意しなければならない FAITH，SOCKS64 に比べ，TOWNS は代理サーバが不要で，未知のプロトコルも変換できる．

企業等の防火壁内のサイトで限られたプロトコルしか通す必要がなければ，防火壁の役割も兼ねている FAITH か SOCK64 が便利である．既に SOCKS を導入しているサイトならば SOCKS ゲートウェイを SOCK64 化するだけでよいので，導入コストが 3 つの中で最も低いといえる．SOCKS を導入していないサイトでは FAITH の方が導入コストが低い．

6. 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

インターネットの急速な成長に伴い，IPv4 アドレスの枯渇が問題になっている．128 ビットのアドレス空間を持つ IPv6 はこの問題を解決するだけではなく，様々な新しい機能を提供している．IPv4 から IPv6 へのインターネットの移行は非常に重要だが，現在採択されているデュアル・スタックとトンネリングを用いた移行機構では十分であるとはいえない．本研究では，ヘッダ変換を利用した移行技術 TOWNS を用いて，現在の移行機構を補完し，より円滑な移行を提供した．また，この際に問題になる IPv4 アドレスと IPv6 アドレスの対応づけをネームサーバを用いることで解決した．

TOWNS を実際に利用して，6bone 上の IPv6 計算機と通信することができた．また，TOWNS は提案中の他のトランスレータと比べると，高速性，汎用性に優れているという利点を持つということが，比較検討の結果，明らかになった．

6.2 今後の課題

TOWNS の実装，仕様はまだプロトタイプに過ぎない．実際にインターネットで運用されるためには，様々なトラブルに耐えうる頑健性を持ち，また高速性を向上させる必要がある．

具体的には以下の課題があげられ，検討が必要である．

マップサーバ

境界ゲートウェイが複数あるサイトでは，一時的に割り当てるアドレス空間を複数に分割し，それぞれの経路を向けることによって，サイト外に出ていくゲートウェイを指定できる枠組がTOWNSでは提供されている．このため要求されたIPv6アドレスに応じて，マップサーバはどのIPv4アドレスを割り当てるかのポリシが反映できる必要がある．現在のマップサーバは初歩的なアドレス割り当てしか提供していないので，改良が必要である．

変換アドレス要求，回答プロトコル

現在の仕様はもっとも簡潔なヘッダ形式となっている．識別子フィールドなどを用意して，安全性を高める必要がある．

また，変換ゲートウェイをより高速に動作させるために，マップ・サーバが能動的に変換ゲートウェイのキャッシュの更新や削除ができる枠組が必要と思われる．

謝辞

まず最初に，本研究を進めるにあたり，指導教官として研究の場を提供して下さった奈良先端科学技術大学院大学の福田晃教授，山本平一教授，および，最所圭三助教授にお礼を申し上げます．また，指導と助言を頂いた奈良先端科学技術大学院大学の山本和彦先生に深謝します．

本研究の初期段階から有用な議論をして頂いたシャープ株式会社の島 慶一さんに感謝します．WIDE プロジェクト IPv6 分科会のみなさんには，メーリングリストや研究会，ワークショップでを様々な意見を頂き，お礼を申し上げます．

2 年間の IPv6 の研究を通して，新しいインターネットの誕生とその成長に立ち会うのはとても楽しい作業でした．最後になりましたが，素晴らしい経験と様々な機会を与えて下さった WIDE プロジェクトに心より感謝の意を表します．

参考文献

[1] J. Postel. Internet Protocol, RFC791. September 1981.

[2] G. Huston. Observations on the Management of the Internet Address Space, RFC1744. December 1994.

[3] R. Atkinson. Security Architecture for the Internet Protocol, RFC1825. August 1995.

[4] R. Atkinson. IP Authentication Header, RFC1826. August 1995.

[5] R. Atkinson. IP Encapsulating Security Payload (ESP), RFC1827. August 1995.

[6] R. Gilligan and E. Nordmark. Transition Mechanisms for IPv6 Hosts and Routers, RFC1933. April 1996.

[7] P. Francis and K. Egevang. The IP Network Address Translator (Nat), RFC1631. May 1994.

[8] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS Protocol Version 5, RFC1928. April 1996.

[9] Y. Rekhter, R. Moskowitz, D. Karrenberg, G. de Groot, and E. Lear and. Address Allocation for Private Internets, RFC1918. February 1996.

[10] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification, RFC1883. January 1996.

[11] F. Baker. Requirements for IP Version 4 Routers, RFC1812. June 1995.

[12] J. Postel. Internet Control Message Protocol, RFC792. September 1981.

[13] R. Hinden and S. Deering. IP Version 6 Addressing Architecture, RFC1884. January 1996.

[14] 陣崎 明 and 小林 伸治. SOCKSによるIPv4-IPv6 Gateway. WIDEプロジェクト研究会, Decenber 1996.

[15] J. Postel. Internet Offical Protocol Standards, RFC1800. July 1995.

付録

A. 拡張ヘッダの詳細

拡張ヘッダの使用方法を具体的に説明する．IPv6 ヘッダ，拡張ヘッダ内には，「次ヘッダ」というフィールドが設けられている．そこには，次に続くヘッダの内容を [15] で規定された 8 ビットの整数値で示している．

図 24に数珠つなぎ型のヘッダ構造の一例を示す．

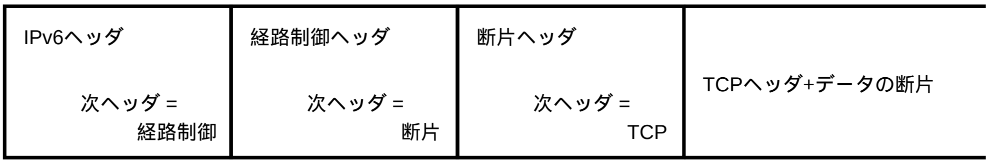


図 24 数珠つなぎ型のヘッダ構造

[10] では，以下の拡張ヘッダが規定されている．また，この順序でそれぞれが多くとも 1 回だけ現れることを推奨している．

中継点オプション・ヘッダ

途中経路を含む各計算機，ルータ上で参照される．

終点オプション・ヘッダ (1)

次述の経路制御ヘッダで指定された計算機と，最終目的アドレスの計算機において参照される．

経路制御ヘッダ

配送経路を明示的に指定するためのヘッダ．基本的には IPv4 の経路制御オプションと同一の働きをするが，よりきめの細かい設定ができる．

断片ヘッダ

分割されたパケットを再構成するためのヘッダ．IPv6と違い，パケットは途中経路で分割されない．

認証ヘッダ

認証を行うためのヘッダ．

暗号ペイロード・ヘッダ

暗号化を行うためのヘッダ．このヘッダが付いていた場合，次のヘッダ以降は暗号化されて，ペイロードを盗聴できないようにする．

終点オプション・ヘッダ (2)

最終目的アドレスの計算機だけが参照する．

上位層ヘッダ

上位層のプロトコル・ヘッダ．

B. アドレス表記法

IPv4では、オクテットごとにピリオドで区切ったアドレス表記法が使われていた。しかし、IPv6ではアドレス長が4倍になったため、この表記法では長くなりすぎる。そこで、IPv6のアドレスは4桁の16進数ごとに“:”で区切ることになった[13]。

```
0800:1234:0000:0000:0000:0000:00AB:CDEF
```

連続した0000は“::”で省略できる。

```
0800:1234::00AB:CDEF
```

また、区切り単位ごとに上位桁の0は省略できる。

```
800:1234::AB:CDEF
```

ただし、複数箇所を “::” にすることはできない。これはアドレスが曖昧になり、一通りに定まらないからである。例えば、以下のようなアドレスを考える。

```
0123:4567:89ab::cdef::0123
```

これは以下のどちらにも解釈できるので、曖昧なアドレスである。

```
0123:4567:89ab:0000:0000:cdef:0000:0123
0123:4567:89ab:0000:cdef:0000:0000:0123
```

最後の32ビットについてはIPv4の表記方法が使える．

```
::10.0.0.1
```