修 士 論 文

IPv6 ネットワーク上で
IPv4 アプリケーションを利用可能にする
アドレス変換技術の開発

平成 25 年度修了

三重大学大学院工学研究科
博士前期課程 電気電子工学専攻
通信工学研究室
井上 雄太

目 次

図 目 次

表 目 次

第 1 章

序論

1.1 背景

インターネットは様々なインターネットプロトコルで構成されており, インターネットプロトコルの中でも, OSI 参照モデルにおいてネットワーク層に位置付けられる IPv4 がある. IPv4 は 2 の 32 乗個の IPv4 アドレスを持ち, インターネットを利用するユーザはこの IPv4 アドレスを用いて通信を行っている. インターネットへの参加組織やユーザの増大により, 32bit のアドレス長を持つ IPv4 アドレスは将来的には枯渇してしまうことが予想される [1]. IPv4 アドレスの数は約 43 億個と数に限りがあり, 全てのユーザに IP アドレスを割り当てることに対する根本的な解決策が必要となっている. そこで, 128bit のアドレス長を持つ IPv6 が次世代のインターネットプロトコルとして注目されている [2]. IPv6 は 2 の 128 乗個のほぼ無限の IPv6 アドレスを持つことができるため, インターネットユーザが多数増大したとしても, すべてのユーザーにグローバルな IP アドレスを割り当てることが可能となる.

しかしながら, IPv4 と IPv6 にはヘッダ形式の違いやアドレス長の違いにより相互の互換性がないことに加えて, 実用性やコストの観点から既存の IPv4 ネットワークインフラストラクチャーを IPv6 へ移行させることは難しい. そこで, IPv4 と IPv6 の共存期間から IPv6 へ完全に移行する間にアプリケーションなどの資産が動作することを可能とするメカニズムが必要になっている [3] [4].

IPv4-IPv6 共存技術として, 単一機器に IPv4 と IPv6 の異なるプロトコルスタック

を共存させる IPv4-IPv6 デュアルスタック技術 [5] [6] [7] [8] や, IPv4 ネットワークを通して IPv6 ノード同士で通信ができるようにする IPv4-IPv6 トンネル技術 [9] [10] [11], IPv6 から IPv4 の通信や, IPv4 から IPv6 への通信を行うには, IPv6 と IPv4 の間を取り持つ IPv4-IPv6 トランスレータ技術が挙げられる [12] [13] [14] [15]. トランスレータ技術の一部として NAT64 や NAT-PT などが挙げられるが, これらは IPv6 ホスト同士の通信において IPv4 ネットワーク環境を中継する環境での, IPv6 ホストのための変換メカニズムである [17] [18] [19] [20]. しかしながら, これらの方法はそれぞれ課題を持っている. 例えば, IPv4-IPv6 デュアルスタックは IPv4 アドレスと IPv6 アドレスを用いるため, 将来枯渇するであろう IPv4 アドレスを用いることが難しいこと, またトンネルメカニズムはトンネルサーバを必要とするためエンドノードだけを使用することが難しいといった課題が挙げられる [16]. 本研究は IPv4 アドレスが枯渇した後の世界を想定しているため, IPv6 ネットワークのみ存在するものとする. この状況において, デュアルスタックは IPv4 アドレスが枯渇しているためデュアルスタックノードとして動作することができないこと, トンネリングに関しては全てのネットワークが IPv6 で構成されているため仮想的なトンネルを構築する必要がないことに加え, 例えば IPv4 アプリケーションを持つ IPv6 ホストが IPv4 アプリケーションを用いて通信を行うにはトンネルサーバを通信を行う相互のネットワーク上に配置しなければならない手間が挙げられる. トランスレータに関しては, 変換機能を実装した端末のみが IPv4 アプリケーションから IPv6 に対して通信を行えることができる.

既存の IPv4 アプリケーションはプログラムをコーディングする段階で IPv4 ソケットを用いて作成されているアプリケーションが多く, これらの IPv4 アプリケーションは IPv6 ネットワークへ移行した際にはソケットレベルのプログラミングの違いにより IPv4 アプリケーションは IPv6 上で用いることができない. IPv4 と IPv6 の両方に対応しているアプリケーションも存在するが, IPv4 にしか対応していないアプリケーションが大半である. IPv6 に対応していない IPv4 アプリケーションを IPv6 に対応させることは難しいことではないが, 大規模なアプリケーションやソースコードがないアプリケーションなどは IPv6 に対応させるには莫大な費用がかかることが予想される.

そこで, トランスレータ技術の一つであり, エンドノードの変換メカニズムとし

て挙げられているのが, IETF（Internet Engineering Task Force）の RFC2767 で定義されている Bump-In-The-Stack(BIS) と呼ばれる技術が挙げられる [21]. この BIS は既存の IPv4 アプリケーションが IPv6 ネットワークへアクセスできることを可能とする, エンドノードのための IP 変換メカニズムである. BIS を用いることで, IPv6 に対応していない IPv4 アプリケーションも多額のメンテナンス費用がかかることなく, またソースコードがない状況においても IPv6 上で動作させることを可能にする. しかし, RFC2767 には概要しか記載されておらず具体的な実装手法についての記載が一切されていない.

1.2　本研究の目的

IPv4 アドレス枯渇後には IPv6 アドレスのみしか各端末に割り当てが行われないことが予想されることから, IPv6 アドレスのみしか利用できない端末においても, 既存の IPv4 アプリケーション資産を有効利用できる IPv6-IPv4 トランスレーション技術を確立することを本研究の目的とする. 提案方式の実用化を検証するにあたり, 提案方式を Linux OS 上に実装し, 通信スループット性能及び提案方式の負荷を明らかにする. 具体的な性能としては, 通常の IPv6 通信と比較して, 通信スループットが同等なレベルの特性を示すようにパケット処理のオーバヘッドが少ない実装を行う. 提案方式は, 端末内に仮想の IPv4 アドレスを割り当てたインターフェースを作成し, IPv4 アプリケーションが仮想 IPv4 アドレスを利用することにより, IPv6 アドレスしか割り当てられていない端末においても, IPv6 ネットワークを通した通信を実現するものである. 実際の動作において, IPv4 アプリケーションを持った端末は通信相手である IPv6 端末と通信を行う際に名前解決を行う. DNS [24] [25] [26] を用いて名前解決を行う際に, IPv4 アプリケーションは IPv4 アドレスの A レコードで通信相手のアドレスを解決しようとするが [27], IPv6 ネットワークに存在する DNS サーバは IPv6 アドレスの AAAA レコードに対応しているため, この IPv4 アプリケーションが送信した DNS クエリを受け取ることができない [28] [29] [30] [31]. また, DNS サーバが返信する AAAA レコードを IPv4 アプリケーションは受信することができない. これらの問題を解決するため, Linux OS のカーネルモジュールとしての実装を行い, Linux OS 内で処理される IP パケットのヘッダ情報を, カーネルモジュールを用いて変更することを予定としている. さらに, 提案方式の実用上評価を行うため, Linux OS である ubuntu にカーネルモジュールを実装する.

1.3　本論文の構成

第 2 章において IPv4-IPv6 の共存技術と提案方式のアイデアとなる Bump-In-The-Stack について説明する. 第 3 章では, 提案方式における概要と本論文で目的となる IPv4-IPv6 変換モジュールについて説明する. 第 4 章において, 提案方式の実装及び実験結果をもとに有効性を検証する. 最後に第 5 章で本論文のまとめと今後の課題について述べる.

第 2 章

IPv4-IPv6 共存技術の関連研究と Bump-In-The-Stack

2.1 関連研究

IPv4 と IPv6 を共存技術の関連研究について説明する. IP の共存技術としてデュアルスタック, トンネリング, トランスレータについてそれぞれ説明する.

2.1.1 デュアルスタック

IPv6 へ移行するための過渡的な技術の一つであり, IPv4 と IPv6 を共存させる技術としてデュアルスタックがある. IPv4 と IPv6 とは異なったプロトコルであるため, 両方を同時に使用するためにはトランスレータを用意するか, IPv6 パケットをカプセル化するなどの方法を採る必要がある. デュアルスタックは, そうした手間を払うことなく, 一つのルータやマシンが IPv4 と IPv6 のアドレスをそれぞれ持って両プロトコルを混在させることができる点が特徴となっている. IPv4 対応機器と通信を行う際には IPv4 を使用し, IPv6 対応機器と通信を行う際には IPv6 を使用することになる. なお, 現在の IPv6 対応機器の大半は IPv4 にも対応しており, デュアルスタックであると言える. こうした傾向は IPv6 が普及しきるまで続くと予想されており, デュアルスタックのネットワーク機器は今後も普及が進むであろうという予測もある.

図 2.1 にデュアルスタックの動作概要を示す. IPv4 ネットワークと IPv6 ネットワー

クが混在している状況で, IPv4 ノードが IPv6 ネットワークにアクセスする際はデュアルスタックが自身に埋め込まれている IPv6 プロトコルを用いて IPv6 ネットワークに接続する. 同様に, IPv6 ノードが IPv4 ネットワークにアクセスする際はデュアルスタックは IPv4 プロトコルを用いて IPv4 ネットワークに接続をする. このような動作をデュアルスタックが行うことで IPv4 と IPv6 の混在環境においても通信が可能となる. デュアルスタックを用いる利点としては, アプリケーションによっては, IPv4 のみでしか動作しないものもあるため, そのような場合においては, デュアルスタックでは旧環境を残しつつ IPv6 に対応させることが可能なため, IPv6 への移行がしやすくなる [32] [33]. しかし, デュアルスタック実装時には, 単一の機器で IPv4 と IPv6 という二つのプロトコルが同時に動作することになるため, 機器の負荷も相応に増大する. 具体的には, IP パケットの行き先を決定する経路制御処理が, IPv4 と IPv6 それぞれで行われることになるため, CPU やメモリにも負荷がかかる. また, 将来的にはデュアルスタックは IPv4 アドレスと IPv6 アドレスを用いるため, 将来枯渇するであろう IPv4 アドレスを用いることが難しいと考えられる.

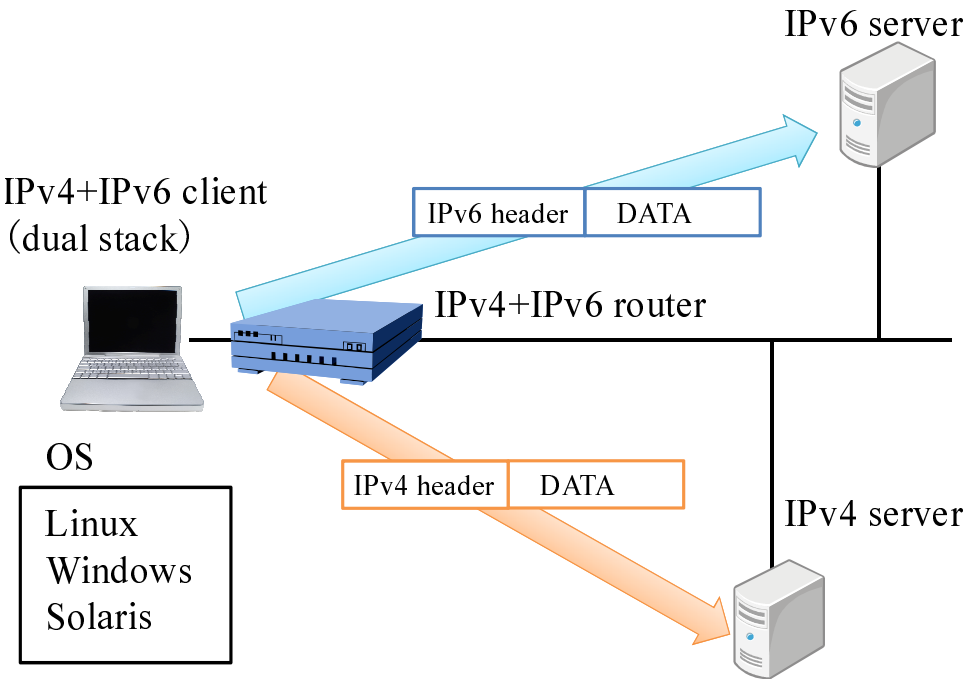

図 2.1 デュアルスタックの動作概要

2.1.2　トンネリング

ネットワーク上に外部から遮断された見えない通り道を作るように見えることからトンネリングと呼ばれる. 通信ネットワーク上の 2 点間を結ぶ閉じられた仮想的な直結回線を確立し, 本来通信を行いたいプロトコルで記述されたパケットを, 別のプロトコルのパケットで包んで送り届けることにより通信を行う. パケットのカプセル化とその解除はトンネルの両端の機器が自動的に行うため, トンネルで結ばれた機器同士は途中の通信方式や経路を気にする必要はなく, あたかもトンネルの両端の機器が直結しているように見える. 具体的には, IPv4 と IPv6 の間でもトンネリング技術を用いることにより, 互いのパケットを通すことができるようになる. 将来的にはインターネットは IPv6 で構築されるため, その中を既存の IPv4 アプリケーションが送信する IPv4 パケットで通すために, IPv4 over IPv6 トンネリング技術を用いるケースが多くなると予想される. なお, パケットのカプセル化／デカプセル化はルータで行う場合が多いため, デュアルスタックとトンネリングを組み合わせることにより, 現在のインターネットで IPv6 をスムーズに共存させることが可能になる [34] [35]. 図 2.2 に IPv4 over IPv6 トンネリングの概要を示す. しかし一方では, トンネルに入った後そのトンネルから抜け出せなくなり無限にパケットがループしてしまう迷子パケット問題や, それに伴う経路制御の問題, また元々のパケットにカプセル化ヘッダが付加されることからオーバヘッドが大きくなってしまうことも課題として挙げられる [36].

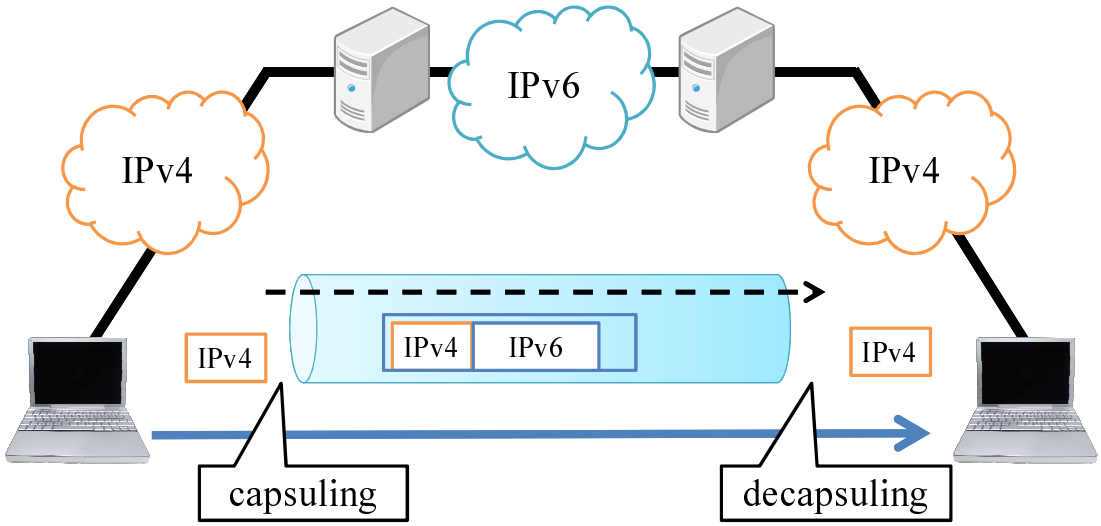

図 2.2 IPv4 over IPv6 トンネリングの概要

2.1.3　トランスレータ

現在, グローバルアドレス空間やプライベートアドレス空間, IPv4 アドレス空間や IPv6 アドレス空間にまたがって通信を行うために, NAT が広く用いられている. NAT の基本的な動作例を図 2.3 に示す. LAN 内にあるプライベートアドレスを持つ host1 がグローバルネットワークにいる host3 と通信を行う際には, NAT 機能を備えたルータを介して送信元のアドレスを host1 が持つ 192. 168. 50. 201 から NAT ルータが持つ 200. 1. 1. 1 に変換させる. その時に NAT ルータは host1 のアドレスと自身のアドレスをマッピングさせることで, 再び外から host1 に送られたパケットを NAT ルータが識別して LAN 内のホストへ送信することができる.

しかし, NAT ではエンドツーエンド通信が課題となっている. このようないわゆる NAT 越え問題 [37] と呼ばれる課題に対して, STUN [38] などでは, UDP Hole Punching 技術 [39] が用いられている. これは, まずプライベート空間のホストが利用する NAT 装置のグローバル IP アドレス, ポート番号をサーバに登録し, それを各ホストが取得する. その後, 各ホストが一度通信を試みることによって NAT 装置がその IP アドレス, ポート番号を記憶するため, それ以降の通信が可能となり UDP 通信を実現する. しかし, 信頼性の高い TCP 通信を実現できないといった機能的な制限がある.

また, IPv6 アドレス空間におけるトランスレーション技術として NAT-PT(Network Address Translation Protocol Translator) がある. NAT-PT は IPv6 ネットワークと IPv4 ネットワークを中継する NAT-PT ゲートウェイにて, SIIT(Stateless IP/ICMP Translation Algorithm) アルゴリズム [40] を用い, IPv6 パケットと IPv4 パケットを変換して中継するものである. NAT-PT では IPv6 ホストの数に対応した IPv4 のアドレスプールを用意し, 動的または静的にアドレスの対応づけを行う. また, IP アドレスとポート番号を用いて IPv4 と IPv6 のマッピングを行うものに NAPT-PT [41] [42] がある.

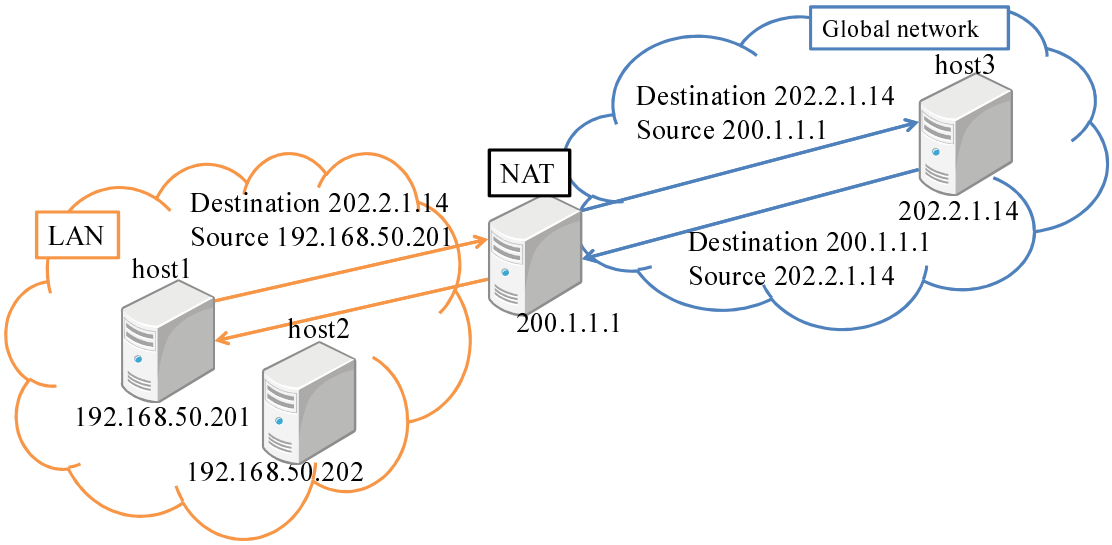

図 2.3 NAT の動作概要

2.1.4　Seamless IP Sublayer

ラベルスイッチング技術 [43] [44] を用いて異なるアドレス空間での通信を可能とする IP 層の拡張技術である [45]. Seamless IP Sublayer(SIPS) と呼ばれる IP 層と TCP/UDP 層に副層として定義することで, 既存の IP 層の機能を拡張し, ヘッダ変換, ホストの特定, ルーチングテーブルの更新, 転送処理などの機能を提供する. TCP/UDP 上で利用される既存のインターネットのアプリケーションに対して, IP と同等のインターフェースを提供するため, これらのアプリケーションをサポートすることが可能である. 図 2.4 に SIPS の構造を示す. SIPS を用いたアーキテクチャでは, 各アドレス空間の端に SIPS 対応ルータと呼ぶ中継装置を配置し, SIPS 導入において定義される SIPS ヘッダを付与したパケットを転送する. 送信ホストは, SIPS 対応ルータより割り当てられた送信先ホストのラベルを SIPS パケットの宛先ラベルに設定し, SIPS 対応ルータに SIPS パケットを送信する. SIPS 対応ルータ間の IP パケットの転送は, IP ネットワークにおける既存のルーチングメカニズムを用いる. つまり, 異なる IP アドレス空間を経由する場合のみ SIPS 対応ルータが転送を行うため, 同一の IP アドレス空間内における IP パケットの転送に関して新たな対応を必要としない.

SIPS による具体的なルーチングの仕組みは, ルックアップテーブルを用いたラベルスイッチングと DNS を用いたラベル管理である.

- ラベルスイッチング
  SIPS 対応ルータは SIPS のラベルを管理するためのルックアップテーブルを持つ. ルックアップテーブルの目的は, 転送先のホストまたは SIPS 対応ルータを特定することと, 双方向のルーチング情報を保持し, 応答パケットを送信可能にすることである. テーブルに格納される情報は, SIPS 対応ルータが生成した送信元ラベル, 送信元の IP アドレス, 転送先の IP アドレスと宛先ラベルである. 送信元のラベル値は転送先の IP アドレス毎に SIPS 対応ルータ内で一意に決められる値である. なお, このテーブルは SIPS パケットの転送方向に対してそれぞれ生成されるため, 上り下りの双方向に SIPS パケットを転送するためには二組のテーブルが必要である. SIPS パケットを中継する SIPS 対応ルータは, 受信した SIPS パケットのヘッダに含まれる宛先ラベルの値を

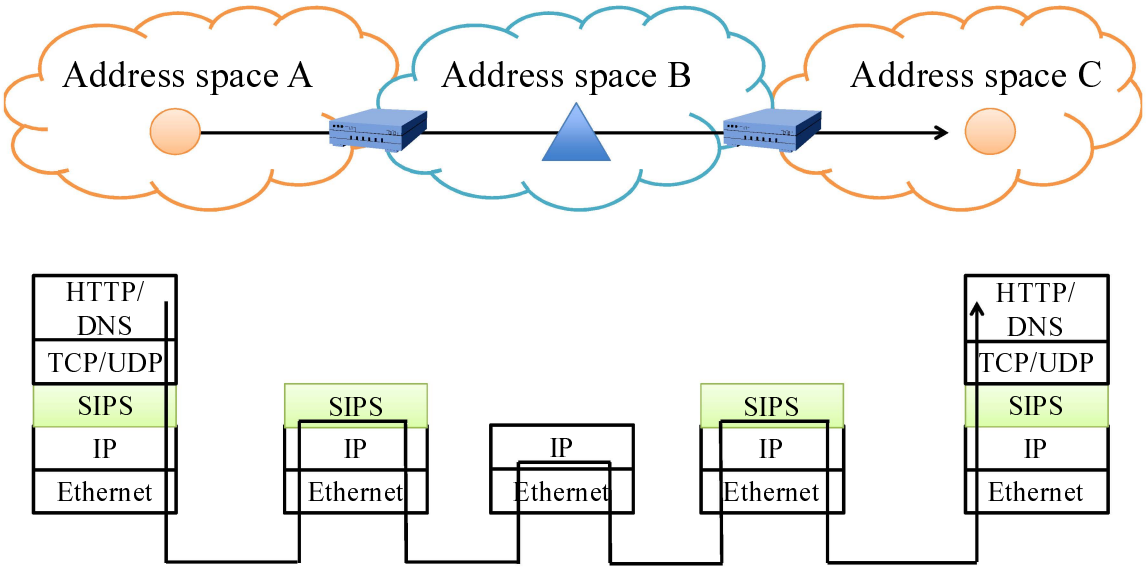

図 2.4 SIPS のアーキテクチャ概要

キーとして, 保持しているルックアップテーブルを検索し, 転送先に IP アドレスと宛先ラベルの情報を取得し, SISPS パケットヘッダの書き換えを行う.

- DNS を用いたラベルの管理
  SIPS では, SIPS 対応ルータが IP アドレスとラベル情報を取得してルックアップテーブルを作成するために DNS を利用する. また, 各ホストが DNS にラベルを登録する手段として Dynamic DNS を用いる. まず, 各ホストはネットワークへ参加して通信を開始する前にホストの FQDN, IP アドレスとラベルの登録要求を SIPS ルータに送信する. 要求を受信した SIPS 対応ルータは, ラベルの生成や IP アドレスの変換を行い DNS サーバにホストの FQDN, SIPS 対応ルータの IP アドレスと生成したラベルを登録する. 各ホストが通信を開始する際には, DNS に登録された通信相手のホストのラベル情報を問い合わせ, 中継する SIPS 対応ルータのルックアップテーブルのエントリを作成する.

以上のルーチング方式を用いることで異なるアドレス空間同士の通信を可能に

する. しかし, 異なるアドレス空間が存在する毎に SIPS に対応したルータを設置しなければならないことや, 大量のパケット送信による Dos 攻撃への対処など, セキュリティ面での考慮が課題となっている.

2.2 Bump-In-The-Stack

IETF より RFC2767 に規定されている IPv4 アプリケーションを IPv6 ネットワークで動作させるための概念であり, ここでは Bump-In-The-Stack(BIS) の概念と通信例について記述する. この BIS は RFC2767 では概要しか記載されておらず, 具体的な実装手法についての記載はないため, 本研究ではこの BIS のアイデアを元に実装手法を提案し実機に実装を行った.

2.2.1 Bump-In-The-Stack の概要

デュアルスタックやトンネリング技術が IPv4/IPv6 共存技術として挙げられたが, アプリケーションレベルに関しては, ほとんどのアプリケーションがスムーズに IP の変換を行うことが出来ない [46]. BIS はデュアルスタックホストに対してサービスを提供し, IPv4 アプリケーションが IPv6 ネットワーク上で利用可能であるメカニズムを提供する. BIS は IPv6 アプリケーションの代わりに translator, extension name resolver, address mapper の三つのモジュールを持っている. 以下それぞれトランスレータ, リゾルバ, マッパーとする. 図 2.5 にモジュールが挿入されたホストの構造を示す.

- トランスレータ
  IPv4 と IPv6 を SIIT で定義されている IP 変換メカニズムを用いて変換する. 具体的には IPv4 アプリケーションから IPv4 パケットを受け取ったとき IPv4 パケットのヘッダを IPv6 パケットのヘッダに変換し, フラグメントした後に IPv6 ネットワークへ送信する. IPv6 ネットワークから IPv6 パケットを受け取った時には対称の動作を行う.

- リゾルバ
  IPv4 アプリケーションのリクエストに対して適切なレスポンスを返却する. 一般的にアプリケーションは通信したいホストの A レコードを解決するためにネームサーバに対してクエリを送信する. このリゾルバはそのクエリをフックした後, ホストの名前解決のために A と AAAA レコードの両方のクエリを作成しサーバへクエリとして送信する. もし A レコードが解決されるの

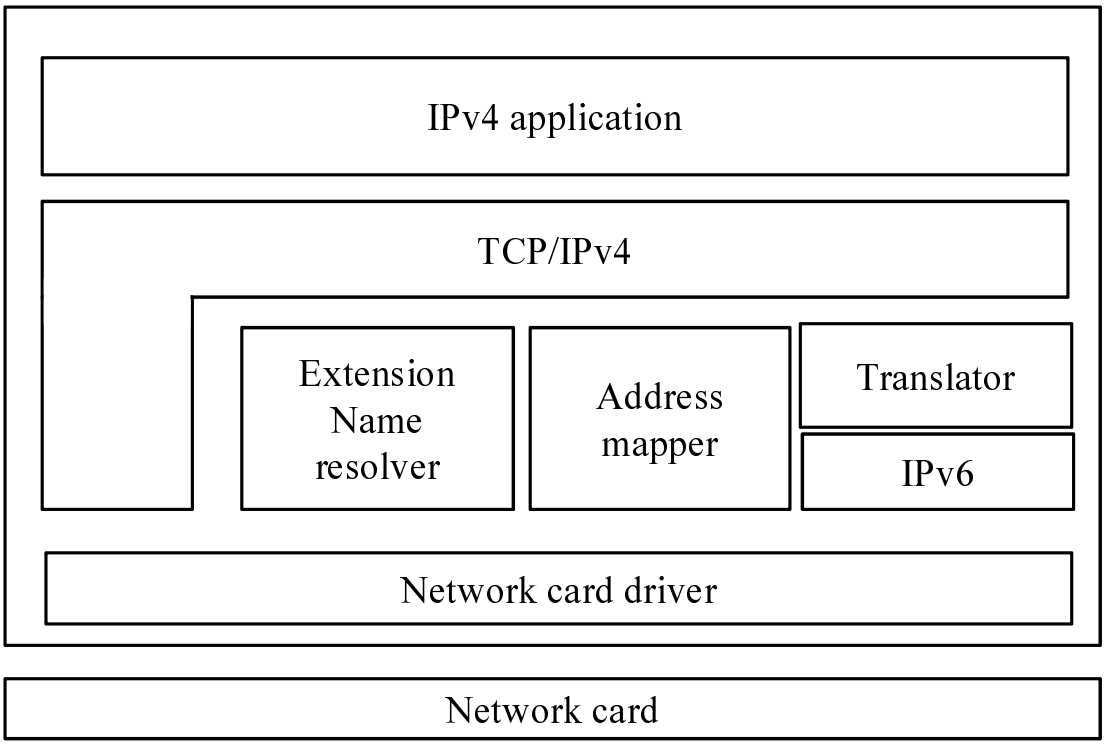

図 2.5 BIS のホスト構造

であればトランスレータによって IP 変換の必要はないが, AAAA レコードのみが利用可能である場合は IPv6 アドレスと対応した一つの IPv4 アドレスがマッパーに割り当てられ, その IPv4 アドレスをもとに A レコードのレスポンスを作成しアプリケーションへ返却する.

- マッパー

  IPv4 アドレスのプールを維持する. また IPv6 アドレスとそれに対応した IPv4 アドレスをペアとしたテーブルも維持する. リゾルバもしくはトランスレータが IPv6 アドレスに対応した IPv4 アドレスを割り当てる際, マッパーはそれを選択しプールの中から IPv4 アドレスを返却する. また, テーブルは新しいエントリ毎に動的に更新される. マッパーが情報を登録するタイミングとしては, リゾルバが通信相手の AAAA レコードを入手しそれが IPv6 アドレ

スの情報としてまだマッピングされていない時と, トランスレータが IPv6 パケットを受け取りマッピングエントリにその IPv6 ソースアドレスがない時である.

2.2.2　BIS の基本動作

図 2.6 に BIS を用いた送信元ホストの動作概要を示す.

1. まず初めにアプリケーションは host6 の名前解決を行うため A レコードでネームサーバにクエリを送信する.
2. リゾルバはそのクエリをフックしてから, また新たに A と AAAA レコードを解決するためのクエリを作成した後サーバへ送信する.
3. この場合, AAAA レコードのみが解決されるため, リゾルバはマッパーに IPv6 アドレスに対応する一つの IPv4 アドレスの割り当てをリクエストする.
4. マッパーはプールの中からその IPv4 アドレスを選択しリゾルバへ返却する.
5. リゾルバはその割り当てられた IPv4 アドレスのための A レコードを作成しアプリケーションへ返却する.
6. 通信相手の名前解決ができたアプリケーションは host6 へ IPv4 パケットを送信する.
7. その IPv4 パケットはトランスレータに届いた後, IPv4 パケットを IPv6 パケットに変換しようとする. しかし, IPv4 送信元アドレスと送信先アドレスの変換の仕方を知らないため, トランスレータはマッパーに対してマッピングエントリがあるか否かをリクエストする.
8. マッパーが自身のマッピングテーブルをチェックし対応するエントリを見つけてから, IPv6 送信元アドレスと送信先アドレスをトランスレータに返却する.
9. トランスレータはその IPv4 パケットを IPv6 パケットに変換してから必要に応じフラグメントを行い, IPv6 ネットワークへ送信する.

以上のような段階を踏むことにより, IPv4 アプリケーションは IPv6 ネットワーク上においても利用可能となる.

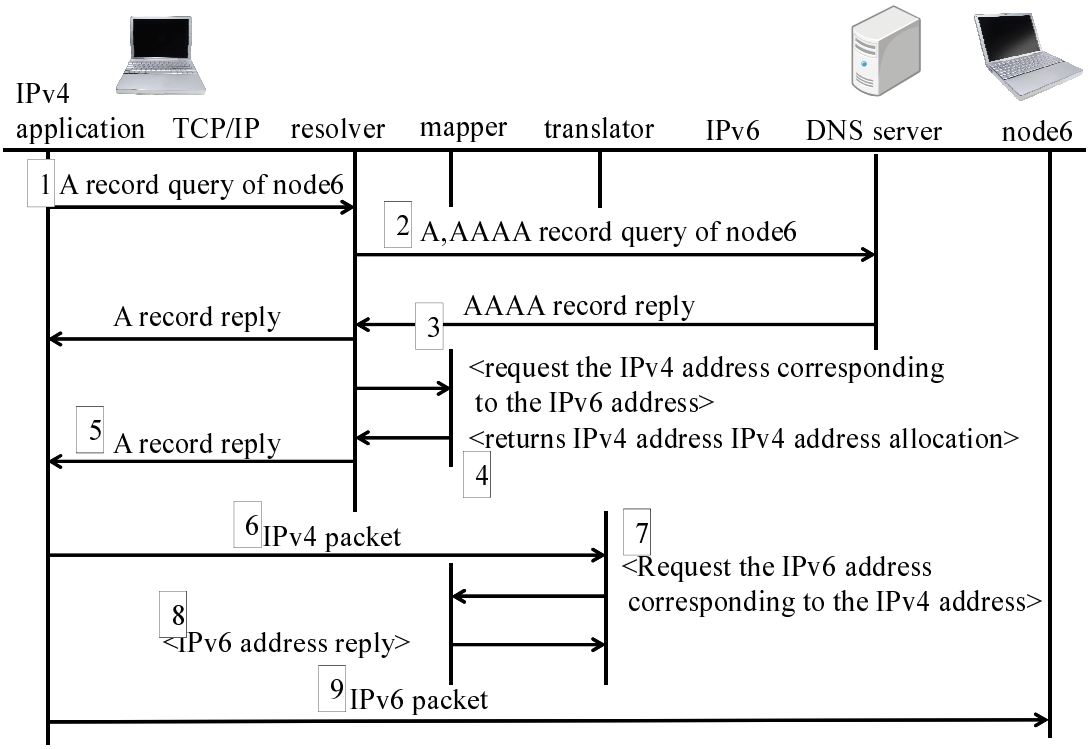

図 2.6 BIS を用いた送信元ホストの動作概要

第3章

IPv6ネットワーク上で IPv4アプリケーションを利用可能にするアドレス変換技術

本章では, IPv6 ネットワーク上で IPv4 アプリケーションを利用可能にするアドレス変換技術の提案を行う. IPv4 アプリケーションは IPv4 アドレスが枯渇した後の IPv6 ネットワーク上ではヘッダ形式やアドレス長の違いにより利用することができないため, IPv4 アプリケーションのためのアドレス変換技術をカーネルモジュールとして挿入することにより IPv4 と IPv6 の通信を実現する. この手法にはパケットをフック, アドレスを変換, DNS パケットを変換するメカニズムが必要であり, Linux のカーネルモジュールとして元のカーネルに組み込むことで IPv4 アプリケーションが IPv6 ネットワーク上で動作することを可能としている.

3.1 提案法における概要

提案方式の大きな目的は, IPv4 アプリケーションが IPv6 ネットワーク上でも高いスループットを所持した上で動作することである. 図 3.1 は提案法の概要図であり, 環境は IPv4 アプリケーションを持った送信元端末と通信相手が持つ IPv6 アプリケーション, DNS サーバにより構成されている. IPv4 ネットワーク枯渇後の世界を想定しているため, DNS サーバには IPv6 ネットワークの世界で用いられている

AAAA レコードだけが格納されており, それぞれの端末に割り当てられている IP アドレスは IPv6 アドレスのみとする. また, 提案法では, カーネルモジュールとしての実装を想定しているため, カーネルのソースコードの修正なしで目的を達成することが可能となる.

BIS とは異なり, 本研究では送信元端末に仮想インターフェース tun を作成し, 送信元の仮想アドレスとして仮想 IPv4 アドレスを割り当てる [47] [48] [49]. tun は大抵トンネルを構築するために使われるが, 実装において仮想インターフェースは仮想 IPv4 アドレスが割り当てられているため, tun メカニズムは利用可能となる. 提案法の実装は二つの仮想 IPv4 アドレスを与え, 一つは仮想的なネットワークインターフェースのための送信元 IPv4 アドレスと, IPv4 アプリケーションのための宛先端末に対応した宛先 IPv4 アドレスの二つである. このように IPv4 アプリケーションは仮想 IPv4 アドレスを用いることで接続を確立することができる. 送信元端末と通信相手の物理的な通信間では, これらの仮想 IPv4 アドレスは物理的な IPv6 アドレスに対応して変換される. 結果として通信相手である IPv6 アプリケーションは物理的な IPv6 アドレスを用いて通信することができ, また一方で IPv4 アプリケーションは仮想 IPv4 アドレスを用いて通信することが可能となる.

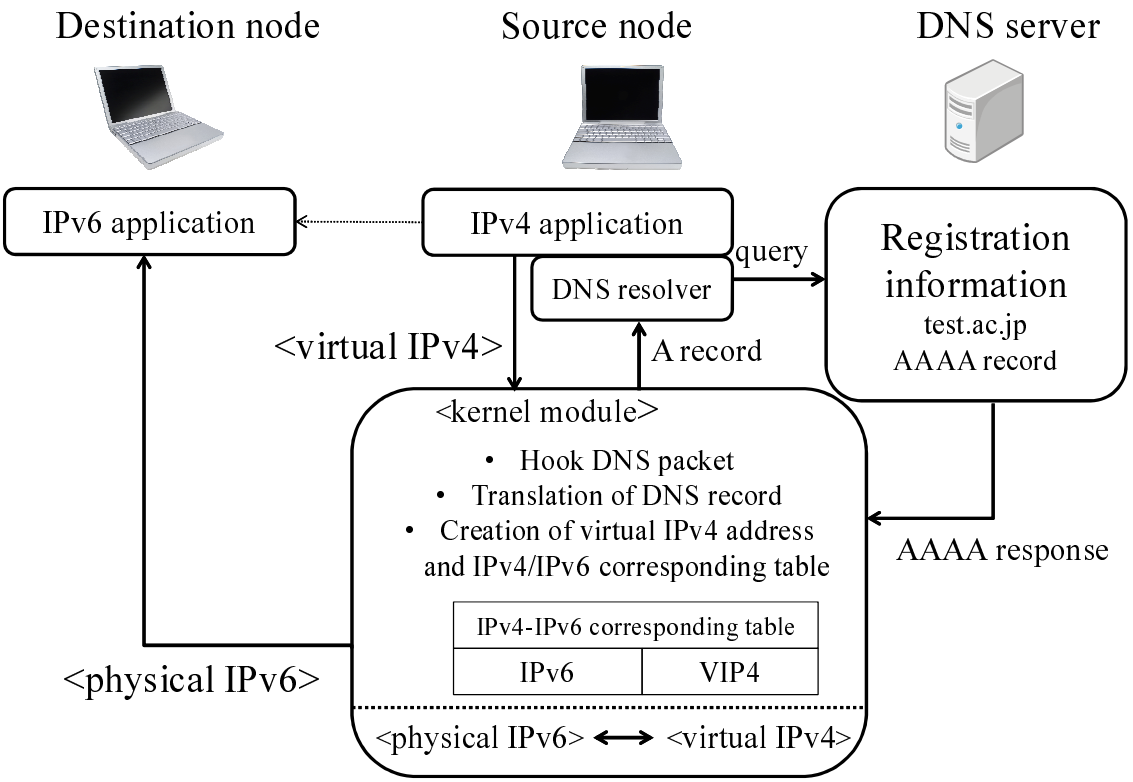

図 3.1 提案法のシステムモデル

本論文において, 提案するカーネルモジュールの機能は以下の通りである.

- IP パケットのフック
- IPv4/IPv6 変換
- DNS メッセージ変換
- 仮想アドレスと対応テーブルの作成

以上の機能の概要をそれぞれ説明する.

3.2　IP パケットのフックと変換

提案方式のカーネルモジュールは Linux ネットワークタスク上で IP パケットをフックする. 初めに IPv4 アプリケーションが送信する IPv4 パケットである DNS クエリをフックし, IPv6 ネットワーク上に適した形である IPv6 パケットに元のパケットの IP ヘッダを書き換えて DNS サーバに送信する. また, 変換テーブルを作成するために, DNS の AAAA レスポンスの IP パケットをフックして通信相手の IPv6 アドレスに対応した仮想 IPv4 アドレスを割り当てる. さらに, カーネルモジュールは仮想 IPv4 アドレスと物理 IPv6 アドレスを対応させた情報を IPv4/IPv6 変換テーブルに登録する. IPv4 と IPv6 間でパケットの IP 変換を達成するために, カーネルモジュールは送信された IPv4 パケットをフックし, 変換テーブルを参照し IPv4 から IPv6 へヘッダを変換する. 最後に, 変換された IPv6 パケットを Linux カーネルに返却する. 結果として, 送信元の端末は IPv6 宛先端末へ IPv4 パケットから変換された IPv6 パケットを送信する. また, 通信相手からの IPv6 パケットにおいては, 提案法を実装した端末は IPv6 パケットを受信した際にそのパケットをカーネルモジュールがフックし, IPv6 パケットから IPv4 パケットへ変換する.

3.3　DNS メッセージ変換

DNS は主に FQDN と IP アドレスのマッピングを制御するために使用されており, インターネットを使った階層的な分散型データベースシステムである. 具体的にインターネットに接続されている全ての端末には, それぞれ固有の IP アドレスが割り当てられており, 自身が通信したい端末と接続を図るには, その端末の IP アドレスを知る必要があるが IP アドレスは数字の羅列で人の立場から見ると個々の端末の IP アドレスを全て把握することは難しい. このため人間では覚えにくい IP アドレスを分かりやすい形で記述したのが FQDN でそれをマッピングさせて管理しやすいシステムを作ったのが DNS である. 図 3.2 に DNS メッセージフォーマットを示す. DNS メッセージにはヘッダ, 質問, 回答, オーソリティ, 追加情報の五つのセクションから成り立つ.

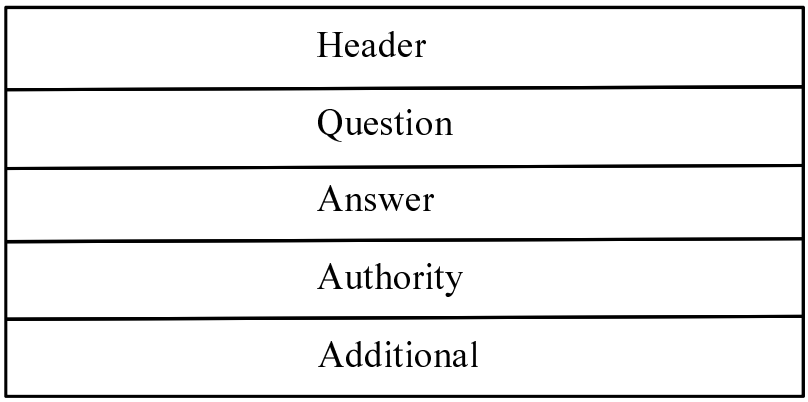

図 3.2 DNS メッセージフォーマット

図 3.3 にヘッダセクションフォーマットを示す. ヘッダセクションはどの DNS メッセージにも必ず存在し, 他のどのセクションが存在するかを示すフラグや, DNS メッセージが運ぶリソースレコード数などが格納される. 図 3.4 に DNS の質問セクションフォーマットを示す. フォーマット内にある QNAME は問い合わせの対象となるドメイン名を格納するフィールドであり, QTYPE と呼ばれるフィールドは 16 ビットでリソースレコードの種類を示す. このリソースレコードには様々な種類が存

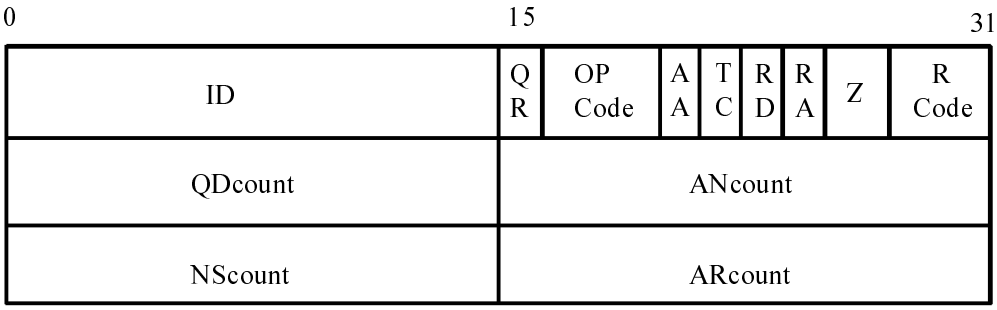

図 3.3 DNS ヘッダフォーマット

在し, 本論文で扱うメインとなるリソースレコードのタイプは IPv4 アドレスを格納している A レコードと, IPv6 アドレスを格納する AAAA レコードがある. 本論文の DNS クエリにおいてこの QNAME には”test.ac.jp”がドメイン名として格納されており, QTYPE には IPv4 アプリケーションが送信するであろう A レコードを格納している. また, 図 3.5 にリソースレコードのフォーマットを示す. このリソースレコードは DNS の回答セクションに, クエリの回答となるリソースレコードが格納される. Name にはリソースレコードのドメイン名が記載され, TYPE ではリソースレコードの種類を示す. また, CLASS はリソースレコードのクラスを示し, TTL はキャッシュに保存されるリソースレコードの有効期限を示す. RDATA にはリソースレコードの値が格納され, リソースレコードが A レコードであれば IPv4 アドレスがこのフィールドに格納される.

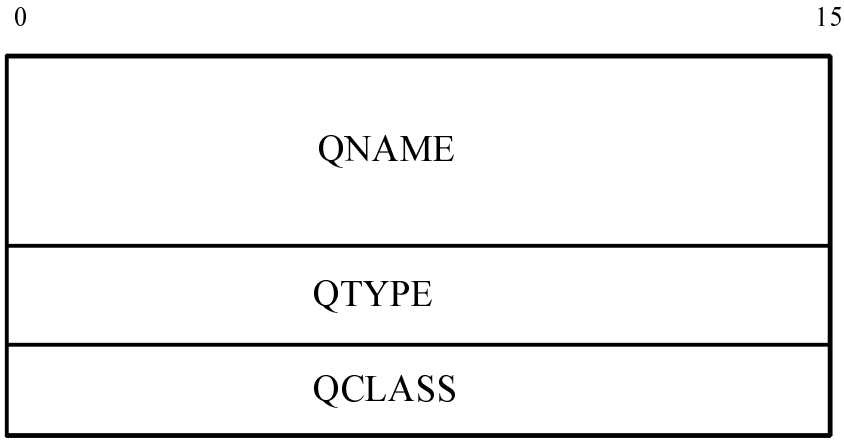

図 3.4 DNS 質問セクションフォーマット

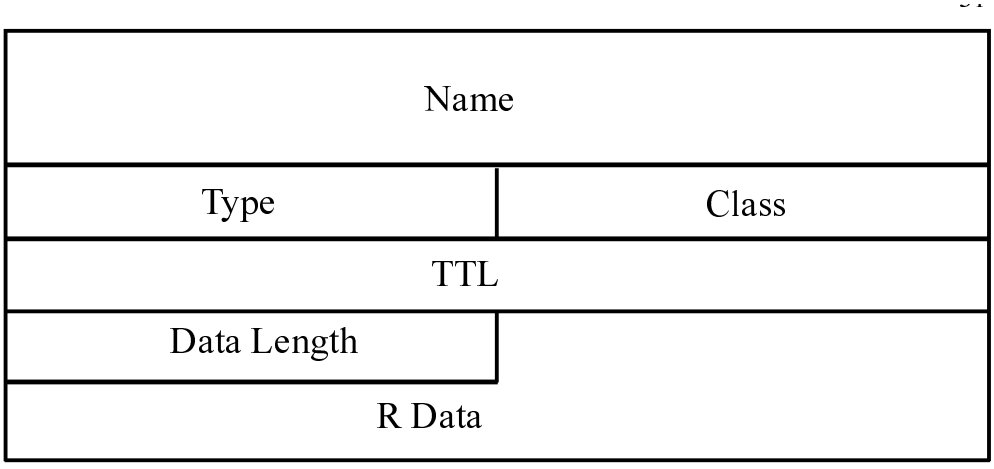

図 3.5 DNS リソースレコードフォーマット

本論文では DNS メッセージ変換は DNS パケット送信時と DNS パケット受信時に行われるため, それぞれの変換方法を記載する.

- DNS パケット送信時
  図 3.6 に DNS パケット送信の様子を示す. 本論文では IPv4 アプリケーションが通信開始を行うことを想定しているため, DNS に対して通信相手の IP アドレスを取得する際に A レコードのクエリを送信する. このときに提案法は IPv4 アプリケーションが送信した A レコードのクエリを IPv6 ネットワークに適した AAAA レコードに変換する. 具体的に変更する箇所としては, DNS の質問セクション内の QTYPE に格納されているリソースレコードを A レコードから IPv6 ネットワークに適した形の AAAA レコードに変換する. このような変換処理を施すことで DNS サーバに対して IPv6 ネットワーク上に適した形でパケットを送信することができる.

- DNS パケット受信時
  図 3.7 に DNS パケット受信の様子を示す. DNS サーバからの AAAA レスポンスに対して, リソースレコードに含まれている RData から通信相手の物理 IPv6 アドレスを取り出し, その IPv6 アドレスに対応した仮想 IPv4 アドレスを作成する. このとき作成した通信相手の IPv6 アドレスとそれに対応した仮想 IPv4 アドレスのペアは IP 変換の際に必要な情報となるため, この対の情報を変換テーブルに登録する. さらに IPv4 アプリケーションのために, DNS サーバから受け取った AAAA レスポンスの代わりとして仮想 IPv4 アドレスが含まれた A レスポンスを新たに作成する. 具体的には回答セクションに含まれるリソースレコードの Type を A レコードに変換し, RData に作られた仮想 IPv4 アドレスを格納することで IPv4 アプリケーションは DNS A レスポンスを入手することができる. 結果として, IPv4 アプリケーションは通信相手の IPv6 アドレスに対応した仮想 IPv4 アドレスを入手することができる.

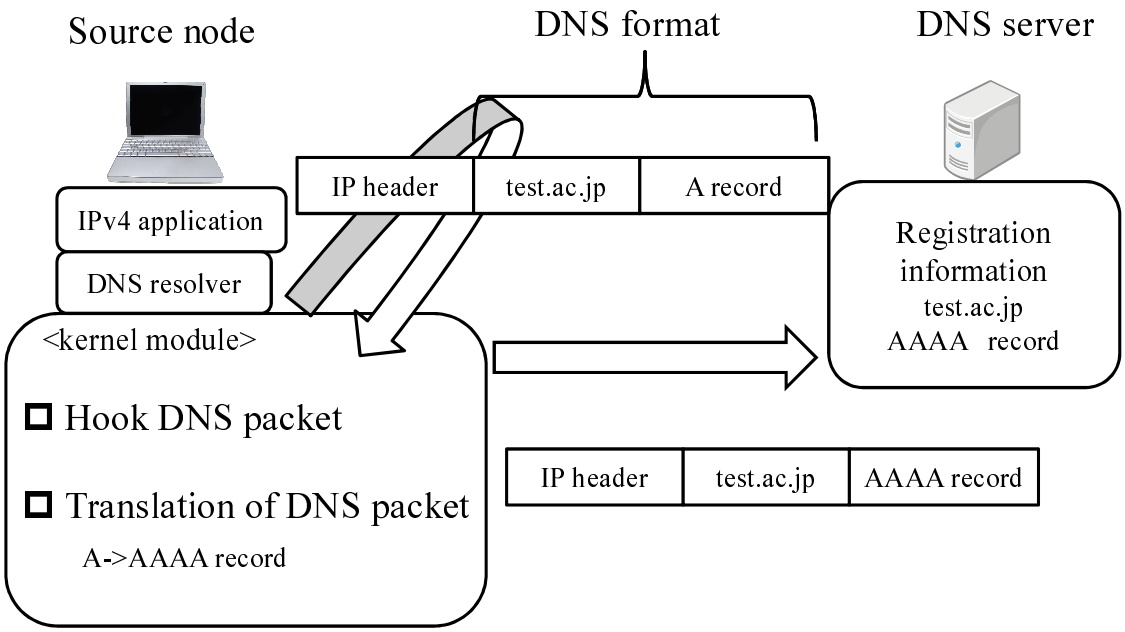

図 3.6 DNS パケット送信

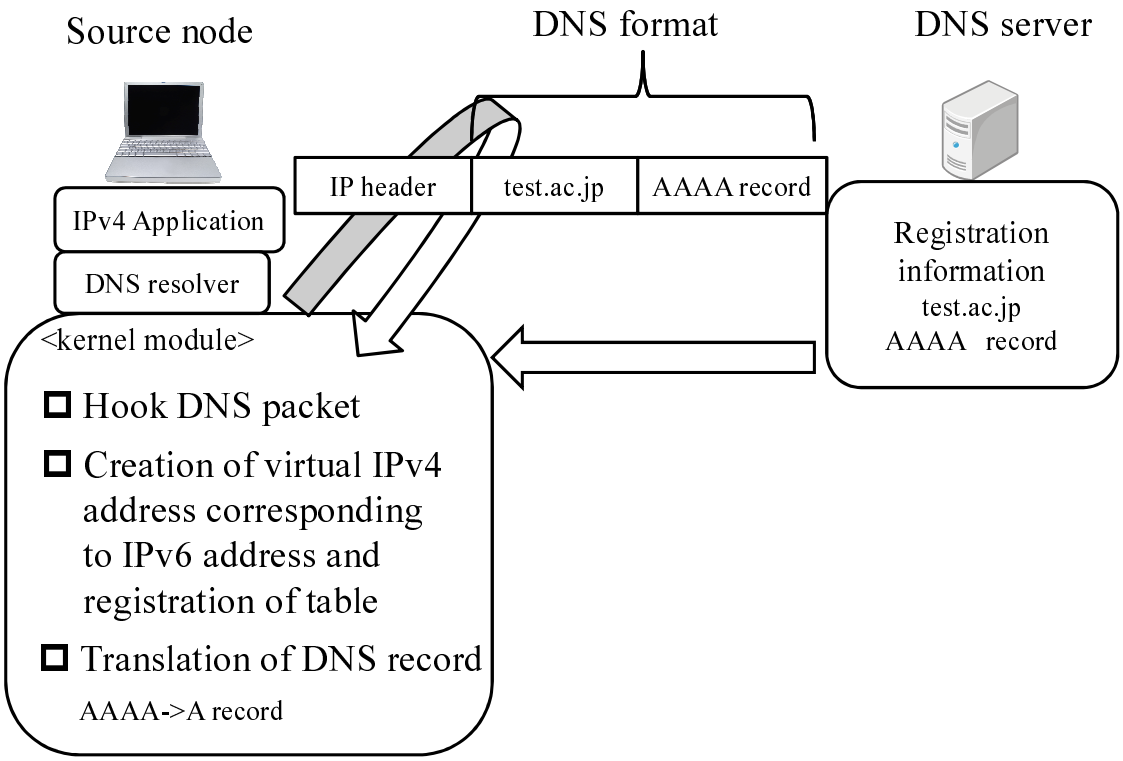

図 3.7 DNS パケット受信

また, 今回の DNS メッセージ変換処理は A レコードと AAAA レコードの変換のみであったが, 必要であれば以下に示すリソースレコードの情報にも対応できる.

DNS TYPEs

RFC 1035: Domain Names - Implementation and Specification RDATA format

NONnon-compressed domain-name

DN:domaon-name DN1

SOA:SOA format DN2 + 20B

MINFO:MINFO format DN2

MX:MX format 2B + DN

QRY:request only w/o compressed domain-name

DN . . compressed domain-name

B . . binary data in bytes

References)

DNS_TYPE_A 1 // RFC 1035 Host address (NON)

DNS_TYPE_NS 2 // RFC 1035 Authoritative name server (DN)

DNS_TYPE_MD 3 // RFC 1035 Mail destination (Obsolete) (DN)

DNS_TYPE_MF 4 // RFC 1035 Mail forwarder (Obsolete) (DN)

DNS_TYPE_CNAME 5 // RFC 1035 Canonical name (DN)

DNS_TYPE_SOA 6 // RFC 1035 Start of a zone of authority (SOA)

DNS_TYPE_MB 7 // RFC 1035 Mailbox domain name (EXPERIMENTAL) (DN)

DNS_TYPE_MG 8 // RFC 1035 Mail group member (EXPERIMENTAL) (DN)

DNS_TYPE_MR 9 // RFC 1035 Mail rename domain name (EXPERIMENTAL) (DN)

DNS_TYPE_NULL 10 // RFC 1035 Null RR (EXPERIMENTAL) (NON)

DNS_TYPE_WKS 11 // RFC 1035 Well known service Description (NON)

DNS_TYPE_PTR 12 // RFC 1035 Domain name pointer (DN)

DNS_TYPE_HINFO 13 // RFC 1035 Host information (NON)

DNS_TYPE_MINFO 14 // RFC 1035 Mailbox or mail List information (MINFO)

DNS_TYPE_MX 15 // RFC 1035 Mail exchange (MX)

DNS_TYPE_TXT 16 // RFC 1035 Text strings (NON)

DNS_TYPE_RP 17 // RFC 1183 Responsible person (MINFO)

DNS_TYPE_AFSDB 18 // RFC 1035 AFS database record (MX)
DNS_TYPE_SIG 24 // RFC 2535 Signature (MX)
DNS_TYPE_KEY 25 // RFC 2535 key record (NON)
DNS_TYPE_GPOS 27 // RFC 1712 Geographical Position (NON)
DNS_TYPE_AAAA 28 // RFC 3596 IPv6 address record (NON)
DNS_TYPE_NXT 30 // RFC 3755 next (NON)
DNS_TYPE_SRV 33 // RFC 2782 Service locator (NON)
DNS_TYPE_NAPTR 35 // RFC 2915 Naming Authority Pointer(NON)
DNS_TYPE_KX 36 // RFC 2230 Key eXchanger record (MX)
DNS_TYPE_CRET 37 // RFC 4398 Certificate record (NON)
DNS_TYPE_A6 38 // RFC 2874 DNS Extensions to Support
// IPv6 Address Aggregation and
// Renumbering (NON)
DNS_TYPE_DNAME 39 // RFC 2672 Non-Terminal DNS Name
// Redirection (DN)
DNS_TYPE_OPT 41 // RFC 6891 OPT/DNSSEC (NON)
DNS_TYPE_APL 42 // RFC 3123 Address Prefix List (NON)
DNS_TYPE_DS 43 // RFC 4034 Delegation signer (NON)
DNS_TYPE_SSHFP 44 // RFC 4255 SH Public Key Fingerprint (NON)
DNS_TYPE_IPSECKEY 45 // RFC 4025 IPsec Key (NON)
DNS_TYPE_RRSIG 46 // RFC 4034 Signature (NON)
DNS_TYPE_NSEC 47 // RFC 4034 Next-Secure record (NON)
DNS_TYPE_DNSKEY 48 // RFC 4034 DNS Key record (NON)
DNS_TYPE_DHCID 49 // RFC 4701 DHCP identifier
DNS_TYPE_NSEC3 50 // RFC 5155 NSEC record version 3 (NON)
DNS_TYPE_NSEC3PARAM 51 // RFC 5155 Parameter record for use
// with NSEC3 (NON)
DNS_TYPE_TLSA 52 // RFC 6698 Transaction Signature (NON)
DNS_TYPE_HIP 55 // RFC 5205 Host Identity Protocol (NON)
DNS_TYPE_SPF 99 // RFC 4408 Sender Policy Framework (NON)

/— Q TYPEs, Meta TYPEs —/
DNS_TYPE_TKEY 249 // RFC 2930 secret key record (QRY)
DNS_TYPE_TSIG 250 // RFC 2845 Transaction Signature (QRY)
DNS_TYPE_IXFR 251 // RFC 1996 Incremental Zone Transfer (QRY)
DNS_TYPE_AXFR 252 // RFC 1035 a transfer of an entire zone (QRY)
DNS_TYPE_MAILB 253 // RFC 1035 mailbox-related records (MB/G/R) (QRY)
DNS_TYPE_MAILA 254 // RFC 1035 mail agent RRs (Obsolute) (QRY)
DNS_TYPE_ALL 255 // RFC 1035 all records (QRY)
/— data RRTYPEs —/
DNS_TYPE_CAA 257 // RFC 6844 Certification Authority Authorization (NON)
DNS_XTYPE_DLV 32769 // RFC 4431 DNSSEC Lookaside Validation record (NON)
DNS_XTYPE_TA 32768 // RFC 4431 DNSSEC Trust Authorities (NON)
/ No-suppored types
DNS_TYPE_X25 19 RFC 1183 for X. 25 PSDN address
DNS_TYPE_ISDN 20 RFC 1183 for ISDN address
DNS_TYPE_RT 21 RFC 1183 for Route Through
DNS_TYPE_NSAP 22 RFC 1706 for NSAP address
DNS_TYPE_NSAP_PTR 23 RFC 1706 for NSAP address
DNS_TYPE_EID 31 Endpoint Identifier
DNS_TYPE_NIMLOC 32 Nimrod Locator
DNS_TYPE_ATMA 34 ATM Address
DNS_TYPE_SINK 40 SINK
DNS_TYPE_NINFO 56 NINFO
DNS_TYPE_RKEY 57 RKEY
DNS_TYPE_TALINK 58 TALINK
DNS_TYPE_CDS 59 Child DS

3.4　具体的な動作例

図3.8に端末間の通信動作例を示す. 送信元端末にはfc00::ebのIPv6アドレスと仮想インターフェースのために10. 1. 0. 235の仮想IPv4アドレスを割り当てる. また通信相手にはfc00::32のIPv6アドレスを割り当てる. そしてDNSサーバにはAAAAレコードとして通信相手のIPアドレスであるfc00::32を登録情報として登録する. DNSサーバはドメイン名として”test. ac. jp ”をセットしている. 以下に通信手順の例を示す.

1. IPv4アプリケーションは通信相手の端末のIPv6アプリケーションと通信しようと試みるため, IPv4アプリケーションは通信相手のIPアドレスを取得する必要がある. IPv4アプリケーションはDNSリゾルバを用いることで通信相手のIPアドレスを入手する. リゾルバはIPv4アプリケーションに対応したAレコードをtest. ac. jpのFQDNとしてDNSサーバに送信する. このときカーネルモジュールはDNSリゾルバが送信したDNS Aクエリをフックし, そのDNSパケットのタイプがAレコードであればAAAAレコードにタイプを変更し, DNSサーバが所持しているレコードタイプに対応した形でDNSクエリを送信する.

2. IPv4アプリケーションからDNS AAAAクエリを受けとったDNSサーバは, その端末に対して通信相手のIPv6アドレスfc00::32として格納したAAAAレスポンスを返却する

3. 送信元端末に挿入されたカーネルモジュールはDNSサーバからのAAAAレスポンスをフックする. そして格納されていたfc00::32の通信相手のIPv6アドレスの情報から, それに対応した仮想IPv4アドレスを作成し, 物理IPv6アドレスと仮想IPv4アドレスの対応情報を変換テーブルに登録する. そして, カーネルモジュールは作られた仮想IPv4アドレスをもとに新たにIPv4アプリケーションに対応したAレコードのDNSレスポンスを作成しIPv4アプリケーションに返却する

4. Aレスポンスを受け取ったIPv4アプリケーションは, その仮想アドレスを用いることによって通信相手との通信を試みる. IPv4アプリケーションはIPv4

パケットを送信するとき, カーネルモジュールは IPv4 パケットをフックして IPv4/IPv6 変換テーブルを参照して IPv4 パケットから IPv6 パケットに変換を行う.

5. 通信相手は IPv6 パケットを受け取る. 逆に通信相手から送信元端末へ通信を開始するとき, IPv6 アプリケーションは IPv6 パケットを送信する. カーネルモジュールは同様にこの IPv6 パケットをフックし変換テーブルを参照して IPv6 パケットを IPv4 パケットに変換する. IPv4 と IPv6 の変換は送信元端末内でローカルに行われるため, IPv6 のみで実行されているように見える.

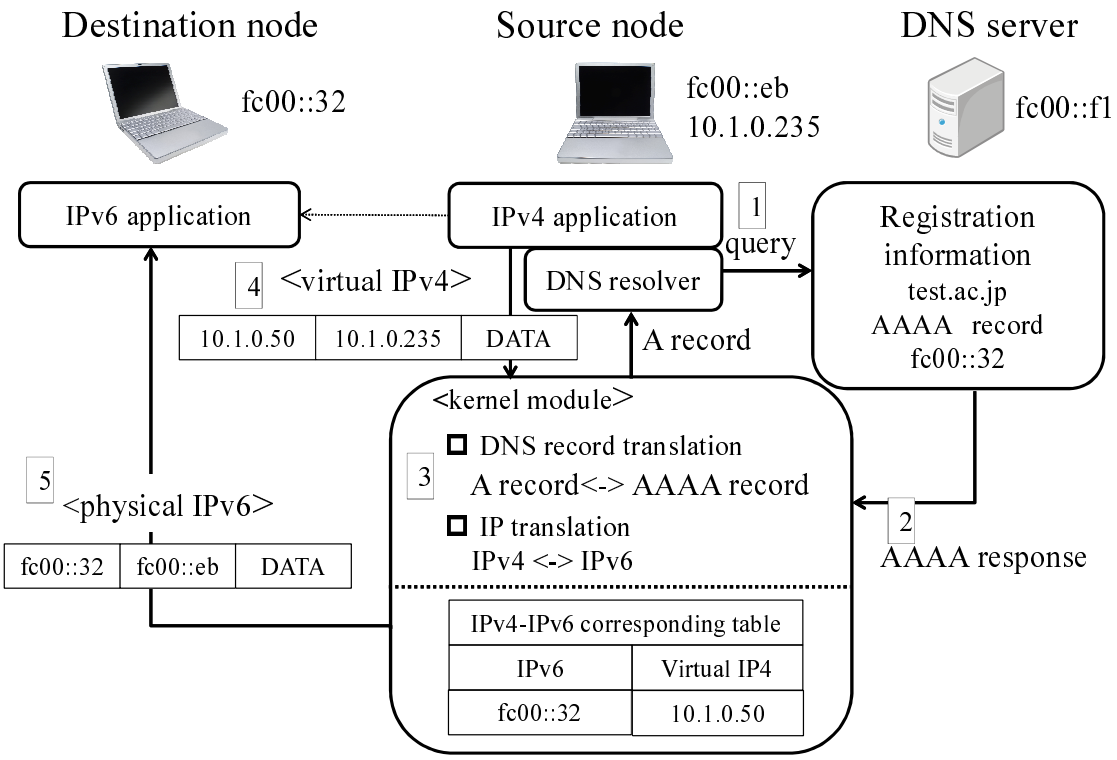

図 3.8 通信動作例

第 4 章

実装

本章では, 本研究において用いた機器の実装環境および, 動作確認, 実験時に用いた諸元について示す.

4.1 実装

4.1.1 Netfilter の概要

この提案法において, 各パケットのためのソケットバッファを処理するために, Linux 機能である netfilter を使用する. Netfilter は多くのルーティングアクションの引き金として本実装によって使用される. Netfilter は Linux プロトコルスタック内でいくつかのフックポイントとして構成されている. つまり, ユーザ定義のカーネルモジュールがこれらのフックに関数をコールバックすることを可能にする. 図 4.1 に netfilter の構造を示す. Netfilter は五つのフックポイントがあり予め定義されている. それぞれのフックポイントでは扱うパケットが異なり, 例えばローカルプロセスへパケットを送信するときは NF_LOCAL_IN のチェインがトリガーとして動作し, 自ホスト内で転送作業が行われる場合は NF_FORWARD のチェインが呼び起こされる.

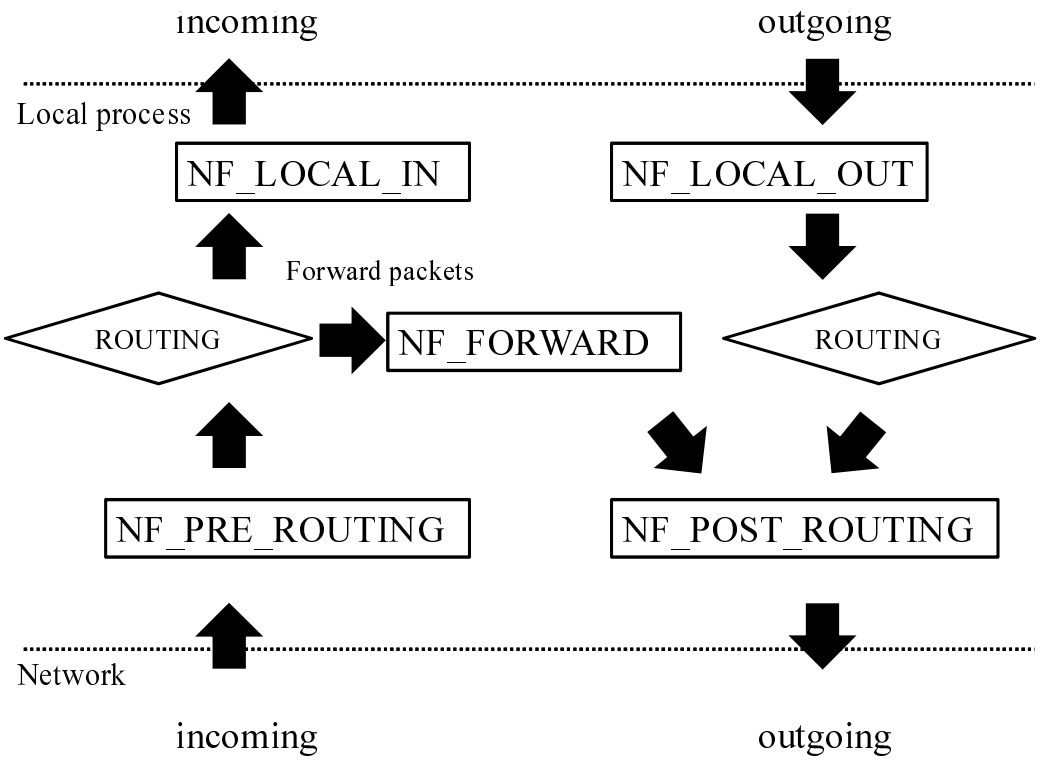

図 4.1 netfiter の構造

4.1.2　提案法の実装

図 4.2 に提案法におけるカーネルモジュールのパケットデザインを示す. 提案法は IPv4 アプリケーションを IPv6 ネットワーク上で利用可能にする実装をカーネル空間で行っている. カーネル空間では, パケットのフックや DNS メッセージの変換, IP 変換などの機能を実装することにより, 双方の通信が可能であることを実現している. 本提案手法では, パケットのフック, DNS と IP 変換機能を実装した. 以下に端末の動作概要を説明する.

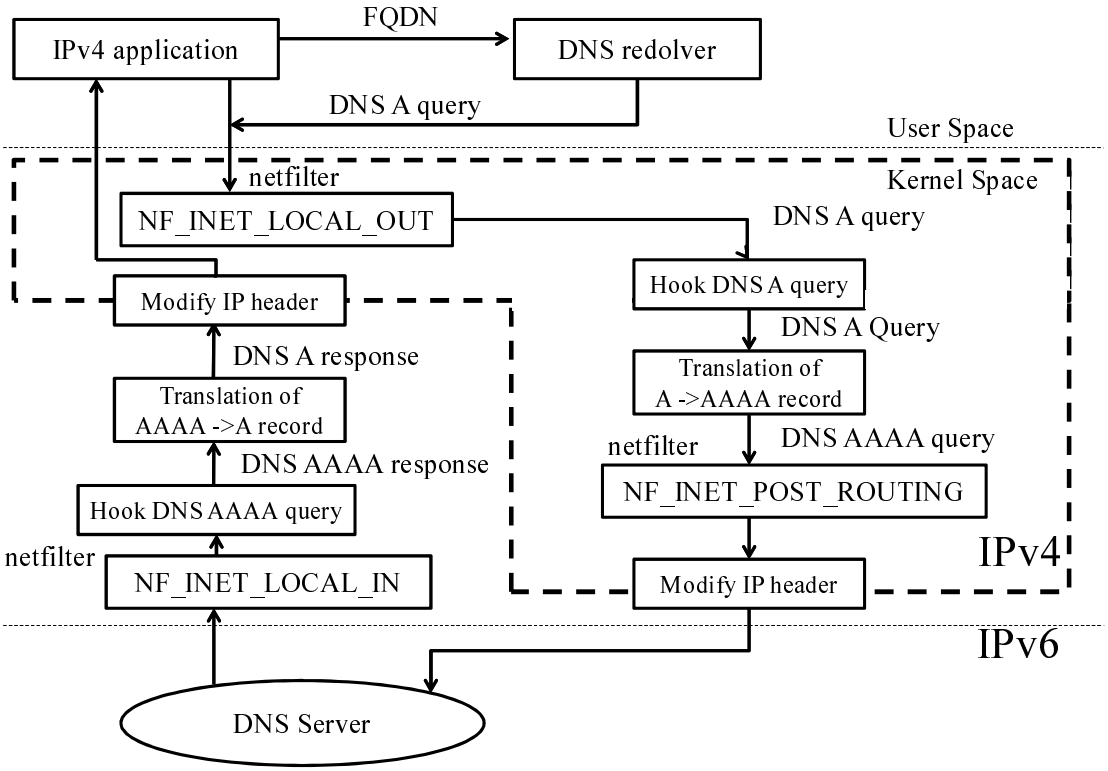

図 4.2 カーネルモジュールのパケットデザイン

- パケットのフック

  提案方式においてカーネルモジュールは, netfilter を用いることで自ホスト

を通るパケットをフックする. IPv4 アプリケーションが名前解決のために送信する IPv4 アドレスの DNS A クエリは NF_INET_LOCAL_OUT のチェインでフックする. 変換処理を施した後は元のチェインに返却する. DNS サーバからの IPv6 アドレスの AAAA レスポンスは NF_INET_LOCAL_IN のチェインでフックしレコード変換処理を行いフックしたチェインに返却する. また, IPv4 アプリケーションが通信相手に送信する IPv4 パケットは, netfilter の NF_INET_POST_ROUTING で IP 変換処理を行い, また通信相手からの IPv6 パケットは NF_INET_LOCAL_IN で IP 変換を行い元のチェインに返却する. フックポイントは図 4.1 に記載するように複数存在するが, 本論文の実装においては NF_INET_LOCAL_OUT と NF_LOCAL_IN, NF_INET_POST_ROUTING の三カ所のみの実装としている.

- IP パケットの変換
  基本的に, IPv4 アプリケーションが送信する IPv4 パケットは外部に出る時に IPv6 パケットへと変換し, 外部から送信される IPv6 パケットは IPv4 パケットに変換されるように実装されている. IPv4 から IPv6 パケットに変換する際には, IPv4 ヘッダを IPv6 ヘッダに変換することで IPv4/IPv6 変換を実現している. IPv6 から IPv4 パケットに変換する際も同様にヘッダ情報を書き換えることで変換を実現している.

- DNS パケットの変換
  IPv4 アプリケーションが名前解決を行う際に送信する IPv4 アドレスの DNS A レコードを, IPv6 ネットワーク上に適した形に変換する. 送信するクエリのレコードタイプはアプリケーションに依存するため, A レコードを含むクエリを送信する DNS パケットのみポインタをずらし AAAA レコードに書き換える. また, 受信した AAAA レスポンスは, DNS の質問セクションである AAAA クエリ部分を取得する. その後, 回答セクション内にあるリソースレコードを取り出し, DNS パケットの再設定を行う. クエリのポインタを取得した後にタイプを A に設定し, 回答セクションに含まれるリソースレコードを A レコードに設定し仮想 IPv4 アドレスを設定することで IPv4 アプリケーションのための DNS A レスポンスを作成することができる. A レスポンスを

作成する際に, バイトオーダーを適切な値に書き換えてから IPv4 アプリケーションに返却する.

- 仮想アドレスと変換テーブルの作成
  カーネルモジュールを送信元端末に組み込んだとき, カーネルモジュールは複数の仮想 IPv4 アドレスを作成しプールする. この際に, 送信元端末が持つ IPv6 アドレスと tun0 に割り当てられた仮想 IPv4 アドレスは変換テーブルに登録される. 名前解決において AAAA レスポンスを受信したとき, AAAA レコードに格納されている IPv6 アドレスとプールされた仮想 IPv4 アドレスの対応づけを行う. また, 対応づけを行った物理 IPv6 アドレスと仮想 IPv4 アドレスは IPv4/IPv6 変換テーブルに登録される. 名前解決を行う毎に, 変換テーブルを参照し, もし物理 IPv6 アドレスに対応した仮想 IPv4 アドレスの割り当てが行われていない場合に, それぞれの IPv6 アドレス一つ一つに対して仮想 IPv4 アドレスが割り当てられ, その都度変換テーブルに登録されるように実装を行った. IPv4 アプリケーションが通信相手と通信する際に用いる送信元アドレスは tun0 に割り当てられた仮想アドレスを用い, 宛先アドレスには DNS より取得し新たに作成した A レスポンスに格納されている仮想 IPv4 アドレスを用いる. 通信を行う際に, これらの仮想アドレスは変換テーブルを参照することにより物理的な IPv6 アドレスに変換されて送信される. パケットを受信する際には, IPv6 アドレスから変換テーブルを参照し IPv4 アプリケーションのための仮想 IPv4 アドレスに変換する.

4.2　動作確認

本提案手法では送信元端末に様々な機能が実装されたカーネルモジュールを組み込むことで動作確認を行った. 具体的にはパケットのフック, DNS 変換, IP 変換の機能をカーネルモジュールにおいて実装することによって, IPv4 アプリケーションが IPv6 ネットワーク上において利用可能とした. 実験諸元を表 4.1 に示す.

表 4.1 評価諸元

OS	Linux
Distribution	Ubuntu 10.04
Kernel version	linux-2.6.32-38-generic
application	iperf, own program
protocol	TCP
measurement speed	100Mbps
Number of measurement	10

動作確認に関して, 以下の手順の動作を確認した.

- パケットのフック
  netfilter を通過するパケットが制御できた.
- IPv4 アプリケーションの名前解決
  DNS サーバへ IPv6 ネットワーク上に適した形でパケットを送信した.

- DNS レスポンスの受信
  AAAA レコードから仮想 IPv4 アドレスの作成と対応づけを行った.
  仮想 IPv4 アドレスから A レスポンスの作成を行った.
- IPv4 アプリケーションの通信
  受信した A レスポンスを用いて通信相手との通信を行った.

4.2.1　スループットの測定

実装したカーネルモジュールの性能を評価するために iperf [51] と呼ばれるソフトを用いてスループットの測定を行った. 今回の測定では, 送信元端末にカーネルモジュールを挿入したときと, カーネルモジュールを挿入せずデフォルトの IPv6 を用いたときの 2 種類を比較して実験を行った. 評価の目的は, ヘッダの伸長または短縮は, 大きなオーバーヘッドが生じることがあるため, パケット操作のオーバーヘッドを確認することであった. 仮想インターフェイスは, 測定中 tun で構成されている. MSS サイズが減少すると, 総パケットサイズに対するヘッダサイズの比が大きくなるため, スループットのオーバーヘッドは, MSS サイズに依存し得る. そのため, MSS の特定のサイズでスループットを評価した. この評価から, 提案の実装では, パケット操作のオーバーヘッドを決定することができた.

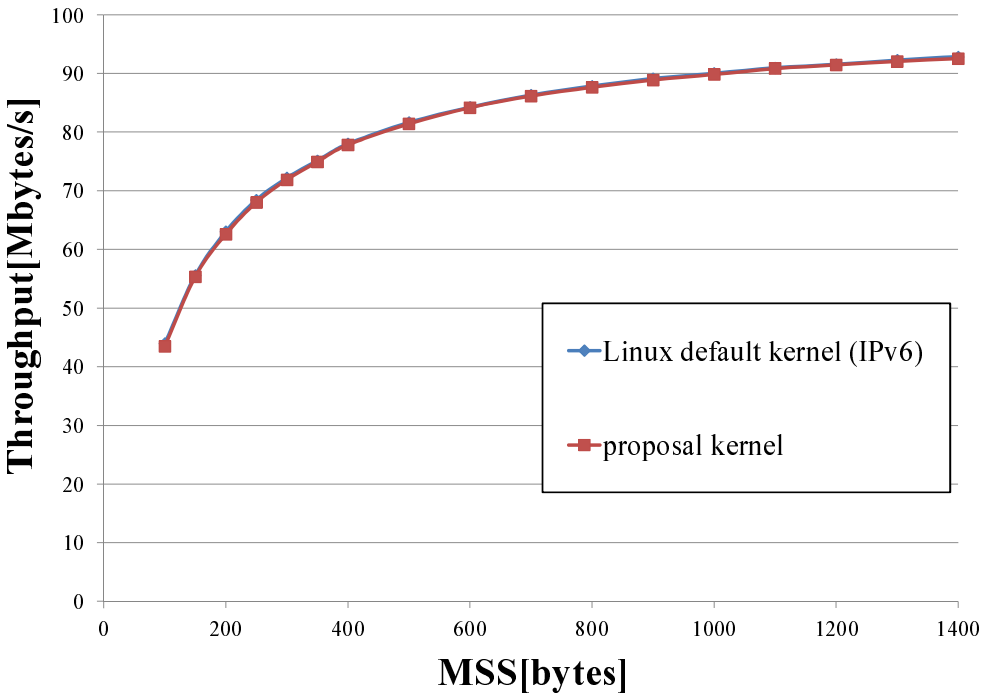

図 4.3 スループット評価

図 4.3 は, MSS の大きさが変化するようなスループット性能を示す. 結果は 10 回

の測定の平均値であり, 開発したカーネルモジュールのスループットは, 一般的な Linux カーネル IPv6 としての性能とほぼ同じレベルであることを示している. また, MSS サイズが減少するにつれて, 合計パケットサイズに対するヘッダサイズの割合が増えるため, MSS が小さいときほどオーバーヘッドが大きくなる.

4.2.2　IPv4 アプリケーションが名前解決にかかる時間

IPv4 アプリケーションが通信相手の IP アドレスを得るのにカーネルモジュールを用いたときと, 通常の名前解決を行いどの程度の遅延が発生しているのかを測定した. IPv4 アプリケーションから DNS 問い合わせをかけて, レスポンスが返ってくるまでの時間を測定するプログラムを作成した. 具体的には, リゾルバの処理＋変換処理＋ RTT ＋サーバ処理の合計を測定するプログラムを用いて測定を行った. 測定は 10 回行い平均をとったところ, 通常の IPv6 カーネルを用いた名前解決にかかった時間は 793. 1[μ s] に対して, カーネルモジュールを挿入した場合に関しては 843. 3[μ s] であった. 結果から, RTT とサーバの処理に関する遅延はほとんどないと考えられるため, パケットのフックと DNS レコードタイプの変換, IP パケットの変換にかかった時間がおよそ 50. 2[μ s] となり少しではあるが遅延が発生していることが測定できた. この遅延に関してはほぼ無視できる時間であるため本研究の提案法は有効であると考えられる.

第 5 章

結論

5.1 本論文のまとめ

本論文では, IPv4 アドレスが枯渇した後の IPv6 ネットワークにおいて, 既存の IPv4 アプリケーションを IPv6 ネットワーク上で利用可能にするアドレス変換技術の提案を行った.

提案方式におけるカーネルモジュールは, IPv4 アプリケーションが IPv6 ネットワーク上で通信相手と通信ができる機能をパケットフック, DNS メッセージ変換, IP 変換, 変換テーブルの作成として持たせた. 提案法のカーネルモジュールを用いれば, カーネルモジュールを挿入した Linux 端末において, IPv6 に対応していない既存の IPv4 アプリケーションもエンドツーエンドで IPv6 端末と通信することが可能となる.

実装においては, 提案方式を Linux OS の netfilter 用のカーネルモジュールとして新たに作成することにより, 既存の Linux のカーネルコードの修正を必要としない実装を行った. また, 開発したカーネルモジュールを用いて通信スループットを測定した結果, 既存の IPv4 アプリケーションが Linux デフォルトの IPv6 と遜色なく動作していることを確認した. 本研究では, 通信相手の名前解決のために IPv4 アプリケーションが送信する DNS パケットを IPv6 ネットワーク上に適した形に修正する必要があるため, その処理にかかる時間を Linux IPv6 と比較し測定した結果, 遅延時間の差はほとんどなく変換処理において影響がほとんどないことを確認した. IP パケットのフックやアドレス変換, テーブル登録などすべてカーネル内

で処理することにより, オーバヘッドの少ない処理を実現している.

これらの結果から提案法は, IPv4 アプリケーションのための IPv6 ネットワークにおける名前解決や相互の通信において, デフォルトの Linux IPv6 同士の通信とほぼ同等のレベルの通信ができることを実現し, 通信における遅延時間においても滞りなく通信できる方式であることが確認できた.

5.2　今後の課題と展望

今現在の実装においては, アプリケーション層でペイロードに埋め込まれた IP アドレスを変換でするることができないという課題を持っている. また, カーネルモジュールを挿入したときにプールされる仮想 IPv4 アドレスはプライベートアドレスで構成されているため巨大なアドレス空間はプールのために使われる. しかし, もし通信元端末が多大な数の IPv6 端末と通信することがあれば割り当てるはずの仮想 IPv4 アドレスが枯渇してしまう恐れがある. この点においては, 変換テーブルの対応情報が最も古いエントリや, あまり使われていないエントリを削除することで解決できる課題であると考えられる. 以上のことから, 提案するモジュールを IPv4 アプリケーションに組み込めば, 膨大な数の IPv6 に対応していない IPv4 アプリケーションを IPv6 上で利用可能にすることができる. またそのことから, ソースコードの公開されていない IPv4 アプリケーションの IPv6 対応も実現し, 大規模なアプリケーションの IPv6 対応にも巨額の費用をかけずに実現することができる.

参 考 文 献

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

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

[3] I. Raicu,S. Zeadally. "Impact of IPv6 on End user Applications", to apper at 10th International Conference on Telecommunications, ICT'2003,Tahiti Papeete, French Polynesia,February 23,2003.

[4] K. K., Ettikan, et al. " Application Performance Analysis in Transition Mechanism from IPv4 to IPv6". Research & Business Development Department, Faculty of Information Technology, Multimedia University (MMU), Jalan Multimedia, June 2001.

[5] E. Nordmark, R.Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005

[6] A.Durand, R.Droms, J.Woodyatt, Y.Lee, "Dual-Stack Lite Broad Deployments Following IPv4 Exhaustion", RFC6333, August 2011.

[7] Kenjiro Cho, Matthew Luckie, Bradley Huffaker, "Identifying IPv6 network problems in the dual-stack world", NetT '04 Proceedings of the ACM SIGCOMM workshop on Network troubleshooting,research, theory and operations practice meet malfunctioning reality pp.283-288, 2004

[8] Qinhua Zheng, Ting Liu, Xiaohong Guan, Yu Qu, Na Wang, "A new worm exploiting IPv4-IPv6 dual-stack networks", WORM '07 Proceedings of the 2007 ACM workshop on Recurring malcode pp.9-15, 2007

[9] W. Simpson, "IP in IP Tunneling", RFC 1853, October 1995.

[10] C. Perkins, "IP Encapsulation within IP", RFC2003, October 1996

[11] Tushar M., Raste, D., B. Kulkarni, "Design and implementation scheme for deploying IPv4 over IPv6 tunnel" Department of Computer Science and Engineering, Walchand College of Engineering, Sangli, India 14 January 2005.

[12] K. Egevang, P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, May 1994.

[13] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[14] Affifi,H., Tountain,L., "ENST Bretagne,Methods for IPv4-IPv6 Transition". IEEE 1999.

[15] Yao-Chung Chang, Reen-Cheng Wang, Han-Chieh Chao*,Jiann-Liang Chen"Performance Investigation of IPv4/IPv6 Transition Mechanisms" Taiwan NICI IPv6 Steering Committee,R.

[16] R. Gilligan, E. Nordmark, " Transition Mechanisms for IPv6 Hosts and Routers," Request for Comments 1933, Internet Engineering Task Force, April 1996.

[17] M. Bagnulo, P. Matthews, I. van Beijnum, " NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," IETF draft, Mar. 2009.

[18] G. Tsirtsis, P. Srisuresh, "Network Address Translation-Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[19] F. Baker, X. Li, C.Bao, and K. Yin, "Framework for IPv4/IPv6 Translation", IETF draft Aug. 2010.

[20] Cristian Perez Monte, Maria Indes Robles, Gustavo Mercado, Carlos Taffernaberry, Marcela Orbiscay, Sebastian Tobar, Raul Moralejo, Santiago Perez, "Implementation and Evaluation of Protocols Translating Methods for IPv4 to IPv6 Transition"Journal of Computer Science and Technology Vol.12, No.2, August 2012.

[21] K.tsuchiya, H.Higuchi, Y.Atarashi, "Dual Stack Hosts using the"Bump-In-the-Stack"Technique (BIS)", RFC2767, February 2000.

[22] Kristen Accardi,Tony Bock,Frank Hady,Jon Krueger " Network processor acceleration for a Linux* netfilter firewall", ANCS '05 Proceedings of the 2005 ACM symposium on Architecture for networking and communications systems, pp.115 - 123.

[23] Zhao Da-yuan, Jiang Yi-xin, Lin Chuang, Li Yan-xi, "Implementation and performance evaluation of IPSec VPN based on netfilter",Wuhan University Journal of Natural Sciences January 2005, Volume 10, Issue 1, pp.98 - 102.

[24] P. Mockapetris, "DOMAIN NAMES - CONCEPTS AND FACILITIES", RFC 1034, November 1987.

[25] P. Mockapetris, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", RFC 1035, November 1987.

[26] R. Elz, R. Bush, "Clarifications to the DNS Specification", RFC 2181,July 1997.

[27] R. Braden, "Requirements for Internet Hosts – Application and Support", RFC1123, October 1989.

[28] S. Thomson, C. Huitema, "DNS extensions to support IP version 6", RFC1886, December 1995.

[29] P. Danzig, K. Obraczka, and A. Kumar, "An analysis of wide-area name server traffic: A study of the Internet domain name system", Proc. ACM SIGCOMM, pp.281 - 292 1992.

[30] E. Cohen and H. Kaplan, "Proactive caching of DNS records: Addressing a performance bottleneck", Proc. Symp. Applications and the Internet (SAINT), pp.85 - 94 2001.

[31] M. Crawford, C. Huitema, "DNS Extensions to Support IPv6 Address Aggregation and Renumbering", RFC 2874, July 2000.

[32] Eun-Young Park, Jae-Hwoon Lee, Byoung-Gu Choe, "An IPv4-to-IPv6 dual stack transition mechanism supporting transparent connections between IPv6 hosts and IPv4 hosts in integrated IPv6/IPv4 network", Communications, 2004 IEEE International Conference on Volume:2, 20-24 June 2004, pp.1024 - 1027 Vol.2

[33] Kawarasaki, Y., Shibata, T., Takahashi, T., "IPv4/IPv6 SIP interworking methods in dual-stack network", Communications, 2003. APCC 2003. The 9th Asia-Pacific Conference on Volume:3,21-24 September. 2003, pp.1124 - 1128 Vol.3

[34] Jun Bi, Jianping Wu, Xiaoxiang Leng , "IPv4/IPv6 Transition Technologies and Univer6 Architecture", IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.1, January 2007.

[35] 丁農, 宮崎純生, 島村祐一, "Ipv4 over IPv6 トンネル振分けにおける DNS 適用方式", 電子情報通信学会, pp.11 - 16, July, 2009.

[36] Hyun-Ho Choi, Dong-Ho Cho, "Mobility management based on mobile IP in mixed IPv4/IPv6 networks", Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th Volume:3,6-9 October. 2003, pp.2048 - 2052 Vol.3.

[37] 内藤克浩, 上醉尾一真, 西尾拓也, 水谷智大, 鈴木秀和, 渡邊晃, 森香津夫, 小林英雄, "NTMobile における移動透過性の実現と実装", 情報処理学会論文誌, Vol 54, No.1 pp. 1- 14, January. 2013.

[38] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[39] P. Srisuresh, B. Ford, D. Kegel, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)", RFC 5128, March 2008.

[40] E. Nordmark, "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[41] Jivika Govil, Jivesh Govil, Kaur, N., Kaur, H.,"An examination of IPv4 and IPv6 networks : Constraints and various transition mechanisms", Southeastcon, 2008. IEEE, pp.178 - 185, 3-6 April 2008.

[42] T. Vazao, L. Raposo, J. Santos, "Migration to the New Internet - Supporting Inter Operability Between IPv4 and IPv6 Networks", Telecommunications and Networking - ICT 2004 Lecture Notes in Computer Science Volume 3124, pp. 678-687, 2004.

[43] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[44] Zhong Ren, Chen-Khong Tham, Chun-Choong Foo, Chi-Chung Ko, "Integration of mobile IP and multiprotocol label switching", Communications, 2001. ICC 2001. IEEE International Conference on Volume:7, pp.2123 - 2127 vol.7, June, 2001.

[45] 角野 宏光, 内田 良隆, 石川 憲洋, 峰野 博史, 水野 忠則, "異なるアドレス空間をシームレスに接続する IP 層の拡張の提案と実装", 電子情報通信学会論文誌 B Vol. J93-B, No.10, pp.1397 - 1407, 2010.

[46] Joo-Chul Lee, Myung-Ki Shin, Hyong-Jun Kim, "Implementing NAT-PT/SIIT, ALGs and consideration to the mobility support in NAT-PT environment", Advanced Communication Technology, 2004. The 6th International Conference on Volume:1,9-11 February. 2004, pp.433 - 439.

[47] Baumgart, I., Heep, B., Krause, S., "OverSim: A scalable and flexible overlay framework for simulation and real network applications", Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference,9-11 September. 2009, pp.87 - 88.

[48] Ishizu, K., Murakami, H., Miyamoto, G., Tran, H.N, "Design and Implementation of Cognitive Wireless Network based on IEEE P1900.4", Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2008. SECON Workshops '08. 5th IEEE Annual Communications Society Conference,16-20 June 2008, pp.1 - 8.

[49] Blywis, B., Gunes, M., Juraschek, F., Schmidt, P., "DES-SERT: A framework for

structured routing protocol implementation", Wireless Days (WD), 2009 2nd IFIP,15-17 December. 2009, pp.1 - 6.

[50] http://www.netfilter.org, retrieved: January.,2014.

[51] http://iperf.sourceforge.net, retrieved:January., 2014.

謝 辞

本研究を遂行するにあたり, 多忙な時間を割いてご指導ならびに御助言を下さった小林英雄教授 , 森香津夫教授 , 内藤克浩助教に深く感謝いたします. また , 研究室の設備などでご協力くださった山本好弘技術職員 , ならびに通信工学講座学生諸氏に深く感謝いたします.

研究業績

- Yuta Inoue, Katsuhiro Naito, Kazuo Mori and Hideo Kobayashi, “ Development of the kernel module for Bump in the Stack, ” The 2nd International Symposium for Sustainability by Engineering at MIU (IS2EM 2012), CO-９, November 2012
- 井上雄太, 内藤克浩, 森香津夫, 小林英雄, “ IPv6 ネットワークにおいて IPv4 アプリケーションを利用可能にするカーネルモジュールの開発, ” 電気関係学会東海支部連合大会, F4-1, 2013 年 9 月