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 導入環境を改善する IPv4/IPv6 混在通信における DNS 名前解決処理方式

北村 浩† 阿多 信吾‡ 村田 正幸¶

† 日本電気株式会社／電気通信大学 〒211-8666 神奈川県川崎市中原区下沼部 1753
‡ 大阪市立大学大学院工学研究科 〒558-8585 大阪府大阪市住吉区杉本 3-3-138
¶ 大阪大学大学院情報科学研究科 〒565-0871 大阪府吹田市山田丘 1-5
E-mail: † kitamura@da.jp.nec.com, ‡ ata@info.eng.osaka-cu.ac.jp, ¶ murata@ist.osaka-u.ac.jp

あらまし IPv4 アドレスの枯渇を契機に、IPv6 の導入が加速され、一つの通信ノードに IPv4 及び IPv6 の複数のアドレスが設定される環境における、IPv4/IPv6 混在通信が一般的になることが予想される。ドメイン名（ホスト名）と IP アドレスの対応付けを行っている DNS の観点からすると、一つのドメイン名に対して A(IPv4)レコード及び AAAA(IPv6)レコードの複数のレコードが対応付けられるのが一般的になる。ドメイン名から IP アドレスを取得する DNS 名前解決処理（正引き）では、一つのドメイン名から、A 及び AAAA の種類の異なる複数のレコード（IP アドレス）情報を取得する必要がある。これを行う現状 DNS 名前解決処理では非効率で、多くの問題を発生する方式が採用されている。また、一般ユーザにとっては、IPv4 及び IPv6 アドレスの違いは強く意識する必要がなく、IP アドレスの種類に依存しないドメイン名情報から IP アドレスを取得する DNS 名前解決処理は、IPv6 導入を容易に行えるかを左右する重要な処理でもある。本論文では、IPv6 導入における重要な機能である IPv4/IPv6 混在通信環境における DNS 名前解決処理の課題を明確にすると共に、効率的に行える処理方式に関して議論する。

キーワード IPv4/IPv6 混在環境、DNS 名前解決、IPv6 導入環境改善、通信アーキテクチャ

Simplified DNS Query Methods under IPv4/IPv6 Mixed Environment

Hiroshi KITAMURA† Shingo ATA‡ and Masayuki MURATA¶

† NEC Corporation / University of Electro-Communications
1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, 211-8666 Japan
‡ Osaka City University 3-3-138 Sugimoto, Sumiyoshi-ku, Osaka-shi, Osaka, 558-8585 Japan
¶ Osaka University 1-5 Yamadaoka, Suita-shi, Osaka, 565-0871 Japan
E-mail: † kitamura@da.jp.nec.com, ‡ ata@info.eng.osaka-cu.ac.jp, ¶ murata@ist.osaka-u.ac.jp

Abstract With a trigger of the IPv4 address exhaustion, it is naturally supposed that the IPv6 introductions will be accelerate to various network environments. Also, it will become general that both IPv4 and IPv6 addresses are set to one communication node and IPv4 and IPv6 mixed communications become popular. From a viewpoint of the DNS, it becomes general that both A record (for IPv4) and AAAA record (for IPv6) are associated with one domain name. In regular DNS resolving that obtain IP address(es) with a domain name argument, it is needed to achieve to obtain heterogeneous IP addresses (IPv4 and IPv6) (heterogeneous records (A and AAAA)) with a domain name argument. With the current DNS resolving mechanism, it is possible to achieve above features. However, it is inefficient and problematic. Also, for general end users, it is not necessary to be strongly conscious of the difference of IPv4 and IPv6 addresses. Therefore, this DNS resolving becomes a significant operation that judges the IPv6 introduction to the network is done smoothly or not. This paper clarifies problems of the current DNS resolving mechanism under IPv4/IPv6 mixed environment that is significant operation for the IPv6 introduction and discusses efficient and improved DNS resolving mechanisms.

Keyword IPv4/IPv6 mixed environment, DNS resolving, IPv6 introduction improvement, Communication architecture

1. はじめに

IPv4 アドレスの枯渇を契機に、様々な通信環境への IPv6 導入が加速されることが予測され、IPv4 と IPv6 が混在する通信環境が一般的になると考えられる。そういった環境では、一つの通信ノードに対し IPv4 及び IPv6 両方のアドレスが設定されることになる。つまり、種類の異なる複数アドレスが設定されることになる。この状態をドメイン名（ホスト名）と IP アドレ

スの対応付けを行っている DNS[1][2]への情報登録の観点から見てみると、ノードを識別する一つのドメイン名に対して A レコード情報（IPv4 アドレスに対応）及び AAAA レコード情報（IPv6 アドレスに対応）の両方が対応つけられて登録されることになる。

このような IPv4 と IPv6 が混在する環境は、複数の種類の情報が混在する環境であり、従来の IPv4 一種類だけで構成されていた環境で用いられてきた技術や機能だけでは対応できずに、新しい技術や機能の研究開発が進んできた[3][4]。現状 DNS 名前解決処理でも新しい機能を取り込み、IPv4/IPv6 混在環境へ対応してきた[7]。しかし、その対応は十分ではなく非効率で、多くの問題が発生する方式になっている。このことは IPv6 導入の阻害要因にもなりかねない問題であり、この視点でも改善が求められている。

また、一般のユーザはドメイン名情報を頼りに通信相手を指定して通信するのが通常の通信方法であり、IPv4 及び IPv6 アドレスの違いは強く意識する必要がない。IPv4/IPv6 混在環境にあっても、ドメイン名情報はアドレスの種類の違いに関係のない唯一の情報であり、ドメイン名情報を介在して異なる種類のアドレスに関連付けられているともいえる。こういった特性は IPv6 の円滑な導入に大いに役立つ。IP アドレスの種類に依存しないドメイン名情報から、IP アドレスを取得する DNS 名前解決処理は、IPv6 の導入を容易に行えるかどうかを左右する重要な処理でもある。

本論文では IPv4/IPv6 混在通信環境における現状の DNS 名前解決処理の課題を明確にすると共に、IPv6 の導入環境を改善することができる、効率的な新しい DNS 名前解決処理方式に関して議論を行う。

2. 現状の DNS 名前解決処理方式と課題の分析

現状の DNS 名前解決がどのように行われているかを、従来の IPv4 単独の環境から IPv4/IPv6 が混在した環境へ段階的に述べ、現状の課題の分析を行う。

2.1. 従来の IPv4 単独環境での単純な DNS 名前解決方式

図 1に IPv4 だけの一種類のアドレスのみが存在した環境における、従来の名前解決処理の方式を示す。

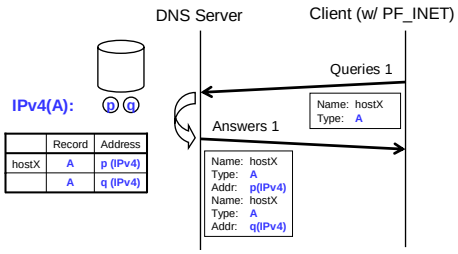

図 1　IPv4 単独環境における DNS 名前解決

一つのドメイン名 hostX に対して、二つの IPv4 アドレス(p,q) が 二つの A レコードのエントリとして登録されているという典型的な状態における DNS 名前解決処理について示している。

極めて自然な方法である。最小のセットである Query/Answer の一対メッセージだけで二つのエントリ情報が同時に一度で取得されている。

2.2. 現状の IPv4/IPv6 混在環境における DNS 名前解決処理方式

次に、現状の IPv4/IPv6 混在環境における DNS 名前解決処理に関して示す。図 2に今後の議論の元になる IPv4/IPv6 混在環境における DNS への情報登録の典型的な状態を示す。

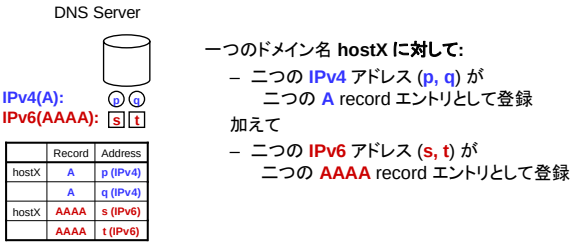

図 2　IPv4/IPv6 混在環境での DNS への情報登録の典型的な状態

当然ではあるが、この状態では A レコード（IPv4 に対応）と AAAA レコード（IPv6 に対応）という異なる 2 種類のレコードが存在する。具体的な名前解決処理の詳細については後述するが、一種類の IPv4 だけのものと比べると複雑な処理になっている。複雑になっている最大の理由は、IPv6 の黎明期である初期段階における状況や方針に強く依存すると考えられる。

現状の IPv4/IPv6 混在環境における DNS 名前解決処理をどのように行うかの仕様を策定した際の要件として、以下のことが挙げられていた。

1. 既に実装があり動作している IPv4 だけの DNS 名前解決処理に影響を与えないこと。
（つまり IPv4 用の DNS 名前解決処理パケットとは独立したパケットを用いて動作すること）
2. DNS Server 側が IPv6 のことを知らず、IPv6 に対応していない環境でも正しく動作すること。
3. 一つのドメイン名に対して、A レコードと AAAA レコード両方の情報が取得できる場合は、AAAA レコード情報が優先されて使われること。
（それならないと、後発の IPv6 が選ばれて通信することがなくなるから)

これらの要件を考慮して、IPv4/IPv6 混在環境における現状の DNS 名前解決処理は A レコード用と AAAA レコード用にそれぞれ独立した、2 対の Query/Answer メッセージによって実現されている。

2 対のメッセージによって実現されるために、そのメッセージの発行の順番及び発行方法が直列か並列かにより、表 1に示すように 4 種類の型が知られている。

表 1　名前解決メッセージ発行の型

Type 名	1st Query	2nd Query	Serial / Parallel	知られている実装など
4-6 Serial	for A record	for AAAA record	Serial	Windows Vista/7 FreeBSD
6-4 Serial	for AAAA record	for A record	Serial	Windows XP RFC4472 shows:
4-6 Parallel	for A record	for AAAA record	Parallel	Some Linux
6-4 Parallel	for AAAA record	for A record	Parallel	

図 3はAレコード取得のためのメッセージを最初に発行し、その取得が終わったらAAAAレコード取得のためのメッセージを発行するタイプの処理方式で、Windows Vista/7 及び FreeBSD などで採用されている。現在最も広く用いられ、発生する問題も少ない処理方式であると考えられている。

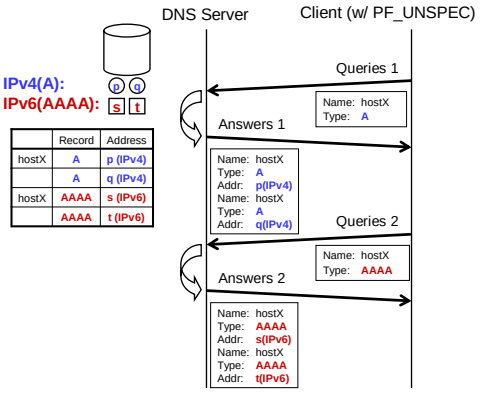

図 3　4-6 (A first) Serial Type

図 4は AAAA レコード取得のためのメッセージを先に発行するタイプのもので RFC4472[7] にこの方式の仕様の記述が存在したり、Windows XP などで採用されていたりする方式である。

この方式は、AAAA レコードが優先されるべきという要件を考慮し、メッセージの発行順番としても先に行うようにしたもので、その当時の考えでは素直な実装であったのだが、現在ではこの方式は問題ある方式であるという理解が進んでおり、推奨される方式になっていない。

この方法が推奨されていない理由は DNS サーバ側が IPv6 に対応していない環境であった場合に、先行して行う AAAA レコード取得のためのメッセージのところで停滞してしまい、その後に行う A レコードが取得できなかったり、できたとしても長い時間がかかったりするという問題が発生するからである。

AAAA レコードが優先されるべきという要件は、メッセージの発行順番として実現するのではなく、getaddrinfo() などの API をユーザアプリケーションが呼んだ際の戻り値の順位として実現されればよい。図 3に示すような 4-6 Serial Type などの現状の API の実装ではその処理がなされており、API 内部では、AAAA レコードの順位を上げるための適切な情報の順位制御処理がなされ、実際のメッセージの発行順番とは違う順位に並び変えたものを戻り値にしている。

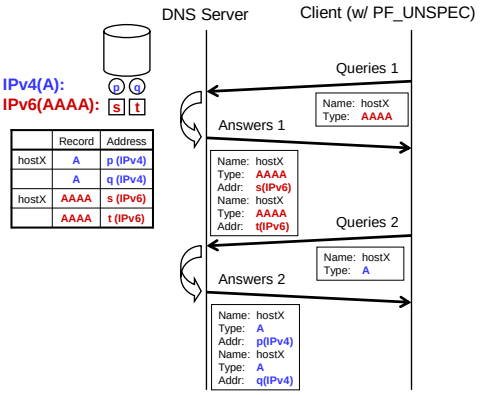

図 4　6-4 (AAAA first) Serial Type

図 5はAレコード取得用のメッセージが先行、図 6はその逆で AAAA レコード取得用のメッセージが先になったもので、A レコード取得用のメッセージとは並列に発行されているものである。並行して発行されているので、どちらのメッセージを先行して処理するかではあまり差のない方式である。これらは Linux（glibc の版が 2.10 以上）などで採用されている実装で、これらの方式もそれなりに広く普及している方式になっている。

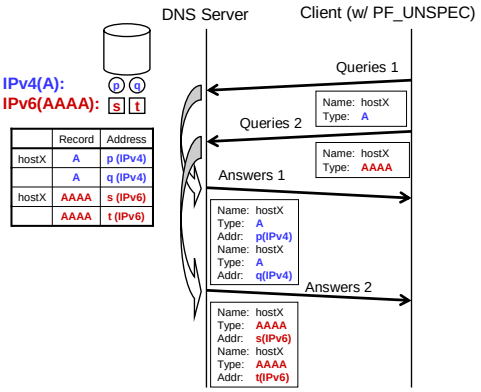

図 5　4-6 (A first) Parallel Type

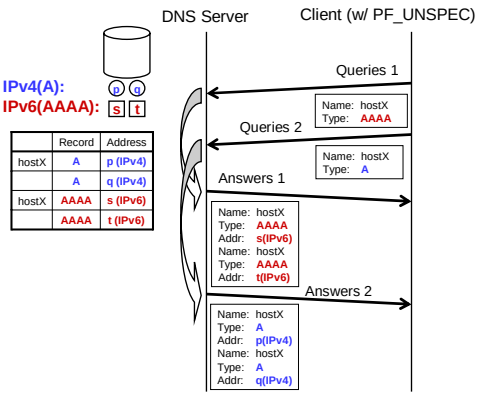

図 6　6-4 (AAAA first) Parallel Type

2.3. 現状の 2 対のメッセージ方式のよる DNS 名前解決処理方式の課題

ここまで示してきたように、IPv4/IPv6 混在環境における、現状の DNS 名前解決処理に採用されている方式は、ユーザアプリケーションが呼び出す API は 1 つなにのも関わらず、その実体は分割された 2 対の Query/Answer のメッセージを使った方式として実現されている。歴史的な経緯を考えるとこの方式を採用したことも理解できないわけではないが、多くの課題を抱えている問題のある方式だといえる。それら課題の特性をまとめてみると以下のように整理される。

1. 複雑である
   明らかに単純な処理手順になっていない。
   2 対であるため、仮に一方のメッセージが通信の中で失われた場合、2 対の間で連携した処理をする必要があるかなどを考えると、その失われ方とその回復をどうやって行うかなどの組み合わせは非常に多様で複雑になる。
2. 非効率である
   1 対のメッセージで実現できることが最も効率が高い単純な処理方式であることは明確である。従って、2 対のメッセージを使うということを採用する段階で既に非効率な方式を選んでいる。
3. 処理を完了し結果を得るまでに時間がかかる
   2 対のメッセージを直列で処理する場合は特にそうであるが、一つ目のメッセージの処理結果を待ってから二つ目のメッセージの処理を行うことになるので、1 対のメッセージで処理する場合に比べれば明らかに長い時間がかかる。
4. ２倍の量のトランザクションになる
   1 対のメッセージに比べれば 2 対のメッセージは２倍の量のトランザクションになるのは当然である。絶対的なトラフィック量の観点からみれば、ネットワーク上には大量のデータ転送処理などがあるだろうから、名前解決処理に要するメッセージが産むトラフィック量は無視できるようなものなのかもないが、トランザクション量の観点としては確実に２倍になる。

3. 新しい DNS 名前解決処理方式の提案

上述した課題の整理を受ければ、問題が発生しにくく効率の高い、新しい DNS 名前解決処理方式はどのようなものにすべきかは明白である。(2 対メッセージに分割して実現するのではなく)1 対の Query/Answer のメッセージだけで、1 つのドメイン名に対応付けられている全て(IPv4 及び IPv6)のアドレス情報（A 及び AAAA レコード）を同時に一度で取得できる方法を用いればよい。

現状の DNS 名前解決処理方式が選択された IPv6 の黎明期の当時と現在の状況を比べると、以下に述べる点で違いがあると考えられる。

1. DNS サーバ側の IPv6 対応が進んだ
   現在では、多くの DNS サーバは IPv6 に対応しており、非対応の DNS サーバの数はわずかであると想像され、IPv6 非対応 DNS サーバのことを重視して対応する必要性が下がった。
2. DNS 名前解決処理を発行するクライアント側の IPv6 対応も進んだ
   広く使われているメジャーなクライアント OS の IPv6 化が進み、ほとんどのクライアントが IPv6 に対応した DNS 名前解決処理を行うことができる。
3. 非効率で問題を起こす名前解決処理が無視できなくなってきている
   IPv6 に対応したノードの数が少ない時代は、このことは、影響の少ないマイナーな問題として軽視されていたが、現在の状況では、無視できない状況になってきている。
   更には、こういた状況を放置することは、IPv6 の導入促進をあるいは、今後のインターネットの健全は発展を阻害する要因になるかもしれないと思われている。

こういった状況の違いは、1 対の Query/Answer のメッセージだけで DNS 名前解決処理を行う方式に切り替えることを技術的にも可能にするとともに、世の中の流れとしてもそれが求められている。

IPv4/IPv6 混在環境において、1 対の Query/Answer のメッセージだけで DNS 名前解決処理を行う新しい方式として、大きく分けて 以下に述べる 3 種類の方式を挙げることができる。

1. Two Existing Records Combined 型
2. One New Special Record 型
3. One Existing Record w/ Mapped Transformation 型

以下にこれらの方式の詳細な機能を述べると共にその特徴を分析する。

3.1. Two Existing Records Combined 型

この方式は、図 7に示すように既存の 2 つレコード型（A と AAAA）を一緒にまとめて 1 つの Query に設定する方式である。

長所:

この方式は、既に仕様が決まっている既存のレコードタイプを用いており、その点において機能拡張が不要である。

また、DNS 名前解決のためのパケットフォーマットとしても、プロトコルの設計段階で汎用的な対応である複数のレコードを一緒にまとめて設定するというこ

とが可能なフォーマットを採用している。

すぐにではなく、長期的な視点で物事を見たらということになるが、この方式は理想的にはどうすべきかを求めた方式であるともいえる。

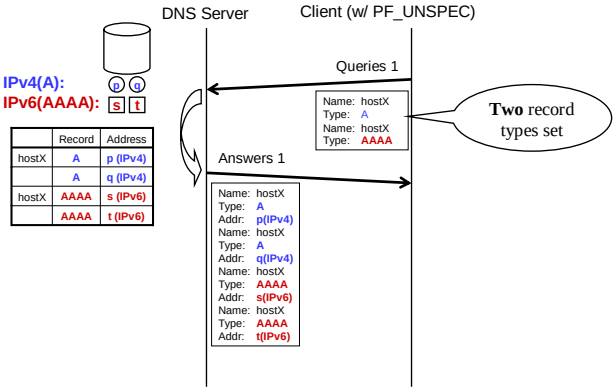

図 7 Two Existing Records Combined 型

短所:

この方式の一番大きな短所はクライアント側の名前解決機能をこの新しい方式に対応したものに更新する必要がある点である。現状のクライアント側の名前解決機能は既に多くのクライアントノードに実装済みのものであるので、それを更新するというのは容易なことではない。

また、DNS パケットフォーマットの設計思想として、複数のレコードを一緒にまとめて設定することが可能になっているといえども、現状 1 Query に複数のレコードが設定されているような例は知られておらず、1 Query には 1 レコードしか設定しないというのが実質的な決まりごとになっており、論理的に可能だけど実績となる前例はないという状態である。

3.2. One New Special Record 型

この方式は、**図 8**に示すように既存にはない新しい特別なレコード型（例えば、AAAA+A）を導入し 1 つの Query に設定する方式である。

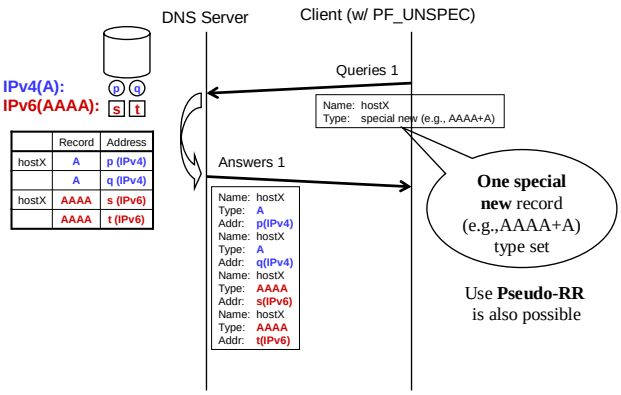

図 8 One New Special Record 型

長所:

この方式は、1 Query には 1 レコードしか設定しないという実質的な決まりごとに従っており、この視点においては導入が容易である。上述した Two Existing Records Combined 型に比べると、この点が改善されていると見ることができる。

長期的な視点で理想的にはどうすべきかを求めた場合と同じであると考えることができる。

短所:

現状の仕様にない新しいレコード型を導入しなければならないので、その点は短所になる。

この方式の一番大きな短所は上述した Two Existing Records Combined 型と同様に、クライアント側の名前解決機能をこの新しい方式に対応したものに更新する必要がある点である。これは実用的な普及を考えた場合に越えなくてはならない大きな問題になる。

3.3. One Existing Record with Mapped Transformation 型

図 9に示すように既存の 1 つのレコード型を 1 つの Query に設定することに加え、IPv4 mapped IPv6 address の特性をうまく利用した変換を加える方式である。この方式は、今までの考え方を大きく変える、独創的な方式である。

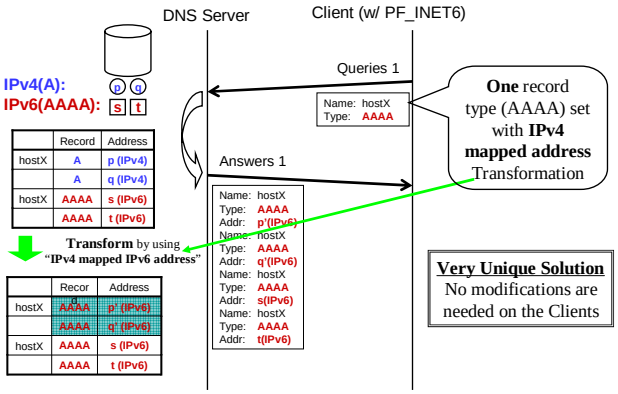

図 9 One Existing Record with Mapped Transformation 型

IPv6 のアドレス空間は極めて広大で IPv4 アドレスは内部に用意に収容することができる。この発想を実現しているのが IPv4 mapped IPv6 address [6]である。噛み砕いて述べると、IPv4 アドレスは（アドレスの上位部分に::ffff に付加することで）IPv6 アドレスの形式で表現することができるということを意味する。つまり、DNS には IPv4 アドレスは A レコードとして登録するという先入観的概念を大きく変えて、IPv4 アドレスは IPv4 mapped IPv6 アドレスと考えて AAAA レコードとして登録しているのと同様な状態として処理する方式である

クライアントの発する Query には 既存の AAAA レコード型を一つ設定して、通常の IPv6 アドレスを AAAA レコードとして登録してある情報に加えて、IPv4 アドレスも IPv4 mapped IPv6 アドレスの AAAA レコードとして取得する方式で、いわば全てのアドレスは IPv6 アドレスであり AAAA レコードがありさえすれば事足りるという方式である。

IPv4 アドレスを IPv4 mapped IPv6 アドレスの AAAA レコードとして DNS に登録することに関しては、最初から変換した AAAA レコードとして登録する静的な処理でも良いし、普通に A レコードで登録しておいて問い合わせがある度に変換する動的な処理でもよい。動的処理の方を選択すれば、登録状態の視点では現状と全く同じになる。

この方式を用いてクライアント アプリケーションが IPv4 mapped IPv6 アドレスを入手した際、元の実体が IPv4 アドレスであったとしても、（AAAA レコードを引数として問い合わせているのだから）情報の扱いとしても当然 IPv6 アドレスのとして扱われることになる。つまり、クライアント アプリケーションの中では、一貫して IPv6 アドレスとして扱われる。そのため、アプリケーションが IPv6 を扱えるものであれば、この方法を用いる場合でも問題を起こすようなことはない。

実体であった IPv4 アドレスに戻して扱われるのはアプリケーションより下位の層であるカーネル内にある IP 層での処理の部分であり、エンドユーザやアプリケーションは下位の IP 層でこの処理がなされていることを特に意識する必要ない。

長所:

既存のレコード型 AAAA を使うため、新しいレコード型を導入する必要がない。Query に設定するのも AAAA の一つのレコード型を設定するだけでよいので、1 Query には 1 レコードしか設定しないという実質的な決まりごとにも従っている。

この方式の一番大きな長所は上述した二つの方式と異なり、クライアント側の名前解決機能をこの新しい方式に対応したものに更新する必要がない点にある。

短所:

いささかトリッキーかなと思われる点を除けば基本的には問題はない方式である。IPv4 mapped IPv6 アドレスという機能の実装に依っているのでこの機能が有効でないクライアントではうまく動作しないことになる。（BSD 系の OS の実装では、この機能自体の実装はなされていても、パラメータ設定のほうで有効にしておらず、default では 利用できない OS がある。）

クライアント側の機能を更新しなくて良いため、サーバ側でやるべきことがあるわけで、その視点では短所といえなくもないが、新しい機能を導入するためにどこかの修正が必要であり、それが膨大な数で多様な実装があるクライアント側でなくサーバ側で実現できることは、短所と呼ぶより長所なのかもしれない。

3.4. 新しい DNS 名前解決処理方式のまとめ

長期的な視点での理想的な DNS 名前解決処理方式を選ぶとするのであれば、Two Existing Records Combined 型及び One New Special Record 型は いずれもその有力な候補になる良い方式ではないかと思われる。しかし、クライアント側の名前解決処理実装をこの新しい方式に対応したものに更新する必要があり、既存の膨大な数のクライアントの実装を更新するのは、実行上の視点では大きな問題になってしまう。

この視点からすると、クライアント実装の変更が不要な One Existing Record with Mapped Transformation 型は、この独創的な概念を理解するには時間がかかるかもしれないが、致命的に不都合になることはなく、実行性の高い有力な方式であると考えることができる。

このあたりは IETF における標準化の議論などと連係して自ずと答えが出てくるのではと考えられる。

4. まとめ

IPv4/IPv6 混在通信環境において重要な役割を果たす処理あり、場合によっては今後加速することが見込まれる IPv6 の導入を阻害してしまう要因になる可能性ある、現状の DNS 名前解決処理方式の課題を明確にした。また、それらの課題を克服し IPv6 の導入環境を改善することができる、効率的な新たな DNS 名前解決処理方式を提案した。

本方式の実装は、DNS サーバの実装を更新するという自然で単純な方法で比較的容易に実現することができるが、それ以外にも可能な実装方法が存在する。そのような実装方式などの検討も含め、今後とも機能強化や検証などの研究活動を継続していく予定である。

文　献

[1] P. Mockapetris, "Domain names - concepts and facilities", RFC 1034, November 1987

[2] P. Mockapetris, "Domain names - implementation and specification", RFC 1035, November 1987

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

[4] S. Thomson, C. Huitema, V. Ksinant and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003

[5] M-K. Shin, et al., "Application Aspects of IPv6 Transition", RFC 4038, March 2005

[6] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006

[7] A. Durand, et al., "Operational Considerations and Issues with IPv6 DNS", RFC 4472, April 2006