卒業論文 1998 年度 (平成10年度)

次世代ネットワーク環境移行のためのプロトコル変換機構に関する研究

慶應義塾大学 環境情報学部
廣瀬 謙治

kenji@sfc.wide.ad.jp

指導教員

村井 純

徳田 英幸

楠本 博之

中村 修

概要

次世代ネットワーク環境移行のための
プロトコル変換機構に関する研究

本研究は，IPv4 から IPv6 への移行をより円滑に進めるために，プロトコル変換機構を実装し，IPv4 ホストと IPv6 ホストの相互通信を可能にすることを目的とする．

インターネットプロトコルである IPv4 (Internet Protocol Version 4) は，その規模性の問題から限界を迎えつつある．そこで，次世代インターネットプロトコルである IPv6 (Internet Protocol Version 6) への移行を行わなければならない．このため，IPv6 インターネットが世界規模で構築され始め，IPv4 ネットワークと IPv6 ネットワークの混在環境がすでに形成されている．

しかし，IPv4 と IPv6 の通信互換性がないことから，移行期のインターネットは，移行のために大きなコストが生じ，円滑に移行が行われないという問題が発生している．IPv4 から IPv6 へとネットワーク環境が円滑に移行するためには，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必要である．

このような問題を解決するために，本研究では，IPv4 と IPv6 の透過的な双方向通信を可能にするプロトコル変換機構 HITS (Hirose Integrated Translation System) を提案し，設計，実装，評価を行った．そして，実際に混在環境下で相互通信が行えることを確認した．また，HITS を用いることで，管理コストを軽減していることを実証した．

慶應義塾大学環境情報学部

廣瀬 謙治

Abstract

A Research in a Protocol Translation Mechanism to Support Transition to the Next Generation Network Environment

The purpose of the research is to smooth the transition of the Internet from the IPv4(Internet Protocol version 4) based network to the IPv6(Internet Protocol version 6) based network.

The Internet based on the IPv4 has a severe problem in scalability. Therefore, the Internet has to switch its network layer protocol to the IPv6, the Internet Protocol of the Next Generation.

Also, some researchers already started to construct the IPv6-Internet in global scale. As a result, the Internet is going to be a kind of the mixtured protocol environment.

There is no direct way to communicate between an IPv4 based host and an IPv6 based host. This makes the transition difficult due to the huge communication overhead. Therefore, the transition is difficult and painful for the Internet. To make a smooth transition, it is necessary to make communication between an IPv4 environment and an IPv6 environment easy and effective.

To accomplish the objective, the author proposes a protocol translation mechanism HITS (Hirose Integrated Translation System) which bridges the gap between those incompatible protocols. The author also evaluates the implementation of HITS, and reveals the HITS helps to make the transition smooth.

Keio University,
The Faculty of Environmental Information
Kenji Hirose

目次

図 目 次

表 目 次

第1章.　序論

本章では，まず，本研究の背景として，IPv4とIPv6が混在する現状を述べ，その問題を説明し，更に本研究の目的を述べる．最後に，本論文中で用いられる用語の定義と本論文の構成について述べる．

第1節.　背景

現在のインターネットは，IPv4 (Internet Protocol Version 4)[1] と呼ばれるプロトコルによって運用されている．しかし，インターネットの爆発的な拡大にともなって，IPv4では解決できない問題点が発生している．そこで，IPv4にかわる次世代のインターネットプロトコルとして，IPv6 (Internet Protocol Version 6)[2] が考案された．

IPv4からIPv6への移行が開始され，IPv6の開発やアプリケーションの移植が活発に進んでいる．さらに，IPv6の実験ネットワークが世界規模で構築され，IPv6への移行は現実的なものとなりつつある．

IPv6への移行は，現存するIPv4ネットワークが一斉にIPv6ネットワークへと移行するのではない．インターネット全体のIPv6への移行は，緩やかに行われる必要がある．なぜなら，全世界のネットワークを一斉にIPv4からIPv6へと切替えると，IPv6に対応してないホストやルータが通信不能になり，インターネット全体に大きな混乱を生じてしまい，現実的な移行ではない．IPv6への移行は，まずサイト単位で進行する．その後，IPv6サイト同士が接続され，段階的にIPv6ネットワークが大きくなっていく．また，IETFでの議論の結果，IPv6への移行は，新たなネットワークがIPv6によって構築され，主要なサービスがIPv6によって提供される，という方針で移行を進めることに決まり，実際その方針にしたがって進められている．そして，段階的にIPv6ネットワークが大きくなっていく．最終的には，IPv4ネットワークとIPv6ネットワークの混在環境が構築され，主要なサービスがIPv6によって提供されていることが想定される．しかし，その後もインターネットがIPv6のみによって運用されることは想定されていない．

このように，インターネット全体のIPv6への移行は，緩やかに行われていく．

第 2 節.　問題提起

円滑な移行を進めるためには，IPv4 ネットワークと IPv6 ネットワークの混在環境下で，通信互換性のない両者のホスト同士が相互に通信できなければならない．なぜなら，移行初期段階では，新たなネットワークが IPv6 によって構築されたとしても，主要なサービスは IPv4 によって提供されている．その際，IPv4 ネットワークと IPv6 ネットワークの相互通信性がなければ，IPv6 へ移行することのメリットが失われ，IPv6 への円滑な移行への動機とならない．また，相互通信性がなければ，既存の IPv4 ネットワークが IPv6 へと移行する動機にならない．さらに，ある程度移行の段階が進み，新たに構築された IPv6 ネットワークが IPv4 ネットワークと比較して，大きくなってくると，主要なサービスが IPv6 によって提供されはじめる．すると，相互通信性がなければ，IPv4 インターネットは衰退してしまい，円滑に IPv6 へと移行できないことになる．そこで，IPv4 から IPv6 へとネットワーク環境が緩やかに移行するために，IPv4 ホストと IPv6 ホストが通信を行う手段が必要である．

IPv4 から IPv6 に移行するためには，段階的に IPv6 ネットワークを構築していくとともに，IPv4 ホストと IPv6 ホストが相互に通信できる必要がある．そのための方法として，IETF での議論の結果，デュアル・スタックとトンネリングという 2 つの技術を用いた移行戦略がとられている．

しかし，円滑な移行をさせるための戦略であるデュアル・スタックでは，移行段階において，相互に通信することが困難になると予想される．なぜなら，

1. IPv4 ホストがデュアル・スタックとなるための管理コストが増大する．
2. IPv4 アドレスの枯渇により，相互通信を確立させるためのデュアル・スタックホストに対して，IPv4 アドレスを一意に割り当てることができない．

という 2 つの理由が考えられる．

これらの問題から，デュアル・スタックによる相互通信は確立できない．つまり，既存の IPv4 ネットワークの IPv4 ホストは，デュアル・スタック化されることによって，IPv6 ネットワークの IPv6 ホストと通信することはできる．しかし，新たに構築された IPv6 ネットワークの IPv6 ホストは，たとえ，デュアル・スタック化しても，割り当てる IPv4 アドレスがないため，IPv4 ホストと通信することができない．これらの相互通信可否を表 1.1 にまとめる．括弧内は，通信の際に利用するインターネットプロトコルである．

表 1.1: 相互接続表

	IPv4 ホスト	IPv6 ホスト	デュアル・スタックホスト
IPv4 ホスト	○ (IPv4)	×	○ (IPv4)
IPv6 ホスト	×	○ (IPv6)	○ (IPv6)
デュアル・スタックホスト	○ (IPv4)	○ (IPv6)	○ (IPv6)

第3節. 本研究の目的

表1.1より，デュアル・スタックホストはどのホストとも通信できることがわかる．デュアル・スタックホストを利用することで，IPv4ホストとIPv6ホストの通信を可能にすることができる．

これは，デュアル・スタックホストをIPv6/v4トランスレータとして利用することで，IPv4ホストとIPv6ホストの相互通信を可能にする．また，サイト内の全てのホストをデュアル・スタックホストとする必要がないので，枯渇しているIPv4アドレスをトランスレータを利用することで，有効に利用できる．

つまり，前節で述べた問題を解決し，IPv4ホストとIPv6ホストの相互通信を可能にするためには，IPv6/v4トランスレータが不可欠である．

IPv6/v4トランスレータは，IPv6パケットをIPv4パケットに変換，あるいは，IPv4パケットをIPv6パケットに変換し，IPv6ホストとIPv4ホストの通信を可能にする．

そこで本研究では，以下の2点を目的とした研究を行った．

- 通信プロトコルに依存しない通信の透過性を確保する

  IPv6/v4トランスレータによる新たなネットワークの構築，あるいは，IPv6/v4トランスレータの既存のネットワークへの導入により，近い将来起こるであろうIPv4ネットワークとIPv6ネットワークの混在環境での，通信プロトコルに依存しない通信の透過性を確保する．

- IPv6への円滑かつ迅速な移行を促す

  IPv4ネットワークからIPv6ネットワークへの接続性を提供することにより，IPv6ネットワークで提供されているサービスを享受することができ，IPv6

へのより円滑かつ迅速な移行を促すことができる．

本研究の実現により，IPv4ホストとIPv6ホストの透過的な双方向通信を提供する．また，特にIPv6への移行を促すためにIPv4ホストからIPv6ホストへの接続性を提供する．

第4節. 用語の定義

本論文中で使用される用語をあらかじめ表1.2に定義する．

表1.2: 用語の定義

用語	本論文中で用いられる意味
IPv4ホスト	IPv4プロトコルスタックのみを持つホスト
IPv6ホスト	IPv6プロトコルスタックのみを持つホスト
デュアル・スタックホスト	IPv4，IPv6両方のプロトコルスタックを持つホスト
IPv4ネットワーク	IPv4によって構築されているネットワーク
IPv6ネットワーク	IPv6によって構築されているネットワーク
IPv4サイト	組織内のネットワークがIPv4によって構築されている組織
IPv6サイト	組織内のネットワークがIPv6によって構築されている組織
IPv4インターネット	IPv4ネットワークの集合
IPv6インターネット	IPv6ネットワークの集合

第5節. 本論文の構成

第2章では，IPv6への移行方法と移行時の問題点を明確にするため，移行技術の現状の方法と，それらの問題点について言及する．第3章では，トランスレータの関連研究について述べる．第4章では，その解決方法として，IPv6/v4トランスレータHITS (Hirose Integrated Translation System)を提案する．さらに，HITSの設計について述べる．第5章では，システムの実装目標を示し，HITSの設計に基づいた実装について述べる．第6章では，本システムを運用した評価について述べる．最後に，第7章では，本研究のまとめと今後の展望を示す．

第2章.　IPv6への移行

この章では，IPv4とIPv6を相互接続するための現在の方法を説明し，さらにこの方法において移行の際に生じる問題について述べる．

第1節.　相互接続技術

相互接続技術には次のものがある．

- デュアル・スタック
- トンネリング
- トランスレータ
  - ヘッダ変換
  - TCP中継
  - アプリケーション中継

これらの技術について次に説明していく．

第1.1節　デュアル・スタック

図2.1に示すように，デュアル・スタックとは，一つのホストが複数のネットワーク層のプロトコルを装備することである．つまり，ホストがIPv6の機能を持つと同時に，IPv4の機能を合わせ持つことである．この方式では，IPv4ホストとIPv6ホストどちらの通信も可能なので，相互接続性を確保できる．デュアル・スタックホストは，IPv4ホストと通信するにはIPv4を用い，IPv6ホストと通信するにはIPv6を使用する．

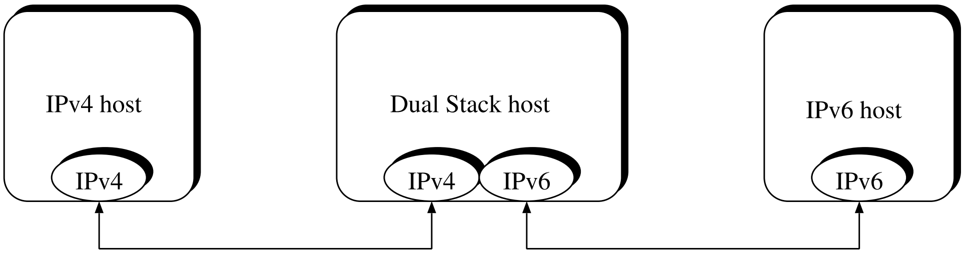


図 2.1: デュアル・スタック

第 1.2 節　トンネリング

トンネリングとは，あるネットワーク層の同種のプロトコルを持つホスト同士が，異なるネットワーク層のプロトコルを利用して通信を行う技術である．トンネリング技術を用いることで，プロトコルの異なる物理的なネットワークを利用して，論理的なリンクを構築することが可能である．この技術は，既存のネットワークを維持したまま，新たなネットワークを構築する際に利用される．

IPv6 への移行の際に利用されるトンネリングは次の 2 種類に分類できる．

- IPv6 over IPv4 トンネル

  図 2.2 に示すように IPv4 ネットワーク上に，IPv6 リンクを構築するためのトンネルである．図 2.3 のように IPv6 パケットに IPv4 ヘッダを付加し，IPv6 パケットをあたかも IPv4 パケットのように見せる．これにより，物理的な IPv4 ネットワークを利用して，論理的な IPv6 ネットワークを構築し，IPv6 パケットを転送することを可能とする．

- IPv4 over IPv6 トンネル

  図 2.4 に示すように IPv6 ネットワーク上に，IPv4 リンクを構築するためのトンネルである．図 2.5 のように IPv4 パケットに IPv6 ヘッダを付加し，IPv4 パケットをあたかも IPv6 パケットのように見せるこれにより，物理的な IPv6 ネットワークを利用して，論理的な IPv4 ネットワークを構築し，IPv4 パケットを転送することを可能とする．

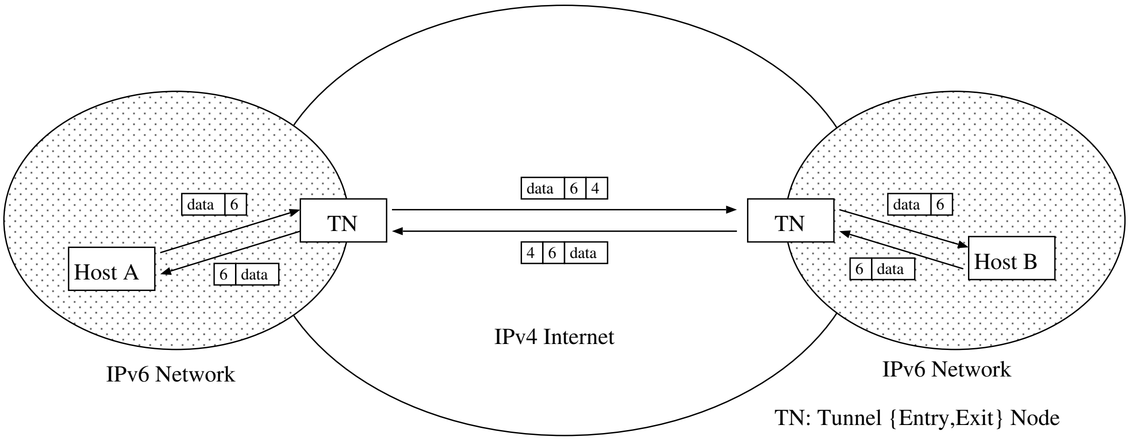


図 2.2: IPv6 over IPv4 トンネル

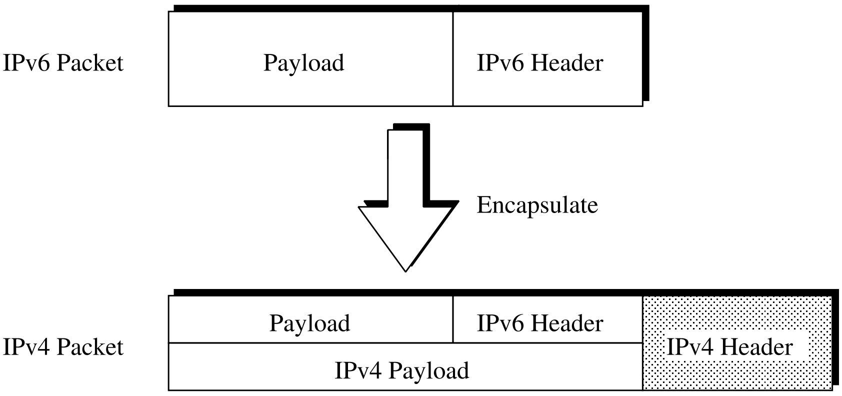


図 2.3: IPv6 パケットのカプセル化

トンネリングを用いることで，直接の接続性がないIPv6ホスト同士，あるいは，IPv4ホスト同士が，異なるネットワークプロトコルをリンク層として利用することによって，通信可能となる．

第 1.3 節　トランスレータ

トランスレータとは，プロトコルの変換を行うことにより，異種のプロトコルを持つホスト同士を，同種のプロトコルを持つホスト同士であるかのように通信させることを可能にする技

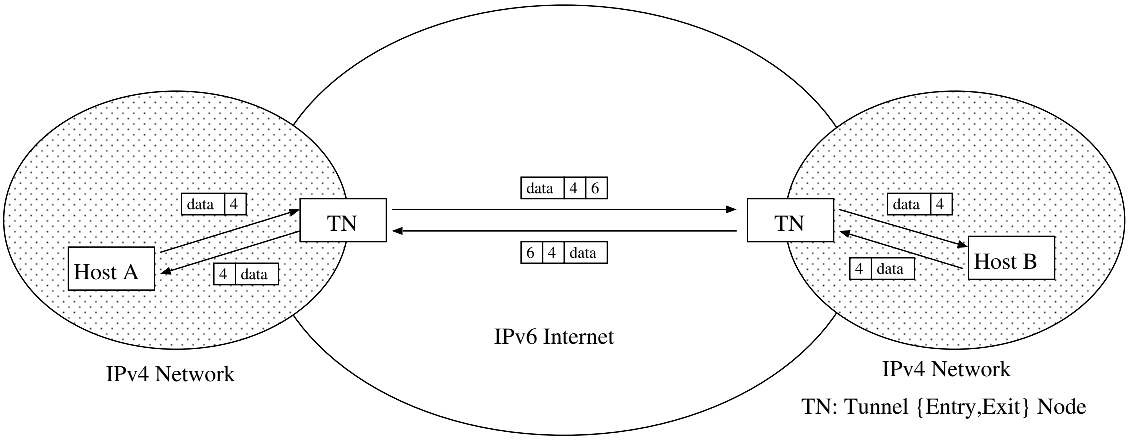


図 2.4: IPv4 over IPv6 トンネル

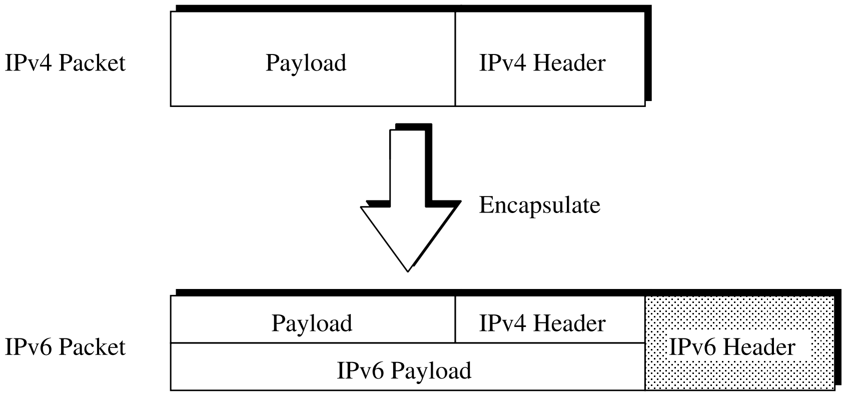


図 2.5: IPv4 パケットのカプセル化

術である．この技術は，過去のプロトコルとの互換性を保つために利用される．

IPv6 への移行に利用されるトランスレータは，IPv6 → IPv4 変換と IPv4 → IPv6 変換を行うトランスレータである．IPv6 への移行に利用されるトランスレータを変換方式で分類すると，ヘッダ変換，TCP 中継，アプリケーション中継の 3 種類に分類される．これは，draft-ietf-ngtrans-translator-01[3] による分類である．この分類についての説明を次に示す．

- ヘッダ変換

  図 2.6 に示すようにヘッダ変換は，途中経路でパケットそのもののヘッダを変換する技術である．その例として，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 が存在する．NAT は，IPv4 から IPv4 へのヘッダ変換を行う．

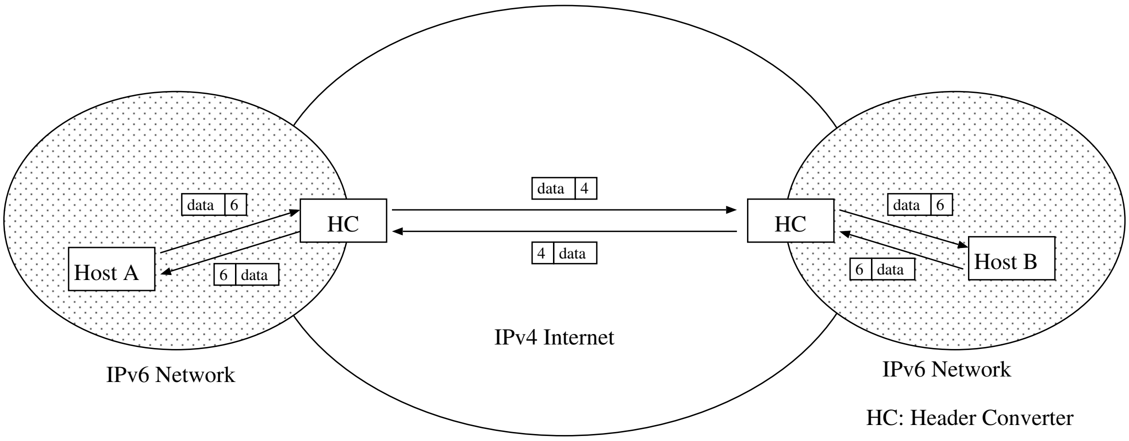


図 2.6: ヘッダ変換

- TCP 中継

図 2.7 に示すように TCP 中継は，中継サーバが，通信の中間に入って TCP の接続を中継することによって，IPv6 と IPv4 の変換を行う方式である．TCP の接続単位に独立して変換が行われる．この方式は，トランスポート層の変換を行う．この方式を利用した実装として，FAITH がある．

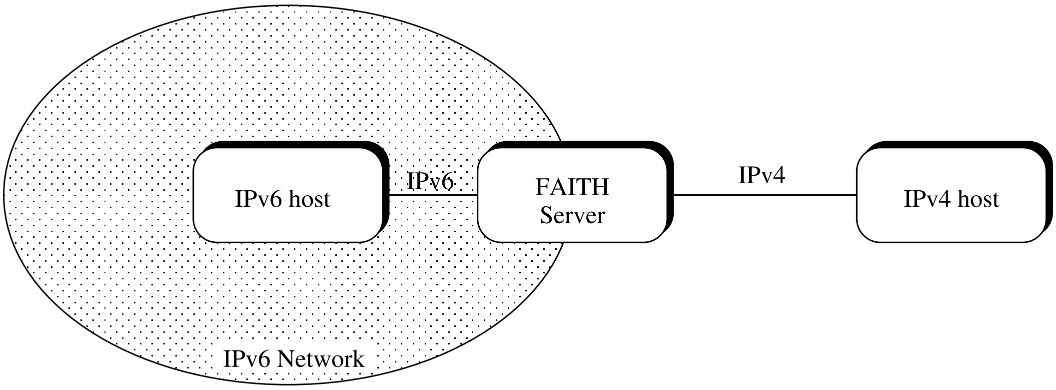


図 2.7: TCP 中継

- アプリケーション中継

  図 2.8 に示すようにアプリケーション中継は，通信経路の途中のゲートウェイで，アプリケーションごとに中継サーバを設置し，アプリケーション層で通信を一旦終える．そして，改めて終点コネクションを確立する方法である．この方式を利用した実装として，SOCKS64[4] や SOCKS による変換 [5] などがある．

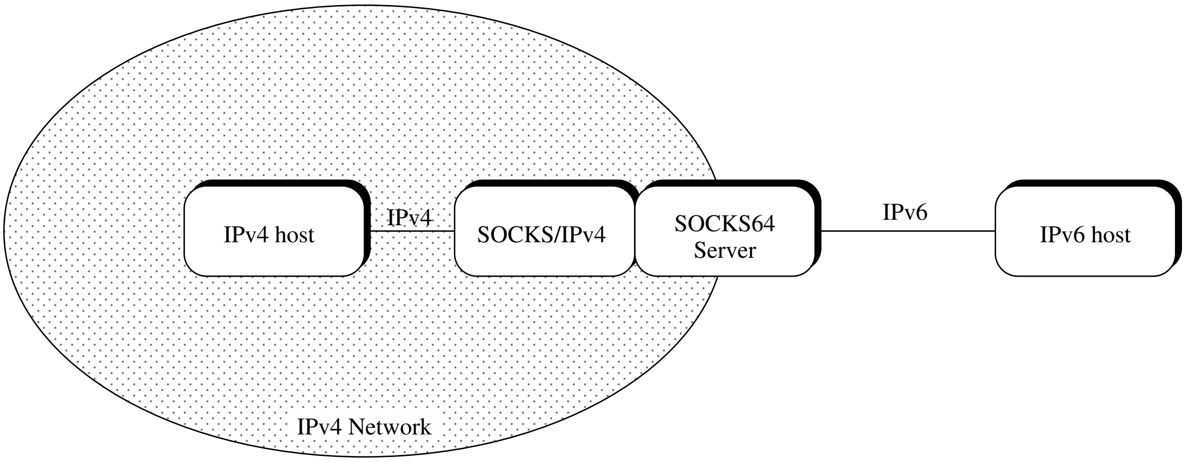


図 2.8: アプリケーション中継

また，IPv6 への移行に利用されるトランスレータを，変換の方向で分類すると，表 2.1 の 4 種類に分類できる．これは論文「Deployment and Experiences of WIDE 6bone」[6] による分類である．

表 2.1: 変換の方向によるトランスレータの分類

種別	変換の方向	ネットワークの大小	適用される時期
トランスレータ A	IPv6 → IPv4	IPv6 < IPv4	移行初期
トランスレータ B	IPv4 → IPv6	IPv6 < IPv4	移行初期
トランスレータ C	IPv4 → IPv6	IPv6 > IPv4	移行後期
トランスレータ D	IPv6 → IPv4	IPv6 > IPv4	移行後期

第 2 節.　移行とその問題点

現在，IPv6 の開発やアプリケーションの移植が活発に進み，IPv6 への移行はより現実的なものとなりつつある．その一方で，移行を円滑に進めるためには，IPv4 ネットワークと IPv6 ネットワークの混在環境下で，両者のホストが相互に通信できる必要がある．

そこで IETF の ngtrans 分科会では，デュアル・スタックによって，IPv4 と IPv6 の相互通信を可能とし，IPv6 over IPv4 トンネルを用いたトンネリングによって，論理的な IPv6 ネットワークを構築するという 2 つの技術を利用した移行戦略が提唱されている．

しかし，IPv4 と IPv6 の相互通信のためのデュアル・スタック戦略は，IPv4 アドレスの枯渇という欠点がある．IPv4 アドレスの枯渇が深刻な事態になるにつれて，新たに増加するホストに IPv4 アドレスを割り当てることが困難となる．したがって，デュアル・スタックホストであるにも関わらず，IPv6 アドレスしか持たないホストが出現する．IPv6 ホストすべてに IPv4 アドレスを割り当てることは難しく，デュアル・スタックによる移行は困難である．つまり，IPv4 から IPv6 への移行のためには，デュアル・スタック以外の IPv4 と IPv6 の相互通信を可能にする技術が必要である．そこで，トランスレータが必要となる．

しかし，既存の移行手法である，トランスレータにも問題がある．まず，ヘッダ変換は，異なるアドレス空間を持つ IPv4 と IPv6 のアドレスを対応づける方法が問題となる．また，FTP のようなペイロード部に IP アドレスを含んでいるような階層違反のプロトコルは，ヘッダ変換だけでは，相互接続することができない．

TCP 中継は，TCP の接続単位に独立して変換が行われるため，あらかじめ利用が想定されてないアプリケーションや UDP の変換は不可能である．

アプリケーション中継は，途中の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異なるアドレス空間を持つ IPv4 と IPv6 のアドレスを対応づける方法が問題となる．

こういった問題から，既存のトランスレータは，次に述べる 4 つの問題が生じている．

- 変換方向が限定されている

  現在ある実装の多くは，図 2.9(2) の移行初期における IPv6 から IPv4 への通信を可能にするもの，図 2.10(3) の移行後期における IPv6 から IPv4 への通信を可能にするもの，図 2.10(4) の移行後期における IPv4 から IPv6 への通信を可能にするものであり，図 2.9(1) の移行初期における IPv4 から IPv6 への

通信を可能にするものはない．つまり，特に移行初期における IPv4 から IPv6 への通信を実現できていないのである．これでは，IPv6 で提供あれているサービスを IPv4 ホストでは享受できず，既存の IPv4 ネットワークが IPv6 へと移行する動機にならない．

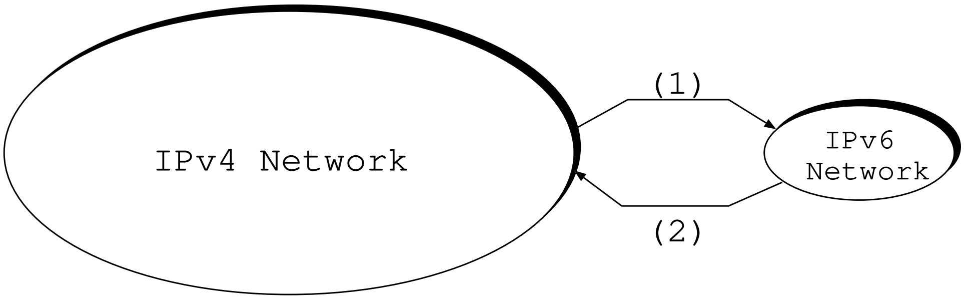


図 2.9: 移行初期

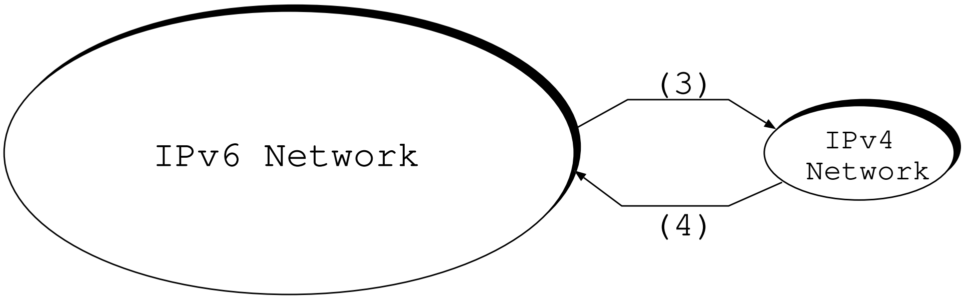


図 2.10: 移行後期

- 移行段階に応じて変更が必要である

　表 2.1 で示しているように，移行段階に応じて，ネットワーク管理者は変換の方向を変更したり，適用するネットワークを適切に設定するコストが必要である．

- IPv4 アドレスの枯渇が考慮されていない

　トランスレータは，変換すべき IPv4 アドレスと IPv6 アドレスの対応が必要であるが，枯渇しつつある IPv4 アドレスを有効に活用されていない．

- 導入コスト

  既存のネットワークにトランスレータを導入する際，あるいは，新規のネットワークをトランスレータを用いて構築する際に生じるコストも考慮しなければならない．なぜなら，導入コストあるいは管理コストが高いと，トランスレータを導入することが困難となる．

第3章.　関連研究

本章では，トランスレータの関連研究について述べる．第 2.1.3 節で述べた，トランスレータの 3 つの方式の代表的な実装である NR60，FAITH，SOCKS64 について述べていく．

第1節.　NR60

日立が開発したトランスレータ機能 [7] 付きのルータである．ヘッダ変換方式を用いたトランスレータで，IPv4 アドレスと IPv6 アドレスの対応付けは，改良ネームサーバによってなされている．

NR60 のトランスレータとしての特徴を次に述べる．

- 利点
  - カーネル空間で実現されているのでパケットの変換が速い．
  - ゲートウェイ上でサーバを立ちあげる必要がない．
  - プロトコルに依存しないヘッダ変換ができるので，未知のプロトコルにも対応できる．
  - クライアントには一切手を加えなくて良い．
- 欠点
  - アドレス対応表を管理する必要がある．

第2節.　FAITH

FAITH は TCP 中継のトランスレータである．ユーザは外部のホスト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ホストと通信する．IPv4 アドレスと IPv6 アドレスの対応付けは，改良ネームサーバによってなされている．

FAITH の特徴を次に述べる．

- 利点
  - 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．
  - クライアントには一切手を加えなくて良い．
  - コネクションを終端するので，きめの細かいアクセス制御ができる．
- 欠点
  - アプリケーション空間で実現されているので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立ちあげるため負荷が集中する．
  - アドレス対応表を管理する必要がある．

第 3 節.　SOCKS64

SOCKS64[4] は米国 NEC が開発した SOCKS[8] を，富士通研究所が 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．本来，SOCKS は Firewall を構築するための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プロトコル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，IPv4 アドレスと IPv6 アドレスの対応づけの問題を解決している．

SOCKS64 の特徴を次に述べる．

- 利点
  - 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である．
  - コネクションを終端するので，きめの細かいアクセス制御ができる．
  - すでに 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．

- 欠点
  - アプリケーション空間で実現されているので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立ちあげるため負荷が集中する．
  - 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．

第 4 節.　まとめ

これらを比較したものを表 3.1 に示す．

表 3.1: トランスレータの比較

機能	NR60	FAITH	SOCKS64
パケット転送速度	速い	遅い	遅い
アドレス対応表	必要	必要	不要
プロトコル毎のサーバ	不要	必要	必要
階層違反プロトコル	困難	可能	可能
導入コスト	低い	低い	高い

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 NR60 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 や SOCKS64 に比べ，NR60 は代理サーバが不要で未知のプロトコルにも対応できる．

しかし，企業内の Firewall で限られたプロトコルのみを通せば良い場合には，FAITH や SOCKS64 の方が便利である．すでに SOCKS を導入しているサイトならば，SOCKS ゲートウェイを SOCKS64 化するだけでよい．

第4章.　HITSの設計

本章では，まず設計方針を述べ，システムの概要，最後に，動作について述べる．

第1節.　HITSの設計方針

前節で述べた問題点を解決し，IPv4からIPv6へのの移行コストを削減し，より円滑な移行を促すためには，第3章で述べたように，既存のトランスレータの機能では不十分である．

そこで，本研究では，上記の項目を満たし，移行期のの管理コストを軽減することのできるプロトコル変換機構として，HITS (Hirose Integrated Translation System) を提唱する．

HITSは，IPv6ネットワークとIPv4ネットワークの双方向通信を目的としたシステムである．HITSの設計方針を次に述べる．

- IPv6/v4双方向通信機能

  IPv4からIPv6への移行を支援するために，通信互換性のない両者の双方向通信の実現，特に移行初期におけるIPv4からIPv6への通信を実現する．

- すべての移行段階において利用できる

  移行段階の変化に関わらず，ネットワーク管理者が設定を変更することなく，利用できる．

- IPv4アドレスの枯渇に対応する

  少ないIPv4アドレスを用いて，より多くのホスト間の通信を可能にする．

- 導入コストを小さくする

  トランスレータを導入するコストを小さくすることで，移行を容易し，より円滑な移行を支援する．

IPv4とIPv6のヘッダ変換を行うには，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，前述したようにIPv6アドレスを一意にIPv4アドレスに割り当てることはできない．

そこで，本システムでは，グローバルな IPv4 アドレスを動的に割り当てる方式で対応づけを行う．この IPv4 アドレスの割り当ては，Fake DNS を通じて，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．

第 2 節.　HITS の概要

前節で述べた設計方針に基づいて，HITS を設計した．HITS の前提条件を次に述べる．

- 図 4.1 に示すように，あるネットワークのゲートウェイとして設置する
- そのネットワークのプライマリ DNS となる
- IPv4 でも IPv6 でも接続可能である．

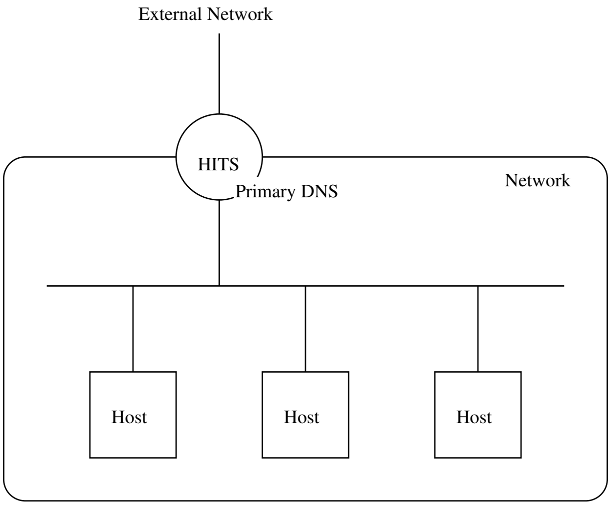


図 4.1: HITS の設置

第3節.　HITSの構成要素

HITSは，Header Converter，Auto Mapper，Fake DNSから構成される．

これまでにも，DNSを利用したトランスレータ[9]は存在したが，本システムでは，IPv4アドレスの枯渇とIPv6への円滑な移行を考慮して，Auto Mapperを用いている．

それぞれの構成要素について説明する．

第3.1節　Header Converter

IPv4とIPv6のヘッダ変換を行う．ヘッダ変換の詳細は，第4.5節で述べる．

Header Converterは，変換受け付けモジュール，変換モジュール，アドレス要求・返答モジュールの3つから構成される．

ヘッダ変換を行う際は，次の手順をふむ．

1. 変換受け付けモジュールは，変換すべきIPv4アドレスとIPv6アドレスの始点アドレスと終点アドレスの対応をアドレス要求・返答モジュールに要求する．
2. アドレス要求・返答モジュールは，Auto Mapperに要求を送信する．
3. アドレス要求・返答モジュールは，Auto Mapperからの返答を受信し，変換受け付けモジュールに返答を行う．
4. 変換受け付けモジュールは，Auto Mapperから，変換すべきIPv4アドレスとIPv6アドレスの始点アドレスと終点アドレスの対応の返答を受ける．
5. 返答された情報を変換モジュールに送信する．
6. 変換モジュールが変換を開始する．

第3.2節　Auto Mapper

Auto Mapperは，アドレスの割当とIPv4アドレスとIPv6アドレスの対応付けの管理を行い，次の2つのモジュールから構成される．

- アドレス割り当てモジュール

  要求に応じて，割り当て可能なアドレスを検索し，アドレスの割り当てを行う．未割り当てのアドレスがあれば，それを割り当てる．無ければ，アドレスの使用期限が来たこととし，アドレスの対応づけを削除し，新たに割り当てを開始する．その後，割り当てた情報をデータベースモジュールに送信する．

- データベース管理モジュール

  IPv4 アドレスと IPv6 アドレスの対応付けのデータベースを管理するデータベースには，ホスト名，IPv4 アドレス，IPv6 アドレス，割り当てられて時刻が登録される．また，再割当を行う際には，古い情報を削除し，新たな情報を追加する．

Auto Mapper では，スプールされているグローバルな IPv4 アドレスを動的に割り当てることにより，少ない IPv4 アドレスを有効に利用でき，IPv4 アドレスの枯渇に対応している．また，グローバルな IP アドレスを割り当てることで，IPv4 と IPv6 の双方向通信が可能となるので，IPv6 への円滑な移行を支援できる．

第 3.3 節　Fake DNS

ホストから要求されたレコード (A or AAAA) がない場合に，Auto Mapper にアドレスの割当を要求する．その返答を受けて，割り当てられたアドレスをホストに返答する．ホストから要求されたレコード (A or AAAA) がある場合は，そのまま返答する．

第 4 節.　HITS の動作

本システムを利用した通信手順について説明する．

第 4.1 節　同種のプロトコルを持つホスト同士の通信

まず，比較対象として，IPv4 ホスト同士が，通常の通信を行う場合について説明する．

サイト A にあるホスト a がサイト B にあるホスト b と通信したいとする．a，b のホスト名，IPv4 アドレスは表 4.1 の通りである．

表 4.1: 通常の通信におけるホスト名と IPv4 アドレス

	ホスト名	IPv4 アドレス
ホスト a	a.v4.ac.jp	1.2.3.4
ホスト b	b.v4.co.jp	5.6.7.8

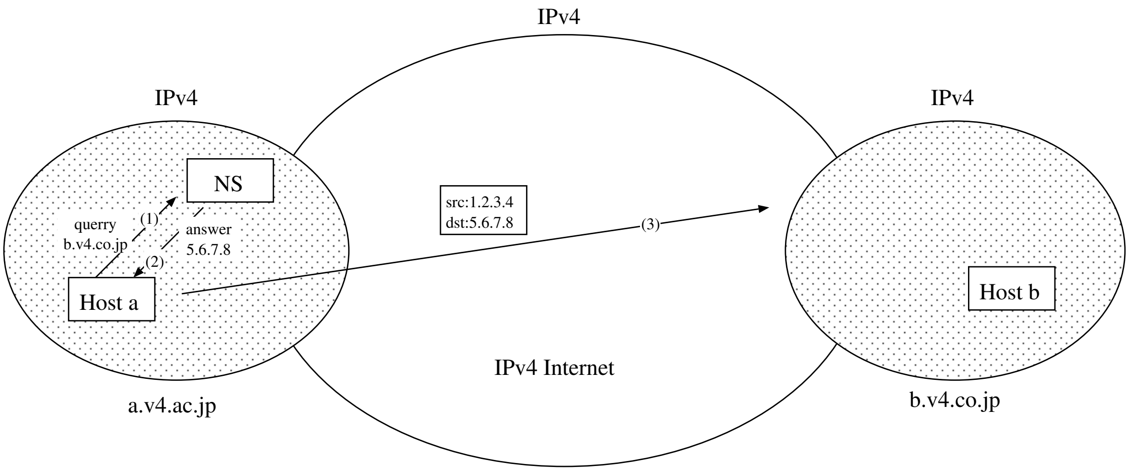


図 4.2: 通常の通信手順

図 4.2 を用いて，通常の通信手段について説明する．図中の括弧は，動作の順番である．

1. a はネームサーバに b の IPv4 アドレス (A レコード) を問い合わせる．
2. ネームサーバは，b の IPv4 アドレス 5.6.7.8 を a に返答する．
3. a は b の IPv4 アドレスがわかり，始点アドレス 1.2.3.4，終点アドレス 5.6.7.8 の IPv4 パケットを送信する．

これが IPv4 ホスト同士の通信方法である．IPv6 の場合は，A レコードではなく，IPv6 アドレスを意味する AAAA レコードを利用して，同様に通信を行う．

第 4.2 節　変換を行う際の通信

次に，IPv4 ホストと IPv6 ホストが，本システムを用いて，通信を行う場合について説明する．IPv6 と IPv4 の相互通信を透過的に行える機構の全体像を図 4.3 に示す．

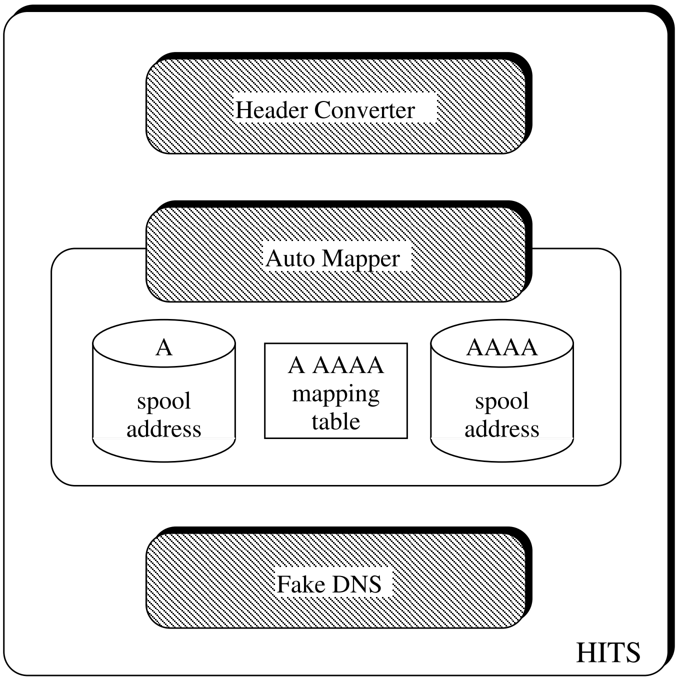


図 4.3: HITS の全体像

IPv6 ネットワークにおける HITS の動作 (IPv6 ホスト → IPv4 ホスト)

まず，図 2.9 の IPv6 ホストから，IPv4 ホストへの通信方法を説明する．図 4.4 にその動作を示す．図中の括弧は，動作の順番である．

(1) IPv6 ホストは，Fake DNS に対して，IPv4 ホストの AAAA レコードを要求する．

(2) Fake DNS は，A レコードしか返答されないことがわかると，Auto Mapper にスプールされている IPv6 アドレスを，IPv4 ホストの AAAA レコードとして，割り当ててもらう．

(3) Auto Mapper は IPv4 ホストの A レコードと割り当てた AAAA レコードの対を保持する．

(4) Fake DNS は IPv6 ホストに IPv4 ホストの仮の AAAA レコードを返答する．

(5) IPv6 ホストはそれを用いて，IPv4 ホストとの通信を開始する．

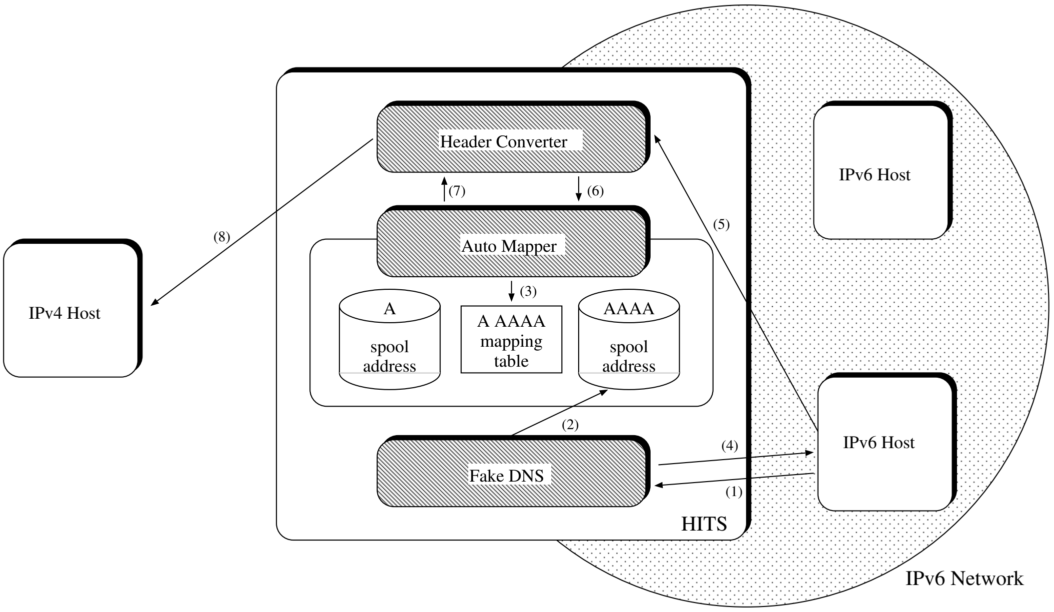


図 4.4: IPv6 ネットワークにおける HITS の動作その 1

(6) Header Converter は，Auto Mapper より変換すべき IPv4 ホストの対を要求する．

(7) Header Converter は，その返答を受け取る．

(8) Header Converter は，ヘッダ変換を行う．

IPv6 ネットワークにおける HITS の動作 (IPv4 ホスト → IPv6 ホスト)

逆に IPv4 ホストから IPv6 ホストへの通信方法について説明する．図 4.5 にその動作を示す．図中の括弧は，動作の順番である．

(a) IPv4 ホストは Fake DNS に IPv6 ホストの A レコードを要求する．Fake DNS は IPv6 ネットワークの DNS になっており，IPv6 ホストの AAAA レコードを TTL=0 で広告しているので，IPv4 ホストは IPv6 ホストの A レコードを必ず，この Fake DNS に対して要求する．

(b) Fake DNS は IPv6 ホストに AAAA レコードしか返答されないことがわかると，Auto Mapper にスプールされているグローバルな IPv4 アドレスを IPv6 ホストの A レコードとして割り当ててもらう．

(c) Auto Mapper は IPv6 ホストの AAAA レコードと割り当てた A レコードの対を保持する．

(d) Fake DNS は IPv4 ホストに IPv6 ホストの仮の A レコードを返答する．

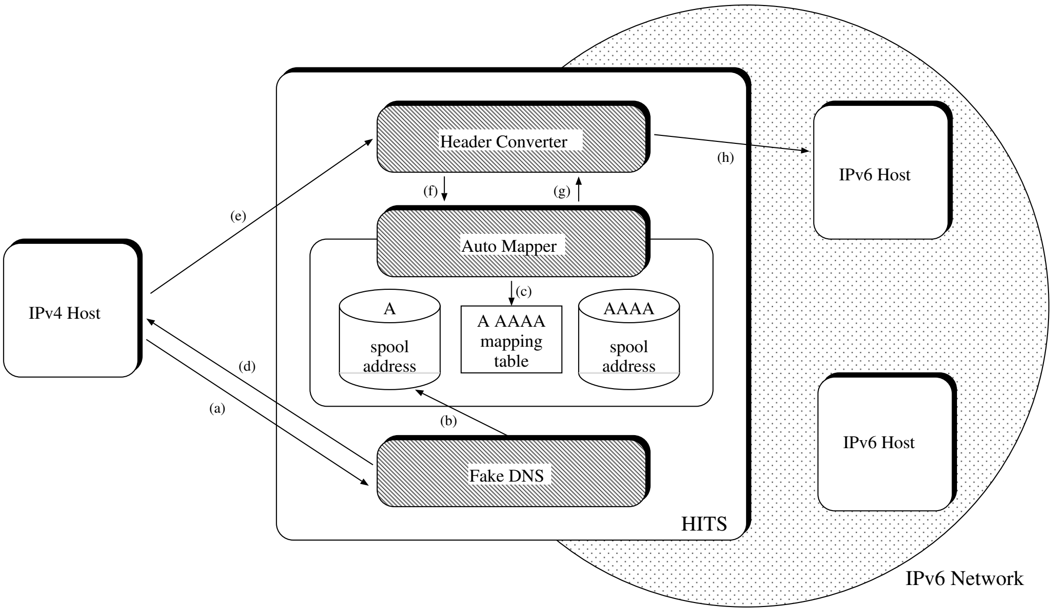


図 4.5: IPv6 ネットワークにおける HITS の動作その 2

(e) IPv4 ホストはそれを用いて，IPv6 ホストとの通信を開始する．

(f) Header Converter は，Auto Mapper より変換すべき IPv6 ホストの対を要求する．

(g) Header Converter は，その返答を受け取る．

(h) Header Converter は，ヘッダ変換を行う．

DNS のキャッシュによるアドレスのギャップ

ここで注意すべき点は，Auto Mapper で再割り当てを行う際には，DNS のキャッシュが消える前に，次のアドレスの割り当てを行わないようにしなければならないということである．

DNS のキャッシュによる IP アドレスのギャップによって通信が確立できない場合について説明する．この際のアドレスは表 4.2 とし，Auto Mapper にスプールされている IPv4 アドレスは 2.2.2.2 のみとする．

図 4.6 のように，IPv4 ネットワークの Host a が IPv6 ネットワークの Host A と通信をしようとしたとする．Host a は Host A と通信を開始するために，Host A の A レコードを要求する．しかし，Host A は IPv6 ネットワークにあり，A レコードを持っていないため，2.2.2.2 を Host A の IPv4 アドレスとして，Auto Mapper に割り当ててもらう．そして，Auto Mapper は Host A の A レコードと AAAA レコードの対応表を持ち，通信が開始される．Host a と Host

表 4.2: DNS のキャッシュによるギャップのホストとアドレス対応表

	IPv4 アドレス	IPv6 アドレス
ホスト a	1.1.1.1	
ホスト A		1234::1
ホスト B		1234::2

A の通信が終了し，次に図 4.7 のように Host B がある Host と通信を開始したとする．すると，Auto Mapper は，Host A に割り当てた IPv4 アドレスを回収し，Host B に割り当てて，持っている対応表を更新し，Host B は通信を開始する．その時に，再び Host a が Host A との通信を再開したとする．DNS のキャッシュが有効になっている場合，広告されているキャッシュ，つまり，Host A は 2.2.2.2 であるというキャッシュを利用して，通信を開始しようとする．ところが，2.2.2.2 という IPv4 アドレスは現在は，Host B に対して，割り当てられているもので，当然，Host a はキャッシュを利用して通信を行うと，Host B と通信することになってしまう．

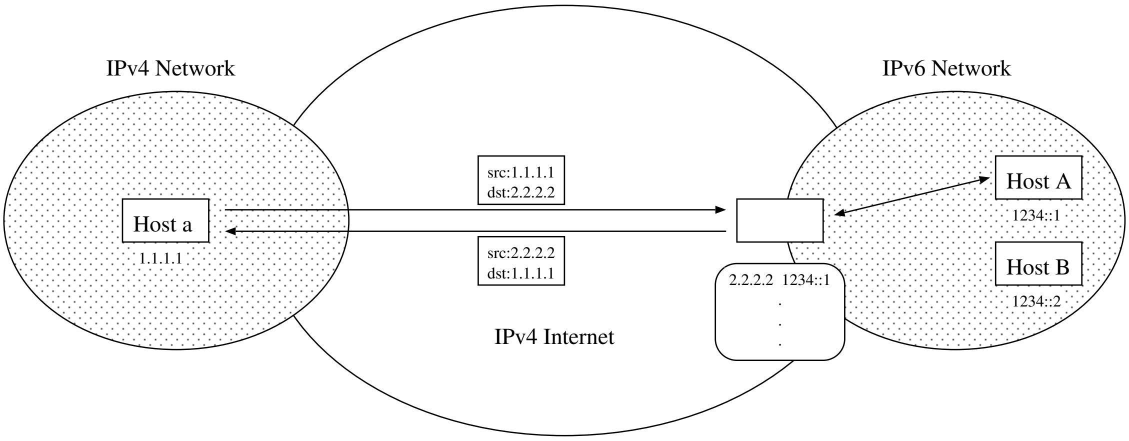


図 4.6: DNS のキャッシュによるギャップその 1

この問題を解決するために，Auto Mapper は再割当を行う際にある程度の時間をおく必要がある．これにより Auto Mapper によるマッピングとキャッシュとのギャップを解消できる．

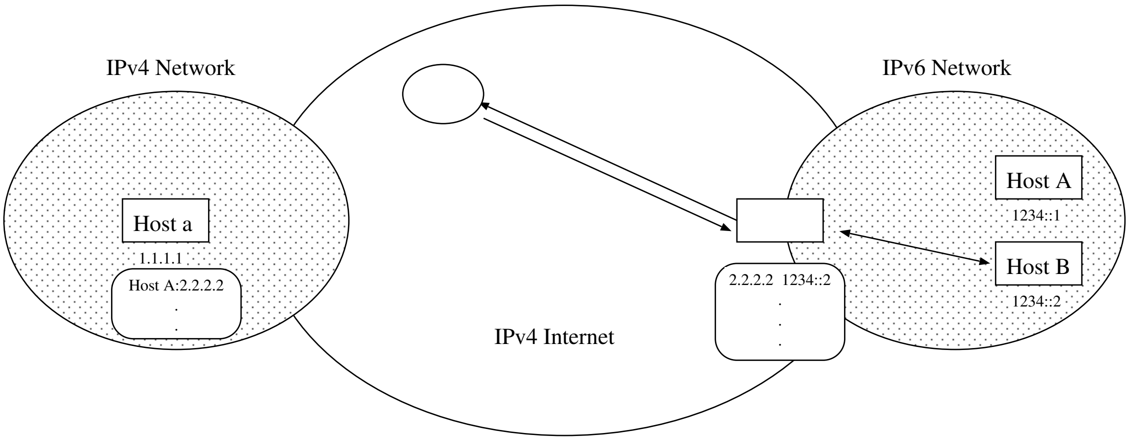


図4.7: DNSのキャッシュによるギャップその2

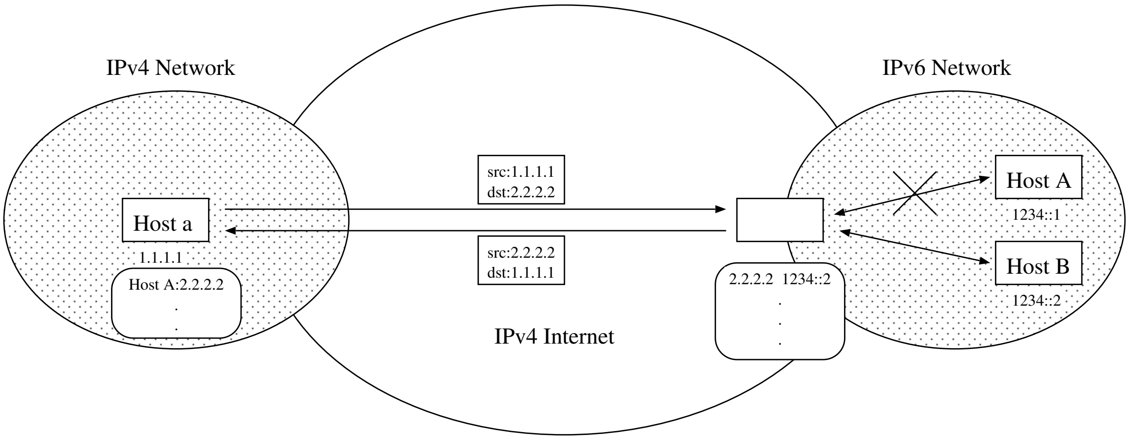


図4.8: DNSのキャッシュによるギャップその3

IPv4ネットワークにおけるHITSの動作 (IPv4ホスト → IPv6ホスト)

次に，図2.10の場合も同様に，動的にアドレスを割り当てることで，本システムにより双方向の通信を提供できる．また，IPv4ホストからIPv6ホストへ通信を行う際には，IPv4ホスト自身のIPv4アドレスとIPv6アドレスを静的に一意に対応づけることも可能である．IPv6ホストからIPv4ホストへ通信を行う際も，互換アドレスを用いて，IPv6アドレスをIPv4アドレスに変換する方法も可能である．

実際に，図2.10のIPv4ホストからIPv6ホストへの通信方法について説明する．図4.9にその動作を示す．図中の括弧は，動作の順番である．

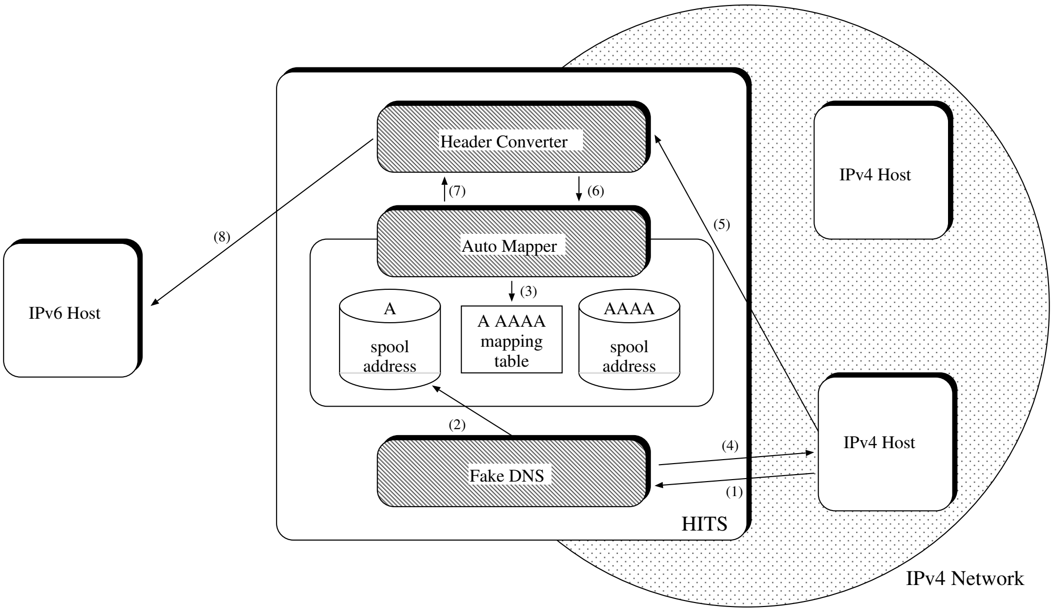


図 4.9: IPv4 ネットワークにおける HITS の動作その 1

(1) IPv4 ホストは，Fake DNS に対して，IPv6 ホストの A レコードを要求する．

(2) Fake DNS は，AAAA レコードしか返答されないことがわかると，Auto Mapper にスプールされている IPv4 アドレスを，IPv6 ホストの A レコードとして，割り当ててもらう．

(3) Auto Mapper は IPv6 ホストの AAAA レコードと割り当てた A レコードの対を保持する．

(4) Fake DNS は IPv4 ホストに IPv6 ホストの仮の A レコードを返答する．

(5) IPv4 ホストはそれを用いて，IPv6 ホストとの通信を開始する．

(6) Header Converter は，Auto Mapper より変換すべき IPv6 ホストの対を要求する．

(7) Header Converter は，その返答を受け取る．

(8) Header Converter は，ヘッダ変換を行う．

IPv4 ネットワークにおける HITS の動作 (IPv6 ホスト → IPv4 ホスト)

逆に IPv6 ホストから IPv4 ホストへの通信方法について説明する．図 4.10 にその動作を示す．図中の括弧は，動作の順番である．

(a) IPv6 ホストは Fake DNS に IPv4 ホストの AAAA レコードを要求する．Fake DNS は IPv4 ネットワークの DNS になっており，IPv4 ホストの A レコードを TTL=0 で広告している

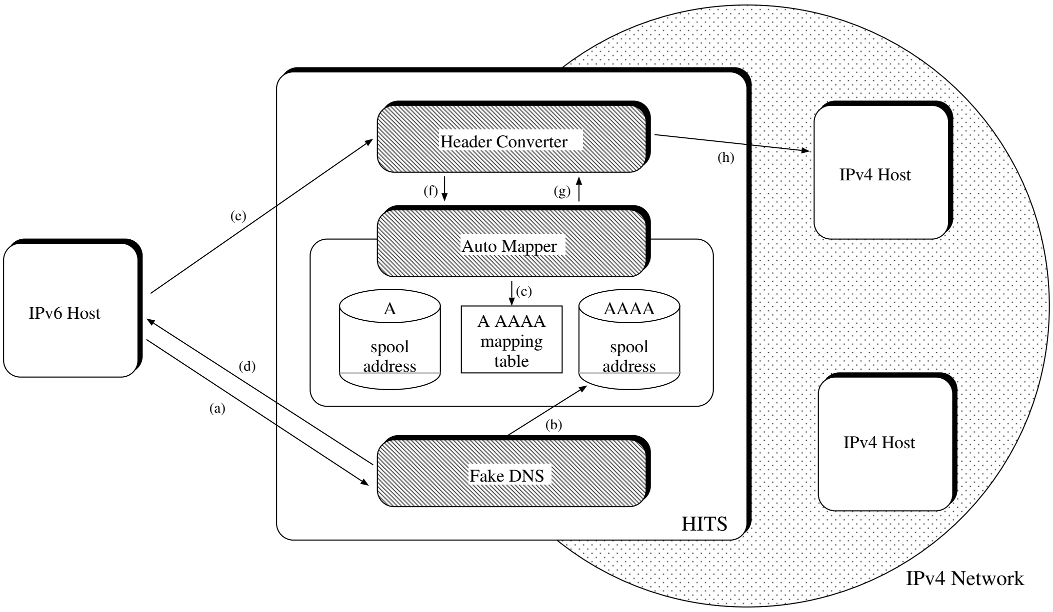


図 4.10: IPv4 ネットワークにおける HITS の動作その 2

ので，IPv6 ホストは IPv4 ホストの AAAA レコードを必ず，この Fake DNS に対して要求する．

(b) Fake DNS は IPv4 ホストに A レコードしか返答されないことがわかると，Auto Mapper にスプールされているグローバルな IPv6 アドレスを IPv4 ホストの AAAA レコードとして割り当ててもらう．

(c) Auto Mapper は IPv4 ホストの A レコードと割り当てた AAAA レコードの対を保持する．

(d) Fake DNS は IPv6 ホストに IPv4 ホストの仮の AAAA レコードを返答する．

(e) IPv6 ホストはそれを用いて，IPv4 ホストとの通信を開始する．

(f) Header Converter は，Auto Mapper より変換すべき IPv4 ホストの対を要求する．

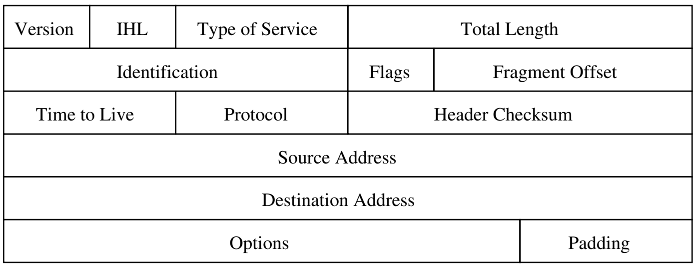
(g) Header Converter は，その返答を受け取る．

(h) Header Converter は，ヘッダ変換を行う．

第 5 節.　ヘッダ変換の詳細

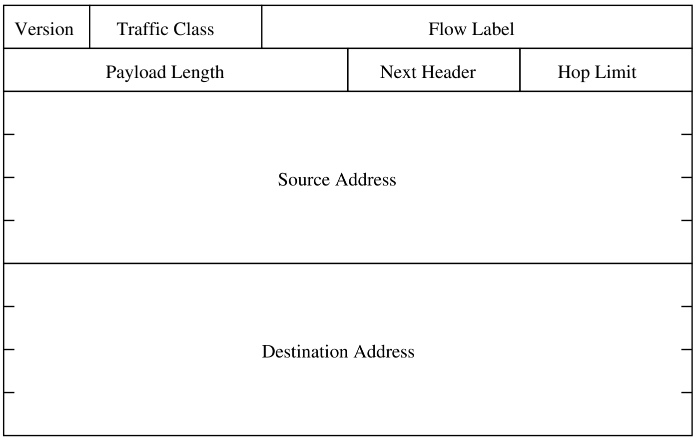
本節では，IPv4 ヘッダと IPv6 ヘッダ，ICMPv4[10] ヘッダと ICMPv6[11] ヘッダの具体的なヘッダ変換の詳細について述べる．

図 4.11 に IPv4 ヘッダを，図 4.12 に IPv6 ヘッダを示す．



<table>
<tr><td>Version</td><td>IHL</td><td colspan="2">Type of Service</td><td colspan="4">Total Length</td></tr>
<tr><td colspan="4">Identification</td><td>Flags</td><td colspan="3">Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td colspan="2">Protocol</td><td colspan="4">Header Checksum</td></tr>
<tr><td colspan="8">Source Address</td></tr>
<tr><td colspan="8">Destination Address</td></tr>
<tr><td colspan="6">Options</td><td colspan="2">Padding</td></tr>
</table>

図 4.11: IPv4 ヘッダ



<table>
<tr><td>Version</td><td colspan="2">Traffic Class</td><td colspan="5">Flow Label</td></tr>
<tr><td colspan="4">Payload Length</td><td colspan="2">Next Header</td><td colspan="2">Hop Limit</td></tr>
<tr><td colspan="8">Source Address</td></tr>
<tr><td colspan="8">Destination Address</td></tr>
</table>

図 4.12: IPv6 ヘッダ

第5.1節　IPv4ヘッダからIPv6ヘッダへの変換

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．

フラグメント禁止フラグが立っていて，IPv4パケットが分割されていない場合は次のように変換を行う．

- Version

  バージョン番号なので「6」に変換する．

- Traffic Class

  初期値の「0 (all zero bits)」を代入する．

- Flow Label

  IPv4では，存在しない概念なので「0 (all zero bits)」を代入する．

- Payload Length

  IPv4パケットの全パケット長から，IPv4ヘッダの「Internet Header Length」フィールドの値を引いた値を代入する．

- Next Header

  IPv4ヘッダの「Protocol」フィールドの値を代入する．もしも「Protocol」フィールドの値が「1 (ICMPv4)」ならば「58 (ICMPv6)」を代入する．

- Hop Limit

  IPv4ヘッダの「Time to Live」フィールドの値から1減らして代入する．

- Source Address

  第4.1節で述べたので省略する．

- Destination Address

  第4.1節で述べたので省略する．

もしも，フラグメントヘッダを加える必要がある場合も，次の例外を除いて，IPv4 ヘッダは上述のように IPv6 ヘッダに変換する．

- Payload Length

  IPv4 パケットの全パケット長から，IPv4 ヘッダの「Internet Header Length」フィールドの値を引き，フラグメントヘッダのための「8」を加えた値を代入する．

- Next Header

  フラグメントヘッダの「44」を代入する．

フラグメントヘッダのフィールドは次のようにする．

- Next Header

  IPv4 ヘッダの「Protocol」フィールドの値を代入する．もしも「Protocol」フィールドの値が「1 (ICMPv4)」ならば，「58 (ICMPv6)」を代入する．

- Reserved

  「0 (all zero bits)」を代入する．

- Fragment Offset

  IPv4 ヘッダの「Fragment Offset」フィールドの値を代入する．

- M flag

  IPv4 ヘッダの「モアフラグメントフラグ」の値を代入する．

- Identification

  下位 16 ビットは，IPv4 ヘッダの「Identification」フィールドの値を代入し，上位 16 ビットは，「0」を代入する．

第5.2節 IPv6ヘッダからIPv4ヘッダへの変換

IPv6フラグメントヘッダの例外を除いて，他のIPv6拡張ヘッダは無視する．IPv6拡張ヘッダそれぞれについて，IPv4ヘッダの「Total Length」フィールドの値の計算をする前に，これらのペイロード長は無視する必要がある．もしも，IPv6フラグメントヘッダがなければ，IPv4ヘッダへの変換は次のように行う．

- Version

  バージョン番号なので「4」に変換する．

- Internet Header Lenghth

  「5 (no IPv4 options)」を代入する．

- Type of Service

  「0 (all zero bits)」を代入する．

- Total Length

  IPv6ヘッダの「Payload Length」フィールドの値にIPv4ヘッダ長を加える．

- Identification

  「0 (all zero bits)」を代入する．

- Flags

  フラグメント禁止フラグが立っていれば「1 (true)」に，それ以外なら「0 (false)」を代入する．

- Fragment Offset

  「0 (all zero bits)」を代入する．

- Time To Live

  IPv6ヘッダの「Hop Limit」フィールドの値から1減らして代入する．

- Protocol

  IPv6ヘッダまたは最後の拡張ヘッダの「Next Header」フィールドの値を代入する．もしも「Protocol」フィールドの値が「58 (ICMPv6)」ならば，「1 (ICMPv4)」を代入する．

- Header Checksum

  IPv4ヘッダが完成したら，そこから計算し直す．

- Source Address

  第4.1節で述べたので省略する．

- Destination Address

  第4.1節で述べたので省略する．

もしも，IPv6パケットがフラグメントヘッダを含んでいても，次の例外を除いて，IPv6ヘッダは上述のようにIPv4ヘッダに変換する．

- Total Length

  IPv6ヘッダの「Payload Length」フィールドの値から，フラグメントヘッダのために8を引いて，IPv4ヘッダ長を加える．

- Identification

  フラグメントヘッダの「Identificaiton」フィールドの下位16ビットの値を代入する．

- Flags

  フラグメントヘッダの「モアフラグメントフラグ」の値を代入し，「フラグメント禁止フラグ」は「0 (false)」にする．

- Fragment Offset

  フラグメントヘッダの「Fragment Offset」の値を代入する．

第 5.3 節 ICMPv4 ヘッダから ICMPv6 ヘッダへの変換

- Echo (Type 0)

  Type を 128 にする．

- Echo Reply (Type 8)

  Type を 129 にする．

- Destination Unreachable (Type 3)

  次に示す Code 以外は，Type を 1 にする．

  - Code 0, 1, 6, 7, 8, 11, 12

    Code 0 (no route to destination) にする．

  - Code 2

    ICMPv6 Parameter Problem (Type 4, Code 1) に変換し，IPv6 ヘッダの「Next Header」フィールドを「6」にする．

  - Code 3

    Code 4 (port unreachable) にする．

  - Code 4

    ICMPv6 Packet Too Big message (Type 2, Code 0) に変換し，「MTU」フィールドの値を IPv4 ヘッダサイズと IPv6 ヘッダサイズの違いで修正する．

  - Code 5

    Code 2 (not a neighbor) にする．

  - Code 9, 10

    Code 1 (communication with destination adminisaratively prohibited) にする．

- Time Exceeded (Type 12)

Typeを4にし「Pointer」フィールドの値を表4.3にする．表にないICMPv4 Pointerは「-1」のICMPv6 Pointerに変換する．

表 4.3: Pointer フィールドの変換

ICMPv4 Pointer	ICMPv6 Pointer
0	0
2	4
8	7
9	6
12	8
16	24

第 5.4 節　ICMPv6 ヘッダから ICMPv4 ヘッダへの変換

- Echo (Type 128)

  Type を 0 にする．

- Echo (Type 129)

  Type を 8 にする．

- Destination Unreachable (Type 1)

  Type を 3 にし，Code フィールドを次のように変換する．

  - Code 0

    Code 1 (host unreachable) にする．

  - Code 1

    Code 10 (communication with destination host administratively prohibited) にする．

  - Code 2

    Code 5 (source route failed)にする．

  - Code 3

    Code 1 (host unreachable)にする．

  - Code 4

    Code 3 (port unreachable)にする．

- Packet Too Big (Type 2)

  ICMPv4 Destination Unreachable(Code 4)に変換し，「MTU」フィールドの値をIPv4ヘッダサイズとIPv6ヘッダサイズの違いで修正する．

- Time Exceeded (Type 3)

  Type 11にする．Codeフィールドは変化させない．

第 5 章.　HITS の実装

本研究では，HITS を導入することにより，移行時のコストの削減と円滑な移行が可能であることを実証するため，HITS の実装を行った．そこで，本章では，HITS の実装について述べる．まず，実装を行った環境について述べ，次に HITS の実装について述べる．

第 1 節.　動作環境

表 5.1 に示す環境において，実装を行った．

表 5.1: 動作環境

計算機	PC/AT 互換機，MMX Pentium 166Mhz，64MB Memory
OS (Operating System)	FreeBSD 2.2.7-Release
IPv6 スタック	KAME[1]

HITS の実装は，KAME に依存した実装となっている．

第 2 節.　Header Converter の実装

Header Converter の実装は，設計に基づいて行った．FreeBSD 2.2.7 上で IPv6 スタックの KAME を拡張して実現した．

拡張部分は表 5.2 に示す 3 つの部分で，それぞれの関係は図 5.1 の通りである．

表 5.2: Header Converter の拡張機能と関数名

機能	関数名
変換受け付けモジュール	hits_trans_accept()
変換モジュール	hits_trans()
アドレス要求・返答モジュール	hits_addr()

それぞれの部分について説明する．

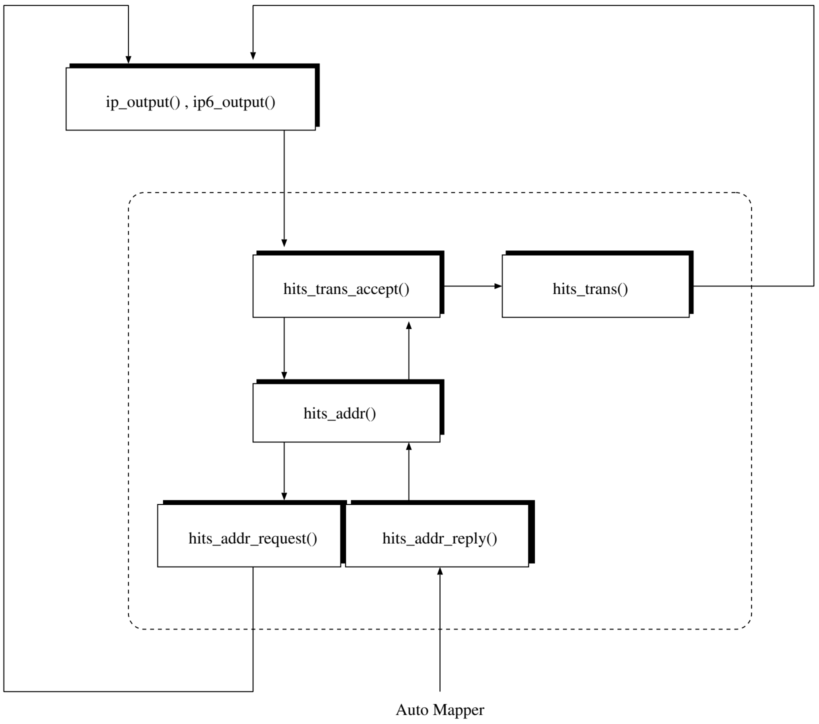


図 5.1: Header Converter

- 変換受け付けモジュール (hits_trans_accept())

  IPv4 パケット出力部 (ip_output())，IPv6 パケット出力部 (ip6_output()) から呼び出される．ヘッダ変換の際にアドレス要求・返答モジュールにそのアドレスを問い合わせ，その返答をうけて，変換モジュールを呼び出す．

- 変換モジュール (hits_trans())

  IPv4 ヘッダを IPv6 ヘッダに，IPv6 ヘッダを IPv4 ヘッダに変換する．ヘッダ変換の詳細については第 4.5 節で述べた．

- アドレス要求・返答モジュール (hits_addr())

  アドレス要求を Auto Mapper に送信し (hits_addr_request())，その返答を受けて (hits_addr_reply())，変換受け付けモジュールを呼び出す．

第3節. Auto Mapperの実装

Auto Mapperは，1つのプロセスとして実装した．Auto Mapperが管理しているIPv4アドレスとIPv6アドレスの対応付けのデータベースはメモリに格納されている．しかし，Auto Mapeerが停止した時のことを考慮し，ファイルにも書き込まれている．これにより，再起動を行った際にも対応づけのデータが保持されている．

データベースは図5.2に示す，構造体を用いて構成される．

```
struct automap_db {
        char    hostname[HOSTNAMELEN];
        char    ipv4addr[IPV4ADDRLEN];
        int     a;
        char    ipv6addr[IPV6ADDRLEN];
        int     aaaa;
        struct timeval map_time;
};
```

図 5.2: automap_db 構造体

automap_db 構造体の説明を表5.3にまとめる．

表 5.3: automap_db 構造体のメンバ

メンバ名	変数型	説明
hostname	char	割り当てたホストの名前が登録される
ipv4addr	char	割り当てたホストのIPv4アドレスが登録される．
a	int	IPv4アドレスが割り当てられた否かを登録する．
ipv6addr	char	割り当てたホストのIPv6アドレスが登録される．
aaaa	int	IPv6アドレスが割り当てられた否かを登録する．
map_time	struct timeval	ホストにアドレスを割り当てた時刻がtimeval構造体形式で登録される．

このデータベースに登録した場合の登録例を図5.3に示す．登録例では，AAAAレコードしか持たないelt.v6.sfc.wide.ad.jpというホスト名をもつホストに，Aレコードとして，203.178.140.44を割り当てたということを登録してある．

```
hostname = elt.v6.sfc.wide.ad.jp
ipv4addr = 203.178.140.44
a        = 1
ipv6addr = 3ffe:501:100c:e320:a00:20ff:fe74:bab8
aaaa     = 0
map_time = 登録時の時刻
```

図 5.3: automap_db 構造体への登録例

Auto Mapper は，さらに，要求に応じて，アドレスを割り当て，返答を行う．

IPv4 アドレスを割り当てる時のアルゴリズムは次の通りである．

1. IPv4 アドレスの要求があると，その IPv6 アドレスに対応する IPv4 アドレスを検索する．
2. 対応する IPv4 アドレスがあれば，それを返答する．
3. 対応する IPv4 アドレスがなければ，未割当の IPv4 アドレスがあれば，それを割り当てて返答する．
4. 未割当の IPv4 アドレスがなければ，アドレスの対応付けを削除し，再度割り当てを行う．

IPv6 アドレスを割り当てる時のアルゴリズムは次の通りである．

1. IPv6 アドレスの要求があると，その IPv4 アドレスに対応する IPv6 アドレスを検索する．
2. 対応する IPv6 アドレスがあれば，それを返答する．
3. 対応する IPv6 アドレスがなければ，未割当の IPv6 アドレスを割り当てて返答する．

割り当て可能なアドレスがなくなると，アドレスの使用期限が来たことになり，そのアドレスの対応付けを削除する．

Auto Mapper が割り当てるアドレスは，図 5.4 のような設定ファイルにより読み込まれ，割り当てられる．

第 4 節.　Fake DNS の実装

Fake DNS に対して，A レコードを要求した場合，

1. キャッシュにエントリが存在するか検索する．

```
ipv4 spool address
203.178.140.43
203.178.140.44

ipv6 spool address
3ffe:501:100c:e321::1
3ffe:501:100c:e321::2
3ffe:501:100c:e321::3
```

図 5.4: Auto Mapper 設定ファイル

  (a) 存在すれば，その A レコードを返答する．
  (b) 存在しなければ，外部のネームサーバに対して，その A レコードの問い合わせを行う．
2. 外部ネームサーバに対して，問い合わせを行う．
  (a) A レコードの返答があれば，それを返答する．
  (b) A レコードの返答がなければ，Auto Mapper に A レコードを要求し，それを返答する．

逆に，Fake DNS に対して，AAAA レコードを要求した場合，

1. キャッシュにエントリが存在するか検索する．
  (a) 存在すれば，その AAAA レコードを返答する．
  (b) 存在しなければ，外部のネームサーバに対して，その AAAA レコードの問い合わせを行う．
2. 外部ネームサーバに対して，問い合わせを行う．
  (a) AAAA レコードの返答があれば，それを返答する．
  (b) AAAA レコードの返答がなければ，Auto Mapper に AAAA レコードを要求し，それを返答する．

第6章.　評価

本章では，HITSを評価するための，実験環境について述べる．次に実際にHITSがトランスレータとして，適切に動作しているかを性能から評価する．さらに，他のトランスレータと比較した管理コストに関する評価を行い，最後に機能についての評価をする．

第1節.　実験環境

HITSを評価するにあたって，本システムを利用した実験環境を構築した．

実験環境のトポロジを図6.1に示す．

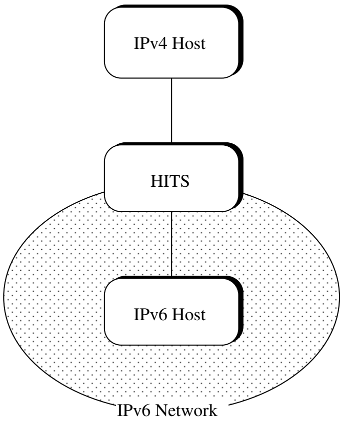


図 6.1: 実験環境

HITSはhits.v6.sfc.wide.ad.jpゾーンのプライマリDNSとなっている．

- HITS (ayu.hits.v6.sfc.wide.ad.jp)

計算機	PC/AT互換機，MMX Pentium 166Mhz，64MB Memory
OS (Operating System)	FreeBSD 2.2.7-Release
IPv6スタック	KAME
IPv4アドレス	203.178.140.43
IPv6アドレス	3ffe:501:100c:e320:260:97ff:fe8b:f122
スプールIPv4アドレス	203.178.140.45,203.178.140.46,203.178.140.47
スプールIPv6アドレス	3ffe:501:100c:e321::1,3ffe:501:100c:e321::2,3ffe:501:100c:e321::3

- IPv4 ホスト (ami.hits.v6.sfc.wide.ad.jp)

計算機	PC/AT 互換機，Pentium 150Mhz，128MB Memory
OS (Operating System)	Microsoft Windows98
IPv4 アドレス	203.178.140.44

- IPv6 ホスト (elt.hits.v6.sfc.wide.ad.jp)

計算機	PC/AT 互換機，MMX Pentium 166Mhz，64MB Memory
OS (Operating System)	FreeBSD 2.2.7-Release
IPv6 スタック	KAME
IPv6 アドレス	3ffe:501:100c:e320:260:67ff:fe2e:2289

第 2 節.　性能に関する評価

HITS の動作確認と動的アドレス割り当て機能に関する評価を行った．

第 2.1 節　動的アドレス割り当て機能に関する評価

Auto Mapper による動的アドレス割り当て機能を評価した．HITS の Auto Mapper を利用しない場合は，図 6.2，図 6.3 に示すように，AAAA レコードのみを持つホストの名前問い合わせはできない．

```
% nslookup -type=a elt.v6.sfc.wide.ad.jp.
Server:  shonan.sfc.wide.ad.jp
Address:  0.0.0.0

Name:    elt.v6.sfc.wide.ad.jp
```

図 6.2: 通常の名前問い合わせその 1

これに対して，HITS の Auto Mapper を利用した場合，図 6.4 のように名前問い合わせに成功する．

```
% nslookup -type=aaaa elt.v6.sfc.wide.ad.jp.
Server:  shonan.sfc.wide.ad.jp
Address:  0.0.0.0

elt.v6.sfc.wide.ad.jp   IPv6 address = 3ffe:501:100c:e320:260:67ff:fe2e:
2289
v6.sfc.wide.ad.jp       nameserver = ferrari.v6.sfc.wide.ad.jp
ferrari.v6.sfc.wide.ad.jp       IPv6 address = 3ffe:501:100c:e320:a00:20
ff:fe74:bab8
ferrari.v6.sfc.wide.ad.jp       internet address = 203.178.140.56
```

図 6.3: 通常の名前問い合わせその 2

```
% nslookup -type=a elt.hits.v6.sfc.wide.ad.jp.
Server:  ayu-hits.v6.sfc.wide.ad.jp
Address:  203.178.140.43

Name:    elt.hits.v6.sfc.wide.ad.jp
Address:  203.178.140.45
```

図 6.4: HITS を利用した名前問い合わせ

第 2.2 節　動作確認

動作確認は，ping を用いて行った．

IPv4 ホストから IPv6 ホストへの ping は，図 6.5 に示すように，HITS を用いることで，IPv4 ホストと IPv6 ホストの通信性が確立された．

逆に IPv6 ホストから IPv4 ホストへの ping は，図 6.6 に示すように，HITS を用いることで，IPv6 ホストと IPv4 ホストの通信性が確立された．

これにより，HITS は双方向通信を実現できたといえる．

```
C:\> ping elt.v6.sfc.wide.ad.jp.
Unknown host elt.v6.sfc.wide.ad.jp.

C:\> ping elt.hits.v6.sfc.wide.ad.jp.

Pinging elt.hits.v6.sfc.wide.ad.jp. [203.178.140.43] with 32 bytes of data:

Reply from 203.178.140.43: bytes=32 time=10ms TTL=254
Reply from 203.178.140.43: bytes=32 time=3ms TTL=254
Reply from 203.178.140.43: bytes=32 time=3ms TTL=254
Reply from 203.178.140.43: bytes=32 time=3ms TTL=254
```

図 6.5: HITS の動作確認その 1

第 2.3 節　HITS のオーバーヘッドに関する評価

IPv4 ホストと IPv6 ホストの透過的な通信を評価するために，HITS を用いた場合と，用いない場合の IPv4 ホスト同士，IPv6 ホスト同士の通信速度を ping によるパケット送出 100 回の平均値を計測した．図 6.1，図 6.2 に計測結果を示した．

表 6.1: HITS を用いた場合の通信速度

IPv4 ホスト→ IPv6 ホスト	2.08ms
IPv6 ホスト→ IPv4 ホスト	2.53ms

表 6.2: 通常の場合の通信速度

IPv4 ホスト→ IPv4 ホスト	1.21ms
IPv6 ホスト→ IPv6 ホスト	1.51ms

HITS を用いたことによる，オーバーヘッドの大きさは，約 1ms ほどである．これは，Header Converter と Auto Mapper によるものであり，比較的 IPv4 ホストと IPv6 ホスト同士の通信は透過的なものとなっている．しかし，改善を加えることで，より透過的な通信を実現できる．

```
% ping6 ayu.hits.v6.sfc.wide.ad.jp.
PING6(56=40+8+8 bytes) 3ffe:501:100c:e320:260:67ff:fe2e:2289 --> 3ffe:501:
100c:e321::1
16 bytes from ayu.hits.v6.sfc.wide.ad.jp, icmp_seq=0 hlim=64 time=51.125 ms
16 bytes from ayu.hits.v6.sfc.wide.ad.jp, icmp_seq=1 hlim=64 time=2.634 ms
16 bytes from ayu.hits.v6.sfc.wide.ad.jp, icmp_seq=2 hlim=64 time=2.659 ms
^C
--- ayu.hits.v6.sfc.wide.ad.jp ping6 statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 2.634/18.806/51.125 ms
```

図 6.6: HITS の動作確認その 2

第 3 節.　管理コストに関する評価

既存のトランスレータと HITS において，トランスレータ管理に要するコストを比較した．トランスレータ管理とは，次の 3 つの作業であると定義する．

- 設定

  新たにネットワークにトランスレータを導入する．

- 変更

  移行の段階によって変換の方向を変更する．

- クライアントの設定

  トランスレータを利用するためにクライアントを設定する．

第 3.1 節　各方式におけるトランスレータ管理コスト

既存のトランスレータと HITS における管理コストについて説明する．

- HITS

  HITS を設定するには，次の手順をふむ必要がある．

  1. DNS の設定

     設置されているネットワークに存在するホスト名とアドレスを登録する．

  2. スプールされるアドレスの設定

     割り当て可能なアドレスを指定する．

  HITS は変更する必要がないので，とくに手順をふむ必要はない．つまり，一度設置すれば，その後の管理コストを必要としない．

  クライアントの設定は必要としない．

- NR60

  NR60 を設定するには，次の手順をふむ必要がある．

  1. DNS の設定

     接続されているネットワークに存在するホスト名とアドレスを登録し，広告する．

  2. スプールされるアドレスの設定

     割り当て可能なアドレスを指定する．

  NR60 を変更する場合，次の手順をふむ必要がある．

  1. DNS の設定

     接続されているネットワークに存在するホスト名とアドレスを登録し，広告する．

  2. スプールされるアドレスの設定

     割り当て可能なアドレスを指定する．

  クライアントの設定は必要としない．

- FAITH

  FAITH を設定するには，次の手順をふむ必要がある．

  1. DNS の設定

     DNS を FAITH モードにする．

  2. FAITH 用 prefix の設定

     変換すべき prefix を指定する．

  3. アプリケーションの設定

     変換すべきアプリケーションを指定する．

  FAITH は IPv6 から IPv4 への一方向のみしか変換できないので，変更する必要がない．

  クライアントの設定は必要としない．

- SOCKS64

  SOCKS64 を設定するには，次の手順をふむ必要がある．

  1. SOCKS64 サーバの設定

     中継サーバの設定をする．

  2. クライアントの設定

     クライアントを SOCKS 対応にする．

  SOCKS64 は特に設定を変更する必要はない．

  SOCKS クライアントを設定する必要がある．

第 3.2 節　管理コストの比較と評価

前節で述べた 4 方式について，トランスレータ設定・変更コスト比較を行った結果を，表 6.3 にまとめる．

NR60 を用いた場合は，設定に 2 手順，変更に 2 手順を踏む必要がある．さらに，移行の段階によって設定を変化させなければならない．

FAITH を用いた場合は，設定に 3 手順かかってしまう．また，IPv6 から IPv4 への変換のみに対応するものなので，IPv6 への移行時に常に用いることができない．

表 6.3: トランスレータの管理コスト比較

方式	設定コスト	変更コスト	クライアントの設定コスト
HITS	2手順	なし	なし
NR60	2手順	2手順	なし
FAITH	3手順	×	なし
SOCKS64	2手順	1手順	あり

SOCKS64は，クライアントに変更を加えなければならないため，1手順×ホスト数の管理コストがかかってしまう．

HITSでは，設定コストは2手順かかるものの，移行段階に応じた設定の変更は必要なく，また，クライアントの設定も必要としない．つまり，一度設置すれば，その後の管理コストを必要としない．

よって，既存のトランスレータとHITSを比較した結果，トランスレータの設定に際して，設定工数からみて，HITSが最も設定コストを削減できる方式であることがわかる．

第4節. 機能に関する評価

既存のトランスレータと機能面で比較すると，HITSには次に述べる利点がある．

- 双方向通信機能

  HITSの設置してあるIPv6ネットワークのIPv6ホストから外部のIPv4ホストへの通信と外部のIPv4ホストからIPv6ネットワークのIPv6ホストへの通信，HITSの設置してあるIPv4ネットワークのIPv4ホストから外部のIPv6ホストへの通信と外部のIPv6ホストからIPv4ネットワークのIPv4ホストへの通信を可能にしている．

- 枯渇しているIPv4アドレスへの対応

  トランスレータはIPv4アドレスとIPv6アドレスをどう対応付けるかが問題だが，枯渇しているIPv4アドレスを有効に利用するため，HITSでは，動

的にIPv4アドレスを割り当てることを可能にしているため，少ないアドレスで，より多くのホスト間の通信を可能にする．

- 両方向の変換に対応

  HITSはデュアル・スタックホストであり，また，移行の段階によって，変換の方向を設定し直す必要がないため，移行初期から移行後期までのすべての移行段階に対応可能である．

- ホスト単位の変更が不必要

  サイト内にある各ホストに変更を加えることなく，トランスレータを利用できる．

以上の機能に関する比較を表6.4にまとめる．

表6.4: トランスレータの機能に関する比較

機能	HITS	NR60	FAITH	SOCKS64
双方向通信	○	×	×	×
アドレスの動的割り当て	○	×	×	×
全移行段階への対応	○	△	×	×
導入の容易さ	○	△	△	×

この結果から，HITSは既存のトランスレータと比べて，円滑な移行を行うために必要な機能を持つことが実証された．

第 7 章.　結論

本章では，本論文のまとめと今後の課題について述べる．

第 1 節.　本研究のまとめ

本研究では，IPv6 への円滑な移行を実現するために，IPv6/v4 トランスレータ HITS の設計，実装，評価を行なった．

IPv6 インターネットへの移行が進行すると，IPv4 インターネットと IPv6 インターネットの混在環境となる．その中で，重要になるのが，IPv4 ホストと IPv6 ホストの相互通信を可能にすることである．しかし，デュアル・スタック戦略による相互接続では，枯渇しつつある IPv4 アドレスを全てのデュアル・スタックホストに割り当てることができず，相互通信ができない可能性がある．

そこで，本研究では，IPv6 への円滑な移行を実現するために，現状のデュアル・スタック戦略で対応できない移行期全ての状況に対応する，IPv6/v4 トランスレータ HITS を提案した．HITS は，動的にアドレスを割り当てる機能をもち，少ない IPv4 アドレスを切替えて利用することで，より多くのホスト間での通信を可能にする．そして，HITS を実装し，他のトランスレータと比較して，移行の際に生じる管理コストが軽減できるかを評価した．

その結果，移行の時期に応じた設定の変更の必要のない HITS は，トランスレータ管理コストを軽減させることができ，IPv6 への移行をより円滑に実現することに寄与できる．

第 2 節.　本研究の成果

本研究の成果として，次のものがあげられる．

- IPv4 ホストと IPv6 ホストの相互通信

  通信互換性のない両者の双方向通信を可能とした．

- 管理コストの軽減

  移行の時期に応じた設定の変更の必要のない HITS は，トランスレータ管理コストを軽減させせることができる．

- IPv6 への移行支援

  HITS を既存のネットワークに導入することにより，または，新規のネットワークに導入することにより，容易に IPv6 インターネットと IPv4 インターネットの相互通信できるため，IPv6 への移行を支援することができる．

第 3 節.　今後の課題

本研究における今後の課題を述べる．

HITS の実装はまだ完全ではなく，今後の課題として，次の 3 つについて検討する必要がある．

- 透過通信性の向上

  Header Converter と Auto Mapper によるトランスレータ機能のオーバヘッドを軽減し，変換をより高速にすることで，IPv4 ホストと IPv6 ホストの透過通信性を向上させ，トランスレータ導入によるコストを軽減させせることができる．

- インターネットでの運用

  HITS は，実験ネットワーク上では，管理コストを軽減できた．今後，インターネットでの運用を行うために規模性の検証を行うとともにさまざまなトラブルに対処できるようにする．また，その運用によって浮上した問題点を改善する．

- プロトコル変換の正当性

  大規模運用を行うために，複数のアプリケーションによる HITS のプロトコル変換の正当性を検証し，実用性を向上させる．

謝辞

本研究を進めるにあたり，暖様々なアドバイスをして下さった，慶應義塾大学環境情報学部の村井 純博士，徳田 英幸博士，楠本 博之博士，中村 修博士に感謝します．

また，本研究に対して，細部にわたる指導と助言を頂いた慶應義塾大学政策・メディア研究科の関谷勇司学士，土井裕介学士にお礼申し上げます．

並びに，さまざまな助言を頂いたWIDEプロジェクトIPv6 WGの皆様，常日頃から多大なる協力と心遣いを頂いた徳田・村井・楠本・中村研究室IPng-WGの皆様，本論文を執筆するにあたって論文の内容や実装について様々な意見を提供して下さった慶應義塾大学村井研究室の諸氏に感謝します．

また，論文執筆に集中できるようにと様々な面で気をつかってくれた，慶應義塾大学環境情報学部の川喜田佑介君，佐藤雅明君，論文執筆のBGMとなったTMN，globe，浜崎あゆみに心から感謝します．

最後に，論文執筆を支援してくれたすべての人に深い敬意と感謝の意を表します．

慶應義塾大学 環境情報学部
廣瀬 謙治
kenji@sfc.wide.ad.jp

参考文献

[1] J. Postel. Internet Protocol. RFC 791, September 1981.

[2] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[3] K. Yamamoto and M. Sumikawa. Categorizing Translators between IPv4 and IPv6. Internet-Drafts, works in progress.

[4] A. Jinzaki and S. Kobayashi. SOCKS64: An IPv4-IPv6 intercommunication gateway using SOCKS5 protocol. Internet-Drafts, works in progress.

[5] H. Kitamura. A SOCKS-based IPv6/IPv4 Translator Architecture. Internet-Drafts, works in progress.

[6] Kazuhiko YAMAMOTO, Akira KATO, Munechika SUMIKAWA, and Jun MURAI. Deployment and Experiences of WIDE 6bone. September 1998.

[7] K. Tsuchiya, M. Sumikawa, K. Watanabe, Y. Atarashi, T. Miyamoto, K. Yamamoto, and J. Murai. A Communication Mechanism between IPv4 and IPv6. Internet-Drafts, works in progress.

[8] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS Protocol Version 5. RFC 1928, April 1996.

[9] 角川宗近. 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. Feburary 1997.

[10] J. Postel. Internet Control Message Protocol. RFC 777, April 1981.

[11] A. Conta and S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 2463, December 1998.