# IPv6 エニーキャストのルーティング実験

2005MT005 馬場 隆章　　2005MT063 松川 侑資
指導教員　後藤 邦夫

## 1　はじめに

IPv6 に移行するにあたり, 新しいアドレスであるエニーキャストアドレスが実装された.

しかし, エニーキャストアドレスをネットワーク上で利用する際には実際の利用例も少なくまだ問題点が多い. 実際にエニーキャストアドレスが実装されているカーネルは FreeBSD だけしかなく, まだ Linux カーネルなどでは実装中の段階である. そこで, 本研究では同一セグメント内でのエニーキャストアドレスの実装とともにさまざまなネットワークを構成し, エニーキャストを使用した場合どのような通信が行われ, またどのような問題点があるのかを実験する.

なお馬場は主にプログラム作成を担当し, 松川は実験の環境構成を担当した.

## 2　エニーキャストアドレス

この節ではエニーキャストアドレス [4] について説明する. エニーキャストアドレスはインターネット上の複数のノードに同一の IP アドレスを割り当てたものである.

ユニキャストアドレスは 1 対 1, マルチキャストアドレスは 1 対多, エニーキャストアドレスは 1 対多のうち 1, という通信になる. この多数のうち 1 というのは, 送信元となるノードから複数ある宛先であるエニーキャストアドレスの中から一番近いものということである. ここでの近さの定義は普通はルーティングプロトコルにより認識されたメトリックに依存するが, 本研究での行う同一セグメント内での実験では, 一番はじめに応答したノードが有効となる.

### 2.1　アドレス空間

エニーキャストアドレスはグローバルユニキャストアドレスと同じアドレス空間であるだけでなく, ユニキャストアドレスと同じ構造をとるため, ユニキャストアドレスを複数のノードに割り当てるとそのアドレスがエニーキャストアドレスとなる.

このためユニキャストアドレスとエニーキャストアドレスは明確な区別のしようがなく, エニーキャストアドレスとして用いられるさいそのアドレスを割り当てられたノードはそれがエニーキャストアドレスであるという明示的な情報をあたえた上で設定しなければならない.

### 2.2　エニーキャスト処理の実装の違い

現在エニーキャストアドレスが実装されているのは FreeBSD だけである.

FreeBSD の場合次のように ifconfig コマンドによってインタフェースにどのアドレスがエニーキャストアドレスであるかを設定することが出来る.

```
# ifconfig インタフェース inet6 v6 アドレス/マスク長 alias anycast
```

なお現在エニーキャストアドレスは宛先アドレスとしてのみ使用され, FreeBSD を使用したとしてもエニーキャストアドレスはルータにのみ使用される.

しかし, Linux カーネルはまだエニーキャストアドレスをサポートしておらず, また ip コマンドにエニーキャストの記述があるものの, まだ設定できないのが現状である. Linux についてもエニーキャストアドレスが使えるようにしたいが, そのためには OS の改造が必要であり, OS 改造は困難である. そのため, 本研究では OS に手を加えずにエニーキャスト処理をする手法を提案する.

具体的には, エニーキャストアドレス宛のパケットが OS で処理される前に OS 外でそのパケットを横取りし, 別のプログラムで処理する. Linux での実装方法は 3 節で述べ, 実際にエニーキャストアドレス通信を行った場合どのような問題があるかを調べたルーティング実験の結果を 4 節で述べる.

## 3　Linux におけるエニーキャストホスト実装手法の提案

本研究では QUEUE の拡張である NFQUEUE[2] と, IPv6 における ICMPv6 のアドレス解決で使用される近隣要請メッセージと近隣広告メッセージを利用して Linux にエニーキャストを独自拡張して実現する. 詳しくは 3.3 で述べる. なお, NFQUEUE によるパケットの横取りには本研究室で開発中のネットワークエミュレータ GINE(Goto's IP Network Emulator)[5] を利用する.

### 3.1　NFQUEUE

NFQUEUE は, パケットをユーザ空間のプログラムやアプリケーションへキューイングするために用いられる. 例えば, ネットワークアカウンティングや, パケットのプロキシやフィルタリングなどを行うアプリケーションへの使用が考えられ, NFQUEUE ターゲットを使うとパケットを別々のキューに入れることができる. なお, ユーザ空間は最大 65536 個まで作ることができる.

### 3.2　ICMPv6

IPv4 から IPv6 に移行するにあたり, ICMPv6 の重要性が増し, IPv6 の根幹にかかわる以下のような重要な役割を担うようになってきた.

1. 近隣探索とアドレス自動生成
2. パス MTU 探索
3. マルチキャストリスナー探索

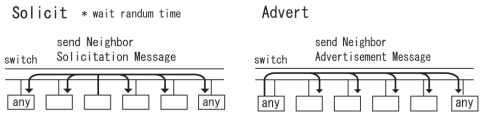


図 1　同一セグメント

### 3.3　システムの概要

本システムではエニーキャストアドレスからの近隣広告メッセージを図 1 のようにマルチキャストに変換することにより, 同一セグメント内で Linux によるエニーキャストアドレスの実装を実現する. ノード間の通信にははじめにアドレス解決のための近隣要請メッセージと近隣広告メッセージのやりとりがなされるが, 本来近隣広告メッセージはユニキャストで送信されるためマルチキャストに変化する処理が必要になる. なお全体の処理のシーケンス図を図 2 に, エニーキャストノード内のパケットの横取り処理のシーケンス図を図 3 に示す.

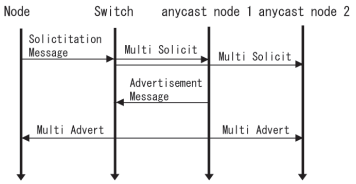


図 2　同一セグメントのシーケンス図

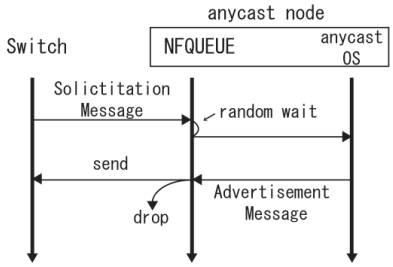


図 3　エニーキャストノード内のシーケンス図

他のエニーキャストアドレスからの近隣広告メッセージのマルチキャストを受け取ったノードはその情報を記録しておき, その情報が有効な間は近隣広告メッセージを自粛させ (drop), 有効な情報がなければ自分のマルチキャストを送信する (send). そうすることにより同一セグメント内で有効な一意のエニーキャストアドレスを設定することができるようになる. なお, どのノードが有効になるかはマルチキャストをランダム時間待たせてから送信することにより実現した.

#### 3.3.1　横取りしたパケットの処理内容

次に横取りしたパケットの処理について述べる. なお, システム全体の流れを図 4 に示した. まず ip6tables で ICMPv6 のパケットは全て NFQUEUE を用い横取りする. 横取りしたパケットは全て今回作成したプログラムに送られるが, ここではわかりやすいように NIC から入ってきたパケットの横取り処理を処理 A, OS から出たパケットの横取り処理を処理 B とし, 処理 A, 処理 B ともに行う処理を共通処理と位置づけて各処理内容に付いて説明していく.

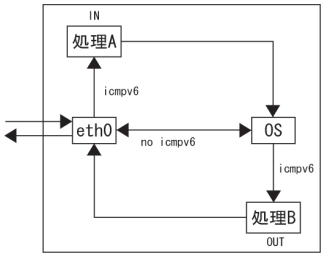


図 4　システム全体の流れ

#### 3.3.2　共通処理

処理 A, B ともに横取りしたパケットの送信元のリンクローカルアドレスを割出し, 図 5 に共通処理の流れを示す. これは, 自分以外のエニーキャストアドレスからのマルチキャストを記録しておくときのために必要だからである. 次に横取りしたパケットが近隣要請メッセージであるか, 近隣広告メッセージであるか調べ, もし違っていたらスルーして, そのまま OS にパケットを戻す. 以上が NFQUEUE でパケットを横取りした場合の共通の処理である.

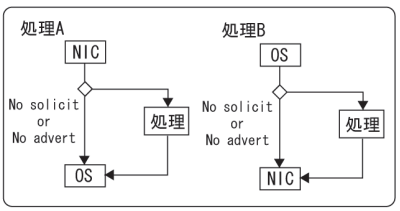


図 5　共通処理

#### 3.3.3　処理 A

処理 A では, NIC から入ってきたパケットを処理する.

横取りしたパケットが以下の条件を満たす場合, パケットをランダム時間待たせた後 OS に戻す処理をする.

- 近隣要請メッセージ
- ターゲットアドレスがエニーキャストアドレス

横取りしたパケットが以下の条件を満たす場合, そのパケットの情報を保持する.

- エニーキャストアドレスからマルチキャストされた近隣広告メッセージ

それ以外のパケットなら, なんの処理もせずパケットを戻す. 図 6 に処理 A の流れを示す.

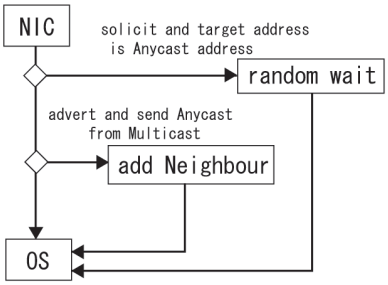


図 6　処理 A

#### 3.3.4　処理 B

処理 B では, OS から出ていくパケットの処理をする.

横取りしたパケットが近隣広告メッセージだった場合, 以下の処理をする.

- エニーキャストアドレスから出たパケットなら他のエニーキャストアドレスからマルチキャストがきているかきていないかを調べる
- 他のノードからの情報が何もなければ自分の近隣広告メッセージをマルチキャスト宛に書き換え, チェックサムも再計算しパケットを戻す
- 他のエニーキャストノードが既にマルチキャストを送っていた場合は自分の送信しようとしていた近隣広告メッセージを破棄する

それ以外のパケットなら, なんの処理もせずパケットを戻す. 図 7 に処理 B の流れを示す.

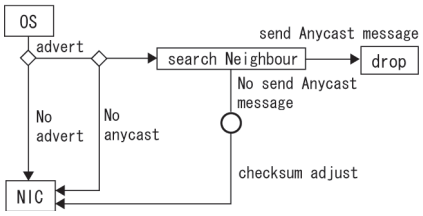


図 7　処理 B

### 3.4　実装実験の結果

次にこれまで述べてきた手法を用いて実際に同一セグメント内でエニーキャスト通信を行った結果を示す. 同一セグメント内にエニーキャストノード 2 つと, ユニキャストノード 1 つのネットワークで実験を行った. 表 1 にエニーキャストノードの IPv6 アドレスと MAC アドレスを示し, 次にユニキャストノードで/sbin/ip コマンドを実行して OS に届いた近隣広告メッセージがどの MAC アドレスから届いたのかを調べることにより, エニーキャストノードが切り替わっていることを確認する.

表 1　同一セグメント内のアドレス設定

| node | IPv6 address | MAC address |
|---|---|---|
| anycast node 1 | 3fff::1 | 00:10:18:31:6e:25 |
| anycast node 2 | 3fff::1 | 00:0e:7b:ef:65:66 |

/sbin/ip コマンドの実行 1

```
[kg2007@localhost ~]$ /sbin/ip -f inet6
neighbor show
3fff::1 dev eth0 lladdr 00:10:18:31:6e:25
REACHABLE
```

/sbin/ip コマンドの実行 2

```
[kg2007@localhost ~]$ /sbin/ip -f inet6
neighbor show
3fff::1 dev eth0 lladdr 00:0e:7b:ef:65:66
REACHABLE
```

以上の結果より, エニーキャストアドレス 3fff::1 の MAC アドレスが変わっていることがわかるので, 同一セグメント内でのエニーキャストノードの実装ができていることがわかる.

### 3.5　問題点

このシステムは NFQUEUE によるパケットの横取りを前提として構築しているため, nfnetlink_queue をサポートしているカーネルでなければ実装できない. しかし, パケットの横取りさえできればどの OS でもこの手法を取り入れることができるので NFQUEUE のようなパケットの横取りができる仕組をもつ OS ならば, Linux 以外の OS でもこの手法は実現可能であり, 同一セグメント内でのエニーキャストアドレスを実装することができる.

## 4　ネットワーク構成と各ネットワークの評価

今節ではルーティングプロトコルデーモンである Quagga[1] を用いて PC をルータにみたてた複数のネットワークを構成し, ping6 で通信を確認した. ルーティングプロトコルにはホップ数をメトリックとして使用する IPv6 対応の RIPng を使用し, ノードは全て実機である.

### 4.1　Quagga

Quagga はよりオープンな開発体系を目指して前身の GNU Zebra のプロジェクトから派生し, 現在も活発的に開発が行われている. Fedora では GNU Zebra に代わって Quagga が収録されており, 全ての実験用 PC に Fedora 8 を使用した.

本研究で Quagga を使用した理由は,

1. Fedora にインストールするのが簡単であったこと.
2. IPv6 用のプロトコルである RIPng をはじめ, 豊富なルーティングプロトコルをサポートしていること.

などである.

### 4.2 エニーキャスト動作実験と同ホップ数実験

まず, 実際にエニーキャストアドレスの特徴である, 近いノードと通信が出来るのかどうかを実験した.

実験方法として, ユニキャストノード (以下 host A) からみてそれぞれホップ数の違う二つのノードにエニーキャストアドレスを割り当て, host A からエニーキャストノードと通信を行う実験を試みた.

実験の結果, host A から見てエニーキャストノードへのホップ数の近いノードとの通信が確認できた. これは, ルーティングプロトコルに RIPng を使用しているため, ホップ数の小さいノードと通信したのだと考えられる. もし仮に, ルーティングプロトコルを OSPFv3 にした場合, 通信経路の帯域幅を考慮するので, また違った結果が得られると考えられる.

次に一番近い距離に複数のノードがあった場合どちらのノードを優先して通信をするのかを調べる同ホップ数実験を行った.

実験方法は, エニーキャスト動作実験のエニーキャストノードを host A から見てホップ数が等しくなるようにネットワークを構成し, エニーキャストノード宛に通信実験を行った. そのさい先にエニーキャストアドレスを割り当てたノードを host B とし, 後からエニーキャストアドレスを割り当てたノードを host C とした.

実験の結果, 先にエニーキャストアドレスを割り当てた host B が有効となり, 後から割り当てた host C とは通信を行わなかった. これは有効とすべき最短ホップ数のノードが複数あった場合, 先に検知したノードを優先するという Quagga の RIPng のルーティングテーブルの作成方法の仕様であると考えられる.

### 4.3 パケットハイジャック

この実験は, エニーキャストアドレスの特性である, 必ず近いノードと通信をする, という特性についての問題点を実際に検証するための実験である.

今まで通信していたノードよりも, より近いノードが現れた場合, そちらのノードを優先して通信してしまうという問題が起きるか調べる.

図 8 のネットワークでは 3 台の PC(router) で Quagga を起動し, エニーキャストアドレスを 3fff::1/64 として, host B, host C の順に割り当て host A からエニーキャスト宛てに通信を試みた. その結果, host B との通信が確認できた. そして host B との通信の最中に router3 にもエニーキャストアドレスを割り当てた.

その結果, host A から host B への通信が router3 へ切り替わっていることがわかった. これはルーティングテーブルが更新されて一番近いノードが替わったことが原因だと考えられる.

このため, エニーキャストアドレスのノードを増やす場合は通信経路が切り替わることを考慮する必要があることがわかる. さらには悪意のあるユーザがノードにエニーキャストアドレスを割り振り, パケットハイジャック (横取り) をしてしまう危険性もあることが実際に確認できた [3]. エニーキャストアドレスを使用するさいにはエニーキャストの認証機能のようなよりセキュアな通信プロトコルの存在が必要であることがわかった.

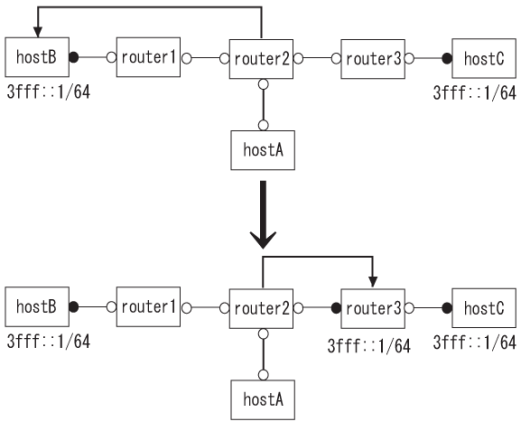


図 8 パケットハイジャック

## 5 おわりに

今回の研究ではエニーキャストアドレスの基本的な動作の確認, 検証及び同一セグメント内での Linux カーネルへのエニーキャストアドレスの実装に成功した. しかし, まだ比較的小さなネットワークでしか実験ができていないのでもう少しノード数を増やし, ネットワークを拡大して実ネットワークを想定した実験を行う必要がある. さらに ping6 でしか通信実験が済んでおらずプロトコルごとに通信形式に差があるため, プロトコルごとにネットワークにどのような影響をあたえるかを実験していく必要がある. また, 本研究ではルーティングプロトコルの RIPng でしか実験ができておらず, OSPFv3 などのルーティングプロトコルごとの実験もしていく必要がある.

## 参考文献

[1] Quagga Software Routing Suite (accessed June. 2008). http://www.quagga.net/.

[2] libnetfilter queue project (accessed October. 2008). http://www.netfilter.org/.

[3] Abley, J. and Lindqvist, K. E.: Operation of Anycast Services, *RFC4786* (December 2006).

[4] Hinden, R. M. and Deering, S. E.: IP Version 6 Addressing Architecture, *RFC1884* (December 1995).

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