# IPv4-to-v6 トランスレータの実現とネットワークエミュレータ上での評価

2006MI108 森 千恵　　2006MI204 畑佐 宏輝
指導教員　後藤 邦夫

## 1 はじめに

アジア諸国を中心にインターネット利用機器が急速に増加し, そのため IPv4 アドレスの不足が深刻となった. IPv4 アドレスの在庫が近い将来にも枯渇するといわれており, その恒久的な対策として IPv6 の導入があげられる. しかし IPv4 と IPv6 には互換性がないため, IPv4 のみのユーザが IPv6 の移行が完了したネットワークから取り残されてしまうことが予想される [1].

本研究ではそのような状況のために IPv4 のみのネットワークから IPv6 のみのネットワークへ通信することのできるよう IPv4-to-v6 トランスレータ (以下トランスレータ) を作成をし, そのトランスレータの評価をする. IPv4 と IPv6 の互換性のない原因はヘッダの構造が違うことにあるため, Ethernet ヘッダ, IP ヘッダ, TCP や UDP などの上位層ヘッダの書き換えをし, また IPv4host と IPv6host の IP アドレスをそれぞれのネットワークで理解できるように, 変換表と改造した DNS を用いる. より実ネットワークに近い環境を構築するためにエミュレータを用い, 実験を試みた.

なお, 実験の環境構成とヘッダ変換プログラムを森が担当し, DNS と変換表を畑佐が担当した.

## 2 システム概要

本節ではトランスレータとネットワークエミュレータ GINE の概要を紹介する.

本研究の IPv4 ネットワークは, 割り当てを受けた IPv4 アドレスでインターネットを利用してきた組織 or ISP を想定する. 本研究では IPv4 アドレスには IPv4 プールと重複しないプライベートアドレスを用いた. プールの詳細は 3.3 節で説明する. IPv6 ネットワークは, IPv6 グローバルアドレスを受けて接続するが, IPv4 しか使えない LAN または顧客ネットワークがあると想定する. 本研究の IPv6 アドレスには documentation 用の IPv6 アドレスを用いた.

### 2.1 トランスレータ

本研究でのトランスレータとは, IPv4 ネットワークと IPv6 ネットワークの間で TranslatorGateway(以下 TG), NameServer, アドレス変換表 (以下変換表) を利用し, プロトコルやパケットをそれぞれのネットワークで理解できる形に変換しながら, 双方の通信をする技術のことを意味する. パケットをそれぞれのプロトコルのフォーマットに合わせる際, ヘッダなどを再構成するので, もとのパケットをペイロードとして新しいヘッダを追加するトンネリング技術とは区別される.

トランスレータの既存技術には TOWNS[5], FAITH, SOCKS64 などがある. TOWNS は translation 方式をベースにしており, その他は application-level gateway 方式を採用している. 本研究では既存技術として存在する TOWNS を参考にトランスレータを作成する.

なお, 本研究で作成するトランスレータと TOWNS との違いとして, TOWNS では TG, MapServer(以下 MS), NameServer がそれぞれ独立しており, それぞれの間を ICMP を改造したパケットを用いてやりとりをしているが, 本研究では MS をオブジェクトとして TG と NameServer が管理する. つまり本研究ではトランスレータの一連の動作が 1 つのプログラムにまとめられている. よって, TOWNS より通信の手間が省けより通信の高速化が実現できると予想される. また本研究では NameServer を改造 DNS, MS を変換表と表した.

### 2.2 GINE(GOTO's IP Network Emulator) の利用

GINE[4] は本研究室で開発中のネットワークエミュレータである. 本研究ではエミュレートをするシステムとして GINE クラスである PFPacket を使用する. PFPacket とはデータリンク層でのフレームを横取りする GINE の機能である. PFPacket は, 特殊な名前空間である Network NameSpace(以下 NS) 上での使用が容易であることから本研究では PFPacket を利用することにした. NS とは, ネットワークスタックを仮想化するものである.

PFPacketIn では仮想ネットワークデバイスを監視し, RAW ソケットを生成, bind する. そして元のインタフェースから PFPacketIn でコピーしたインタフェースへ enqueue する. PFPacketOut では指定したソケットに繋いで dequeue する. つまり, PFPacketIn でフレームを横取りし, PFPacketOut でデータリンク層で横取りしたフレームを書き戻すという操作を行っている. これによりフレームの横取りの一連の操作を成り立たせることができる.

本研究では, GINE を TG での GINE 内ライブラリクラスの使用や, 実験での仮想ホストの生成に用いる.

## 3 トランスレータの構成と通信手順

本節では, TOWNS を参考に作成したプログラムの構成について説明する.

トランスレータは以下の要素で構成される.

- 変換ゲートウェイ (TG)
- 改造 DNS
- 変換表

### 3.1 変換ゲートウェイ (TG)

TG は IPv4 と IPv6 のパケットを相互に変換する機能である．IPv4 と IPv6 は互換性のないプロトコルであるため, IPv4, IPv6 間で通信をするためにはそれぞれのプロトコルやパケットをそれぞれのネットワークで理解できる形に変換する必要がある．

TG では以下の処理をする．

- IP ヘッダ変換
- Mac アドレス情報取得および Ethernet ヘッダ変換
- チェックサム計算
- 上位層ヘッダ変換

なお, 本研究ではヘッダの変換を用いた相互通信を主としており, FTP 通信などに必要となるペイロードの変換は省略する．また, IPsec に関しても同様に今回のヘッダ変換とは別の処理をすることとなるため, 本研究からは省略した．拡張ヘッダも TCP や UDP などのプロトコルと別の処理を必要とするため, 本研究では未実装である．

IP ヘッダ変換では, それぞれのヘッダ構造に対応した IP ヘッダの書き換えをする．

IPv4 から IPv6 へのヘッダ変換を以下に示す．

- version → ’4’ から’6’ にする．
- Traffic Class → デフォルト値の’0’ にする．
- Flow Label → IPv4 では存在しない概念なので’0’ を代入．
- Payload Length → IPv4 のパケット長から, 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．本研究では割愛するが, 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．
- Next Header → IPv4 のプロトコルを代入．本研究では割愛するが, 変換後に拡張ヘッダが現れるのならばその番号を代入．
- Hop Limit → IPv4 の TTL を代入．
- Source/Destination address → 変換表に問い合わせる．

一般的に通信は双方向であるので, IPv6 ヘッダを IPv4 ヘッダに変換できる必要がある．IPv6 ヘッダから IPv4 ヘッダへの変換は以下のようになる．

- version → ’6’ から’4’ にする．
- IHL → (IPv6 にはない概念なので，再計算) 変換後のオプションを含めたヘッダの長さを 2bit 右にシフト．
- Type of Service → IPv4 にはほとんど使われていないフィールドなので, ’0’ を代入．
- Total Length → IPv6 のペイロード長に IPv4 のヘッダ長 (シフト前) を加えたものを代入．
- ID，Flag，Fragment Offset → IPv6 の拡張ヘッダに断片ヘッダが存在すれば変換する．なければ’0’ を代入 (本研究では’0’ を代入)．
- Time to Live → Hop Limit を代入．
- Protocol → 次ヘッダを代入．
- Header Cheaksum → IPv6 には存在しないので再計算して代入．
- Source/Destination address → 変換表に問い合わせる．
- option → 本研究では割愛

本研究では拡張ヘッダに関する処理は未実装であるため, 実験ではフラグメントヘッダが生成されないようペイロード長を調節した．

送信先 Mac アドレスは OS から ARP, Neighbor discovery 情報を取得することで知ることが出来る．ping または ping6 コマンドでその IP アドレス宛に通信を確認したあと, コマンド ip neigh show の出力をプログラム上で読み込み, その情報を宛先 Mac アドレスに書き換える．また, 上位層のプロトコルを識別する Ethernet ヘッダのタイプ番号も変換の必要がある．

IP ヘッダを書き換えるため IPv6 から IPv4 へ書き換えた場合, チェックサムを計算し直す必要がある．上位層のチェックサム計算に用いる擬似ヘッダに, IP ヘッダの宛先 IP アドレスと送信元 IP アドレスの情報が含まれているため, IP ヘッダを書き換えたあと上位層のチェックサム再計算の必要がある．そのため, プログラム上で一度擬似ヘッダを生成し直し, チェックサムの計算をし, その値を上位層のチェックサムに書き換えなければならない．なお, 本研究では上位層として TCP, UDP, ICMP, ICMPv6 を扱っている．

上位層ヘッダでは, UDP や TCP は IPv4 と IPv6 でその違いはない．しかし ICMP と ICMPv6 は IPv4 と IPv6 で構造は似ているが, 割り当てられたタイプとコードがかなり異なる．そのため, ICMP と ICMPv6 のヘッダ変換をする必要がある．

### 3.2 改造 DNS

DNS は名前解決のサービスのことであるが, IPv4 アドレスは A レコード, IPv6 アドレスは AAAA レコードと，レコードが異なる．更に IPv4 から IPv6 の通信で, IPv4host は IPv6 アドレスが理解できず, IPv6host は IPv4 アドレスが理解できない．そこで本研究の改造 DNS は, IPv4host から聞かれる FQDN(Fully Qualified Domain Name) に対し, キャッシング DNS サーバである unbound-1.3.4[2] のライブラリ関数を利用して，A レコードを AAAA レコードに変換し，外部 DNS に FQDN の IP アドレスを聞きにいく．そして, 返ってきた返答を A レコードに変換し, IPv6 アドレスに対する IPv4 アドレスを一時的に割り当て，IPv4host に返答する働きを持つ．

### 3.3 変換表

IPv4 アドレス空間よりも IPv6 アドレス空間の方が大きいため, IPv4 から IPv6 へは静的なアドレス変換が可能であるが IPv6 から IPv4 への静的なアドレス変換

は難しい．そのため IPv6 アドレスと IPv4 アドレスの相互変換をするさいは変換表を用いて変換する方法を採用する．

変換表は仮の IPv4 アドレスと IPv6 アドレスの対応をデータとして管理し，アドレス変換のさいに利用する．IPv6 アドレスから仮 IPv4 アドレス関連付けについて，仮 IPv4 アドレスの割り当てのない IPv6 アドレスが引数として与えられた場合，仮 IPv4 アドレスをアドレスプールとして設定した 10.0.0.0/8 のプライベートアドレスを利用し，ホスト部を IPv6 アドレスの MD5 digest ハッシュ値末尾 32bit としたオブジェクトを生成する．また，IPv6 アドレスプールを 2001:0:ffff::/48 とする．IPv4 アドレスを仮 IPv6 に変換するさい，このアドレスプールを利用して末尾に IPv4 アドレスを付与し，仮 IPv6 アドレスを生成する．例えば，IPv4 アドレスが 172.16.0.2 の場合，仮 IPv6 アドレスは 2001:0:ffff::172.16.0.2 となる．

### 3.4 通信手順

トランスレータを用いた IPv4host から IPv6host への通信手順について説明する．

1 対 1 の host の通信手順を図 1 のようなネットワーク例を用いて説明する．この通信のシーケンス図を図 2 で表す．

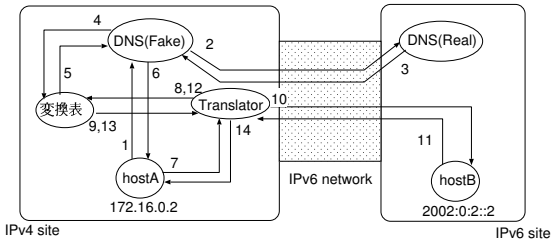


図 1　IPv4 から IPv6 への通信手順

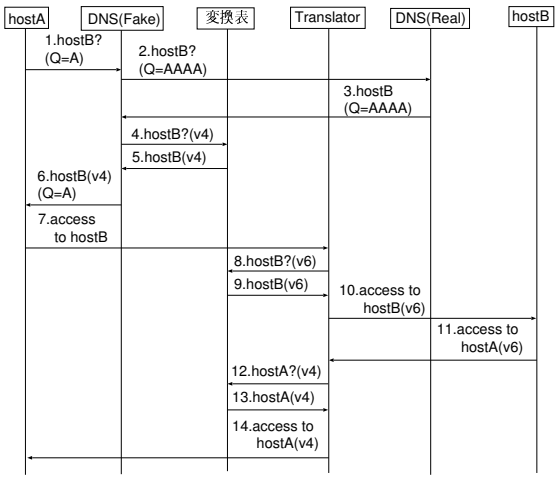


図 2　IPv4-IPv6 通信シーケンス図

送信元の hostA は IPv4 のみが利用できる host で，宛先の hostB は IPv6 のみが利用できる host であるとする．hostA 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るとする (仮 IPv6 アドレス)．仮 IPv6 アドレスの生成方法は 3.3 節で述べたとおりである．各 host の host 名と IP アドレスは表 1 のように設定する．

表 1　IPv4 アドレスと IPv6 アドレスの関係

| | host 名 | IPv4 address | IPv6 address |
|---|---|---|---|
| host A | hostA.<br>myroot | 172.16.0.2 | 2001:0:ffff<br>::172.16.0.2<br>(仮 IPv6<br>アドレス) |
| host B | hostB.<br>myroot | 10.x.y.z<br>(仮 IPv4<br>アドレス) | 2002:0:2::2 |

図 1 の TG，変換表，改造 DNS がトランスレータの部分となる．

TG は IPv4 と IPv6 のヘッダ変換を行い，改造 DNS は宛先ホストの IP アドレスの問い合わせをし，そして変換表は IPv4 アドレスと IPv6 アドレスの対応づける．

通信手順を説明する．

1. hostA が DNS(Fake) に hostB.myroot の IP アドレスを問い合わせる (A レコード)．
2. DNS(Fake) がレコードを AAAA レコードに書き換えて，外部ネームサーバ DNS(Real) に hostB.myroot の IP アドレスを問い合わせる．
3. hostB.myroot の IP アドレス，2002:0:2::2 が AAAA レコードで DNS(Fake) に返ってくる．
4. DNS(Fake) が変換表に 2002:0:2::2 に対応する IPv4 アドレスを問い合わせる．
5. 変換表には 2002:0:2::2 に対応する仮 IPv4 アドレスがまだ生成されていないので，10.x.y.z(仮 IPv4 アドレス) を生成し，これを返す．仮 IPv4 アドレスの生成については 3.3 節で述べたとおりである．
6. DNS(Fake) がレコードを A レコードに書き換えて，hostA に仮 IPv4 アドレス 10.x.y.z を返す．
7. hostA が 10.x.y.z に対してパケットを送信する．
8. Translator が変換表に仮 IPv4 アドレス 10.x.y.z に対応する IPv6 アドレスを問い合わせる．
9. 変換表は 10.x.y.z に対応する hostB の IP アドレス 2002:0:2::2 を返す．このとき hostA の IPv4 アドレスを仮 IPv6 アドレス 2001:0:ffff::172.16.0.2 とする．仮 IPv6 アドレスの生成については 3.3 節で述べたとおりである．
10. Translator がヘッダ変換したパケットを hostB へ送る．
11. hostB が 2001:0:ffff::172.16.0.2 に対してパケットを送信する．
12. Translator が変換表に仮 IPv6 アドレス 2001:0:ffff::172.16.0.2 に対応する IPv4 アドレス

を問い合わせる.

13. 変換表は 2001:0:ffff::172.16.0.2 の末尾 32bit を抜き取り, それを IPv4 アドレスとした 172.16.0.2 を返す. hostB の IP アドレスを仮 IPv4 アドレスである 10.x.y.z とする.
14. Translator がヘッダ変換したパケットを hostA へ送る.

## 4 実験の構成

NS を 2 つ作成し, 実機を 2 つ表現する. ひとつは IPv4 のみの host(172.16.0.2), もう一つは IPv6 の host(2002:0:2::2) である. この 2 つある NS の間に作成したトランスレータを置く. この通信構成イメージを図 3 で表す.

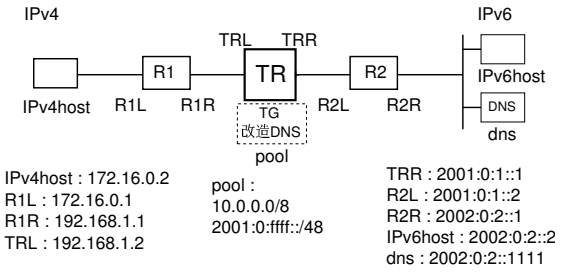


図 3 IPv4-IPv6 通信

2.2 節で説明した PFPacket でフレームを横取りし, トランスレータで書き換えをしてから宛先ホストにパケットを送る.

本研究で実験をする際トランスレータから送信元と宛先の MAC アドレスがフレームヘッダを変えずに送信しようとしても送信することが出来ないため, トランスレータを外部ルータ 2 つで挟み, MAC アドレス処理はルータの ARP や Neighbor discovery に任せて通信する.

## 5 実験、評価

本節では実現したトランスレータの動作確認と性能評価実験について述べる.

### 5.1 実験環境

本研究の実験で使用した PC のスペックを表 2 に示す.

表 2 使用した PC のスペック

| | |
|---|---|
| PC | Power Edge 840 |
| OS | Ubuntu 9.04 Jaunty Jackalop 64 bit |
| CPU | Intel(R) Xeon(R) CPU X3220 @ 2.40GHz (Quad Core) |
| メモリ | 2.0GB |

### 5.2 性能評価

本研究で実現したトランスレータのスループットを iperf[3] を用いて測定した. スループットを 10 回測定し, その平均値を算出した.

UDP の測定においては, オプションで送信する UDP の帯域を指定している. iperf では UDP を指定すると、クライアントから送信するトラフィックがデフォルトで 1Mbps となるためである.

4 章のように 1 ホスト対 1 ホストの通信実験ができるネットワークを構築した. 1 ホスト対 1 ホストのネットワークにおける実験結果を表 3 に示す.

表 3 1 ホスト対 1 ホストのスループット測定結果

| | 最小 (Mbps) | 最大 (Mbps) | 平均 (Mbps) |
|---|---|---|---|
| UDP | 231 | 232 | 231.8 |
| TCP | 291 | 368 | 328.1 |

このことから, 本研究で作成したトランスレータは高いスループットを実現するしていることが分かる. TCP の方がスループットが高い理由は, TCP には輻輳制御があるためであると考えられる.

## 6 おわりに

本研究では, IPv4 から DNS に聞かれる FQDN に対してレコードの変換をして返答を返せること, またヘッダ変換やチェックサム計算, 上位層変換を変換することで, IPv4host と IPv6host の通信が可能であることを示した.

今後の課題としては以下のものがあげられる.

1. IPv6host から IPv4host への通信を可能とする.
2. IP ヘッダ変換での拡張ヘッダと fragmentation の変換処理, 経路制御ヘッダの変換ルール.
3. FTP 通信などに必要となるペイロードの変換処理.
4. IPsec に関する処理.
5. DNS での cache サーバや逆引き処理の実装.

## 参考文献

[1] 社団法人日本ネットワークインフォメーションセンター (accessed August. 2009). http://www.nic.ad.jp/.

[2] nlnetlabs.nl (accessed October. 2009). http://www.nlnetlabs.nl/.

[3] Iperf (accessed September. 2009). http://dast.nlanr.net/projects/Iperf/.

[4] Ihara, A., Murase, S. and Goto, K.: IPv4/v6 Network Emulator using Divert Socket, *Proc. of 18th International Conference on Systes Engineering(ICSE2006)*, Coventry, UK, pp.159–166 (Sep.2006).

[5] 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 の相互互換, 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 (1997).