IPv6トラブルシューティング
家庭ネットワーク・サーバ編

2014/11/19

NTTソフトウェア株式会社
高宮紀明

本資料に記載の会社名、製品名は、それぞれの会社の商標もしくは登録商標です。

自己紹介

- 1999 年よりIPv6 にかかわり始める。
- 第一回TAHI プロジェクト相互接続試験に参加
- USAGI プロジェクト参加（～2008）
- IPv6 普及・高度化推進協議会参加
  - アプリケーションのIPv6対応検討SWG に参加
  - InternetWeek2013
    （Asterisk のIPv6 対応について）
  - IPv6 Summit in SAPPORO 2014
    （IPv6 アプリケーションサービスの作り方）
- 業務：NGN対応ソフトウェア開発・販売（2008～）

agenda

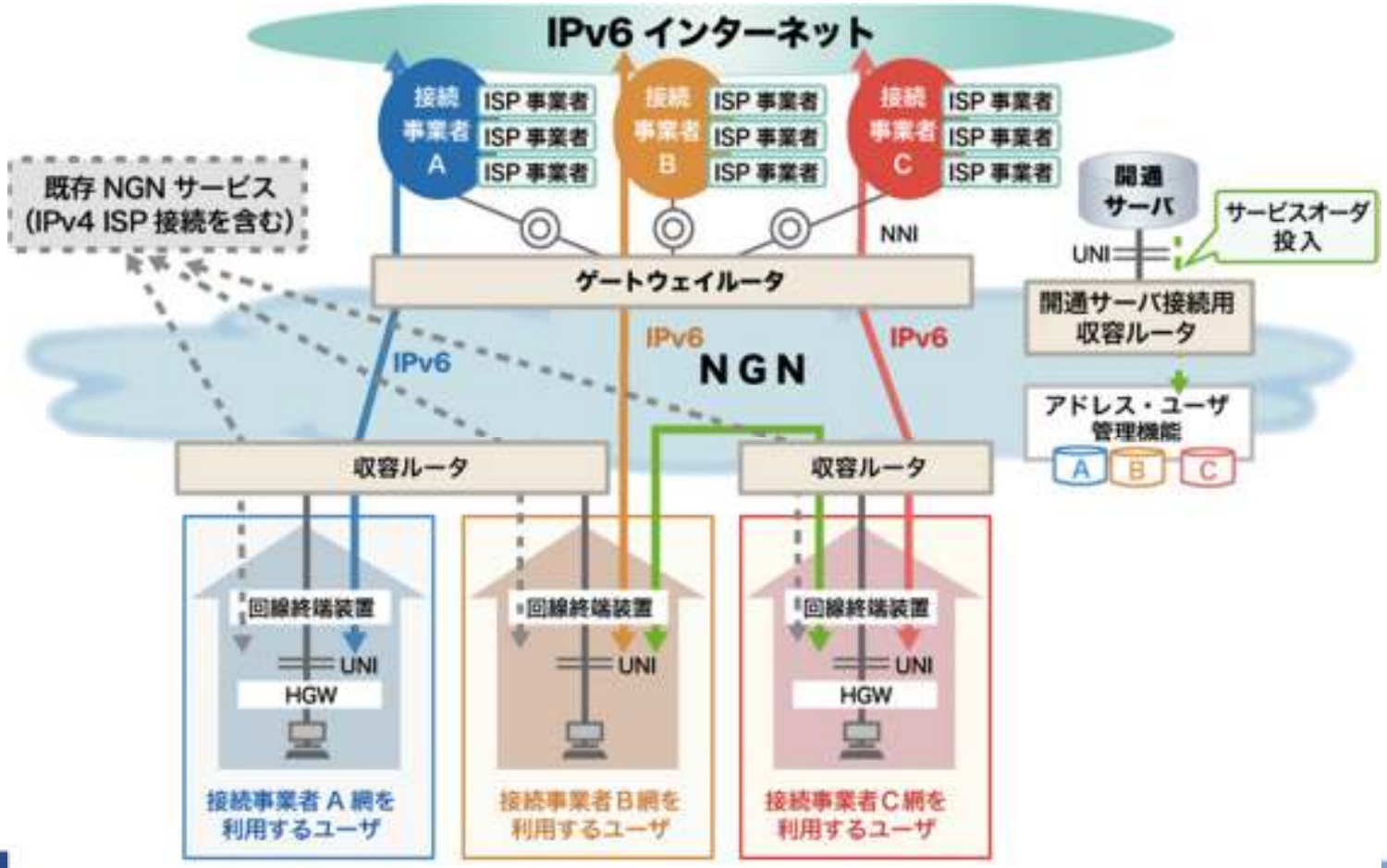
- 家庭ネットワークの IPv6 接続について
- サーバの公開
- デュアルスタックでの課題
- ネットワークの課題
- その他の課題

家庭ネットワークの IPv6 接続について

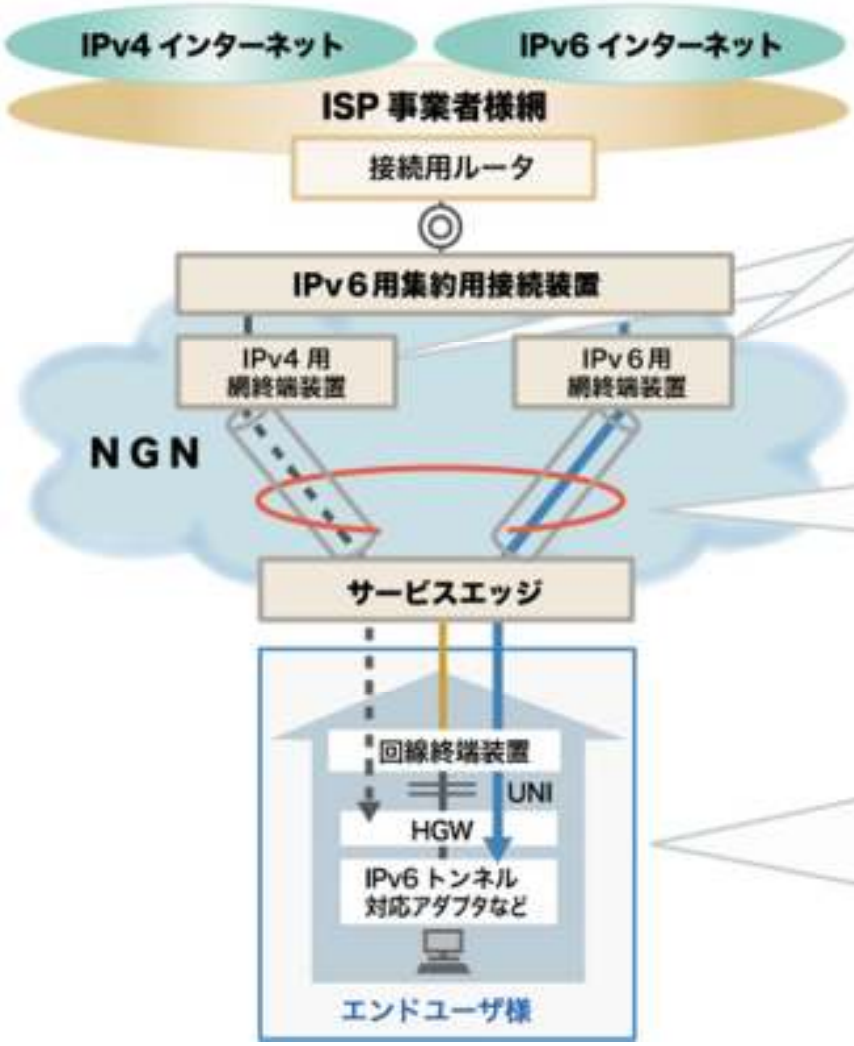
- 接続方法
  - いまや、IPv6 サービスを提供しているプロバイダは多く、つなぎ方にはいろいろな方法があります。
  - つなぎ方は大別して2つ。
    - IPoE 方式
      - キャリア回線の IP 接続をつかってそのまま IPv6 にアクセスする方式。
    - トンネル方式
      - PPPoE や IPv6 over IPv4 トンネルを使用して IPv6 ネットワークにアクセスする方式。
  - Windows Vista 以降では、Teredo による端末から IPv6 接続環境を自動的に（勝手に？）作ることも可能

IPoE による接続方式

- 現在は 16 社までが接続事業者として運用可能
  - http://www.ntt-east.co.jp/release/detail/20120926_01.html



トンネル接続



- IPv6 用集約用接続装置は IPv4 用網終端装置も収容可能。
- IPv4 用網終端装置と IPv6 用網終端装置は物理的に異なる。

- IPv4 と IPv6 で異なる ISP 識別子（サブドメインを利用した設定も含む*）を使いトンネルを確立し、トンネルごとにセッションを利用する。

*サブドメインの設定要否は ISP 事業者様で選可能です。

- トンネル方式を利用する全てのエンドユーザ様は IPv6 用網終端装置へトンネルを確立する装置（**IPv6 トンネル対応アダプタ**）などが必要。
- トンネル方式を確立する装置等から IPv6 用網終端装置へ PPPoE(IPv6) を確立した後、エンドユーザ様の端末（またはネットワーク）から、NTT東日本・NTT西日本が提供するフレッツ光ネクストを経由し、それぞれを契約している接続事業者網に接続する。

IPv6 サービス提供プロバイダ

- JAIPA（一般社団法人日本インターネットプロバイダー協会）に、各プロバイダの対応がまとめられています。
  - http://www.jaipa.or.jp/ipv6/index.html
  - プロバイダによっては無料
  - トピック
    - v6 プラスサービスの開始（@nifty、BIGLOBE 等）
      - http://buffalo.jp/taiou/kisyu/network/ipv6/
  - NTT東西からも情報があります。
    - http://flets.com/next/ipv6/ （NTT東日本）
    - http://flets-w.com/isp/ipv6/ （NTT西日本）

Windows のデフォルト設定について

- Windows はデフォルトで Teredo が使用可能
- DNS で IPv6 アドレスが引けた場合、アクセスが遅い場合がある
  - Microsoft の Teredo サーバの停止
    - https://isc.sans.edu/diary/Microsoft+Teredo+Server+%22Sunset%22/16153
    - http://en.wikipedia.org/wiki/Teredo_tunneling
      - As of IETF88, Microsoft plans to deactivate their Teredo servers for Windows clients in the first half of 2014 (exact date TBD), and encourage the deactivation of publicly operated Teredo relays.
  - Teredo を使う場合は、IPoE あるいはトンネル形式などの IPv6 アクセスがないことになるため、ネットワークの見直しを検討する。
  - 場合によっては、IPv6 機能を停止させることの検討が必要。

アクセス網の IPv6 ネットワークについて

- IPv6 インターネットに接続していなくても、一見 IPv6 アドレスが割り当てられている場合がある。
- 実はアクセス網の閉域網アドレスの場合があるので要注意。
  - 参考：http://www.attn.jp/maz/p/i/policy-table/
    NTT東西のIPv6閉域網向けポリシーテーブル設定
- IPv4/IPv6 のアドレスはポリシテーブルに従っているため、必要に応じてポリシテーブルを変更する。
  - Windows の場合は
    http://www.vwnet.jp/Windows/w7/IPv4/IPv4PriorityUP.html

アクセス網の IPv6 ネットワークについて

- NTT 東西閉域網
  - 2001:c90::/32、2001:d70::/30、2001:a000::/21、2404:1a8::/32、2408::/22
- ポリシテーブル

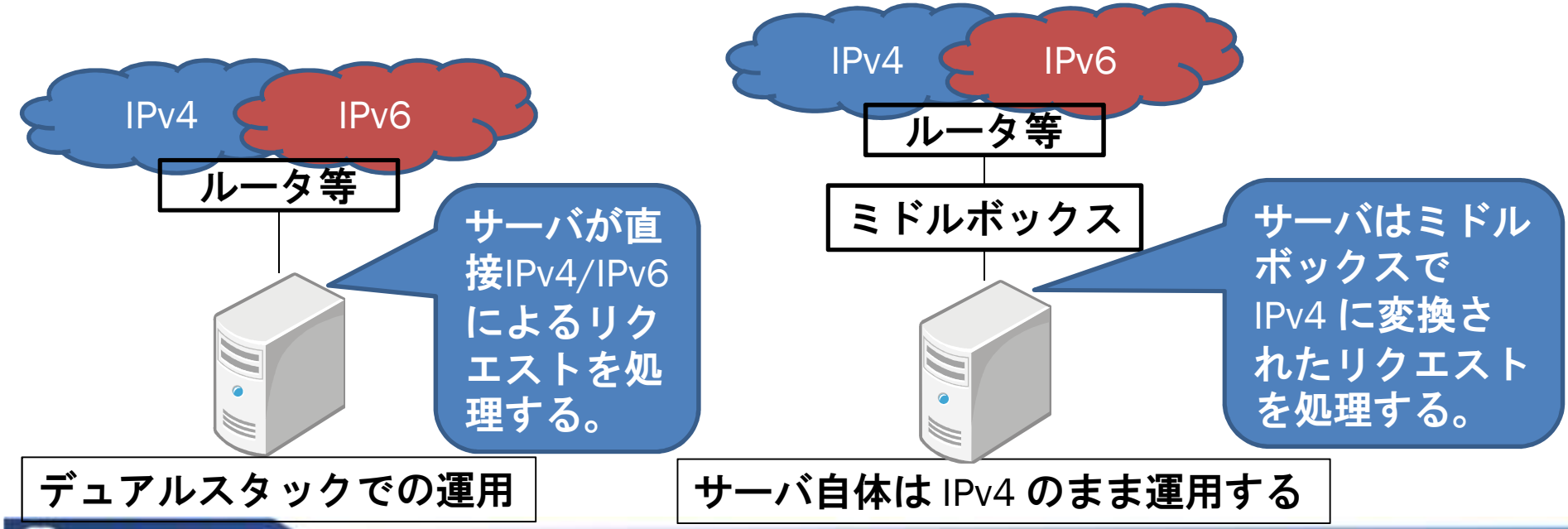
```
C:¥Windows¥System32>netsh interface ipv6 show prefixpolicies
アクティブ状態を照会しています...

優先順位     ラベル    プレフィックス
----------  -----  --------------------------------
        50      0  ::1/128
        40      1  ::/0
        30      2  2002::/16
        20      3  ::/96
        10      4  ::ffff:0:0/96
         5      5  2001::/32
C:¥Windows¥System32>ping localhost

yms-15 [::1]に ping を送信しています 32 バイトのデータ:
::1 からの応答: 時間 <1ms
::1 からの応答: 時間 <1ms
::1 からの応答: 時間 <1ms
::1 からの応答: 時間 <1ms
```

IPv6 ネットワークの構築

- いずれのネットワーク接続を使用している場合でも、いったん手元のルータでセグメントを分けて、家庭内の端末に IPv6 アドレスを割り当てる。
- その時に設定しなければならないのは、主に以下の項目（1点目以外は IPv4 とほぼ同じ）
  - Router Advertisement の設定
  - （DHCPv6 の設定）
  - フィルタリング
  - DNS の設定

サーバの公開について

- IPv6 サーバの公開手段
  - サーバ自体をデュアルスタックで運用する。
  - サーバ自体は IPv4 で運用し、サーバとインターネットの間にトランスレータもしくはリバースプロキシ等（ミドルボックス、RFC3234）を置く。



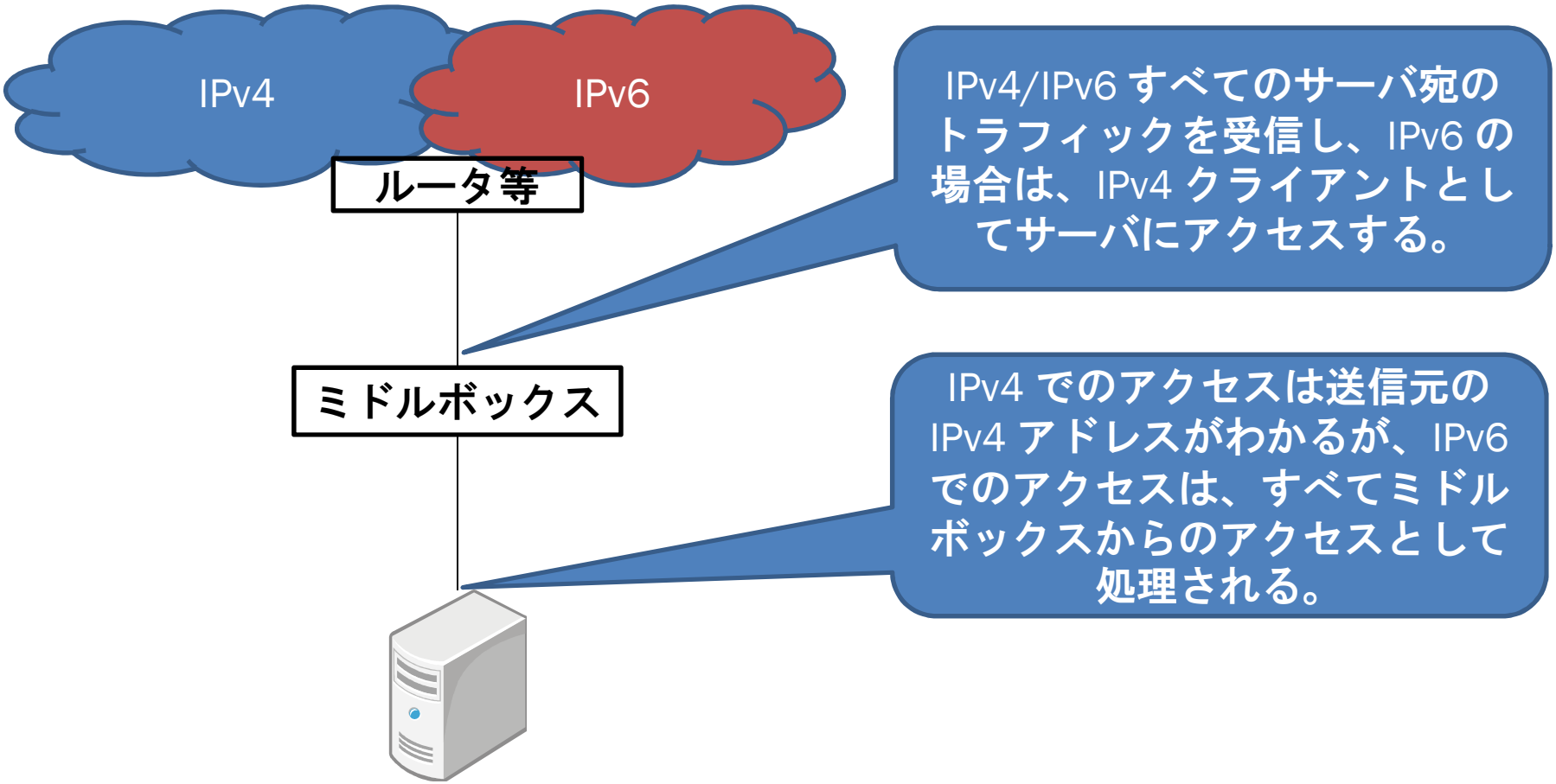
ミドルボックスについて

- IP レベルの変換装置
  - IPv4/IPv6 プロトコルトランスレータ専用機
  - IPv4/IPv6 変換機能付きファイアウォール製品
  - IPv4/IPv6 変換機能付きロードバランサ
- アプリケーションレベルの変換装置（ALG）
  - リバースプロキシ
  - CDN（CDN が IPv4/IPv6 変換装置の役割を果たす）

ミドルボックスの運用

- サーバが IPv6 に対応していない場合はミドルボックス経由での通信が必要となる。
- 公開するサービスが少なければ、リバースプロキシ等のALGを、公開するサービス毎に設置すればよい
- ミドルボックスのメリット
  - サーバ本体は IPv4 のままでよい（製品など、手を入れられないサービスなど）
- ミドルボックスのデメリット
  - ALG の場合、サービス毎に用意する必要がある。
  - サーバの IPv6 アクセス状況が分かりづらい。

例:リバースプロキシの接続環境

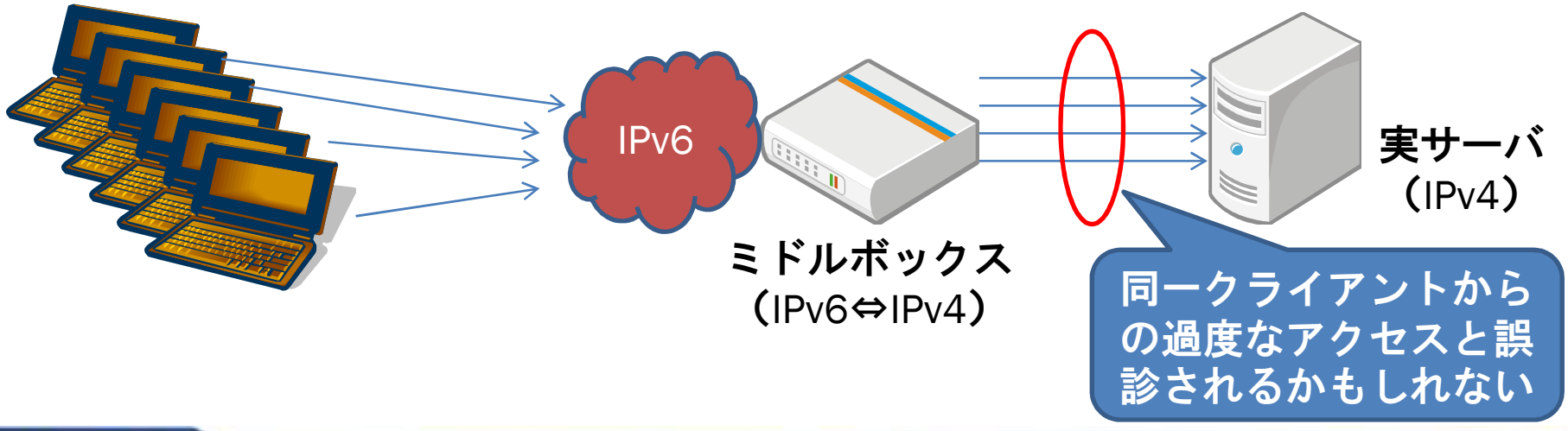


ミドルボックスの課題（1）

- 課題
  - 前述のとおり、IPv6 でインターネットからアクセスがあった場合、ミドルボックスが接続元となる。
- 対策
  - 接続元の IP アドレスがわかるようにする。
    Apache の場合、mod_remoteip を有効にするなど、HTTP ヘッダ経由でクライアントの IPv6 アドレスが取得できるようにする。
  - ログファイルについては突合して解析しやすくするために、
    - ミドルボックスとサーバの時刻を合わせる。
    - ミドルボックスの接続元ポート番号を記録し、突合させやすくする。
  - などの対策を講じる

ミドルボックスの課題（2）

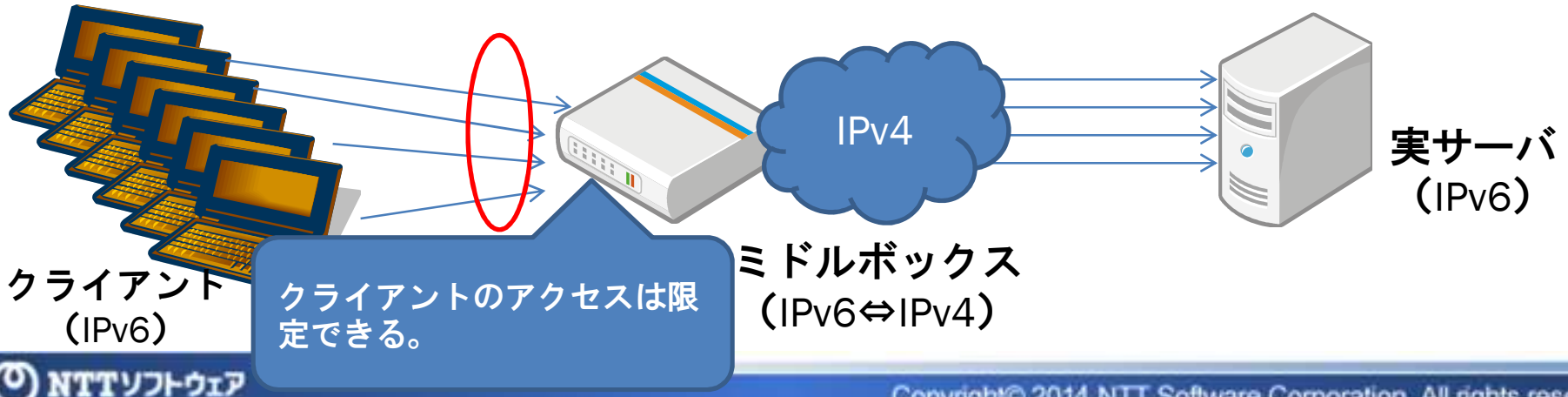
- 課題
  - 実サーバから見るとミドルボックスから大量にアクセスがあるように見え、DoS アタックのように見える。
- 対策
  - アクセス量に応じた、サーバのチューニングを行う（同一クライアント IP アドレスからのアクセス許容量など）
  - ミドルボックスでのログを確実にとる



ミドルボックスの課題(3)

- 課題
  - ミドルボックスが踏み台に使用される可能性がある。
- 対策
  - ミドルボックスの用途に応じて、適切な設定を行う。

ミドルボックスの用途	サービスの適用範囲	ミドルボックスの接続先
サーバの IPv6 化	インターネット全体からのアクセス	対象となるサーバのみ
IPv4 クライアントの IPv6 サーバへのアクセス	自分のネットワーク内の端末のみ	インターネット全体へのアクセス



ミドルボックスの課題(3)

- 踏み台確認例
  - ミドルボックス経由で意図しないサーバへのアクセスがないか、確認する。

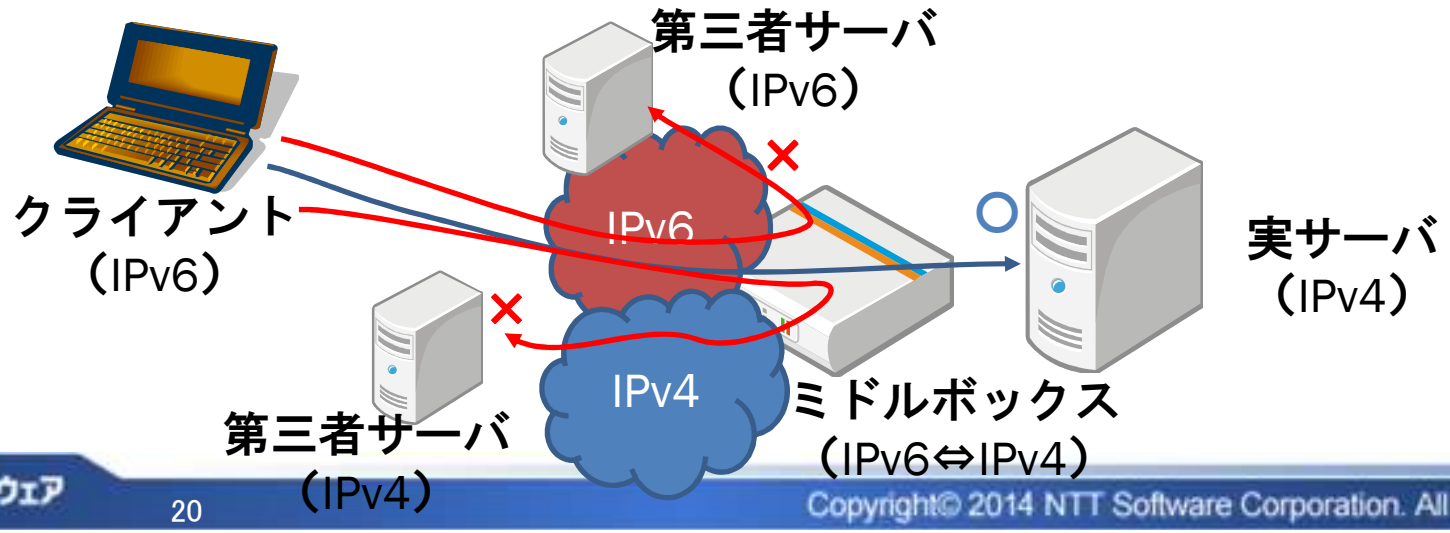
【実行例】

意図しない IPv4 サーバへのアクセスがないか、確認する。

```
% telnet <ミドルボックスの IPv6 アドレス> 80
GET http://<第三者のWebサーバのIPv4アドレス>/ HTTP/1.0
```

意図しない IPv6 サーバへもアクセスがないか、確認する。

```
% telnet <ミドルボックスの IPv6 アドレス> 80
GET http://<第三者のWebサーバのIPv6アドレス>/ HTTP/1.0
```

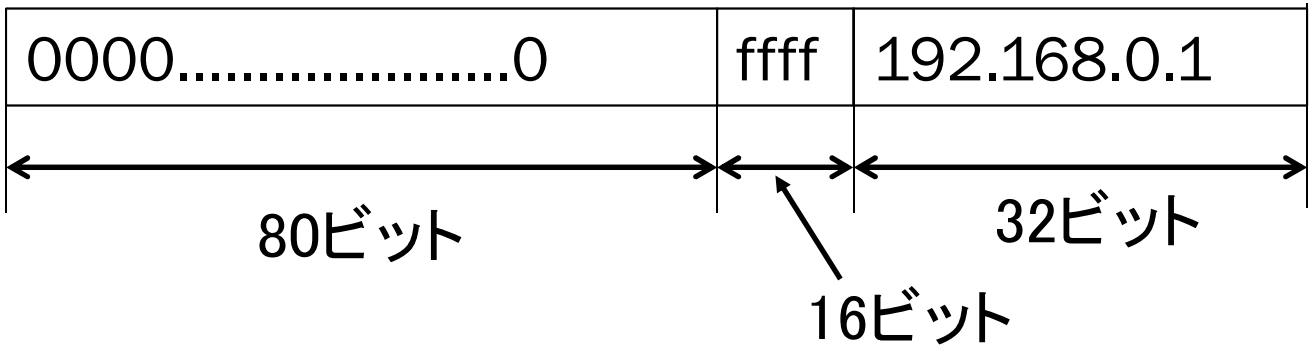


デュアルスタック環境での課題（1）

- 課題
  - サーバを IPv6 対応とすることによる、IPv4 でアクセスしたクライアントの表現が変わってしまう。
- 対策
  - IPv4 でアクセスしてきた端末が、IPv4 射影アドレス（::ffff:0.0.0.0/96）で表現されることが原因であることが多い。
  - 対策としては
    - このまま運用し、射影アドレスで表現されている場合は IPv4 によるアクセスとして扱うようにサーバを修正する。
      （アドレスタイプチェックのマクロとして、IN6_IS_ADDR_V4MAPPED() が /usr/include/netinet/in.h で定義されている
    - IPv4 接続と、IPv6 接続のソケットを分離する。

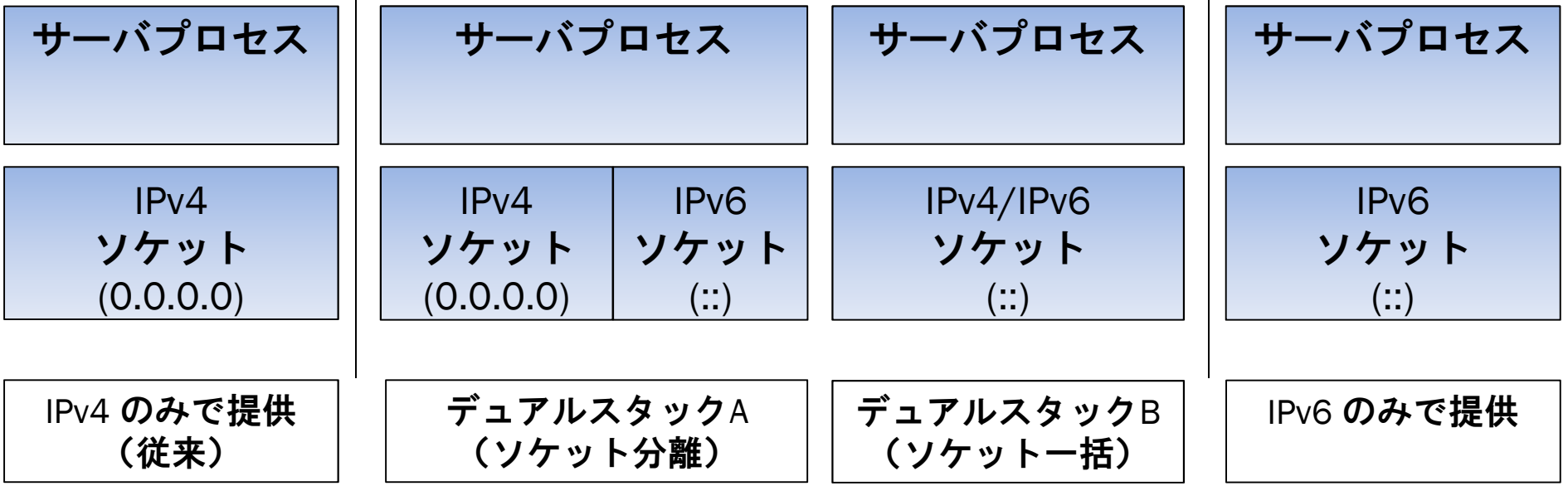
- IPv4 射影アドレス（IPv4 mapped）
  - ::ffff:0.0.0.0/96
  - IPv4 しか対応していないホストと通信するときに、内部的に使用される
  - 送信元・宛先アドレスとしては利用されない

IPv6 射影アドレスフォーマット



デュアルスタック環境での課題(1)

・ デュアルスタック構成について



- デュアルスタック構成については、AとBでどちらがとれるかは OS の実装に依存する。
  - A 案で実装した場合は、IPv4/IPv6 をそれぞれに対応したソケットで処理する。
  - B 案で実装した場合は、IPv4 でアクセスしてきた相手を IPv4 射影アドレスで処理する。
- Linux の場合、ソケットオプションを使用することにより、両方の方式を選択することが可能。

デュアルスタック環境での課題(1)

- ソケットの分離について(詳細)
  - サーバソケットを開く場合、IPv4 のみ、IPv6 のみでオープンするときの挙動を確認すること。
- 例
  - sshd の場合、/etc/ssh/sshd_config を
    - ListenAddress 0.0.0.0 および ListenAddress :: の 2 行を設定する場合、IPv4/IPv6 のソケットをそれぞれ開き、処理を行う(CentOS、Debian でのデフォルト動作)

```
【Debian での sshd のポートの状態(CentOS では、どちらも tcp として表示される】
% netstat -an |grep :22
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
%
```

    - ListenAddress :: のみ記述した場合は、IPv6 のみ処理を行う。
    - ListenAddress 0.0.0.0 のみ記述した場合は、IPv4 のみ処理を行う。

デュアルスタック環境での課題（1）

- IPv4/IPv6 ソケットの分離について
  - システムのデフォルト設定は、
    /proc/sys/net/ipv6/bindv6only
    の値を確認すること
    - 1 のとき、IPv6 のみ bind する（0 の場合、0.0.0.0 で bind されたポート番号は、:: で bind することができない）
  - CentOS{5,6,7}、Debian（Wheezy）はデフォルト 0（::/0 を bind することで、IPv4/IPv6 を処理する）
  - アプリケーションごとに変更する場合は、BSD ソケット関数を使用して変更する。

bindv6only **をアプリ内だけで有効にする例：**

```
int s = socket(AF_INET6, SOCK_STREAM, 0);
int on = 1;
ret = setsockopt(s, IPPROTO_IPV6, IPV6_V6ONLY, &on, sizeof(on));
```

デュアルスタック環境での課題（1）

- それでも残る課題
  draft-itojun-v6ops-v4mapped-harmful-02 (IPv4-Mapped Address API Considered Harmful)
  - 実装の複雑化
    - 多くのOSではIPv4射影アドレスを無効化できる
    - IPv4射影アドレスを無効化した場合、IPv6でのみ動作するようになるアプリケーションも
  - アクセス制御が複雑化
    - IPv4 射影アドレス用の設定が必要になる場合も
      - 射影アドレスのアクセス制御を IPv6 として扱う必要があるものがある。
    - 同一のIPv4ホストとの通信でも、OSやアプリケーションによって見え方が異なる
  - コードの移植性が低下

ネットワークの課題(1)

- 課題
  - 意図しないRA(Router Advertisement)
    - IPv6接続性がないネットワークのはずなのに、グローバルIPv6アドレスがついている
    - IPv6のデフォルトルートが見覚えのないIPv6アドレスに変更されている
- 想定原因
  - 事故や攻撃などの理由で、正規のRA以外のRAを受信したRAには認証がないため、容易に詐称されうる
  - よくあるケース:Windowsの「インターネット接続の共有(ICS) 」が6to4を有効化

ネットワークの課題(1)

- 対策
  - 経路表を確認し、デフォルルートが正しいか確認
  - 不正RAの監視・対応ツールの利用
    - NDPMon – IPv6 Neighbor Discovery Protocol Monitor
      http://ndpmon.sourceforge.net/
    - rafixd(意図しない RA の lifetime を 0 にして、無効化する)
      http://www.kame.net/dev/cvsweb2.cgi/kame/kame/kame/rafixd/
- 参考
  - RFC 6104 Rogue IPv6 Router Advertisement Problem Statement
  - RFC 6105 IPv6 Router Advertisement Guard

ネットワークの課題(1)

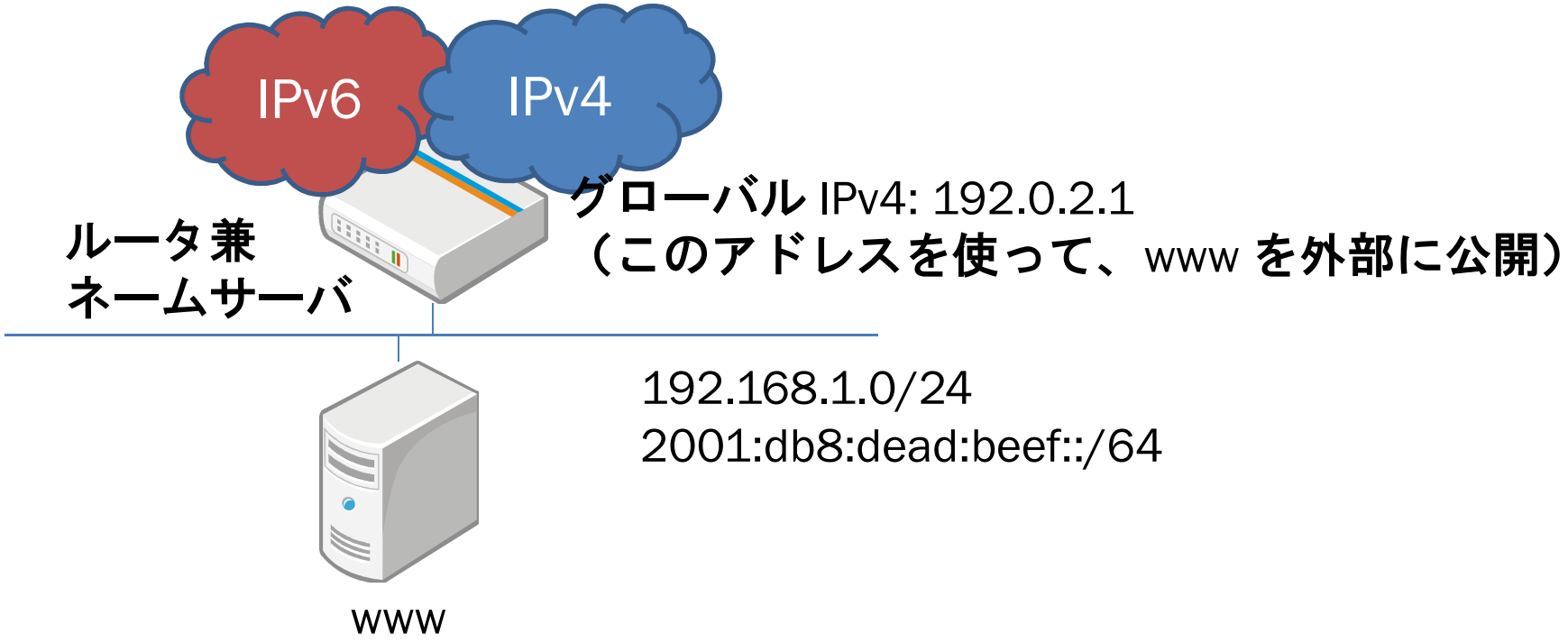
- RA の監視
  - L2 スイッチなどでも監視する。
  - RA の優先度を設定させる(RFC4191 で規定)
    - Linux の場合、radvd で実装済み。
- RA を止めたい場合
  - サーバアドレスであれば RA による自動設定を使わず、静的な割り当てでも十分

ネットワークの課題(2)

- DNS サーバの管理
  - IPv6 アドレスを直接打つのは大変
    →DNS サーバによりホスト名で管理しやすいようにする。
  - とくに、自分でドメインを持っている場合など、
    - IPv6 アドレスは外部から検索されてもよいが、IPv4 アドレスはプライベートなので検索されたくない。
- 対策(BIND9 の場合)
  - DNS の設定ファイル(ゾーンファイル)を公開用とプライベートに分けて運用する。
    - view の運用

ネットワークの課題(2)

- IPv4/IPv6 のデュアルスタック構成
- IPv4 アドレスは一つ(固定)



ネットワークの課題(2)

- view の活用

named.conf（**関連部分のみ抜粋**）

```
acl my-network {
    192.168.1.0/24;
    2001:db8:1::/48;
};
view "internal" {
    match-clients { my-network; };
    :
    zone "mydomain.org" {
        type master;
        file "internal.zone";
        allow-query {
            any;
        };
    }
    :
};
```

```
view "external" {
    match-clients { any; };
    :
    zone "mydomain.org" {
        type master;
        file "internal.zone";
    };
    :
};
```

ネットワークの課題(2)

プライベート用ゾーンファイル（関連部分のみ抜粋）

```
$TTL 86400
@       IN      SOA     ns1.hogehoge.org.  root.ns1.hogehoge.org. (
                        1 ; Serial
                        7200        ; Refresh (2h)
                        3600        ; Retry   (1h)
                        604800      ; Expire  (7d)
                        86400 )     ; Minimum TTL (24h)
                IN      NS      ns1.hogehoge.org.
ns1             IN      A       192.0.2.1
                IN      AAAA    2001:db8:dead:beef::1
www             IN      AAAA    2001:db8:dead:beef::80
                IN      A       192.168.1.2
```

公開用ゾーンファイル（関連部分のみ抜粋）

```
$TTL 86400
@       IN      SOA     ns1.hogehoge.org.  root.ns1.hogehoge.org. (
                        :（上記と同じ）
                        )
                IN      NS      ns1.hogehoge.org.
ns1             IN      A       192.0.2.1
                IN      AAAA    2001:db8:dead:beef::1
www             IN      AAAA    2001:db8:dead:beef::80
                IN      A       192.0.2.1
```

動作確認

```
host1% nslookup
> www.example.org
Server:         192.168.1.1
Address:        192.168.1.1#53

www.example.org   canonical name = host1.example.org.
Name:   host1.example.org
Address: 192.168.1.2
> set type=aaaa
> www.example.org
Server:         192.168.1.1
Address:        192.168.1.1#53

www.example.org   canonical name = host1.example.org.
host1.example.org has AAAA address 2001:db8:dead:beef::1
>
```

DNS サーバはローカルのサーバを使用する。

返す IPv4 アドレスは LAN で有効な IP アドレス

IPv6 アドレスを返却。

動作確認

```
> server 8.8.8.8
Default server: 8.8.8.8
Address: 8.8.8.8#53
> www.example.org
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
www.example.org   has AAAA address 2001:db8:dead:beef::1

Authoritative answers can be found from:
> set type=a
> www.example.org
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   www.example.org
Address: 192.0.2.1
>
```

DNS サーバは外部のサーバを使用する。

IPv6 アドレスは先ほどと同じ。

返す IPv4 アドレスはインターネットで有効な IP アドレス

ネットワークの課題(3)

- 課題
  - 名前解決に時間がかかるようになった
- 想定される原因
  - DNS権威サーバのトランスポートの問題
    - IPv6では応答しないが、IPv4では応答する
  - DNSのペイロードが大きくなった
    - ゾーンファイルにAAAAレコードを記述することで、パケット長が 512 バイト以上に
    - リゾルバやFirewallがEDNS0に対応していない
    - UDPからTCPにフォールバックしている

ネットワークの課題(3)

- 解決策
  - 解決に時間のかかるサーバは hosts ファイルを利用する。
    - Linux 等では /etc/hosts
    - Windows では、C:¥WINDOWS¥system32¥drivers¥etc
    - ただし、localhost(::1)の定義には注意が必要
      - CentOS 系は、127.0.0.1、::1 ともに localhost で参照可能
      - Debian 系は、::1 は ip6-localhost として参照される。
  - DNS サーバの確認
    - DNSのトランスポートの確認
    - EDNS0対応の確認
    - クエリの挙動確認

ネットワークの課題(3)

- DNS サーバのトランスポート確認

```
dig soa example.com @権威サーバのIPv6ドレス
dig soa example.com @権威サーバのIPv4ドレス
```

- EDNS0 の確認

```
dig +bufsize=2048 soa example.com @権威サーバのIPv6かドレス
dig +bufsize=2048 soa example.com @権威サーバのIPv4かドレス
```

```
% dig +bufsize=2048 soa org @2001:500:2::c

; <<>> DiG 9.7.3 <<>> +bufsize=4096 soa org @2001:500:2::c
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32229
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 13
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;org.                           IN      SOA

;; AUTHORITY SECTION:
org.                    172800  IN      NS      a2.org.afilias-nst.info.
:
```

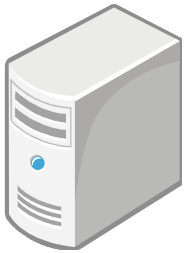
ネットワークの課題(3)

- DNS クエリの挙動確認
  - AAAAクエリを先に実施するOS
    - Windows XP、Linux
  - Aクエリを先に実施するOS
    - Windows Vista、Windows 7、FreeBSD、Mac OS X
- ただし、Linux でも、ディストリビューションによって挙動が変わるので注意が必要。
  - CentOS5 系は AAAA/A の順で問い合わせる（問い合わせ毎にポート番号が異なる）
  - CentOS 6 系は、A/AAAA の順で問い合わせる（問い合わせのポート番号が同じ）

ネットワークの課題(3)

CentOS 5 系

Query AAAA www.example.com
51525/udp → 53/udp



Query Response AAAA www.example.com
51525/udp ← 53/udp

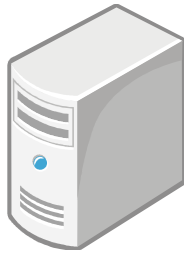
Query A www.example.com
42840/udp → 53/udp

Query Response A www.example.com
42840/udp ← 53/udp

CentOS 6/Debian Wheezy 系

Query A www.example.com
64738/udp → 53/udp

Query AAAA www.example.com
64738/udp → 53/udp



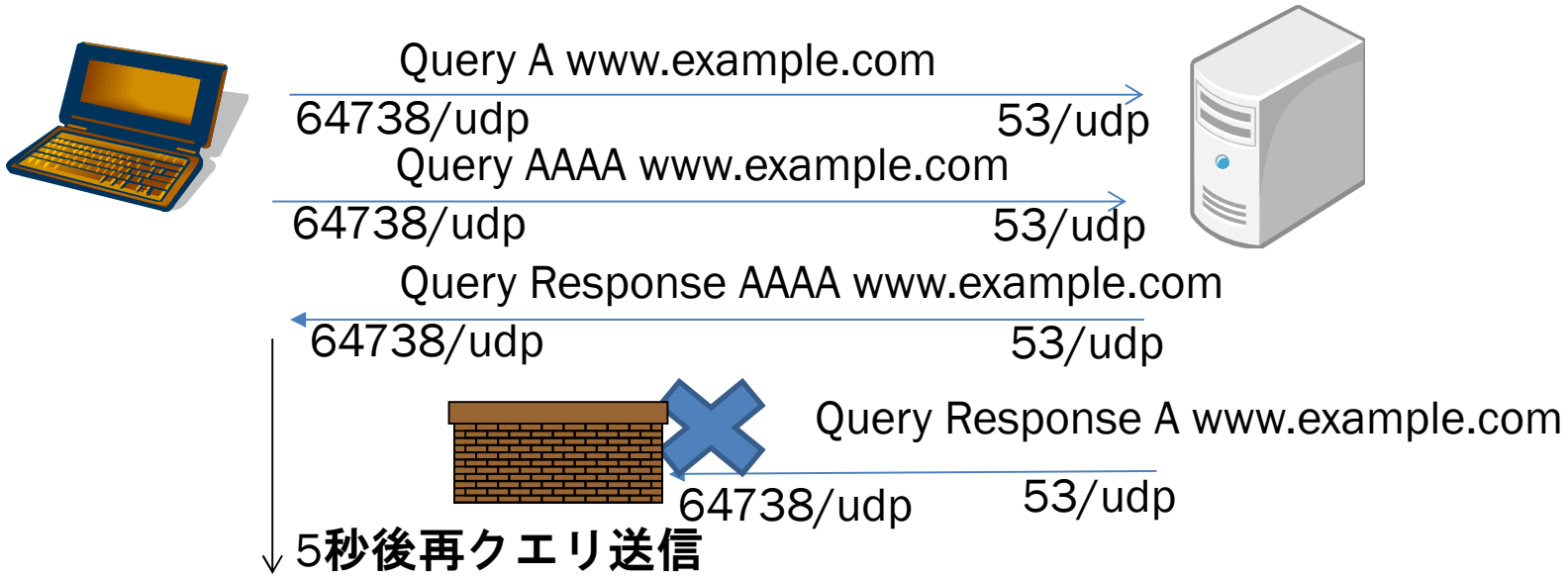
Query Response AAAA www.example.com
64738/udp ← 53/udp

Query Response A www.example.com
64738/udp ← 53/udp

レスポンスの順番は DNS サーバに依存する

ネットワークの課題(3)

- ファイアウォール経由の問い合わせの場合、ファイアウォール製品によっては同一ポートからのクエリを同一のセッションとみなされ、返信が落とされてしまうものがある。

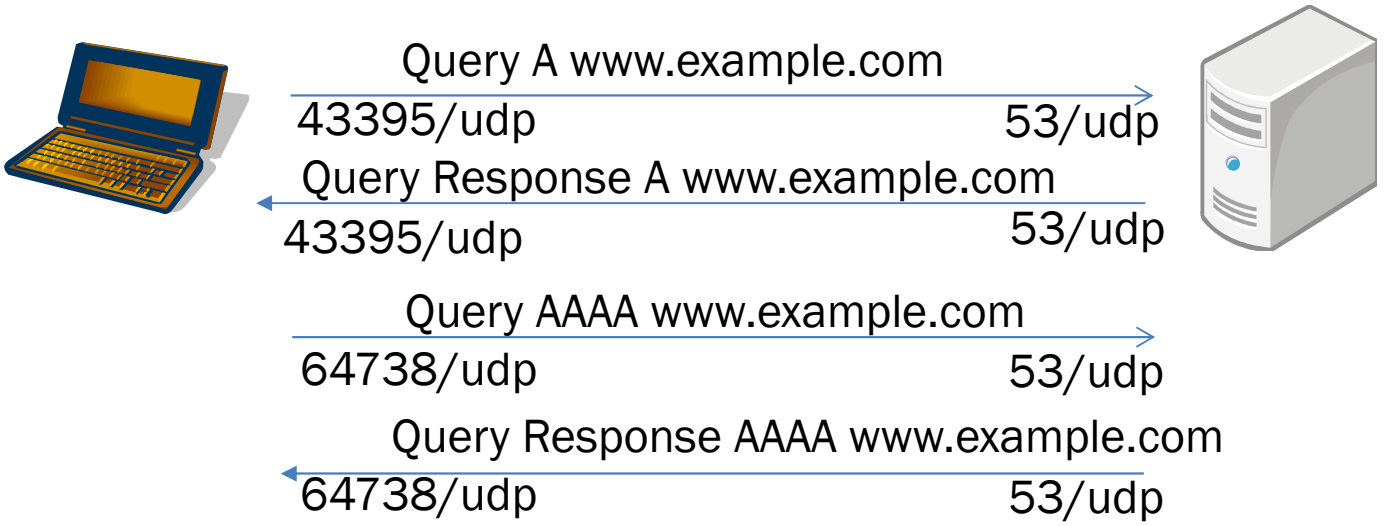


ネットワークの課題(3)

- 対策
  - /etc/resolv.conf を修正する。
    - options timeout:1 を設定する。
      (挙動が同じだが、再送間隔が短くなる)
    - options single-request-reopen を設定する。

resolv.conf **の記述例**

```
nameserver 2001:db8:dead:beef::1
options single-request-reopen
```

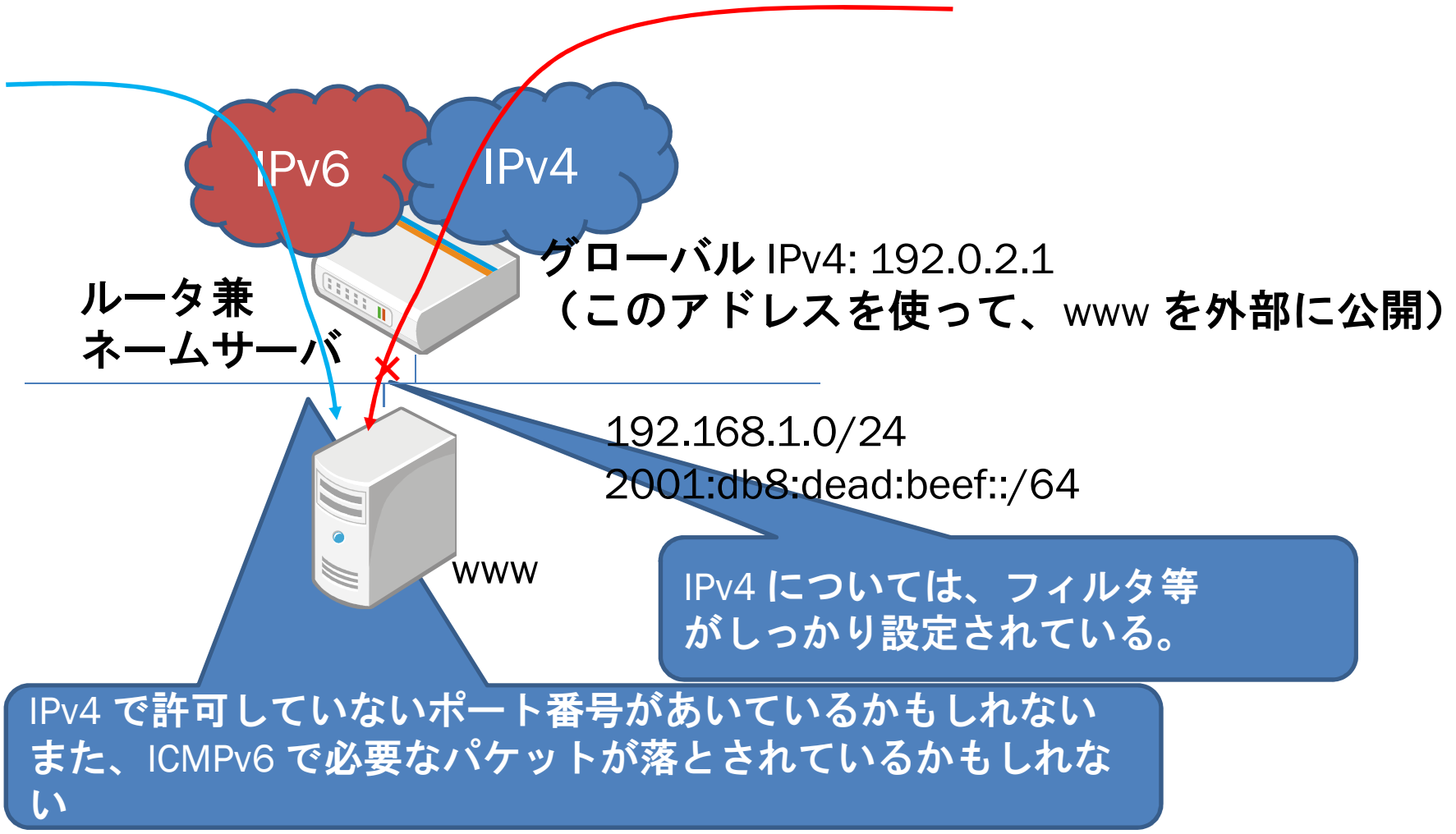


ネットワークの課題(4)

- ICMPv6 の過度なフィルタにより遮断してはならない ICMPv6 メッセージを止めてしまう。
  - Destination Unreachable (Type 1)
    - All codes
  - Packet Too Big (Type 2)
  - Time Exceeded (Type 3)
    - Code 0 only
  - Parameter Problem (Type 4)
    - Codes 1 and 2 only
- ガイドラインとして、RFC4890「Recommendations for Filtering ICMPv6 Messages in Firewalls」を参照
- TCP/UDP においては、IPv4 と同程度のフィルタを忘れずに設定すること。
  - IPv4 はしっかり守られているが、Teredo など端末自らトンネルを張るような場合は抜けとなる場合が多い。
  - VPN ソフトウェアなど、IPv4 しか考慮されていないアプリケーションを使っているときも要注意。

ネットワークの課題(4)

・ 意図しないIPv6 通信



ネットワークの課題(4)

- 課題
  - IPv6 のアクセス制御が不十分で、意図しない通信ができてしまう。
  - 不十分とは？
    - IPv4 と同じレベルのセキュリティになっていない
    - ICMPv6 のうち、落としてはならないタイプのパケットが通信できる形態となっている。
  - 対策
    - アクセス制御はかならずチェックし、IPv4 あるいはサービスの仕様通りになっていることを確認する。
    - ICMPv6 のフィルタ条件については、次ページのタイプを通過させるように修正する。

ネットワークの課題(4)

- アクセス制限の記述パターン
  - サーバ(アプリケーション)により書き方が様々
  - Postfix
    IPv6 アドレスは、[...] で囲む必要がある
    例： mynetworks = 127.0.0.0/8 [::1]/128
  - Delegate
    IPv6 アドレスのコロンは、_ で表記する
    例：-P _:80
  - Apache
    IPv6 アドレスは128 ビットすべて表記する
    例：2001:db8:dead:beef::/64
    （×2001:db8:dead:beef/64）

参考文献

- IPv6 普及・高度化推進協議会
  IPv4/IPv6 共存WG
  - http://www.v6pc.jp/jp/wg/coexistenceWG/index.phtml
- InternetWeek の過去の資料
  - https://www.nic.ad.jp/ja/materials/iw/2010/proceedings/s9/iw2010-s9-02.pdf
  - https://www.nic.ad.jp/ja/materials/iw/2011/proceedings/t2/

参考：発表で表示したプログラム例

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>

main(int argc, char **argv)
{
    int s;
    int ret;
    int on;
    struct sockaddr_in6 serv;
    struct sockaddr_in6 dst;
    int dstlen;
    char v6addr[INET6_ADDRSTRLEN];

    memset(&serv, 0, sizeof(serv));
    serv.sin6_family = AF_INET6;
    serv.sin6_port = htons(8080);

    s = socket(AF_INET6, SOCK_STREAM, 0);

    on = 1;
    ret = setsockopt(s, SOL_SOCKET, SO_REUSEADDR,
                     &on, sizeof(on));

    printf("ret = %d¥n", ret);
```

```
 /* IPv6 のみ Listen したい場合 */
    if (argc > 1) {
        on = 1;
        ret = setsockopt(s, IPPROTO_IPV6,
                         IPV6_V6ONLY, &on,
                         sizeof(on));
        printf("ret = %d¥n", ret);
    }

    ret = bind(s, (struct sockaddr *)&serv,
               (socklen_t )sizeof(serv));
    printf("ret = %d¥n", ret);

    ret = listen(s, 30);
    printf("ret = %d¥n", ret);

    ret = accept(s, (struct sockaddr *)&dst,
(socklen_t *)&dstlen);
    printf("ret = %d¥n", ret);
    printf("dst.sin6_family = %d¥n",
           dst.sin6_family);
    inet_ntop(dst.sin6_family, &dst.sin6_addr,
              v6addr, INET6_ADDRSTRLEN);
    printf("dst.sin6_addr = %s¥n", v6addr);
    close(s);
}
```