社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

IPv6 ネットワークにおけるエニーキャスト通信実現のための プロトコル設計と実装

土居　聡† 阿多　信吾†† 北村　浩††† 村田　正幸†††† 宮原　秀夫†

† 大阪大学 大学院情報科学研究科
〒 560–8531 大阪府豊中市待兼山町 1–3
†† 大阪市立大学 大学院工学研究科
〒 558–8585 大阪府住吉区杉本 3–3–138
††† NEC ネットワークス開発研究所　〒 108–8557 東京都港区芝浦 2–11–5
†††† 大阪大学 サイバーメディアセンター　〒 560–0043 大阪府豊中市待兼山町 1–30
E-mail: †{s-doi,miyahara}@ist.osaka-u.ac.jp, ††ata@info.eng.osaka-cu.ac.jp, †††kitamura@da.jp.nec.com, ††††murata@cmc.osaka-u.ac.jp

あらまし　IPv6 の持つ新しい機能の 1 つとして，エニーキャスト通信機能がある．エニーキャスト通信を実現するために規定されているエニーキャストアドレスは，特定の機能（サービス）に対して割り当てられるアドレスであり，クライアント側はエニーキャストアドレスを指定しさえすれば，対応する機能を提供する複数のサーバの中から 1 台の適切なサーバが自動的に選ばれ，そのサーバに対して通信することが可能となる．しかし，エニーキャスト通信については基本的仕様が規定されているのみで，その実装，実現方法について解決すべき問題が多く存在する．そこで本稿では，エニーキャストアドレスを利用する場合の制限事項を明らかにし，その解決のための手法を設計，実装し，既存のアプリケーションがプログラムを変更することなく，エニーキャストアドレスを利用できることを検証した．
キーワード　IPv6, エニーキャスト，ICMPv6，アドレス解決

A Protocol Design and Implementation for the Anycast Communication in the IPv6 Network

Satoshi DOI†, Shingo ATA††, Hiroshi KITAMURA†††, Masayuki MURATA††††, and Hideo MIYAHARA†

† Graduate School of Information Science and Technology, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka 560–8531, Japan
†† Graduate School of Engineering, Osaka City University
3–3–138 Sugimoto, Sumiyoshi–ku, Osaka 558–8585, Japan
††† Development Laboratories, NEC Corporation　2–11–5 Shibaura, Minato–ku, Tokyo 108–8557, Japan
†††† Cybermedia Center, Osaka University　1–30 Machikaneyama, Toyonaka, Osaka 560–0043, Japan
E-mail: †{s-doi,miyahara}@ist.osaka-u.ac.jp, ††ata@info.eng.osaka-cu.ac.jp, †††kitamura@da.jp.nec.com, ††††murata@cmc.osaka-u.ac.jp

Abstract　Anycast is defined as one of new IPv6 features, which supports service-oriented address assignments in IPv6 networks. Anycast address is not determined by the location of node, but by the type of service presented at the node. In anycast communication, the client can automatically obtain the appropriate node corresponding to a specific service without knowledge of the location of the server. However, there are some technical problems to be solved in the current anycast definitions. In this paper, we first clarify such restrictions in anycast communications, and then design a new architecture for anycast communications so that every application can support anycast communications without any modifications of their program. We have implemented our designed architecture on the UNIX-based operating system, and verified that our proposed method smoothly supports anycasting without change of existing applications.
Key words　IPv6, anycast, ICMPv6, address resolution

1. はじめに

IPv6 には，IPv4 にはなかった新しい機能がいくつか加えられている．この新機能の一つとして，エニーキャストアドレス通信機能がある [1]．エニーキャストアドレスとは，特定の機能に対して割り当てることのできるアドレスであり，クライアントはこのエニーキャスト通信機能を利用することにより，実際にどのノードからサービスを提供されるかを知る必要なく最適なサーバからサービスを受けることが可能となる [2]．例えば，プロキシや DNS などの各サービスごとに既知のエニーキャストアドレスを決めておけば，あらかじめオペレーティングシステム上で設定しておくことができるため，従来のネットワーク管理者によるホストごとの設定は必要なくなる．

このように，エニーキャストアドレスは機能的に興味深い側面を持った技術ではある．しかしながら現在，エニーキャスト通信についてはその基本的仕様のみが定義されている段階である．このため，実際の運用を考えた場合，エニーキャストアドレスとユニキャストアドレスの区別ができない，エニーキャストアドレスはパケットの送信元アドレスとして利用できない，通信中の相手が変化する可能性がある，といったエニーキャストアドレスを用いた通信特有の問題がある．これらの問題を解決しなければ，エニーキャスト通信を行うことはできない．

本稿では，エニーキャスト通信実現のための技術課題を明らかにし，これらを解決するための新たなプロトコルを設計，実装し，その効果を検証する．そして，既存のアプリケーションを改変することなくエニーキャスト通信を実現するために、共有ライブラリの動的な置換による手法を取り入れた，エニーキャスト/ユニキャストアドレス解決モデルを提案する．さらに設計した方式を実装し，実機を用いた評価を行うことによって，従来は利用できなかった，`telnet` 等のコネクション型アプリケーションの宛先にエニーキャストアドレスを利用できることを示す．

以下，まず 2 章では，エニーキャストアドレスの利点および応用例について説明し，さらにエニーキャストアドレスを用いた通信を実現するために必要な技術課題について述べる．次に 3 章では，エニーキャストアドレスを用いた通信を行うために必要となる，アドレス解決モデルを新たに提案する．さらに 4 章では提案したアドレス解決手法を FreeBSD 上に実装し，実機を用いた評価結果を示す．最後に 5 章で，まとめと今後の課題を述べる．

2. エニーキャストアドレスを用いた通信

本章では，エニーキャストアドレスの性質を示し，期待されるサービスについて述べる．さらに，エニーキャスト通信実現のために解決すべき技術課題について明らかにする．

2.1 IPv6 におけるアドレスタイプとエニーキャストアドレス

IPv6 ではアドレスに以下の 3 つのタイプが定義されている．

（1） ユニキャストアドレス
（2） マルチキャストアドレス
（3） エニーキャストアドレス

表 1 に各アドレスタイプにおける通信形態をまとめる．ここで

表 1　各アドレスタイプの通信形態

アドレス	通信形態	対象メンバ	アドレス設定の対象
ユニキャスト	1 対 1	1	単一インターフェース
マルチキャスト	1 対 多	多	グループ
エニーキャスト	1 対 1	多	機能（サービス）

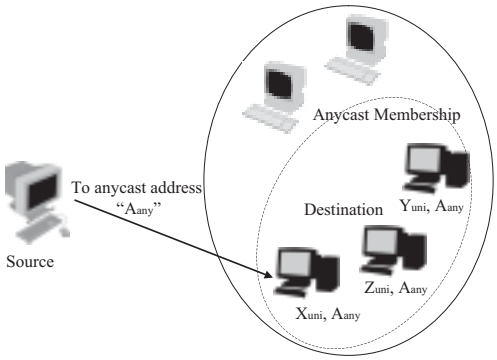

図 1　エニーキャストアドレス

取り上げるエニーキャストアドレスとは、同一の機能を提供する複数のノードのインターフェース（エニーキャストメンバーシップ）に割り当てられるアドレスであり，エニーキャストアドレスを指定したパケットは，同一の機能を提供するメンバーシップのうち,「最適な」ノードのインターフェースへと届けられる．どのインターフェースが「最適」であるかは，経路制御メカニズムによって決定される（図 1）．

2.2 エニーキャストアドレスの応用例

複数のインターフェースが同一のエニーキャストアドレスを持つ場合，どのインターフェースにパケットを届けるかについて適当なポリシーを与れば，最適なインターフェース（たとえば，送信側から最も近いノードなど）にパケットを送ることができる．また，同一のサービスを提供するノードをグループ化し，そのグループに属するノードのインターフェースにエニーキャストアドレスを割り当てることによって，サービスにアドレスを対応させることができる．以下に，考えられるエニーキャストアドレスの応用例とその利点について説明する．

（1） 最適サーバの自動選択

経路制御を適切に行うことで，複数サーバのうち最適なサーバと通信することができる.「最適」さの評価基準としては，経由するルータの数（ホップ数），サーバの負荷などが考えられる．

（2） アドレスのサービスへのマッピング

DNS サーバやプロキシサーバなど，一般的に広く利用されているサービスを提供するサーバは，そのサービスごとに特定のエニーキャストアドレスをあらかじめ定めることで，利用者はあらかじめ決められたエニーキャストアドレスを用いるだけでサービスを受けることができる．さらに，これらのエニーキャストアドレスをアプリケーションの標準値として規定することにより，利用者は一切の設定を行うことなくサービスを利用することができる．例えば，DNS サーバに対してある決まったエニーキャストアドレスを割り当てておき，クライアント側でそのアドレスをオペレーティングシステム上で設定しておけば，プラグアンドプレイで最適な DNS サーバを利用することができる．

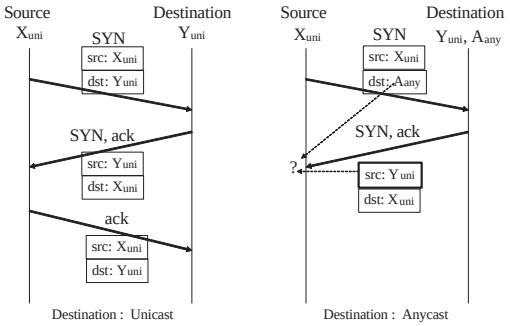

図 2　TCP 3 ウェイハンドシェーク

（3） 通信の信頼性の向上とサーバの負荷分散

経路制御機構によって，エニーキャストアドレスを宛先に持つパケットは，対応するインターフェースのうち最適なノードのインターフェースに届けられる．経路制御を負荷に応じて行えば，サーバの負荷分散も実現可能となる．

2.3　エニーキャスト通信における制限事項

エニーキャストアドレスを用いた通信を行う際，エニーキャストアドレスの特性から，注意すべき点がいくつか考えられる．エニーキャスト通信の特性として，以下が考えられる．

（1） 仕様上の特性

- エニーキャストアドレスはパケットの送信元アドレスとして利用できない

この仕様により，セッションを持つようなプロトコルではエニーキャストアドレスは利用できない．例えば，TCP [3] アプリケーションにおいてエニーキャスト通信を行う場合，3 ウェイハンドシェーク時に，SYN セグメントの宛先アドレスに指定されたエニーキャストアドレスと，その SYN – ACK の送信元アドレスが異なるため，TCP コネクションが確立できない（図 2).

- エニーキャストアドレスとユニキャストアドレスの区別ができない

IPv6 では，エニーキャストアドレスはユニキャストアドレスのアドレス空間から選ばれるという仕様になっており，エニーキャストアドレスに対して特にアドレス空間を設定していない．このため，送信ノードが宛先アドレスを調べただけでは，そのアドレスがエニーキャストであるかどうかはわからない．

（2） 通信中の相手が変化する可能性

「最適さ」をはかる基準として考えられる，ホップ数，通信相手の負荷などは動的に変化する．そのため，持続的セッションを持つエニーキャスト通信の場合，通信中の相手が変化する可能性がある．

以上のように，エニーキャストアドレスを用いた通信には，解決すべき技術課題が多く存在する．

3.　エニーキャストアドレスによる通信のためのアドレス解決モデルの設計

本章では，2 章で明らかにしたエニーキャストアドレスを用いた通信時に問題となる点の解決策について述べる．

3.1　提案方式への要求事項

今後，エニーキャストアドレスを用いた通信を普及させるためには，既存の枠組みを変更することなく容易に移行できることが重要となる．そのために本稿で提案する通信モデルでは，

- 前章で述べた，エニーキャスト通信上の問題点を解決する
- 既存の技術をできる限り利用する
- 既存の通信プログラムの修正を行わずにエニーキャスト通信をサポートする

ことを目標とする．

3.2　AARP によるアドレス解決

エニーキャストアドレスを用いた通信では，通信相手にはエニーキャストアドレスと実際に通信を行うノードが必ずしも一対一に対応しない．そのため，持続的なセッションを持つような通信，例えば TCP アプリケーションではエニーキャストアドレスを利用できないという問題がある．このことから，提案方式では以下の手順により，内部で宛先エニーキャストアドレスを対応するユニキャストアドレスへと変換することにより，エニーキャスト通信をユニキャスト通信に置き換えるという手法を用いて，前章の問題点を解決する．

（1） 通信開始時に，エニーキャストアドレスを対応するユニキャストアドレスに変換する（以下，アドレス解決と呼ぶ）．

（2） 変換されたユニキャストアドレスを宛先としてパケットを送出する．

本稿では，アドレス解決の手順として新しくエニーキャストアドレス解決プロトコル（Anycast Address Resolving Protocol; 以下 AARP）を提案する．AARP は，DNS で FQDN（Fully Qualified Domain Name）を対応する IP アドレスへと変換する「名前解決」と同様，エニーキャスト通信の開始時もしくは開始前にエニーキャストアドレスをユニキャストアドレスへと変換する「アドレス解決」を行う新しいプロトコルである．

3.3　AARP の実現方法

アドレスがエニーキャストアドレスであるかどうかは，エニーキャストアドレスが割り当てられたノードしかわからないため，エニーキャストアドレスを対応するユニキャストアドレスに解決するためには，実際に対応するノードからのパケットを受信する必要がある．したがって，アドレス解決の方法として以下の 2 つが考えられる．

（1） client initiate ― プローブパケット方式

エニーキャストアドレス宛のプローブパケットを送出し，ノードからの応答を受信することでユニキャストアドレスを取得する．

（2） server initiate ― ピギーバック方式

ノードからの通信パケット上に，エニーキャストアドレスを付加させ（ピギーバック），エニーキャストアドレスとユニキャストアドレスを対応させる．

2 つの方式を表 2 にまとめる．(1) の方式では，プローブパケットによるトラヒックが発生し，ネットワーク資源を消費するという問題がある．一方（2）ではエニーキャストアドレスを付加させるために IPv6 終点オプションヘッダを利用する [4] ことなどが必要となり，アプリケーションの修正が要求される．このため，本稿では（1）のプローブパケットを用いたアドレス解決を行う（図 3）．なお，プローブパケットによるトラヒックは，エニーキャストアドレス解決のキャッシングにより軽減することが可能である．

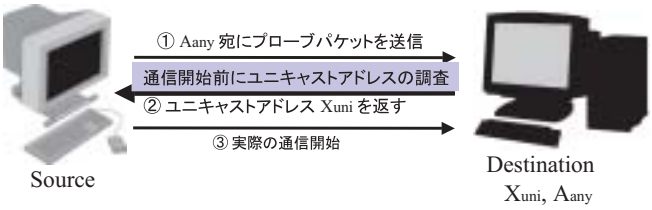

図 3　client initiate ― プローブパケット方式

表 2　2 つの方式の比較

	プローブパケット方式	ピギーバック方式
問題点	最初の一往復のプローブパケット分だけ多くのパケットを必要とする	アプリケーション層，トランスポート層のすべてのプロトコルを修正する必要がある
利点	実装が容易	トラヒック量はユニキャストと同じ

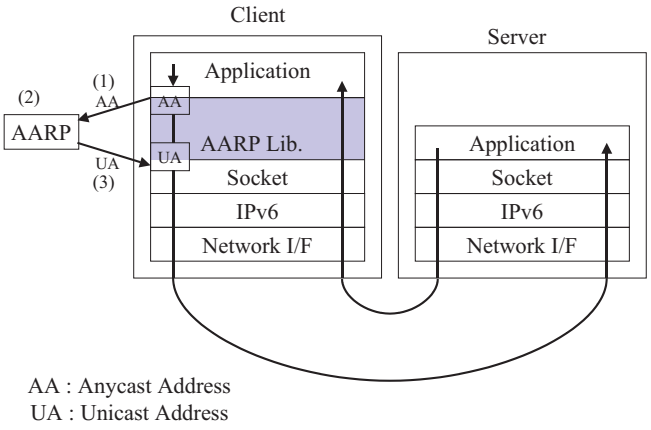

図 4　AARP ライブラリを組み込んだプロトコルスタック

3.4　AARP 中間ライブラリ

本稿では，アプリケーションプログラムを修正せずにエニーキャストアドレスを用いた通信をサポートするため，SOCKS-based IPv6 / IPv4 Gateway [5] 機構の実装方法として用いられている，共有ライブラリを置き換えるための中間ライブラリを作成し，アドレス解決を行う方法を提案する．以下では，この中間ライブラリを AARP ライブラリと呼ぶ．

図 4 に AARP ライブラリを使用した場合の，送受信ノードのプロトコルスタックを示す．AARP ライブラリは，アプリケーションとソケットの間に挿入され，アプリケーションで指定されたエニーキャストアドレスをユニキャストアドレスに変換し，ソケットへ渡す．図 4 における通信手順は以下の通りである．

（1） アプリケーション内で宛先アドレスを AA とした通信が開始されるとき，通信時にコールされる API（TCP を用いたアプリケーションであれば `connect()`）によって，通常の共有ライブラリ上の関数ではなく，AARP ライブラリ上の関数が実行される．

（2） AARP ライブラリは，置き換えられた関数内でアドレス解決を行う．

（3） AARP ライブラリは，得られたユニキャストアドレス UA を宛先として，本来の共有ライブラリの API をコールし，通常のユニキャストアドレスに対する通信と同様の処理を行う．

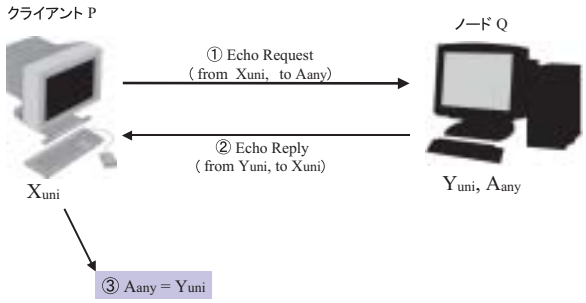

図 5　ICMPv6 Echo Request/Reply によるアドレス解決

4.　AARP の実装と評価

本章では，3 章で述べた AARP ライブラリによるアドレス解決モデルを実機に実装し，評価実験を行う．実験結果より，既存のアプリケーションを改変することなくエニーキャスト通信が実現できることを示す．本稿では，インターネットの代表的なアプリケーションとして TCP を利用した `telnet`，`ftp` および UDP を利用した DNS による評価実験を行い，TCP，UDP の区別なくエニーキャスト通信が実現できることを示す．

4.1　AARP の実装

本稿では，アドレス解決を行うプローブパケットとして，ICMPv6 [6] Echo Request および Echo Reply を用いる．実装で用いたオペレーティングシステム（FreeBSD 4.4 Release）では，受信した ICMPv6 Echo Request パケットの宛先アドレスがエニーキャストアドレスである場合に，対応するユニキャストアドレスを送信元アドレスとして ICMPv6 Echo Reply パケットが返送されるため，プローブパケットとしての機能を有しているからである．

ICMPv6 Echo は，`ping6` プログラムなどで利用されている ICMPv6 の一種で，IPv6 が稼動しているノードは ICMPv6 Echo Request に対して必ず ICMPv6 Echo Reply を返さなければならないという通信規約がある [6]．

図 5 のように，クライアント P はエニーキャストノード Q との ICMPv6 Echo Request/Reply のやり取りによって，ノード Q のユニキャストアドレスを知ることができる．

（1） クライアント P からエニーキャストアドレス A_{any} を宛先として Echo Request を送信する

（2） エニーキャストノード Q はパケットの送信元アドレスにエニーキャストアドレスを設定することができないため，ユニキャストアドレス Y_{uni} を送信元アドレスとした Echo Reply をノード P 宛に送信する．

（3） クライアント P は，受け取った Echo Reply の送信元アドレスから，エニーキャストアドレス A_{any} に対応するユニキャストアドレスは Y_{uni} であることがわかる．

以上の手順は，API（Application Program Interface）を公開することにより実現可能であるが，ICMPv6 によるアドレス解決手法を利用するには，ICMP パケットを受信するために管理者権限が必要となる．そのため，アドレス解決部を API で提供する場合，アプリケーションが管理者権限によって動作する必要があり，既存の設定を変更する必要が生じる．そこで本稿では，プローブパケットを使ったアドレス解決部分をデーモン（daemon）

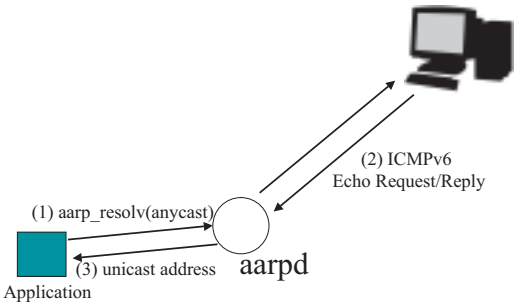

図 6　aarpd

プログラムにより提供し，利用側にはデーモンプログラムとの対話部分の API を公開する方法を提案する．デーモンプログラムは，すでに解決済みのエニーキャストアドレスについてはキャッシングを用いて応答させることによって，プローブで消費されるトラヒックを軽減する．図 6 に具体的な手順を示す．提案方式では，実際のアドレス解決を行うデーモン `aarpd` と，デーモンにアドレス解決を依頼する `aarp_resolv()` API を提供する．`aarp_resolv()` はエニーキャストアドレスを引数とし，ユニキャストアドレスを返す API である．図 6 に示すように，

（1） アプリケーションは `aarp_resolv()` API をコールし，`aarpd` にエニーキャストアドレス解決を依頼する．

（2） `aarpd` は，ICMPv6 Echo メッセージを用いてアドレスを解決する．

（3） `aarpd` は，アプリケーションにユニキャストアドレスを返す．

本稿で提供する中間ライブラリを用いれば，中間ライブラリが `aarp_resolv()` によってアドレス解決を行うため，アプリケーション側でアドレス解決を行う必要はない．

提案方式では，トランスポート層のプロトコルによって通信開始時に呼び出される関数が異なるため，それぞれの場合に分けて，どの関数を置き換えるべきかを考える必要がある．これは，実装を行うシステム上で稼動しているオペレーティングシステムなどによって異なる．本稿では，FreeBSD 4.4 Release 上に実装を行った．まず，TCP 通信の場合，通信開始時の 3 ウェイハンドシェークを起動する関数である，`connect()` を置き換えればよい．次に，UDP 通信であるが，データを送受信する関数である `send()`，`sendto()`，`sendmsg()`，`recv()`，`recvfrom()`，`recvmsg()` が置き換えの候補となる．このうち，本稿で実装を行ったオペレーティングシステムである FreeBSD 4.4 Release では，`send()` は `sendto()` を，`recv()` は `recvfrom()` をそれぞれ内部で呼んでいるため，`sendto()`，`recvfrom()` を置き換えれば `send()`，`recv()` も置き換わる．したがって，`sendto()`，`sendmsg()`，`recvfrom()`，`recvmsg()` を置き換える必要がある．

4.2　TCP 通信の評価

まず，TCP を利用したアプリケーションである `ftp` について，図 7 のネットワーク上で `3ffe:fffe:2:1:ffff::1`（エニーキャストアドレス）に対して，

```
> ftp 3ffe:fffe:2:1:ffff::1
> telnet 3ffe:fffe:2:1:ffff::1
```

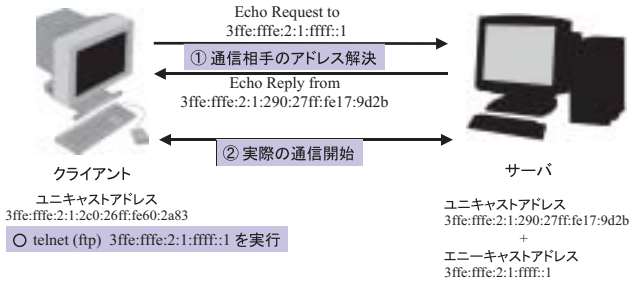

図 7　ネットワーク構成

```
06:41:12.489877 3ffe:fffe:2:1:2c0:26ff:fe60:2a83 > 3ffe:fffe:2:1:ffff::1: icmp6:  echo request
06:41:12.490032 3ffe:fffe:2:1:290:27ff:fe17:9d2b > 3ffe:fffe:2:1:2c0:26ff:fe60:2a 83: icmp6: echo reply
① ICMPv6 Echo Request / Reply によるエニーキャストアドレスからユニキャストアドレスへの変換

06:41:12.490924 3ffe:fffe:2:1:2c0:26ff:fe60:2a83.1447 > 3ffe:fffe:2:1:290:27ff:fe 17:9d2b.23: S
06:41:12.491076 3ffe:fffe:2:1:290:27ff:fe17:9d2b.23 > 3ffe:fffe:2:1:2c0:26ff:fe60 :2a83.1447: S ack
06:41:12.491126 3ffe:fffe:2:1:2c0:26ff:fe60:2a83.1447 > 3ffe:fffe:2:1:290:27ff:fe 17:9d2b.23: . ack
② ユニキャストアドレスへの TCP 3 ウェイハンドシェーク
```

図 8　telnet の結果

```
07:32:48.700560 3ffe:fffe:2:1:2c0:26ff:fe60:2a83 > 3ffe:fffe:2:1:ffff::1: icmp6:  echo request
07:32:48.700716 3ffe:fffe:2:1:290:27ff:fe17:9d2b > 3ffe:fffe:2:1:2c0:26ff:fe60:2a 83: icmp6: echo reply
① ICMPv6 Echo Request / Reply によるエニーキャストアドレスからユニキャストアドレスへの変換

07:32:48.701590 3ffe:fffe:2:1:2c0:26ff:fe60:2a83.1449 > 3ffe:fffe:2:1:290:27ff:fe 17:9d2b.21: S
07:32:48.701743 3ffe:fffe:2:1:290:27ff:fe17:9d2b.21 > 3ffe:fffe:2:1:2c0:26ff:fe60 :2a83.1449: S ack
07:32:48.701795 3ffe:fffe:2:1:2c0:26ff:fe60:2a83.1449 > 3ffe:fffe:2:1:290:27ff:fe 17:9d2b.21: . ack
② ユニキャストアドレスへの TCP 3 ウェイハンドシェーク
```

図 9　ftp の結果

が実行されたときのサーバとの接続確立までのパケットの送受信について， `tcpdump`[7] による出力を，`telnet` は図 8 に，`ftp` は図 9 にそれぞれ示す． `telnet`，`ftp` ともに，

（1） ICMPv6 Echo Request/Reply により，アドレス解決をおこなう

（2） 得られたユニキャストアドレスに対して実際の通信を開始する（TCP 3 ウェイハンドシェークの開始）

という手続きを行っている．この結果から，ICMPv6 Echo Request/Reply によってアドレス解決を行い，ユニキャストアドレスに対して TCP 3 ウェイハンドシェークを行っていることが確認できた．

4.3　UDP 通信の評価

次に，DNS サーバにエニーキャストアドレスを割り当て，クライアントが通信を行う際のホスト名解決を行っている様子を，図 10 のネットワークでの実験によって示す．

クライアント端末の `/etc/resolv.conf`（FreeBSD において DNS サーバのアドレスを設定するファイル）には，図 10 の DNS サーバに割り当てられてあるエニーキャストアドレス `3ffe:fffe:2:1:ffff::5` が，

- `nameserver 3ffe:fffe:2:1:ffff::5`

と書いてあり，図 10 は，このクライアントが

```
> telnet xxx.yyy.zzz（FQDN）
```

を実行したときの様子を表している．その通信手順は以下の通りである．

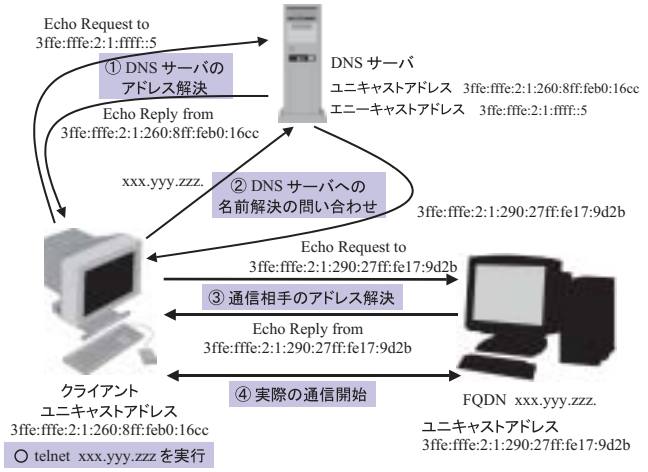

図 10　ネットワーク構成（DNS サーバがある場合）

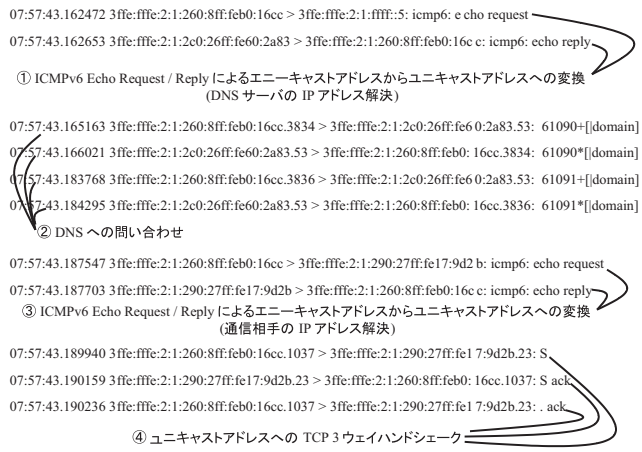

図 11　DNS 名前解決の結果

（1） 自ノードに設定してある DNS サーバのエニーキャストアドレスに対し，ICMPv6 Echo Request/Reply によるアドレス解決を行う

（2） 得られた DNS サーバのユニキャストアドレスに対して，ホスト名 `xxx.yyy.zzz` の名前解決の問い合わせを行う．

（3） 名前解決によって得られた通信相手の IP アドレスに対し，ICMPv6 Echo Request/Reply によるアドレス解決を行う

（4） 得られた通信相手のアドレスに対して実際の通信を開始する

この，ホスト名解決を行ってから `xxx.yyy.zzz` と通信を開始するまでのパケットの送受信の記録を `tcpdump` による出力として，図 11 に示す． この結果から，DNS サーバの IP アドレスとしてエニーキャストアドレスを指定したホスト名解決を実現できることが確認できた．

DNS サーバにエニーキャストアドレスを割り当て，ホスト名解決を問い合わせることができるというのは，本提案通信モデルの大きな利点である．世界中のすべての DNS サーバに，ある決まったエニーキャストアドレスを割り当てておけば，クライアント端末のオペレーティングシステム上であらかじめ DNS サーバのアドレスを設定することができる．そのため，ネットワーク管理者が DNS サーバのアドレスを設定することなく，プラグアンドプレイ時に最適な DNS サーバを利用することができるようになる．

5. まとめと今後の課題

本稿では，IPv6 で新たに導入されたエニーキャスト機能について，その問題点に関する考察を行った．その結果，既存アプリケーションはエニーキャストアドレスを利用した通信ができないという問題があることを示した．そこでこの問題点を解決するため，AARP という，エニーキャストアドレスをユニキャストアドレスへ変換し，実際の通信は変換後のユニキャストアドレスを用いて行う，新たなプロトコルを提案した．また，実装方法を示し，既存のアプリケーションプログラムを変えることなくエニーキャストアドレスを用いた通信を行うことができることも明らかにした．

本稿で実験に用いたネットワークは非常に小さな構成であるため，エニーキャストアドレスを広域ネットワーク上で利用した場合にどのような問題が起こるかについての実験は行っていない．また，ICMPv6 Echo Request/Reply を通信前にやり取りするためのメッセージ量の把握や，それを軽減するためにキャッシュを行う場合のキャッシュの保持時間などは，実際のネットワーク上でその影響を計測し，明らかにしていかなければならない．

文　献

[1] R. Hinden, S. Deering, “IP Version 6 Addressing Architecture,” RFC2373, 1998.
[2] C. Partridge, T. Mendez, W. Milliken, “Host Anycasting Service,” RFC 1546, 1993.
[3] J. Postel, “Transmission Control Protocol,” RFC793, 1981.
[4] J. Bound, P. Roque, “IPv6 Anycasting Service: Minimum requirements for end nodes,” draft-bound-anycast-00.txt (expired), 1987.
[5] H. Kitamura, “A SOCKS-based IPv6/IPv4 Gateway Mechanism,” RFC3089, 2001.
[6] A. Conta, S. Deering, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,” RFC2463, 1998.
[7] “TCPDUMP public repository,” `http://www.tcpdump.org/`.