# Internet Week 2011

# T2: 事例から学ぶIPv6トラブルシューティング

# ～サーバ編～

(株)クララオンライン / USONYX PTE. Ltd.
白畑　真

# 今日の内容

- はじめに
- サービスにおけるIPv6対応手法
- IPv6問題のクイック診断チャート
- まとめ

# はじめに

- 今日の目的
  - デュアルスタックでのサービス運用において、遭遇する可能性のあるサーバとミドルボックス関連の問題と解決策
- 前提知識
  - IPv4/IPv6デュアルスタックで各種サービスを構築できること
  - 今回は特に説明のない限り、サーバ側はLinux、クライアント側はWindowsを前提に説明

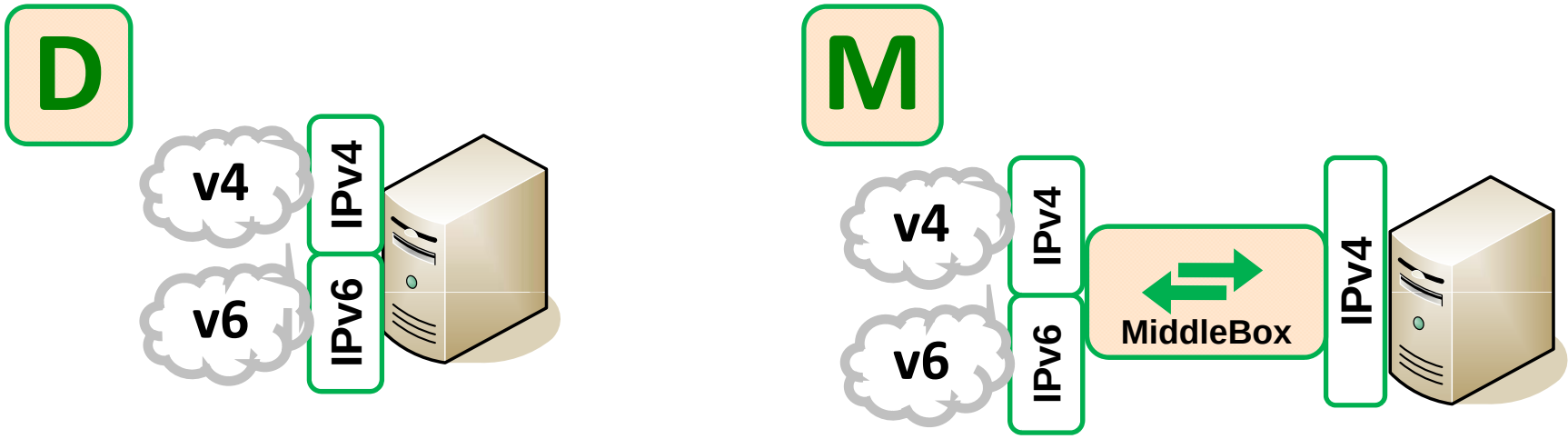
# サービスにおける二つのIPv6対応手法

サーバをIPv4/IPv6<u>デュアルスタック</u>で運用する構成

- IPv4/IPv6変換なし

<u>サーバ自体はIPv4</u>シングルスタックで運用・<u>ミドルボックス</u>でIPv6とIPv4を変換

- ミドルボックス(リバースプロキシ、ロードバランサ、トランスレータ、CDNなど)でIPv4・IPv6を変換

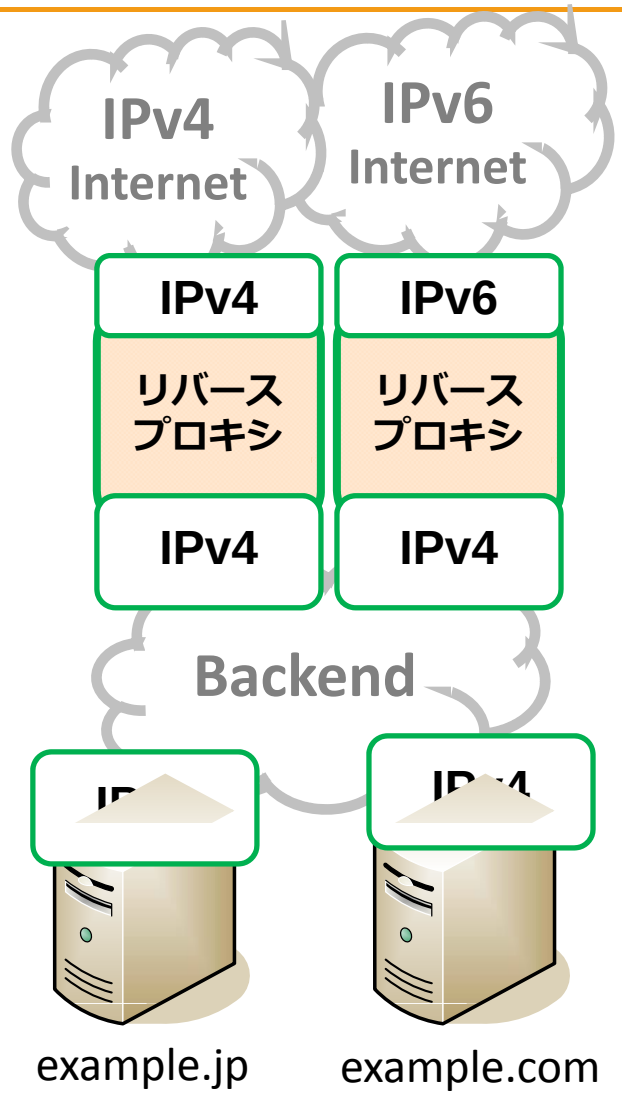



※ネットワーク側はIPv4/IPv6デュアルスタックであることが前提

# ミドルボックスとはなにか

- RFC3234の定義
  - “*A middlebox is defined as **any intermediary device** performing functions other than the normal, standard functions of an **IP router** on the datagram path between a source host and destination host.*”
- IPv4/IPv6プロトコル変換機能を備えた機器
  - プロトコルトランスレータ
  - IPv4/v6変換機能付きFirewall
  - IPv4/v6変換機能付きロードバランサ
- アプリケーションレベルゲートウェイ
  - リバースプロキシ
  - CDN (Contents Delivery Network)

# ミドルボックス: IPv4/IPv6変換用の一般的な構成



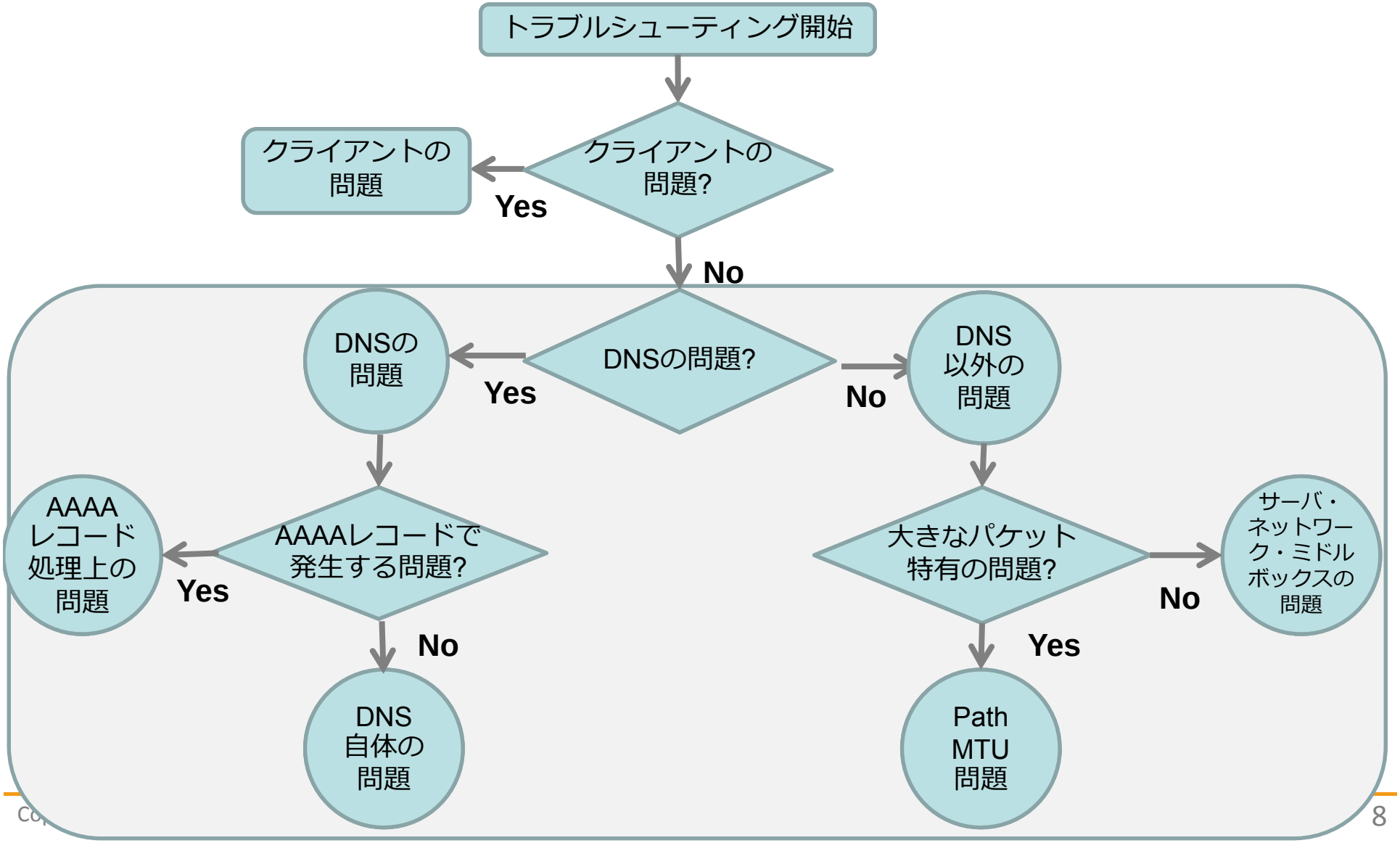

- リバースプロキシ
  - 本来のサーバに代ってインターネット向けにサービスを提供
  - フロントエンド側のリバースプロキシサーバをIPv4/IPv6デュアルスタック、またはIPv6で運用
  - バックエンド側は引き続きIPv4で運用
    - 原理的には逆のパターンもありうる
- メリット
  - サーバ側はシングルスタックでよい
  - アプリケーションレベルでIPv4/IPv6を変換するため、IPv4/IPv6区間でそれぞれMTUが違っても問題ない
- デメリット
  - 利用可能できるプロトコルが限定される
  - 柔軟性 (例:ドメイン名の追加)

# サーバのトラブルシューティング

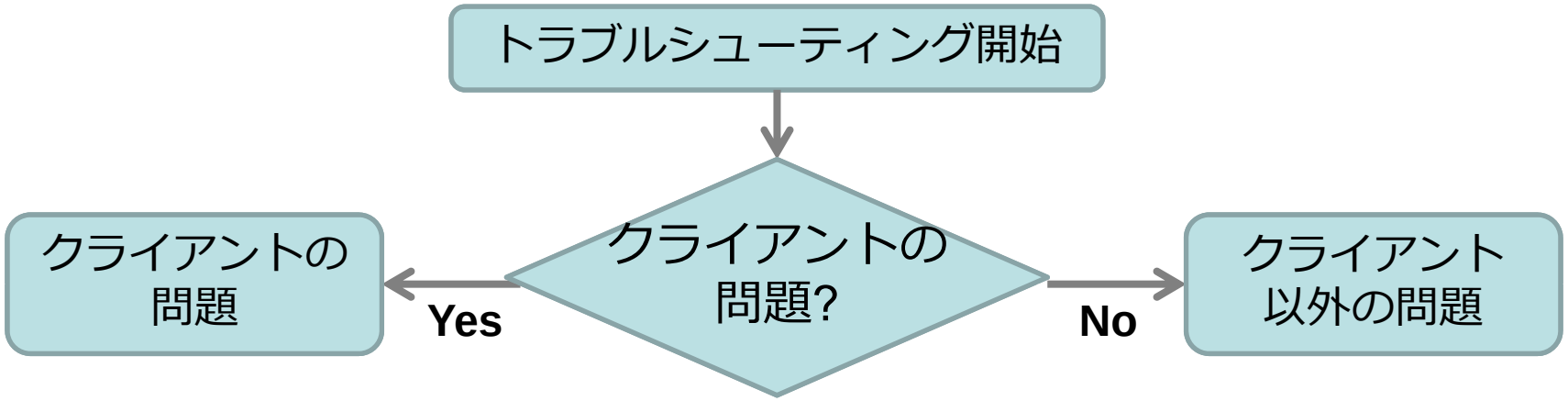
D
- IPv4/IPv6 デュアルスタック
  - サーバまでIPv4/IPv6デュアルスタック
  - IPv4/IPv6変換なし

M
- IPv4/IPv6 変換＋IPv4サーバ
  - サーバはIPv4シングルスタック
  - NAT-PT/リバースプロキシ等でIPv4/IPv6を変換

# 問題切り分けのポイント



# クイック診断チャート:クライアント



# 問題事例1

D M

- **サービスをIPv6対応にしたところ、**
  - **サーバにつながらなくなった**
  - **以前よりも遅くなった**
- 想定原因
  - フォールバック
  - DNS
  - Path MTU ブラックホール
  - IPv6自動トンネル技術(6to4, Teredo)での接続

# 切り分け方法: IPv6のデフォルト設定

- IPv6アドレス:
  - 自動トンネル系
    - 2002::/16 6to4
    - 2001::/32 Teredo
  - 閉域網
    - NTT東西
    - 2001:c90::/32 2001:d70::/30 2001:a000::/21 2404:1a8::/32 2408::/22

# test-ipv6.com

- IPv6/IPv4接続性を確認するためのテストサイト
  - クライアントのWebブラウザやネットワークにおけるIPv6関連の切り分けに有用
  - クライアントからこのサイトにアクセスできるが、他のIPv6対応サーバに接続できない場合、後者に問題がある可能性が高い

## 意図せず有効になるIPv6
# デフォルト設定に注意

- 多くのOSではデフォルト設定でIPv6が有効
  - リンクローカルアドレスが自動的に設定されている
  - 自動トンネル(6to4, Teredoなど) – 特にクライアント向けOS
- RAの広報など、ネットワーク側がIPv6に対応したタイミングで、意図しないうちにIPv6対応になる可能性
- Linuxサーバの場合の例:
  - IPv4ではiptablesでパケットフィルタリングされている
  - IPv6ではip6tablesでパケットフィルタリングされていない
- **サーバ側で直接IPv6サービスを提供しないのであれば、IPv6機能の無効化を検討すべき**
  - きちんとセキュリティ対策を講じた上でIPv6の対応を

# 問題事例1:フォールバック

D M

- 最初にIPv6で接続を試行するが、一定時間内にIPv6で接続できない場合に、IPv4で接続を試行。
- 想定原因
  - クライアントのIPv6接続性に問題がある
    - クライアントはIPv6アドレスを持っているが、IPv6インターネットに接続してない。IPv4インターネットには接続性がある環境
    - Path MTU ブラックホール
  - サーバ側のIPv6サービスに問題がある
    - IPv6ネットワークやミドルボックスで問題が生じている一方、IPv4サービスでは障害が発生していない
    - IPv6のアクセス制御リストの内容がIPv4と異なる
      - 例: Linuxのパケットフィルタリングは、IPv4はiptables、IPv6にはip6tablesという異なるプログラムとなっている

# 切り分け方法:フォールバック

- クライアントにIPv6のみのサイトに接続可能か確認してもらう
  - 例: ipv6.google.com
- クライアントにIPv6閉域網のアドレスが割り振られてないか確認する
  - 参考: NTT東西のIPv6閉域網向けポリシーテーブル設定 http://www.attn.jp/maz/p/i/policy-table/
- 外部から、IPv4とIPv6のアドレスそれぞれに正常性確認を行う
  - telnet サービスのIPv6アドレス 80
  - telnet サービスのIPv4アドレス 80

# 切り分け方法:フォールバック

- IPアドレスベースのアクセス制限
  - IPv4とIPv6で同等かどうか
    IPv4では通信が許可されているが
    IPv6では通信が許可されていない
- DNSベースのアクセス制限
  - IPv4クライアントでは逆引きが**設定されている**ことが多い
  - IPv6クライアントでは逆引きが**設定されない**ことが多い
  - DNSBLも注意が必要

Linuxの設定ファイル
/etc/hosts.allowの例

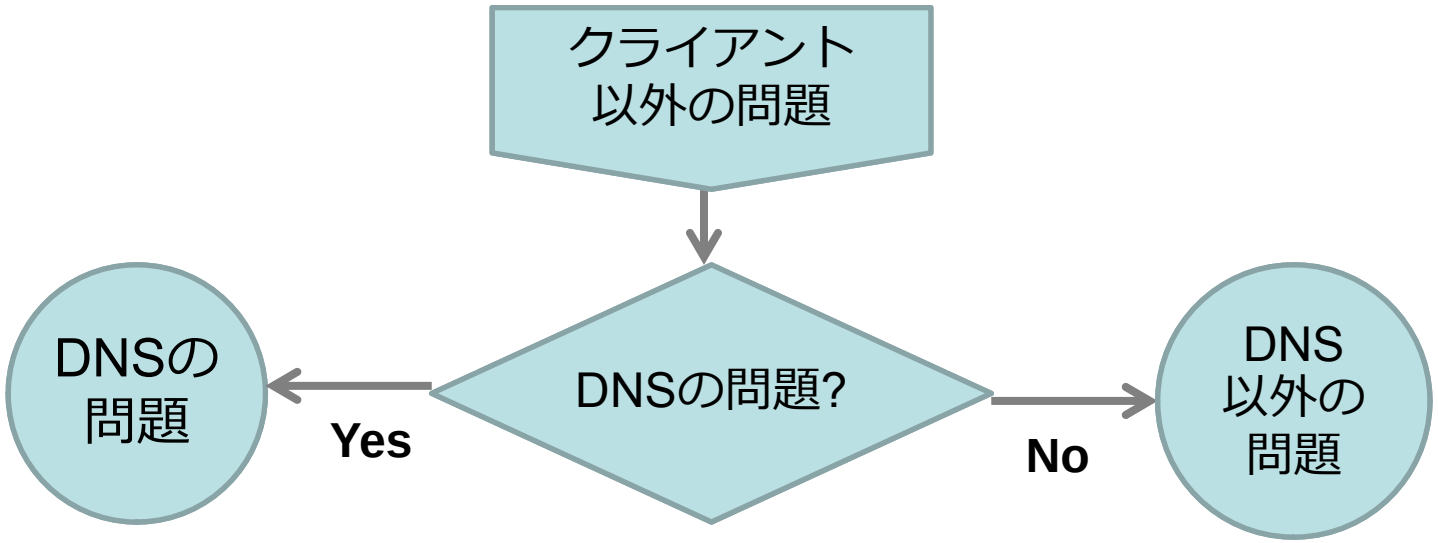
```
ALL: 192.0.2.0/24
ALL: [2001:db8:cafe:123::]/64
```

Apacheの設定ファイル
.htaccessの例

```
Order Allow,Deny
Allow from .example.com
Deny from any
```

ミドルボックスを介した通信を行う構成の場合には、送信元IPアドレスがミドルボックスのアドレスとなることに注意

# クイック診断チャート:DNS



# 問題事例1: DNS

- 名前解決に時間がかかるようになった
- 想定原因
  - 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バイト以上になった
      - 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

# 切り分け方法: DNS (1/3)

- DNSではなくhostsファイルを利用する
  - › hostsファイルにIPv6アドレスだけ、もしくはIPv4アドレスを記載
- ファイルの場所
  - UNIX系OS:
    - /etc/hosts
  - Windows:
    - C:¥WINDOWS¥system32¥drivers¥etc

# 切り分け方法: DNS (2/3)

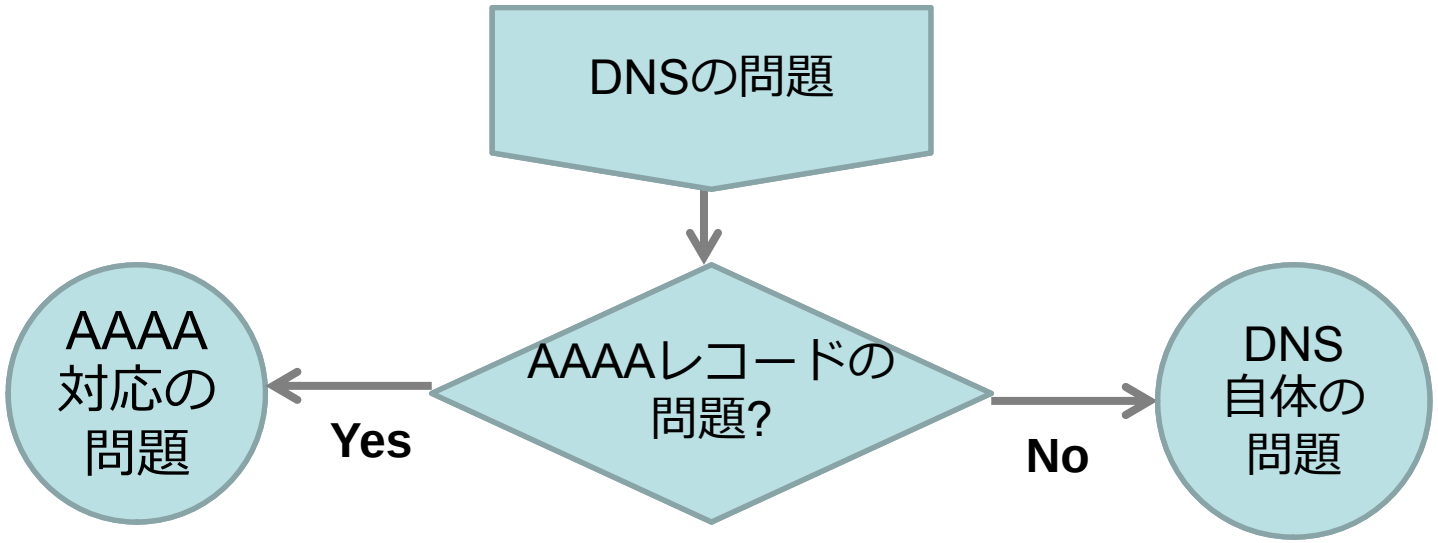
- DNSのトランスポートの確認
  - dig soa example.com @権威サーバのIPv6アドレス
  - dig soa example.com @権威サーバのIPv4アドレス
- EDNS0対応の確認
  - dig +bufsize=2048 soa example.com @権威サーバのIPv6アドレス
  - dig +bufsize=2048 soa example.com @権威サーバのIPv4アドレス

```
; <<>> DiG 9.2.4 <<>> +bufsize=2048 soa example.com @a.iana-servers.net.
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46122
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:

;; ANSWER SECTION:
example.com.            3600    IN      SOA     dns1.icann.org.
hostmaster.icann.org. 2010073215 7200 3600 1209600 3600
```

# クイック診断チャート: AAAAレコード

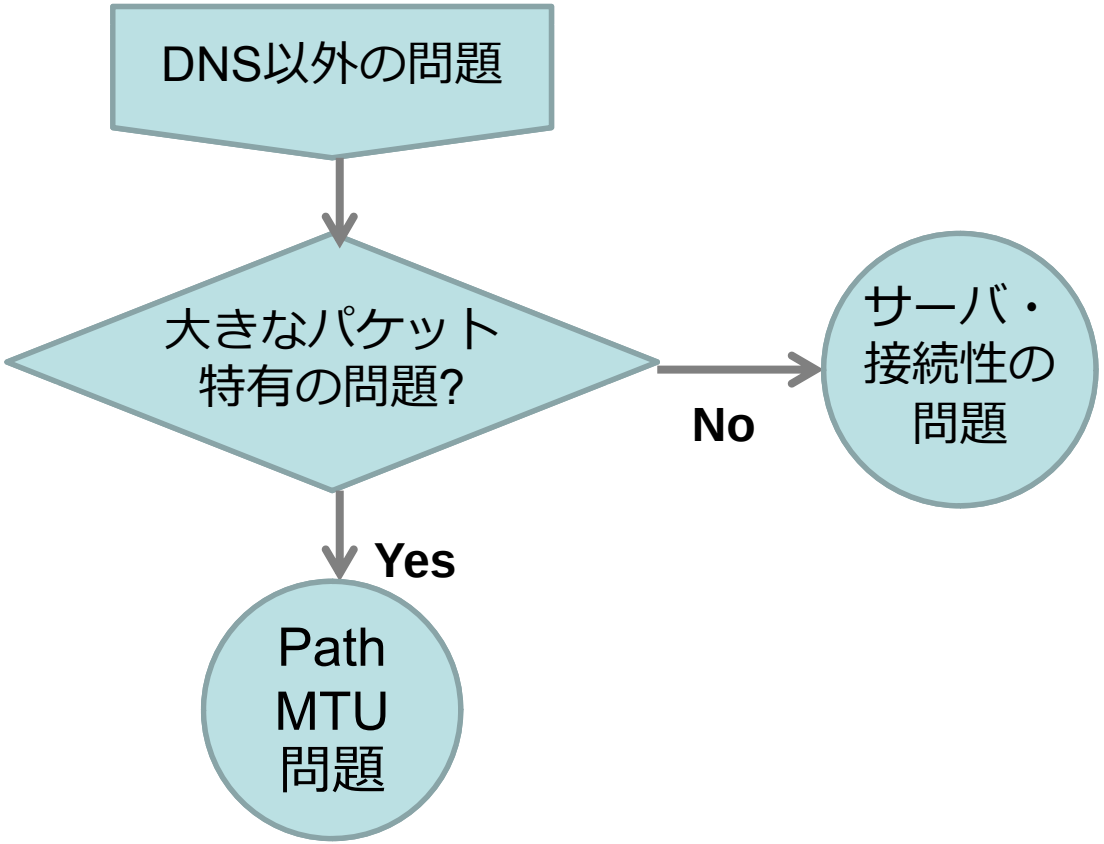


# 切り分け方法: DNS (3/3)

- ミドルボックスの挙動
  - FirewallがDNSのペイロードの内容を確認している
    - 例: Aレコードは通すがAAAAレコードは通さない DNS ALG
  - ブロードバンドルータがDNSのレコードのキャッシュ機能を持っている
    - 例: 正しくないAAAAレコードの内容を応答する
- 同一セグメント内や他のアクセス環境からdig等で確認する

**AAAAフィルター**
ISPのDNSキャッシュサーバなどによってAAAAフィルターが行われている場合、クライアント側ではAAAAレコードの名前を解決できず結果的にIPv6でアクセスできない

# クイック診断チャート: PMTU問題



# 問題事例1: Path MTU ディスカバリ

D M

- 通信開始時に時間がかかる
- 想定原因
  - Path MTUディスカバリが行われている
    - クライアント側のネットワーク環境のMTUとサービス提供側のMTUが異なる
    - サーバのMTUを1280に設定するのも一つの方法
    - IPv4/IPv6変換を行うミドルボックスが導入されている場合、MTUの変換処理に注意(特にNAT-PTの場合)

# 切り分け方法: Path MTU ディスカバリ

- IPv4とIPv6の経路でそれぞれのMTUを調査
  - tracepath
  - tracepath6
- MTUの変更
  - › IPv6の最小MTU: 1280
  - › OSによっては特定経路のMTUのみを変更可能
    - Linuxの場合:

    ip route -6 add 2001:db8:beef::/64 via 2001:db8:cafe::1 **mtu** 1460

# 問題事例2:
# IPv6で通信はできるがIPv4より遅くなった

D M

- 想定原因その1
› 通信先までの経路がIPv4/IPv6で異なる
  - IPv4インターネットでは、日本国内のホスト同士が通信する際、通常は国内で通信が完結
  - IPv6インターネットでは、日本国内のホスト同士の通信でも海外経由となるケースも見られる
    - ただし、徐々に改善している
  - IPv6で一部経路が存在しないため到達できない場合も
    - test-ipv6.comの各ミラーサイトへの接続状況を確認
      http://www.test-ipv6.com/mirrors.html

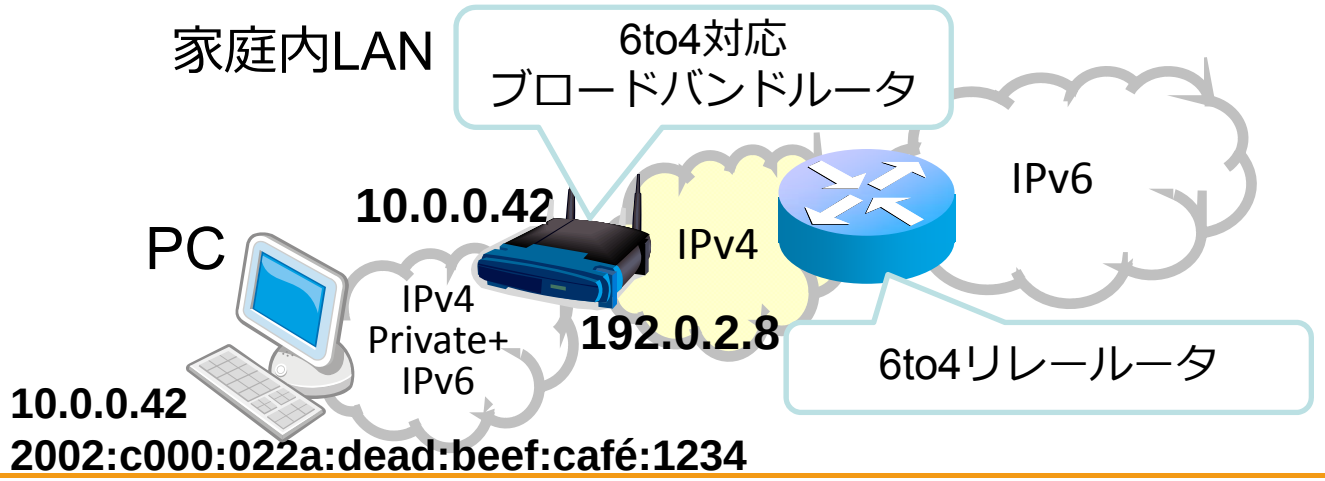
# 問題事例2:
# IPv6で通信はできるがIPv4より遅くなった

- 想定原因その2
- IPv6自動トンネルを利用している
  - IPv4プライベート+6to4の両方アドレスを持っているクライアントがIPv4/v6デュアルスタックのサーバに接続する場合、IPv4が優先
  - 一部のクライアントや環境では6to4がIPv4プライベートアドレスよりも優先される
  - 例: ブロードバンドルータがNAT+6to4対応, クライアントがLinux/Mac OS Xの環境ではIPv4プライベートより6to4が優先される
    - ただし、Mac OSX 10.6.5 から、6to4アドレスの優先度がIPv4プライベートアドレスよりも下がった
      - http://www.gossamer-threads.com/lists/nsp/ipv6/25194
  - Teredoについては、リテラル表記のIPv6アドレスとの通信にのみ利用される

# ブロードバンドルータ経由での6to4接続

- ブロードバンドルータにグローバルIPv4アドレスが割り当てられている場合
- 6to4対応ブロードバンドルータが必要
  - ブロードバンドルータが6to4を終端 + NAT
  - PCとブロードバンドルータ間はNative IPv6
    - ブロードバンドルータがRAを広報

Apple社 AirMac Extreme, TimeCapsule
BUFFALO社 WZR-AMPG300NH
など…



# 6to4はどんな時に利用されるのか?

| サーバ側 | クライアント側 | 標準で利用されるプロトコル |
|---|---|---|
| IPv4 | IPv4+IPv6(6to4) | **IPv4接続** |
| IPv4+IPv6 | | |
| IPv6 | | 6to4によるIPv6接続 |

※ポリシーテーブルを書き換えることで、IPv4/IPv6の両方に対応したサーバに対して、6to4接続を優先利用することも可能

# 切り分け方法

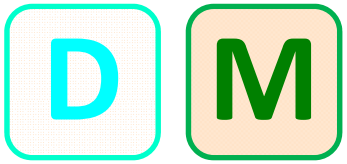
- OSの名称・バージョンを確認する
- IPv4とIPv6で経路を比較
  - traceroute (tracert -4)
  - traceroute6 (tracert -6)
- IPv6アドレスを確認
  - 自動トンネル
    - 2001::/32 Teredo
    - 2002::/16 6to4

## 問題事例2:
## IPv6で通信はできるがIPv4より遅くなった

D M

- 想定原因その3
› IPv6経路上のMTUがIPv4経路上のMTUと比較して小さい
  – MTUが小さい分、スループットが低くなる
  – 現時点のIPv6インターネットには、まだトンネル接続が残っている
    - トンネル区間などのMTUは1280に設定されていることが多い

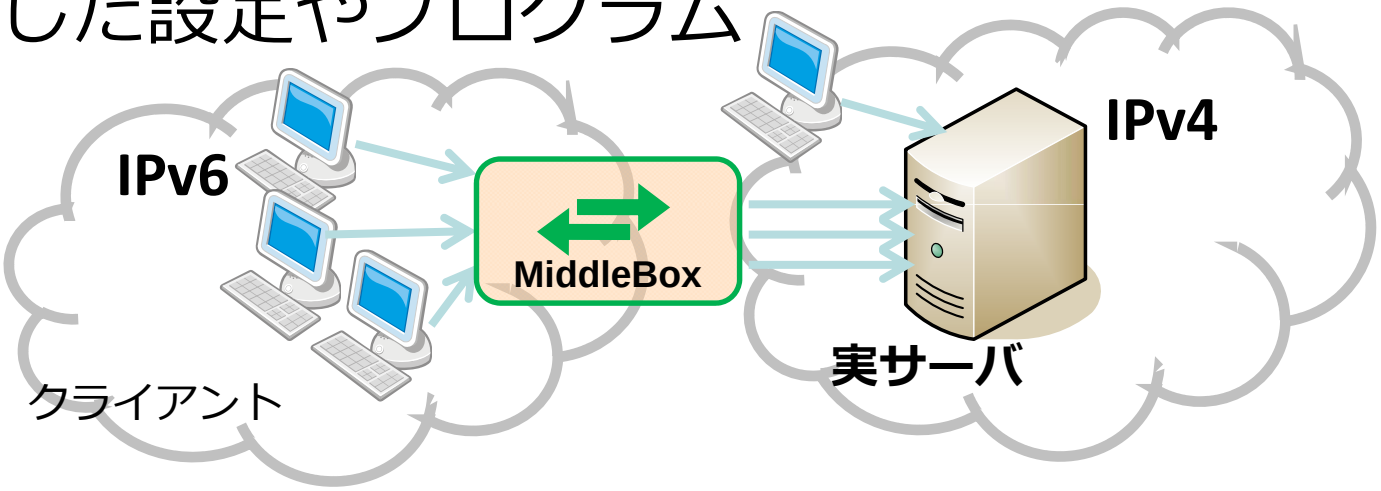
# 問題事例3




- ミドルボックス(IPv4/IPv6変換装置)を経由した場合、アクセス元のIPアドレスが取得できない
- 対策
  - HTTPプロキシやロードバランサを利用する場合、HTTPヘッダ経由で実サーバがクライアントのIPアドレスを取得可能にする
    - X-Forwarded-Forヘッダの値など
      - Apache 2.3のmod_remoteipモジュールなどを利用する方法もある
  - それ以外の場合でも、ログの突き合わせに配慮する
    - Webサーバ側でも、クライアントのポート番号のログを取得するなど

# 問題事例3:ミドルボックスの運用

- 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る
  - Firewallやサーバ側でのDoSアタック対策
- 実サーバ側でのアクセス制限
  - クライアント側のIPアドレスやドメイン名に依存した設定やプログラム



# 問題事例3:ミドルボックスの運用

- ミドルボックスの運用
  - アクセス変換のログ取得を行う
  - プロトコル変換サービスの提供範囲を意識してアクセス制限を設定する
    - オープンプロキシにならないよう注意

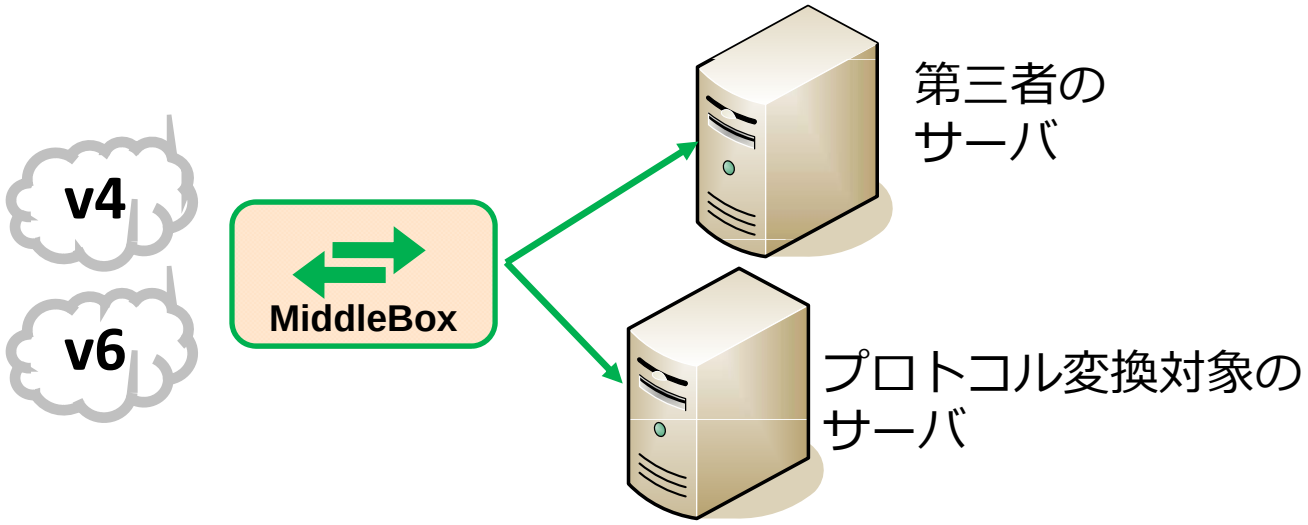
| ミドルボックスの設置場所 | サービスの提供範囲 | ミドルボックス経由でアクセスできるサーバ |
|---|---|---|
| クライアント側<br>(トランスレータ等) | クライアントのネットワークのみ | インターネット全体 |
| サーバ側<br>(トランスレータ,リバースプロキシ,ロードバランサ等) | インターネット全体 | ミドルボックス配下のサーバのみ |

# 切り分け方法3

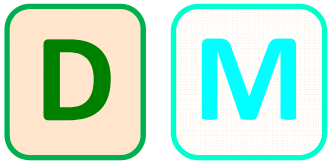
- 外部からミドルボックスにアクセスし、サービス提供外のホストにアクセスできるか確認

```
$ telnet 2001:db8:beef::1 80
GET http://第三者のWebサーバのIPv4アドレス/ HTTP/1.0
```

```
$ telnet 2001:db8:beef::1 80
GET http://[第三者のWebサーバのIPv6アドレス]/ HTTP/1.0
```



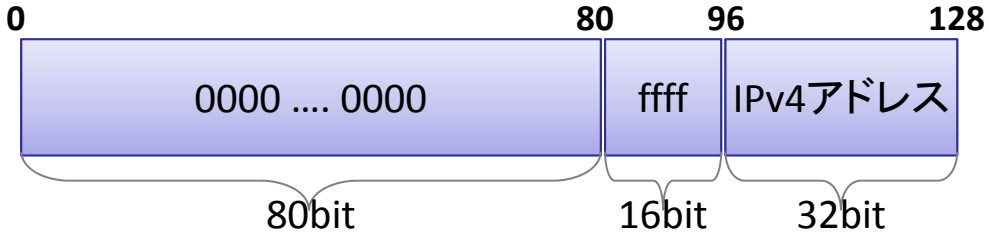
# 問題事例4




- サーバをIPv6対応にしたら、IPv4のアドレス表記や利用するソケットが変わった
- 想定原因
› IPv4の通信において、IPv6のソケット上でIPv4射影アドレスが利用されている
- 解決策
  - そのままにしておく
    - プログラムの改修で対応する
  - IPv4のソケットを利用する
    - アプリケーションの設定変更
    - IPv4用サービスとIPv6サービス用に、別プロセスで起動する
      - OSの設定とソケットのbindの方法によってはできないケースも

# IPv4射影アドレス(IPv4 Mapped Address)とは

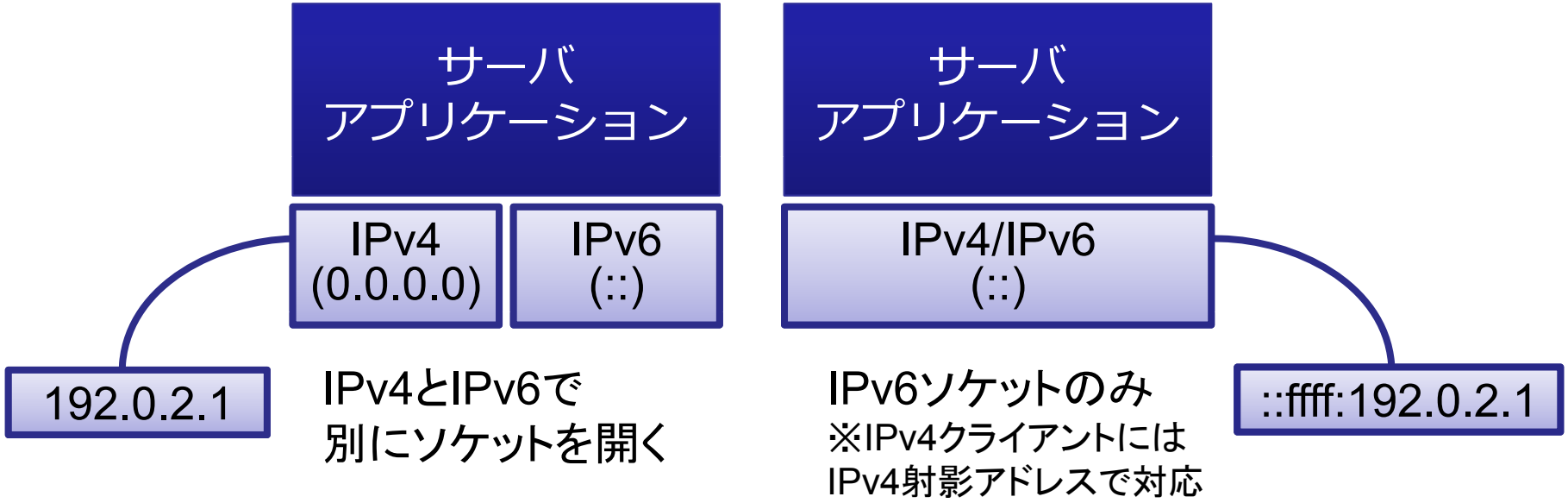
- IPv4アドレスをIPv6アドレスとして表す特殊なIPv6アドレス




- 例: 192.0.2.128 の場合
  ::ffff:192.0.2.128 もしくは ::ffff:c000:280
- IPv6対応アプリケーションが、IPv4/IPv6対応のソケット(“::”)でIPv4のみを持つノードと通信する際に利用
  - ノード内部での利用に限定
  - 送信元・宛先アドレスとしては利用されない

# IPv6対応サーバアプリケーションとソケットの実装方法

- アプリケーションのIPv4/IPv6デュアルスタック対応方式には2種類の方法がある
  - OSやアプリケーションの実装、設定により異なる
  - ログ取得やアクセス制御に互換性がない場合も



# 例: OpenSSHの場合

- IPv4/IPv6で別々にソケットをbind(2)する場合

“netstat –an”コマンドの結果:

```
tcp    0 0 0.0.0.0:22                0.0.0.0:*                   LISTEN
tcp    0 0 :::22                     :::*                        LISTEN
```

**sshd_configファイルの設定:**
ListenAddress 0.0.0.0
ListenAddress ::

- IPv6のみでソケットをbind(2)する場合

“netstat –an”コマンドの結果:

```
tcp    0 0 :::22                     :::*                        LISTEN
```

**sshd_configファイルの設定:**
ListenAddress ::

アプリケーションによっては、設定ファイルやコマンドラインでソケットの構成方法を変更できる場合もある

# 各種OSとIPv4射影アドレス

| OS | IPv4射影アドレスの対応 | OS全体での無効方法化 |
|---|---|---|
| Linux 2.6 | デフォルト状態で有効 | net.ipv6.bindv6only変数を1に変更 |
| FreeBSD 5.x 以降 | デフォルト状態で無効 | net.inet6.ip6.v6only変数を1に変更 |
| Mac OS X | デフォルト状態で有効 | net.inet6.ip6.v6only変数を1に変更 |
| Windows Server 2003, Windows XP 以前 | 無効 | なし |
| Windows Server 2008, Windows Vista 以降 | 有効 | なし |

**サーバアプリケーション側の対応:**
ソースコードを、IPV6_V6ONLY ソケットオプションを設定するように改修することで無効化可能

IPv4 射影アドレスを利用しない環境では、IPv4クライアントからの接続を受け付けるために**IPv6のソケットとは別に**IPv4のソケットを開く必要がある

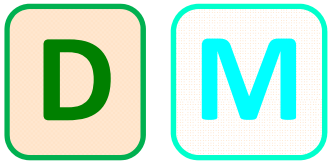
# IPv4射影アドレスの問題点

- draft-itojun-v6ops-v4mapped-harmful-02 (IPv4-Mapped Address API Considered Harmful) の指摘
  - 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
    - 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



41

# 問題事例5: 意図しないRA




- **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

# 問題切り分け: 意図しないRA

- 経路表を確認し、デフォルルートが正しいIPv6アドレスになっているか確認
- 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
- 不正RAの監視・対応ツール:
  - NDPMon - IPv6 Neighbor Discovery Protocol Monitor http://ndpmon.sourceforge.net/
  - rafixd 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

# 問題切り分け: 意図しないRA

- RAを利用しない
  - サーバやルータなどでは静的にIPv6アドレスを設定するのがおすすめ
- RAを利用する場合
  - L2スイッチ等でRAのフィルタリング
  - RAの監視
- RAの優先度を設定
  - RFC4191で優先度が設定できるようになった
  - High・Medium(デフォルト)・Low
    - Highに設定することで過失による影響を軽減

# 問題事例6

D M

- IPv6環境でBondingが正常に機能しない
- 想定原因
› Bondingの設定がIPv4を前提としている
  – Bondingの主な監視方法
    - MII(物理インタフェースの状況)
    - ARPの応答
      – ARPはIPv4固有のプロトコル

→MIIなどIPv4に依存しない方法を採る必要がある

# 切り分け方法6

- Bondingの動作モードを確認
- ARPモニタリングの場合:

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.4.0 (October 7, 2008)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0
ARP Polling Interval (ms): 1000
ARP IP target/s (n.n.n.n form): 192.168.1.1

Slave Interface: eth0
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:22:19:XX:XX:X2

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:22:19:XX:XX:X4
```

# IPv4/IPv6関連のバグとセキュリティホール

IPv6対応とセキュリティ

# IPv6特有のセキュリティホール

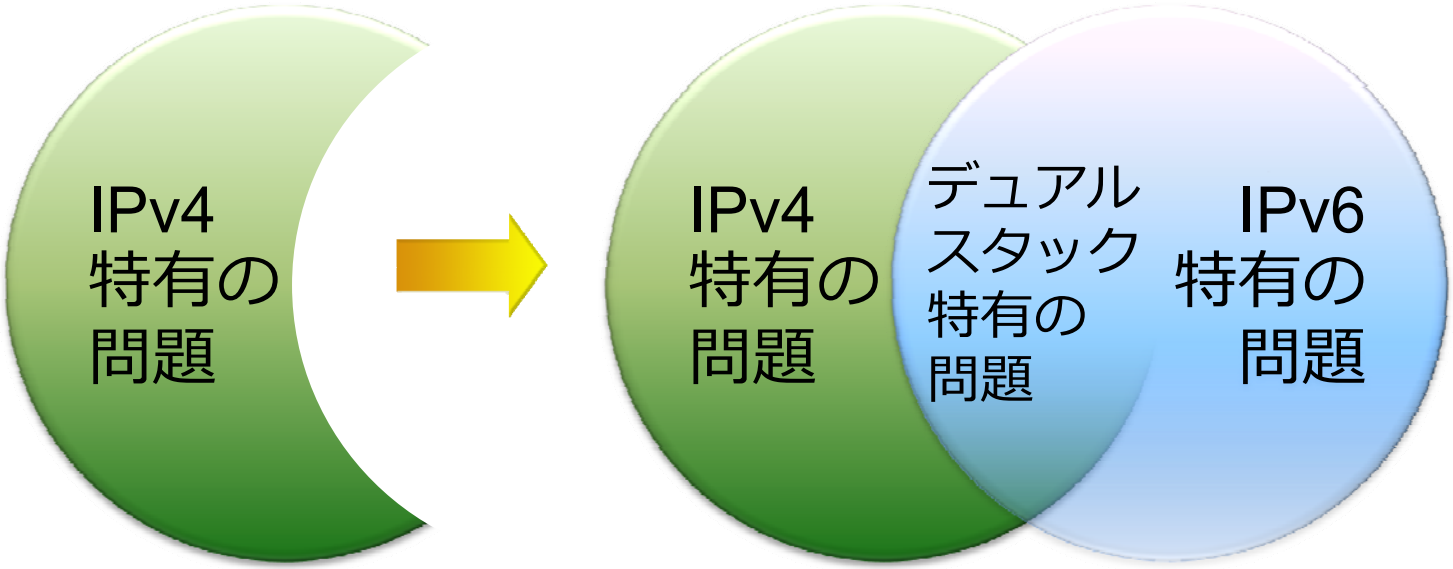
- プロトコル仕様上の欠陥
  - 例: IPv6 Type 0 Routing Header Vulnerability (CVE-2007-2242)
- プロトコル実装上の欠陥
  - 例1: IPv6 Neighbor Discovery Protocol Neighbor Solicitation Vulnerability (CVE-2008-2476)
  - 例2: IPv6 を実装した複数の製品にサービス運用妨害 (DoS) の脆弱性 (JVN#75368899)
    - 大量のIPv6アドレスやNDPエントリを生成させるDoS攻撃
- プロトコルスタックの成熟度
  - 歴史的には、IPv4プロトコルスタックに重大なセキュリティホールが発見されてきた
  - 例: 1996年の Ping of Death攻撃, 2004年のPath MTU Discovery攻撃

# IPv4/IPv6デュアルスタック環境特有のセキュリティホール

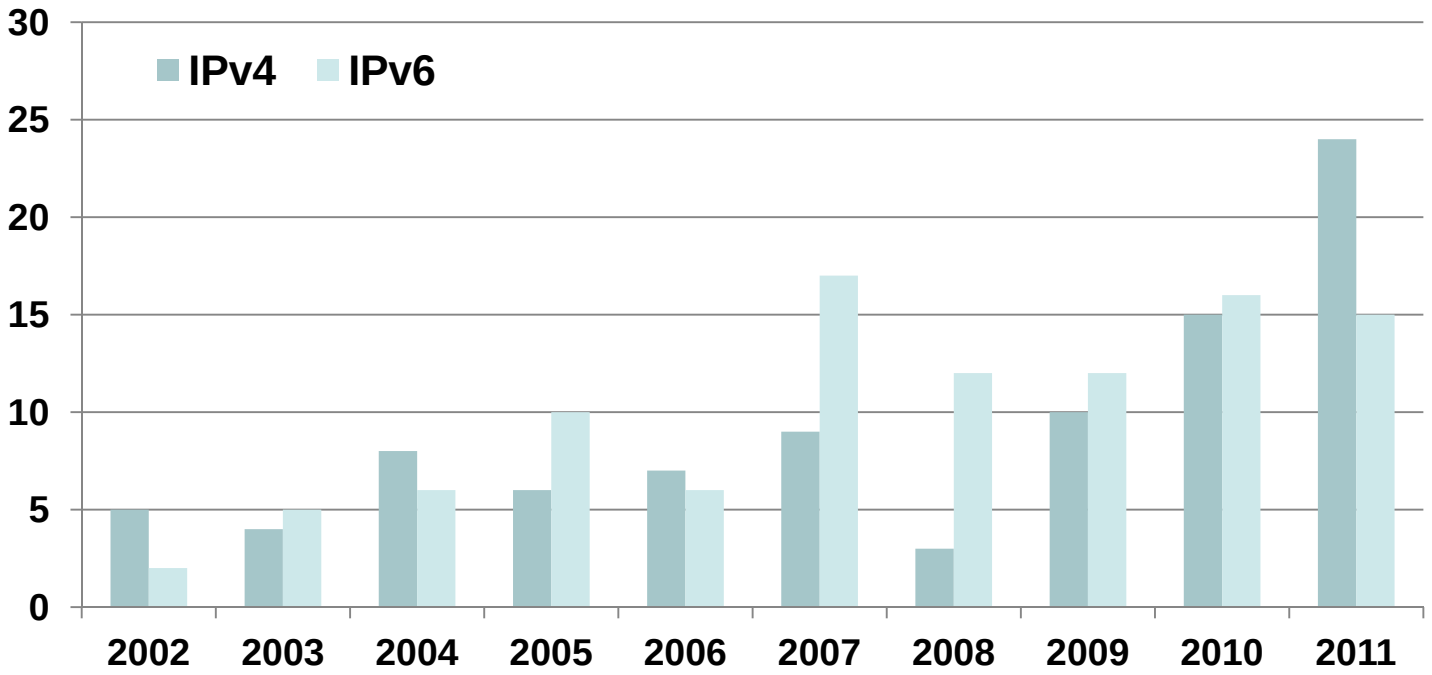
- CVE-2008-1153
  - ネットワーク機器に対して特殊なIPv6パケットを送ることで、当該機器のIPv4のサービスに対してDoS攻撃が成立する脆弱性
- CVE-2006-6263, CVE-2006-6266, CVE-2007-3038
  - IPv4/IPv6共存技術であるTeredoクライアントを踏み台として悪用する攻撃
- CVE-2007-1338
  - CPEのIPv6トンネルの設定不備に関する脆弱性

# サーバのデュアルスタック対応

- IPv6を実装したソフトウェアは、IPv4に比べ歴史が浅いため何らかの欠陥が残っている可能性が高い



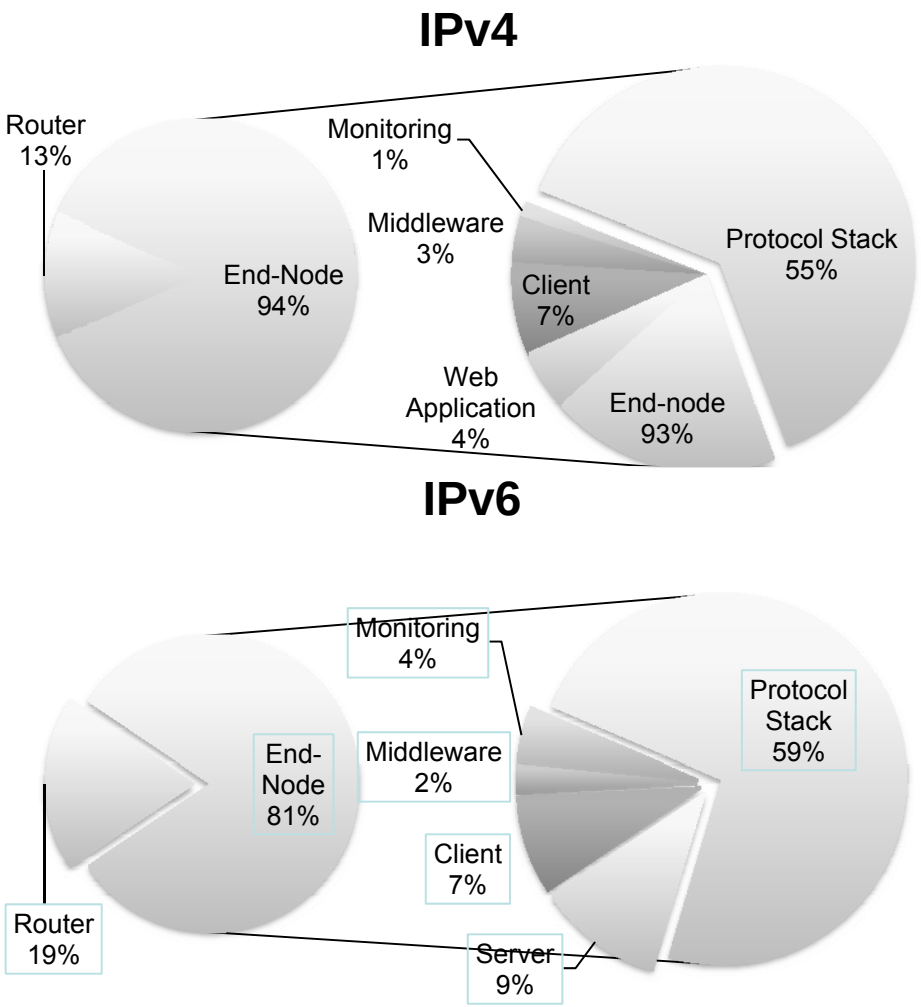
# IPv4/IPv6 プロトコル特有の脆弱性の推移




出典 : MITRE社の CVE Database より発表者が作成, 2011年11月現在

IPv4/IPv6ともにプロトコル特有の脆弱性の発見件数は高い水準で推移

# IPv4/IPv6に関連した脆弱性の傾向




- プロトコルスタックの脆弱性が多い
- 個々のサーバ、クライアントアプリケーションも無視できない
- 脆弱性の例
  - [IPv4関連] 不正な形式のパケット処理に関する問題
  - [IPv6関連] IPv6アドレスの解釈時の問題

# まとめ

- 問題切り分けのポイント
  - 発生箇所: サーバとクライアント、ミドルボックス
  - プロトコル・サービスの切り分け
- 主なチェックポイント
  - フォールバック
  - DNSのIPv6対応
  - Path MTUブラックホール
  - IPv6関連のバグ・セキュリティホール情報にも要注意

# 参考資料: ミドルボックス

# Middlebox: トランスレータ

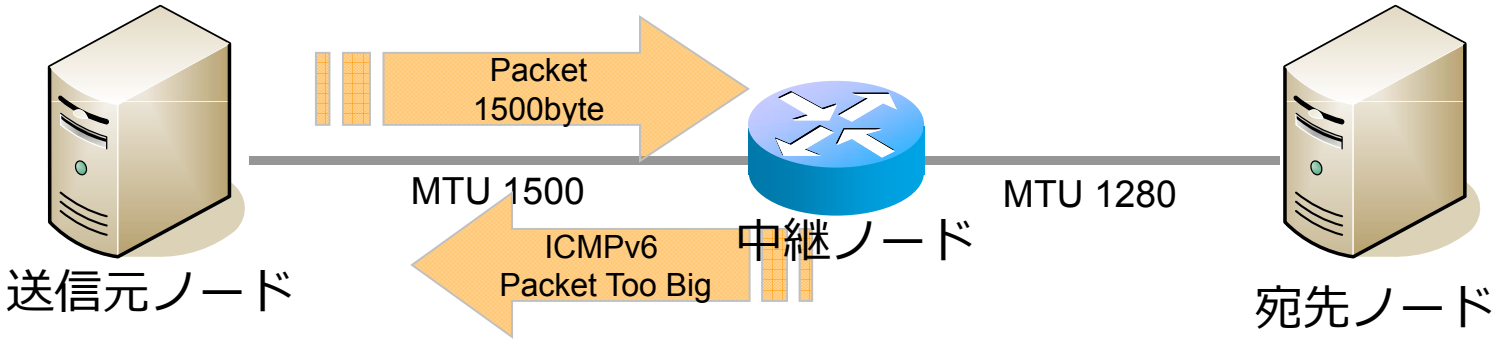
- IPv4とIPv6は回線上で互換性がない
  - IPv4クライアントとIPv6シングルスタックのサーバ
  - IPv6クライアントとIPv4シングルスタックのサーバ

  プロトコルを変換して通信できるようにする必要がある
- IP層で変換を行うNAT-PT方式が主流
  - マスカレード
  - 静的マッピング(1:1)
    - 単一のIPv6アドレスを単一のIPv4アドレスに割り当て

    = IPv4アドレスの節約にはならない
    - サーバ向け
  - 動的マッピング(n:m)
    - DNS-ALGとの連携
    - クライアント向け

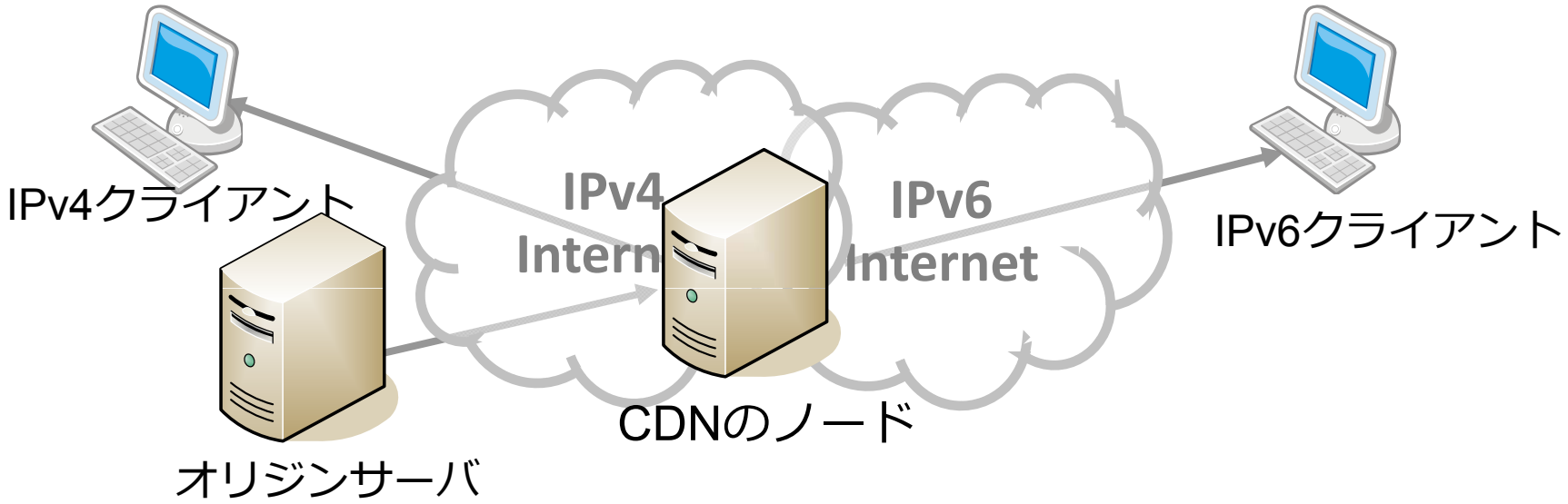
# IPv6とMTU

- IPv6の最小MTUは1280バイト
- IPv6では中継ノードでパケットの分割を行わない
- Path MTU Discovery (RFC1981)
  - 相手先ノードまでの経路上で利用可能な最大MTUを調査
  - 転送しようとするパケットがルータの転送先インタフェースのMTUより大きい場合、送信元に ICMPv6 type=2 (Packet Too Big) エラーを返す
  - Firewall で ICMPv6 type=2 code=0 をフィルタしないよう注意



# Middlebox: CDN

- Contents Delivery Network
  - DNSやBGP Anycastなどを利用し、アクセス元から最適なCDNのノードに誘導
  - 広域に分散した一種のリバースプロキシ
    - CDNのノードがIPv6でサービスを提供すれば、オリジンサーバはIPv4のみに対応すれば良い

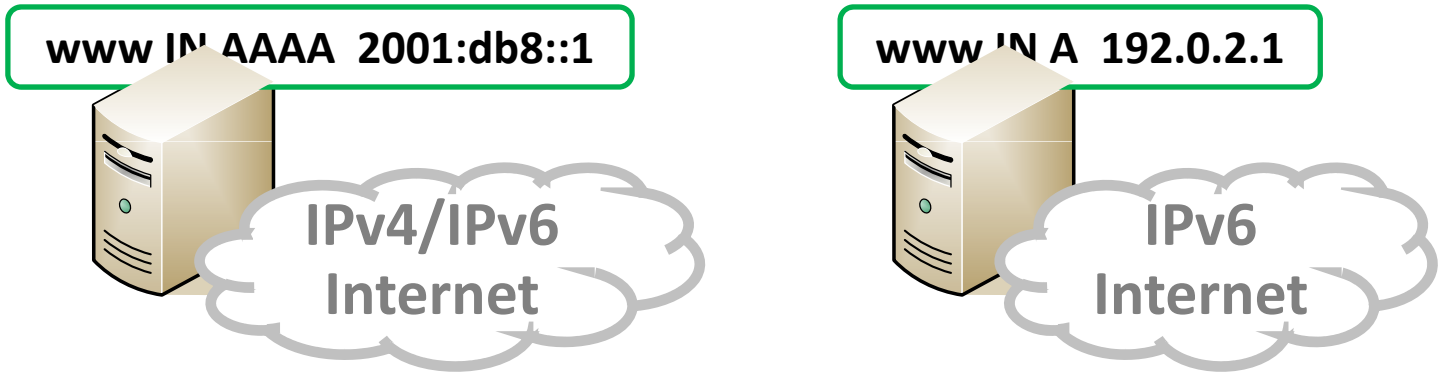


# 参考資料: IPv6環境でのDNS

## DNSのIPv6対応(1/5)

# IPv6環境でのDNS

- 正引き: ホスト名からIPv6アドレスへの変換
  - AAAAレコードでIPv6アドレスを記述
- 逆引き: IPv6アドレスからホスト名への変換
  - ip6.arpa以下のツリーにPTRレコードを記述




**DNSレコードとDNSトランスポートは独立**

1. IPv4/IPv6のいずれかを問わず、**DNSはIPv6アドレスの問い合わせ(AAAAレコード)**に応答する
2. **IPv6の通信で、**DNSがIPv4(Aレコード)とIPv6(AAAAレコード) の問い合わせに応答する

## DNSのIPv6対応(2/5)

# DNSのIPv6対応に必要な機能

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
     - ASPサービスを利用している場合には事業者の対応が必要
   - 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では、UDP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アドレスなどサイズの大きなレコードを格納する際に役立つ
     - 互換性の問題があるため、Root DNS/ccTLD DNSでは避けられている
3. DNS権威サーバのトランスポートのIPv6対応
   - IPv6インターネット経由でDNSクエリに応答

- 主要な実装:
  - 1のみ対応: djbdns
  - 1~3すべて対応: BIND 9, NSD, Microsoft DNS (Windows Server 2003)

# DNSのIPv6対応(3/5)
# 名前の付け方の例

1. IPv4向けとIPv6向けで同じ名前を使う
   - 例: KAME Project
     - www.kame.net (AレコードとAAAAレコードの併用)
       - 203.178.141.194
       - 2001:200:0:8002:203:47ff:fea5:3085
2. 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)
       - 72.14.205.147
       - 72.14.205.99
       - 72.14.205.103
       - 72.14.205.104
     - ipv6.google.com (AAAAレコードのみ)
       - 2001:4860:0:2001::68

# DNSのIPv6対応(4/5)
# 名前の付け方

1. 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、名前解決に失敗する、もしくは遅くなる恐れ
    - DNS検索過程において壊れたDNSサーバが存在する
    - IPv4/IPv6で両方で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
    - Aレコードと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
    - IPv6の閉域網とIPv4インターネットの組み合わせ: マルチプレフィックス問題や、RAによりトンネルが利用できない問題の影響を受ける恐れ

2. 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
  - 壊れたDNSサーバの実装に伴う問題を回避できる
    - AAAAレコードの問い合わせに対する応答を正しく処理できないDNSサーバ
- デメリット:
  - 透過性: ユーザがIPv4/IPv6のいずれを利用するか意識する必要がある

**iDC/ISP以外のDNSサーバにも注意**

DNSサーバを内蔵しているロードバランサやブロードバンドルータ、DNSの内容を検査するFirewallやIDS,IPSについても配慮が必要

# DNSのIPv6対応(5/5)
# ドメイン名のレジストリ/レジストラの対応

※本ページではトランスポートとしてIPv6を利用して再帰的問い合わせを行うケースを想定しています

- Root DNSにIPv6 Glue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リの対応
  - IPv6のホスト登録(GlueレコードとしてAAAAレコード登録)ができるか
  - 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