~IPv6とDNSの正しい付き合い方~

IPv6時代のDNS設定を考える

2009.3.5

IPv6 Operations Forum @ Shinagawa

(株)クララオンライン

白畑　真

自己紹介など

- 白畑　真
  - ホスティング事業者にてネットワークとサーバ周りの雑用など
- DNSとの関わり
  - お客様のドメイン名をホスティング
    - オーソリテイティブサーバとして動作しているサーバが多い
    - リカーシブサーバも提供していますが、台数はごく少数
  - DNSサーバはお客様サーバ(*1)上にインストール、もしくは当社サーバ上で提供

    *1:管理者権限はお客様。自由にゾーンの内容を編集可能
- ご注意
  - 本発表の内容は(株)クララオンラインの公式見解ではなく、個人的な見解です。
  - クララオンラインは現時点でIPv6サービスを提供しておりません。

今日の目的:

IPv4/IPv6共存環境でのDNSの姿について議論する

IPv6環境でのDNS

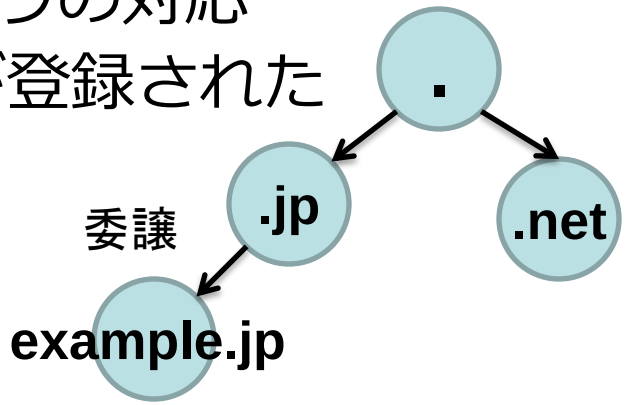
背景

1. リソースレコードのIPv6対応
   - AAAAレコードが記述できること
     - BIND, djbdns, NSD, Microsoft DNSなどほぼ全てのDNSサーバが対応
     - ASPサービスを利用している場合には事業者の対応次第
   - ip6.arpaレコードの逆引きが設定できること
   - DNS権威サーバ自体への到達性はIPv4のみでも良い
2. **EDNS0 (Extension Mechanisms for DNS) 対応**
   - もともとのDNSパケットのデータ長は最大512バイトであるため、512バイト以上のデータ長に対応するための拡張
     - IPv6リソースレコードと直接関係はないが、IPv6などサイズの大きなデータを格納する際に役立つ
     - 互換性の問題から、Root DNS/ccTLD DNSでは現時点では未利用
3. **DNS権威サーバのトランスポートのIPv6対応**
   - IPv6経由でDNSクエリに応答する
     - BIND, NSD, Microsoft DNSなどが対応 (標準のdjbdnsは非対応)

DNSとIPv6トランスポート

背景

- ドメイン名のレジストリ/レジストラの対応
- Root DNSにIPv6グルーレコードが登録された
  - . (root) (2008年2月4日)
  - .jp / .kr (2004年7月20日)



- ドメイン名のレジストラの対応
  - IPv6のホスト登録ができるか
    - グルーレコードとしてAAAAレコードを登録できるか
  - JPRS (.jp), Verisign GRS(.com, .net) など主要レジストリは対応済み
- 対応レジストラー覧:
  - FAQ : DNS : Which DNS Registrars allow me to add AAAA glue for my Domain Name Servers?
    http://www.sixxs.net/faq/dns/?faq=ipv6glue
  - network/IPv6/IPv6対応のレジストラー覧 - Tomocha WikiPlus
    http://wiki.tomocha.net/ipv6_registrar.html

Root DNSからIPv6トランスポートだけで
名前解決ができる環境が整ってきた

DNSサーバの設計

背景

- RFC3901: DNS IPv6 Transport Operational Guidelines
  - **IPv4/IPv6の両トランスポートでDNSデータを持つ**
  - [リカーシブサーバ] 全てのIPv4再帰ネームサーバはIPv4のみか、デュアルスタックであるべき
  - [オーソリテイティブサーバ] 全てのDNSゾーンは、最低でも1つのIPv4到達性のある権威サーバによって提供されるべき
- RFC4472: Operational Considerations and Issues with IPv6 DNS
  - 1.3 名前空間の分断を避ける
  - トランスポートがIPv4でもIPv6でも、同じリソースレコードを提供
    - IPv6での到達性しかない権威サーバのみでゾーン情報を提供した場合、IPv4インターネットからは参照できない
    - AAAAレコードを設定する場合には、本当にIPv6でサービスが利用可能な状態になっていることを確認

とにかく**IPv4トランスポートだけでも名前解決**ができることが重要

議論

論点1

IPv6向けサービスとIPv4向けサービスで名前を分けるべきか

名前の付け方(1/3)

1. **IPv4向けとIPv6向けで別の名前を使う**
   - 例: Google
     - www.google.com (Aレコードのみ)
       - 66.249.89.99
       - 66.249.89.104
       - 66.249.89.147
     - ipv6.google.com (AAAAレコードのみ*1)
       - 2001:4860:c003::68
2. **IPv4向けとIPv6向けで同じ名前を使う**
   - 例: KAME Project
     - www.kame.net
       - 203.178.141.194 (Aレコード)
       - 2001:200:0:8002:203:47ff:fea5:3085 (AAAAレコード)

*1: Google over IPv6プログラム参加ネットワークの場合には同じ名前を利用する。(事前に登録してあるDNSリゾルバからのwww.google.com の問い合わせの場合、AレコードだけではなくAAAAレコードも応答する)
http://www.ripe.net/ripe/meetings/ripe-57/presentations/Colitti-A_strategy_for_IPv6_adoption.Z8ri.pdf

名前の付け方(2/3)

IPv4とIPv6で別の名前を使う

- メリット:
  - レコードの設定の柔軟性が増す
  - **IPv6の導入に伴う諸処の問題回避**
    - AAAA RRを正しく処理できないDNSリカーシブサーバ
    - IPv6トランスポートがない環境でIPv6接続を試行した場合、タイムアウト待ちが発生するクライアント
  - **問題切り分けが容易**
    - IPv6サービスでのみ発生した障害など
- デメリット:
  - 透過性
    - ユーザがIPv4/IPv6を意識してホスト名を使い分ける必要がある
    - 明示的な設定が必要

名前の付け方(3/3)

IPv4とIPv6で同じ名前を使う

- メリット:
  - 透過性:

  ユーザがIPv4/IPv6を意識することなくサービスを利用できる
- デメリット:
  - 一部のユーザ環境から名前解決に失敗・遅延する恐れ
    - DNS検索過程において壊れたDNSサーバが存在するリスク
      - EDNS0を通さないFirewall、AAAAクエリに返事をしないDNSサーバ…
    - IPv4/IPv6で同一のサービスが提供されている場合、通信品質が低いプロトコル(現状ではIPv6)が選択される可能性がある
      - 例:太平洋をまたいだトンネル接続
    - AレコートとAAAAレコードの問い合わせ順序によっては、遅延が発生する場合がある
    - IPv6の閉域網とIPv4インターネットにアクセスできる環境では、マルチプレフィックス問題の影響を受ける可能性がある

JANOG23でいただいた情報

デュアルスタックのSMTP サーバを運用していたら、ある特定のIPv4 Onlyのサーバからメールが届かなかったりしたことがある。これは設定次第で MX レコードの設定で回避することが可能であるが不思議な現象である。

(JPRS 民田さん)

```
$ORIGIN example.jp.
;
@       IN      MX      10 mail
                MX      20 mail4
;
mail            A       192.0.2.1
                AAAA    2001:db8::1
;
mail4           A       192.0.2.1
```

MXの先にv4/v6デュアルスタックのメールサーバを設定する場合は、MX群にIPv4だけのサーバを残すのが安全である。

主張1

(少なくとも当面の間)IPv4向けサービスとIPv6向けサービスで名前を分けるべき

論点2

IPv6で逆引きを設定すべきか

IPv4の状況をよく考えてみると...

- 日本や欧米のISPでは逆引きを設定しているケースが多い(?)
- アジアのISPでは逆引きをしないケースが多いように感じる

逆引きの意義とは

DNS逆引きの用途(1/2)
アクセス認証

- tcp_wrappers(*1)など
  - 二重逆引き(double reverse lookup):
    IPアドレス→ホスト名→IPアドレス
  - IPアドレスの逆引き結果と正引き結果を照合
    - IPアドレスを逆引き
      (3.9.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.0.d.2.0.0.0.0.0.0.0.2.6.2.ip6.arpa.→www.iana.org.)
    - 得られたホスト名を正引き
      (www.iana.org → 2620:0:2d0:1::193)

  *1: -DPARANOID オプション付き(デフォルト)でコンパイルされた場合
- パスワードや電子証明書など他の認証方式の代替とはならないが、ドメイン名とIPアドレスを紐づける上で一定の効果はあるのではないか
  - セキュリティ向上にならないという意見もあり
    http://homepages.tesco.net/J.deBoynePollard/FGA/dns-avoid-double-reverse.html

複数のIPアドレスの一元管理

- 複数のアドレスブロックを持っている組織がアクセス制御を実装する場合
  - IPアドレスプレフィックスベース
    - プレフィックス変更毎に追加・削除が必要
  - ドメイン名ベース
    - アクセス制御の設定自体は不要
    - IPアドレスブロックの追加・削除時には、DNSの正引き・逆引きエントリを更新

Anonymityとユーザ認証

- ユーザの粒度
  - IPv4時代: 動的IPアドレスが前提。接続ごとに割り当てられるIPアドレス空間もばらばら

    →ドメイン名単位でのアクセス制限の例

    (例: Apacheの.htaccess, tcp_wrappersの/etc/hosts.allow等)

.htaccess

Order Deny,Allow
Deny from all
Allow from **exmaple.jp**

/etc/hosts.allow

sshd: .**example.jp**

  - IPv6時代: /48(準)固定IPアドレス
    - ユーザの識別が容易に

DNS逆引きの用途(2/2)

- ネットワーク情報の参照
  - アクセスログ等の解析に利用
    - プロバイダ名や国情報など
    - あくまで参考情報なので、逆引き結果と正引き結果が一致している必要まではない
  - Geolocation技術(IPアドレス等によるアクセス元判定技術)の発展
    - DNS逆引きに対する依存度は下がりつつあるように見受けられる

サーバとクライアントの関係

- 逆引きが設定されていない場合、ホストの識別が容易ではない
  - 直感的にわかりにくい
  - そもそもIPv6は128bitもあるのでアドレスを覚えられない
- サーバにもクライアントの側面がある
  - 例: SMTPサーバ
  - 他のSMTPサーバから見ればSMTPクライアント
- クライアントもサーバも逆引きを設定すべき

主張2

IPv6もIPv4同様に逆引きを設定すべきだ

どのように逆引きを実装するか

逆引きの実装方法(1/2)

1. DNSエントリの自動生成
   - BINDの$GENERATEディレクティブ(ライクな)に近いが単純に/64空間に適用するのは困難。
   - 一定の規則で逆引き/正引きエントリを自動生成できる仕組みが必要(オンザフライ?)
2. ワイルドカードレコードの利用

2001:db8:ff00::/48の逆引きの例:

```
*.0.0.f.0.8.b.d.0.1.0.0.2.ip6.arpa.      IN
      PTR   site.example.com.
```

   - 正引きと一致しない。そもそも正引きのレコードと逆引きのレコードの値は一致させるべきか

逆引きの実装方法(2/2)

3. Dynamic DNS
   - DHCPv6等と連携し、動的に正引きと逆引きを登録することも考えられる
   - 膨大なアドレス空間の中で、必要に応じてリソースレコードを作成
     - IPv6の莫大なアドレス空間全てに対してエントリを作成する必要はない

まとめ

- IPv4+IPv6どちらでも同じ内容のレコード
- NGN+IPv6でIPアドレスの意味が変わる
  - 動的IPアドレスから(準)固定IPアドレスへ
  - IPv4ネットワークとIPv6ネットワークが等価であるとは限らない
- DNSの逆引きに何を期待するのか
  - IPアドレスとドメイン名所有者の紐付け
  - 正引きと逆引きの一致(?)
  - 実装方法
    - アドレス自動生成
    - ワイルドカード
    - Dynamic DNS

ご静聴ありがとうございました