第 XXX 部

IPv6 の欠点の修正

第 30 部
IPv6 の欠点の修正

第 1 章　v6fix WG の概要

IPv6 は、その仕様、実装、運用の大部分が健全であるが、少しの欠点により、ユーザへ悪い印象を与えている可能性がある。たとえば、あるユーザが試しに IPv6 を使い始めた際に、IPv4 を利用する場合と比べ、些細な欠点のために快適さが大幅に劣っているなら、IPv4 へ後戻りしてしまうだろう。

少しの欠点のせいで、このようなことが実際に起こっている。そこで、WIDE Project では、IPv6 の仕様、実装、運用の欠点を修正する活動を始めた。これを取り扱う分科会の名前は、IPv6 Fix（v6fix）である。

IPv6 Fix は、以下のような成果を出すことを目的とする。

- 仕様に問題がある場合は、仕様を修正する。
- 実装に問題がある場合、自分たちで直せるものは直し、直せないものはベンダに通知する。また、どの欠点が修正され、どれがまだであるかという状況を把握できるようにする。
- 運用に問題がある場合は、運用者に通知する。
- 以上を解説した文章を、日本語と英語の両方で書き公開する。

第 2 章　動機と全体像

2.1 背景

IPv6 Fix を始めるきっかけとなった、ユーザに悪印象を与えている事件を 2 つ示す。

1) あるユーザがあるホテルに宿泊した。“ipv6 install” を実行した Windows XP をインターネットに繋ごうとしたが、接続できなかった。そのため、ホテルに問い合わせたところ、「“ipv6 uninstall” して下さい」と言われた。実際に実行してみると、接続できるようになった。

   これは、おそらく AAAA RR （Resource Record）をひくとおかしな返答を返す DNS サーバの問題であろう（4.2 節参照）。

2) ある ISP で BIND 9 をサービスに投入したところ、ユーザから「ブラウザでページにアクセスする際の快適さが失われた」という苦情を受けた。

   これは、BIND 9 が利用できない IPv6 トランスポートを利用しようと試みているのが原因である。

2.2 全体像

アプリケーションが通信を開始する際は、以下のような段階を踏む。

1) 名前に対し DNS をひく
   終点アドレスが得られる。
2) コネクションの確立を試みる
   始点アドレスと終点アドレスの組み合わせを探す。
3) データのやり取り
   経路 MTU などを調整する。

これらの手続きがスムーズに処理されることが重要である。また、利用できる IPv6 の終点アドレスと始点アドレスの組み合わせが見つからない場合は、速やかに IPv4 へ遷移すべきである。

以下、それぞれの段階で見つかっている欠点について説明する。

2.2.1 名前に対し DNS をひく

IPv4 と IPv6 が両方利用できる場合は、名前に対し A RR と AAAA RR を検索すべきである。

問題のあるリゾルバは、IPv6 が利用できないのに AAAA RR をひく（IPv6 が利用できないから AAAA RR をひいても無駄であるし、IPv6 アドレスを得ることにより、利用できないのに利用しようと試みる間違いをアプリケーションが起こす可能性がある。）。

DNS サーバ側に問題がある場合もある。たとえば、AAAA RR を問い合わせると、誤った返答をする DNS サーバが存在する。前述のように、これがホ

テルの事件の原因だと考えられている（4.2 節参照）。

またたとえば、利用できない IPv6 トランスポートで検索を試みる DNS サーバも存在する。前記の BIND 9 がこれにあたる。

2.2.2 コネクションの確立を試みる

次に利用できる終点アドレスと始点アドレスの組を探すべきである。DNS をひいて終点アドレスのリストが得られている。そのリスト中のある終点アドレスに対し、最適な始点アドレスを選択し、コネクションの確立を試みる。成功すれば、その時点で組み合わせの探索を終了する。失敗すれば、ほかの終点アドレスを対象とし、同じ動作を繰り返す。

この過程に関し、IPv6 の仕様上の欠点が指摘されている。それは、RFC 2461 の 5.2 節で定められている “onlink assumption” である。規定してある動作は次の通りである。すなわち、デフォルトルートがない場合、終点アドレスが IPv6 のグローバルアドレスであるなら、その通信相手は同一リンクに存在すると仮定して、通信を試みる。現実的には、多くの場合通信相手は同一リンクに存在しないため、TCP がタイムアウトするまで待たされる（3 章参照）。

また、通信相手に運用上の問題がある場合がある。たとえば、あるサーバがメールと Web のサービスを提供しているとする。メールは IPv6 に対応済みであるが、Web がまだである。そして、以下のように CNAME を使って設定してしまったとする。

```
server.example.com.   IN A      192.0.2.1
                      IN AAAA   2001:DB8::1
www.example.com.      IN CNAME  server.example.com.
mail.example.com.     IN CNAME  server.example.com.
```

サーバへは IPv6 の到達性はある。しかし、Web のサービスは IPv6 で提供されていないため、コネクションを張ろうとすると、TCP の RST が返ってくるまで待たされる。この待ち時間は短いため問題視する必要はないかもしれないが、別の深刻な問題も引き起こす。それは、IPv6 → IPv4 トランスレータの裏からコネクションが張れなくなることである。

この場合、サーバの管理者は、以下のように設定すべきである。

```
www.example.com.   IN A     192.0.2.1
mail.example.com.  IN A     192.0.2.1
                   IN AAAA  2001:DB8::1
```

2.2.3 データのやり取り

コネクションが確立された後、快適な通信となるためには、最適な経路を使うことと、ICMPv6 が送受信できることが必要である。

IPv6 in IPv4 トンネルが不適切に敷設されていると、ほかにもっと速い経路があるにもかかわらず、遅い IPv6 in IPv4 トンネルを利用してしまう場合がある（6 章参照）。

また、ICMPv6 を落とすような実装のファイアウォールや、落とすように設定されたファイアウォールが存在すれば、経路 MTU を調整できないなどの問題が発生する（7 章参照）。

第 3 章　IPv6 の仕様の不備

IPv6 は、IPv4 からのスムーズな移行を踏まえ、慎重に議論が重ねられ、その仕様が固められていったはずだが、実際の運用にあたり、不具合が発見された。その代表例が onlink assumption の問題である。

3.1 onlink assumption 概要

RFC 2461 の 5.2 節は、もしデフォルトルートが存在しない場合、IPv6 ノードは、すべての宛先が、同一のリンクにいると仮定せよと規定している。これは、手動で異なるプレフィクスをつけられた 2 つのノードが、互いに通信する場合に有用だとされてきた。

3.2 onlink assumption の問題点

3.1 節で挙げたとおり、onlink assumption が有用である場合が想定されてはいたが、移行時期においては有害であることが分かってきた。

たとえば、A レコードおよび AAAA レコードが登録されているようなサーバへアクセスする場合を考える。この場合、IPv6 の接続性が存在しているときは問題ないが、移行時期においては、ノードが IPv6 に対応していても、接続しているネットワークが IPv6 に対応しておらず、IPv4 での接続性しかない場合が多々ある。

このような場合、ユーザとしては、あるノードへアクセスする際は、IPv6 での試行に時間をかけるこ

となく、IPv4で接続して欲しい。しかし仕様では、デフォルトルートが存在しない場合は、すべての宛先が同一のリンクに存在すると仮定せよと定義されている。そのため、IPv6でのアクセスを試みる。すると、そのノードは、その宛先に対するNeighbour Discoveryが失敗するまでの間、IPv4での接続を試みない。このため、IPv4にフォールバックするまでに、数秒の遅延が発生することになる。

3.3 それぞれの実装の状況

3.3.1 WindowsXP SP1およびSP2

onlink assumptionが実装されてはいるが、IPv6のネットワークの到達性がない場合の試行順序が

1) IPv4で試行

2) IPv4で失敗した際には、IPv6で試行

となっているため、あるノードへアクセスする際の遅延は発生しない。

3.3.2 Linux

Linux 2.4系およびLinux 2.6.9以前のカーネルでは、onlink assumptionに起因する問題が発生する。この問題の回避方法としては、IPv6の経路が存在しない状態において、各インタフェースごとに設定されている“::/0”な経路を手動で削除することが挙げられる。

3.3.3 BSD

通常の使用においては、問題は発生しない。

3.3.4 MacOS X

対処済であり、問題は発生しない。

3.3.5 Solaris9

onlink assumptionが有効となっており、問題が発生する。

3.4 解決策

すでにIETFではこの問題点が指摘され、仕様の改定に向けて作業が進められており、RFC 2461の改訂の際に、仕様に反映される予定となっている[219]。

第4章 DNSサーバとリゾルバ

4.1 問題点

現在、多くのアプリケーションがIPv6に対応しており、その多くはgetaddrinfo()ライブラリを利用してDNSのA RRおよびAAAA RRの問い合わせを送信する。その結果として得られたアドレスに対し順次コネクション確立を試みて、成功したアドレスを使って以後の通信を進めるというのが典型的な動作である。この際、IPv4とIPv6の両方のアドレスが得られている場合には、先にIPv6アドレスによるコネクション確立を試みる実装が一般的である。

これらの動作は、A RR、AAAA RRの問い合わせがいずれも正確に処理され、接続性のないアドレスへのコネクション確立を試みた場合にはそれがただちに失敗して接続性のあるアドレスとのコネクションを迅速に確立できる、という仮定に依存している。しかし、実運用上においては、仕様に反するDNSサーバ（後述）や、IPv6プロトコル仕様上の不具合（3章参照）、TCPの仕様および実装上の性質（5章参照）などによって、これらの仮定が成り立たない場合がある。そのような場合には、接続性のあるアドレスとのコネクションが成立するまでに長い待ち時間を要したり、接続できるはずのアドレスとのコネクション確立に失敗したりといった問題を引き起こす。Webブラウジングの例でいえば、ページが表示されるまでに長時間待たされたり、表示できるはずのページが表示できない、といった症状として現れる。

AAAAの問い合わせに関するDNSサーバの不正な挙動については、文献[213]で詳述している。これらについては、4.2節と4.3節でも述べる。

一方、上で述べたgetaddrinfo()ライブラリに代表されるDNSクライアント（リゾルバ）の挙動を工夫することで、問題を回避あるいは軽減できる可能性もある。これについては4.4節で説明する。

なお、文献[213]でも述べられているように、一般に「DNSサーバ」とよばれているサーバは、その機能によって権威（authoritative）DNSサーバと

キャッシュサーバに分類される。本章では、とくに断りのない限り、「DNS サーバ」とは権威 DNS サーバを指すものとする。

4.2 不正な挙動をする DNS サーバの事例

本節では、不正な挙動をする DNS サーバについて述べる。詳細は、文献 [213] に記述されている。

あるホストの A RR が存在し、AAAA RR が存在しない場合、AAAA RR の問い合わせに対して DNS サーバは、以下のような応答を返す必要がある。

- response code（RCODE）が 0（つまり no error）
- answer section は空

この応答を受け取ったキャッシュサーバやリゾルバは、スムーズで正しい処理を継続できる。これ以外の応答を返して、問題を引き起こす DNS サーバの事例を以下に述べる。

A) AAAA RR の問い合わせを無視する
IPv6 をサポートした多くのリゾルバは、最初に AAAA RR を問い合わせ、解決できない場合 A RR を問い合わせるフォールバックの機能をもっている。AAAA RR の問い合わせが無視されると、名前解決に多大な時間を要し、ユーザに影響をおよぼす。

B) RCODE 3（“Name Error”、“NXDOMAIN”）を返す
“Name Error” は、問い合わせたホスト名に関する RR が 1 つも存在しないことを意味している。この応答を受け取ったリゾルバは、すぐに名前解決をあきらめるため、A RR の問い合わせにフォールバックしない。したがって、本来なら IPv4 の通信が可能である場合でも、通信不能になってしまう。

C) “Name Error” 以外の RCODE を返す
RCODE 4（“Not Implemented”）、RCODE 2（“Server Failure”）、RCODE 1（“Format Error”）が返ってきた場合、リゾルバは正しい処理をする。キャッシュサーバは、AAAA RR が存在しなかったことをキャッシュしないため、後のリゾルバからの問い合わせのたびに問い合わせるため、DNS サーバの負荷を上げ、ネットワークの帯域を浪費する。

D) 壊れた応答を返す
たとえば、AAAA RR の問い合わせの答えを入れるフィールドに IPv4 アドレスが入った応答が返る場合がある。キャッシュサーバはリゾルバに対して、そのまま答えを渡すものや、RCODE 2（“Server Failure”）にして返すものがある。後者の場合、前項と同様の悪影響がある。

E) Lame Delegation になる
A RR の問い合わせには Authoritative 応答を返すが、AAAA RR には Inauthoritative 応答を返す。あるキャッシュサーバは、ある DNS のゾーンについて、一度 lame と認識した DNS サーバに対してはその後一定期間問い合わせをしないようになる。また、その期間中は、そのゾーン内の問い合わせに対してリゾルバに無条件に RCODE 2（“Server Failure”）を返す。この場合は、たとえリゾルバが A RR にフォールバックしても名前解決に失敗する。

4.3 不正な挙動をする DNS サーバの調査

前節で述べたように、不正な挙動をする DNS サーバの存在がユーザにまで影響を与え、IPv6 へのスムーズな移行を妨げる原因の 1 つとなっている。IPv6 Fix では、このような DNS サーバをインターネットから排除し、健全な IPv6 環境の構築を目指している。本節では、そのための活動内容について述べる。

不正な挙動をする DNS サーバを削減していくために、以下のような方策をとる。

1. 不正な挙動をする DNS サーバを探し出すツールの作成
2. 調査の実施
3. 利用している DNS サーバの種類をサイト管理者へインタビュー
4. ベンダへ調査結果を報告し、実装の改善を求める

2004 年には、調査ツールを作成し、JP ドメインを調査した。なお、JP ドメインの調査は株式会社日本レジストリサービス（JPRS）との共同研究である。

- 調査ツール
  調査ツールは perl で作成され、調査のためのパケットを DNS サーバに対して送信し、返ってきたパケットを解析して、結果を出力する。
  調査は、以下の手順を提供されたドメイン名の数分繰り返す。
  1. 対象となるドメイン名を決定する。
     example.jp
  2. 当該ドメインを管理する DNS サーバを特定する。

ns.example.jp, ns.wide.ad.jp

3. 調査に用いるホスト名を決定する。入力がドメイン名のため、“www” や “ftp” など、典型的なホスト名をドメイン名に付加し、A RR を問い合わせて存在するかを確認する。

   www.example.jp の A RR を
   ns.example.jp へ

4. 3. で調査に用いるホストが特定できた場合、当該ホストの AAAA RR を 2. で見つかった各 DNS サーバに問い合わせて挙動をチェックし、結果を出力する。

   www.example.jp の AAAA RR を
   ns.example.jp へ
   www.example.jp の AAAA RR を
   ns.wide.ad.jp へ

- JP ドメインの調査

JP ドメインの調査は、JPRS が本調査のために用意したサーバマシンと 2004/11 時点での JP ドメインリストを用いて、5 日間かけて実施した。

表 4.1 に JP ドメインの調査結果について述べる（プライバシなどの問題から、結果の統計情報についてのみ述べる）。

ドメインの視点と DNS サーバの視点の両方で集計した。表 4.1 では、それぞれ example.jp、および ns.example.jp/ns.wide.ad.jp に相当する。

不明である理由として、サーバが一時的にダウンしている場合や、上記調査方法 3 にて調査に用いるホスト名を推測しているため、見つからない場合などが挙げられる。

今後は、不正な挙動をする DNS サーバの実装を特定するために、サイト管理者へインタビューし、その結果をベンダへ報告する。また、調査範囲を JP ドメイン以外にも広げ、その際の調査方法を検討する。

表 4.1. JP ドメインの調査結果

	ドメイン	DNS サーバ
問題あり	0.04%	0.11%
問題なし	82.16%	84.39%
不明	17.80%	15.50%

表 4.2. 問題のあるサーバ内の分類

A) AAAA RR の問い合わせを無視する:	4.7%
B) RCODE 3 (“Name Error”) を返す:	4.7%
C) “Name Error” 以外の RCODE を返す:	8.5%
D) 壊れた応答を返す:	0.0%
E) Lame Delegation になる:	82.1%

4.4 リゾルバ側での解決案と実装

4.1 節で述べた DNS リゾルバの典型的な動作は、すべての DNS サーバが仕様に忠実に振る舞い、また IPv6 のコネクション確立が失敗した場合に IPv4 に迅速にフォールバックできている限り、本来は問題を引き起こす原因にはならない。しかし、これまでに見てきたように、実際の運用環境においてはこれらの仮定が成り立たない場面も多く、それが問題を生んでいるのが実情である。さらに、こうした問題が IPv6 への移行を妨げる原因の 1 つともなっている。

本節では、DNS リゾルバ側の動作を工夫することによる対処法について述べる。ここでは、以下の 2 つの方法を提案する。

a. AAAA RR の問い合わせをそれが必要な場面に限定する
b. AAAA RR の問い合わせの待ち時間をある条件のもとで短縮する

以下では、それぞれの方法について詳述し、その実装を紹介する。

4.4.1 AAAA RR の問い合わせを限定する

DNS に関連した問題は、AAAA RR の問い合わせをそれが必要な場面に限定することでその相当部分を軽減または回避できる。たとえば、IPv6 の接続性をまったく持っていない環境においては、IPv6 による通信は不要であり、したがって一般ユーザにとって AAAA RR の問い合わせも不要である。そのような場合には、A RR のみを問い合わせ、得られた IPv4 アドレスを使って通信することで、これまでに述べてきた問題をすべて回避できる。

そこで、ここでは「AAAA RR の問い合わせを必要な場面」を「リンクローカルアドレス以外の IPv6 アドレスを持っている場面[1]」と定義し、その状況でのみ AAAA RR を問い合わせるという変更を提案する。これは、事実上は「グローバル IPv6 アドレス

1 ここでは、表現を簡単にするために、ループバックアドレス（::1）はリンクローカルアドレスの一種であるとみなすことにする。

を持っている場面」とほぼ同義である。

リンクローカル以外の IPv6 アドレスを持たないホストにとっては、実際的なアプリケーションを用いた IPv6 での通信にはほとんど意味がない。したがって、リンクローカルアドレス以外のアドレスの有無による判定は現実的な観点から有効だといえる。また実際、リンクローカルアドレスのみを持つ IPv6 ホストという環境は、IPv6 に対応した OS を稼働させている非 IPv6 ユーザにとっての典型的な設定である。これらのユーザに対して、AAAA RR の問い合わせに起因した問題点の回避策を提供することで、IPv6 に対して短絡的な悪印象を与えずに済み、間接的に IPv6 への移行を円滑にすることに寄与できるであろう。

この方式は、getaddrinfo() ライブラリに対しては、第 3 引数の hints 構造体に AF_UNSPEC ファミリが指定された場合に、リンクローカルアドレス以外の IPv6 アドレスがなければ DNS の AAAA RR 問い合わせを出さないようにすることで実現できる。

なお、RFC3493（IPv6 の API 仕様）の 6 章によれば、hints 構造体における AF_UNSPEC の指定は、呼び出し側においてすべてのファミリに対応していることを示しているのみであり、ライブラリ側での具体的な挙動は特定されていない。したがって、条件に応じて AAAA RR の問い合わせを省略したとしても仕様には反しない。

ところで、提案方式は、getaddrinfo() ライブラリの結果をその後の通信先のアドレスとして利用するということを前提としている。仮にたとえば、このライブラリを DNS の運用・診断ツールの一部として利用し、AF_UNSPEC ファミリを指定した場合に A RR と AAAA RR の両方の問い合わせがなされることが期待されていたとすると、その仮定には反する挙動となる。しかし、上述のように、API の仕様上は、アプリケーション側で問い合わせ結果にこのような仮定をおくことはできないはずである。すなわち、両方の問い合わせが必要なら AF_INET と AF_INET6 を指定して二度 getaddrinfo() を呼び出すといった処理が、API 仕様上は正しいアプリケーション実装であり、問題のある場合に修正すべき対象は本来はアプリケーション側である。また、こうした運用ツールでは getaddrinfo() よりも低レベルのインタフェースを用いることが普通であり、上述の変更が実際に問題になる場面は少ないと考えられる。

4.4.2 AAAA RR の問い合わせ待ち時間を短縮する

IPv6 の接続性を持った環境においては、AAAA RR の問い合わせは必須であり、4.4.1 項で述べた方法は適用できない。ここでは、そのような場合の中で、とくに AAAA の問い合わせを無視する DNS サーバに対応するリゾルバ側の対応方法を提案する。

一般に広く使われている ISC BIND[147] に由来したリゾルバの実装では、まず AAAA RR の問い合わせを出す。それが成功するか時間切れになった時点で A RR の問い合わせを出し、同様に成功するか時間切れになった時点で、AAAA RR の問い合わせ結果とあわせた最終的な応答をアプリケーションに返す。この方法では、問い合わせ先の DNS サーバが AAAA RR の問い合わせを無視している場合、最初の段階で 1 分程度の待ち時間を要した上、結果的には A RR の問い合わせのみが有効だったということになる。

これに対し、提案方式では、以下のようにして不要な待ち時間を軽減する。

- まず A RR の問い合わせを出す。
- 次に AAAA RR の問い合わせを出す。ここで、A RR の問い合わせが成功していた場合には、待ち時間を短縮し、問い合わせが無視されていても不必要に長く待たないようにする。

この方式によれば、「A RR には正しく応答するが AAAA RR は無視する」DNS サーバへの問い合わせについて、最初の段階では正しい応答が迅速に得られる。したがって、AAAA RR の問い合わせについては短縮した待ち時間が適用され、AAAA RR の問い合わせが無視されている期間中、不必要に待たずに済むようになる。

4.4.3 提案方式の実装

前の 2 つの項で述べた提案方式を、KAME プロジェクトから提供しているスナップショットの FreeBSD 用 getaddrinfo() ライブラリとして実装した。この実装は 2004 年 11 月 29 日版以降のスナップショットで利用できる。

この実装では、4.4.2 項における短縮待ち時間を $\max(1\text{秒}, T*2)$（ただし T は A RR の問い合わせが成功した場合に、それに要した時間）としている。これにより、問い合わせ先の DNS サーバが遠隔地にあるなどの理由でもともと遅延が大きい場合にも妥当な長さの待ち時間を設定できると考えられる。す

なわち、「A RR にも AAAA RR にも正しく応答するが、応答までの遅延が大きい」DNS サーバへの問い合わせの場合にも、A RR と AAAA RR の両方の応答が正しく得られると期待できる。

なお、RFC 3484 で規定されている、「アドレス選択の既定順序」を実現するためには、一般的な getaddrinfo() ライブラリは IPv6 アドレスを先に、IPv4 アドレスを後に返す必要がある。4.4.2 項で述べた方式では、DNS の問い合わせについては A RR を先行させているが、実際の実装においては、その後の処理でアドレスを並べ替え、RFC 3484 で規定された順序となることを保証している。

第 5 章　TCP のコネクション確立

5.1 問題点

TCP のコネクション確立においても、IPv4 へのフォールバックに時間がかかる場合がある。4 章で述べたように、通常の IPv6 対応アプリケーションでは、通信相手のアドレスを getaddrinfo() ライブラリにより取得し、複数のアドレスが得られた場合はそれぞれに対して順次コネクションの確立を試みる。多くの実装では、相手が IPv4 と IPv6 のデュアルスタックであれば、IPv6 アドレスが優先されるため、IPv6 での接続を先に試み、失敗した場合に IPv4 での接続にフォールバックする。相手に複数の IPv6 アドレスがあれば、そのそれぞれに対しても接続を試みることになる。

通信相手からリセットが返る、あるいは、ICMPv6 ハードエラーが返る場合は、直ちにコネクションの確立が失敗し、RTT 約 1 回分の時間で迅速に次のアドレスにフォールバックできる。しかし、コネクションの確立が、再送回数がしきい値を越えてたり、タイムアウトで失敗するような場合にはフォールバックに長い時間を要することになる。

TCP がコネクション確立の失敗に時間がかかるケースは、主に 2 通りに分類できる。

(1) 再送回数がしきい値を越える場合

これは、ICMPv6 ソフトエラーが返ってくる場合に起こる。TCP の仕様 [20] では、一部の ICMP エラーをソフトエラーとし、ネットワークの一時的な障害の可能性があるため、接続を中断してはいけないことにしている。そのため、TCP は ICMPv6 によるエラーが返ってきているにもかかわらずこれを無視して再送を続ける。4.4BSD 由来の TCP コードでは、コネクション確立前の再送回数に制限を持つため、通常 10 秒程度で再送回数がしきい値を越えコネクション確立が失敗する。

なお、ソフトエラーは ICMP Destination Unreachable メッセージのうち、

コード 0（network unreachable）、
コード 1（host unreachable）、
コード 5（source route failed）

の 3 種類となっている。

(2) タイムアウトする場合

こちらは、コネクション確立の試みに全く応答がない場合に起こる。4.4BSD 由来の TCP コードでは、コネクション確立用にタイマを持ち、デフォルトでは 75 秒経ってもコネクションが確立できないとタイムアウトする。

問題の本質は、複数のアドレスに対してコネクション確立を試みて、問題があれば透過的にフォールバックする手法の実現の難しさである。最近では、IPv4 だけの世界でも、ひとつの DNS ホスト名が複数のミラーサーバのアドレスにマップされるような使い方が見受けられ、同様の問題を含んでいる。しかしながら、現状のデュアルスタック・インターネットでは、DNS に IPv6 アドレスを登録しているにもかかわらず、IPv6 で到達できないホストが少なからず存在する。そのため、ユーザには IPv6 を利用可能にすると接続に時間がかかるかのような印象を与えてしまう。

本来は、DNS に登録されたすべてのアドレスは到達可能であるべきであり、到達できないホストやネットワークを修正するのが筋である。そちらに関しては 6 章で述べる。しかし、通信相手側の問題を解決することは難しく、現実的には、TCP のフォールバックの高速化に関して、何らかの対策を取って問題を軽減することが求められる。

5.2 対策

TCP のフォールバックの高速化は、接続性問題の対症療法にすぎない。ここでは、IPv6 普及の障害を減らすという観点から、IPv6 から IPv4 にフォール

バックする場合を中心に対策を述べる。

(1) 再送回数がしきい値を越える場合

ICMP ソフトエラーが返る場合には、コネクション確立時に限りこれをハードエラーと同様にみなして迅速にフォールバックする方法が提案されている [112]。ソフトエラーに対する対応が仕様に定められた当時は、通常のホストはひとつの IP アドレスしか持たなかったため、ネットワークやホストに到達できないという ICMP エラーが返って来ても、接続性の回復を期待して再送を繰り返すしか手段がなかった。しかし、IPv4 と IPv6 の両方のアドレスを持つことが普通になった現在では、ICMP ソフトエラーによって接続性の問題が明らかになれば即時に次のアドレスにフォールバックすることもできる。

難しいのは、再送により接続性が回復する可能性とフォールバックにより迅速に接続性が得られる可能性のバランスを見つけることである。これは通信環境で大きく異なる。IPv6 接続性に問題の多いデュアルスタック環境では、ICMP ソフトエラーで即座にフォールバックするのは有効である。一方、物理層において非常に不安定な通信環境では、再送のほうが有効であろう。4.4BSD 由来の TCP でも、ICMP ソフトエラーが返ると、コネクション確立前に限って、通常より早期に終了するという中間的な方法を取っている。通信環境が大きく変わった現在では、これよりは積極的なフォールバックが妥当だと思われるが、不安定な通信環境でも動作する TCP の利点を損なわないよう注意することも必要である。また、今後 IPv6 接続性をめぐる状況が変化しても柔軟に対応できるように考慮しておく必要もあるだろう。

(2) タイムアウトする場合

コネクション確立の SYN パケットにまったく応答がなくタイムアウトする場合にはあまり有効な対策はない。この場合は、タイムアウト以外のイベントがなく、かつ、タイムアウトは極端に短くすることはできない。多少タイムアウトを短くしても、あまり効果は期待できない。

ほかの方法として、複数のアドレスに並列に接続を試みることも考えられるが、この方法はリソースの無駄使いが多く、また、既存のコードへの変更も大きいためあまり現実的ではないと思われる。

また、到達できないアドレス情報をキャッシュして、それ以降利用しないようにすることも考えられる。最初のコネクション確立は時間がかかるが、2 度目以降のアクセスには効果がある。しかし、この方法も必要な変更の大きさの割りに、その効果に疑問がある。

このように、ネットワークから何も情報が得られずタイムアウトする場合には、TCP 側ではあまり細工のしようがないため、ネットワークの問題を解決することが必要になる。

5.3 標準化および実装状況

ICMP ソフトエラーを、コネクション確立前に限り、ハードエラーと同様に扱う提案がされており [112]、IETF TCP Maintenance and Minor Extensions（tcpm）WG で議論されている。IPv6 コミュニティには、IPv6 普及のために必要だという意見が多いが、TCP コミュニティには既存の TCP の挙動を変えることに慎重な意見があり、本原稿の執筆時点（2005 年 1 月）では合意は形成されていない。

議論となっているのは、まず、ICMP ソフトエラーに対する挙動を、システムグローバルな変数とするべきか、ソケットオプションなどでアプリケーションごとに制御できるようにするべきかという点である。前者は既存のアプリケーションを変更せずに導入できる。それに対し、後者は必要なアプリケーションを変更するべきだという立場である。そして、デフォルトの挙動がどうあるべきか、また、それらを仕様で定めるべきか実装にまかせるべきかという議論もある。これらの議論は、IPv6 の普及促進という立場と、TCP は本来どうあるべきかという立場で大きな隔たりがあり、収束には時間がかかりそうである。

実装に関しては、Linux では 1996 年のカーネルバージョン 2.0.0 より、ICMP エラーでただちにフォールバックするようになっている。KAME による実装でも 2004 年 12 月からこの方式を採用している。

5.4 セキュリティへの考慮

ICMP ソフトエラーでフォールバックを速くしても DoS 攻撃の危険性は従来と変わらない。現在でも、ICMP ハードエラーが返ればコネクション確立に失敗するので、これと同じ挙動になるだけで危険性が拡大することはない。

第 6 章　IPv6 インターネットの品質

本格的な IPv6 への移行を実現するには、IPv6 インターネットが IPv4 インターネットと同等以上の品質を持つことが必要条件である。IPv6 インターネットが IPv4 より劣っていれば、普通のユーザはコストと労力を使って移行するはずがない。しかし現実的に、IPv6 インターネットの品質はまだ IPv4 に遠く及ばない。

ユーザは IPv6 を使っていて、ごく一部の問題のあるサイトに出会うと IPv6 自体の問題だと考えてしまう。そのことが IPv6 普及の大きなハードルになってきている。問題の原因としては、実験的な運用、ピアリング不足、不適切なトンネルなど、さまざまな要因がある。

そして、IPv6 普及の抱えるジレンマとして、普及促進のために手軽な IPv6 接続を提供することが、実験的な利用を増やして、結果的に IPv6 インターネットの品質を低下させることも挙げられる。さらに、IPv6 の透過的な設計が問題解決を難しくしている一因になっている。IPv6 は IPv4 と共存し、もし IPv6 での通信に問題があれば自動的に IPv4 にフォールバックするようになっている。そのため、利用者が気がつかないうちに IPv4 で通信できてしまうので、IPv6 ネットワークの問題はしばしば見過ごされることになる。

我々は日常的に IPv6 を使っていて、問題があるのは一部であり、ほとんどのサイトには問題がないことを直観的に知っている。もし、一部のサイトが問題を起こしているだけなら、それらを直せば IPv6 インターネットの品質を大きく全体的に改善できる。

我々はこのように考えて、IPv6 インターネットの品質の現状を把握するため、2004 年の 1 月から IPv6 インターネットと IPv4 インターネットのの品質比較調査を行っている。調査の詳細については [40] を参照されたい。

6.1 リーフサイトにおける問題点

- 実験的な IPv6 導入

  IPv6 を実験的に導入してみたものの、本格運用にいたらない場合。日常的には IPv6 を利用していないため、問題があっても自分達は気がつかない。多くの場合、IPv6 アドレスが DNS に登録されたままになる。

- 不適切なトンネル

  トンネルは IPv6 の導入には欠かせない技術であるが、同様に誤った利用も簡単にできてしまう。なかでも、物理的なトポロジを無視したトンネルによる利用可能帯域の減少や応答時間の増大は、IPv6 を使うと通信品質が低下する問題の大きな要因である。たとえば、6bone 時代の古いトンネルが残っている場合や、国内間の通信が海外のトンネルブローカーを経由する場合も見られる。物理トポロジが変わってしまっているのに、トンネル設定はそのままにされがちで、また、トンネルで構成されたネットワークは障害診断が難しいため、結果的にトンネルの利用は品質低下を招きやすい。

6.2 バックボーンにおける問題点

我々の調査では、バックボーンには比較的問題が少ないが、IPv4 と比較するとまだまだ改善すべき点があることが確認できた。また、バックボーンの場合、ボトルネックとなっている部分を改善することで大きな効果が期待できる。

- ピアリング不足、パス不足

  IPv6 対応の IX はまだ数が少なく、ピアリングも十分になされていない。また、構成機器の制約などで IPv6 で利用できるパスも限られているため、IPv6 での通信は IPv4 に比べて迂回する場合が多い。

品質問題に関しては、問題の把握が容易ではないため、ネットワーク運用者も問題に気づいていないことが多い。我々の調査結果をわかりやすく運用者にフィードバックして、IPv6 ネットワークの品質改善に繋げていくことが課題である。

第 7 章　ファイアウォール

現在の IPv4 インターネットにおいて、一般的に組織のネットワークをインターネットに接続する際、

組織内部のネットワークを組織外の攻撃から防御するために、組織内部ネットワークとインターネットの境界にファイアウォールとよばれる装置を設置する。一般的に組織内部からインターネットへの通信は可能であるが、ファイアウォールによる制限のためインターネットから組織内部へは特定の通信のみが通過できる。このため、インターネットに公開するサーバは、インターネットからアクセス可能なネットワークに配置する。

セキュリティに関して慎重な組織では、インターネット上で公開しているサーバに対してもファイアウォールを配置している。さらに、公開サーバをICMPv4 を用いた DoS 攻撃から防御することを想定して ICMPv4 パケットを通過させないよう設定している場合がある。このため、IPv6 によるインターネット接続においても同様に DoS 攻撃を想定して、ICMPv6 パケットを遮断するよう設定している可能性がある。また製品や実装によっては、デフォルトの設定が ICMPv6 パケットを通過させないようになっている可能性がある。IPv6 ネットワークにおいて ICMPv6 を遮断すると、IPv6 の通信制御に問題が発生する可能性が存在する。

また、通常よりも大きい DNS パケットを通過しないファイアウォールが存在する。この種の DNS パケットの典型例は EDNS0[315] である。これらのファイアウォールの問題から、IPv6 の通信制御が阻害され、IPv6 で適切に通信ができなくなっている。

7.1 ファイアウォールに起因する問題点

遮断されると問題となる ICMPv6 のパケットのタイプとして、次のものが存在する。

- Destination Unreachable（Type = 1）
- Packet Too Big（Type = 2）
- Time Exceeded（Type = 3）

上記の ICMPv6 のタイプのパケットが遮断されると、それぞれ以下の現象が発生して通信上の問題になる。

- Destination Unreachable

  実際の通信の終点アドレスまで到達したのか、通常のパケットが途中で破棄されたのか、経路が無いのか、もしくは通信相手のトランスポートのポートが無いのかが分からない。このため、TCP で通信する場合に、前述の原因でパケットが不到達となっても、早期に対処できず、送信元のアプリケーションは TCP のタイムアウトなどを待ってから処理することになる（5.1 節参照）。
- Packet Too Big

  送信パケットが経路 MTU より大きい場合、この Packet Too Big の ICMPv6 パケットが返ってこないと、経路 MTU を検知できない。そのため、IPv6 では途中の通信路上でフラグメント処理を実施しないので最終的に通信できない。
- Time Exceeded

  送信するパケットの Hop Limit よりも通信相手までのホップ数が多い場合に、Time Exceeded の ICMPv6 パケットが返ってこないと、送信パケットの Hop Limit を調整する手段が失われて通信ができない場合が存在する。たとえば、traceroute などのツールは、最小の Hop Limit をパケットに設定して送信するためまったく機能しなくなる。
- EDNS0

  ICMPv6 パケットではないが、EDNS0 などの通常の DNS パケットよりもサイズの大きなパケットを通過させない場合に、IPv6 における DNS により名前が解決できないという問題も存在する。これは、BIND 9.3 実装を用いた（FreeBSD-5.3R などの）OS であれば、“edns-udp-size” オプションを設定することで回避可能ではある。しかし、これは対症療法であって、本質的な解決のためにはファイアウォールを修正しなければならない。

7.2 ファイアウォール検査ツール

前節の問題点を調査するために、ファイアウォールの ICMPv6 とサイズの長い DNS パケットの遮断問題を検査するツールを作成し、実際の IPv6 インターネット上で動作させて、問題がどのネットワークにおいて存在するのかを調査することになった。また、製品個別の問題点の調査のために、作成したファイアウォール検査ツールを公開することとなった。

本検査ツールは、ファイアウォールの ICMPv6 パケット通過の検査を目的として、前節で述べた ICMPv6 パケットが返るようなパケットを意図的に生成し、ファイアウォールを通過するように送信し、通信相手もしくはファイアウォールの ICMPv6 パケット応答を検査する。本ツールの仕様は、次のとおりである。

- Destination Unreachable

  このタイプの ICMPv6 パケットには、port unreachable と destination unreachable という 2 つのサブタイプが存在し、これらを別々の手法で検査する。サブタイプの port unreachable を検査する場合には、ファイアウォールの先に存在するホストにて使用していないポート番号をパケットに設定して送信する。その後、ファイアウォールか、その先にあるホストからの ICMPv6 パケット応答を検査する。また、サブタイプの destination unreachable を検査する場合には、ファイアウォールの先の存在するアドレスプレフィクスで実際には存在しないアドレスを送信するパケットの終点アドレスに設定して送信する。その後、ファイアウォールもしくは、その先にあるホストからの ICMPv6 パケット応答を検査する。

- Packet Too Big

  検査対象のファイアウォールを通過する通信路上の経路 MTU よりサイズの大きいパケットを、ファイアウォールの先にあるホスト宛に送信する。その後、ファイアウォールもしくは、その先にあるホストからの ICMPv6 パケット応答を検査する。

- Time Exceeded

  検査対象のファイアウォールの先にあるホストまでのホップ数より小さい Hop Limit を送信するパケットに設定して送信する。その後、ファイアウォールもしくは、その先にあるホストからの ICMPv6 パケット応答を検査する。

- EDNS0

  模擬的な DNS Query をファイアウォールの先にある DNS サーバに送信し、DNS サーバもしくは、ファイアウォールからの応答を検査する。

7.3 今後の活動

現在、ファイアウォール検査ツールを作成中であり、完成後に一般に公開したいと考えている。また、本検査ツールは FreeBSD 上に実装しているが、一般の方に広く使用していただくために Windows へ移植したいと考えている。さらに、ファイアウォールベンダの方々に使用して頂いて、その結果を報告していただき、結果をまとめて公開していきたいと考えている。また、公開して使用してもらうだけではなく、自身で独自に IPv6 インターネットにおけるファイアウォールの調査を進めていく。