IPv6対応アプリケーション開発

2016年12月12日

IPv6普及・高度化推進協議会
渡辺 露文

About me

- 渡辺 露文（わたなべ つゆふみ）
  - Twitter: @tsuyu23
  - 普段は某SIer勤務
  - 経歴

自社ISP運用・ツール開発
⇒ データセンターでのシステム構築・運用 ⇒ 研究開発
⇒ 社内インフラの構築・運用 ⇒ 技術調査・技術者教育、OSS利用管理
⇒ セキュリティ

  - IPv6普及・高度化推進協議会 会員
    - IPv6導入に起因する問題検討SWG
    - アプリケーションのIPv6対応検討SWG

Do you know …

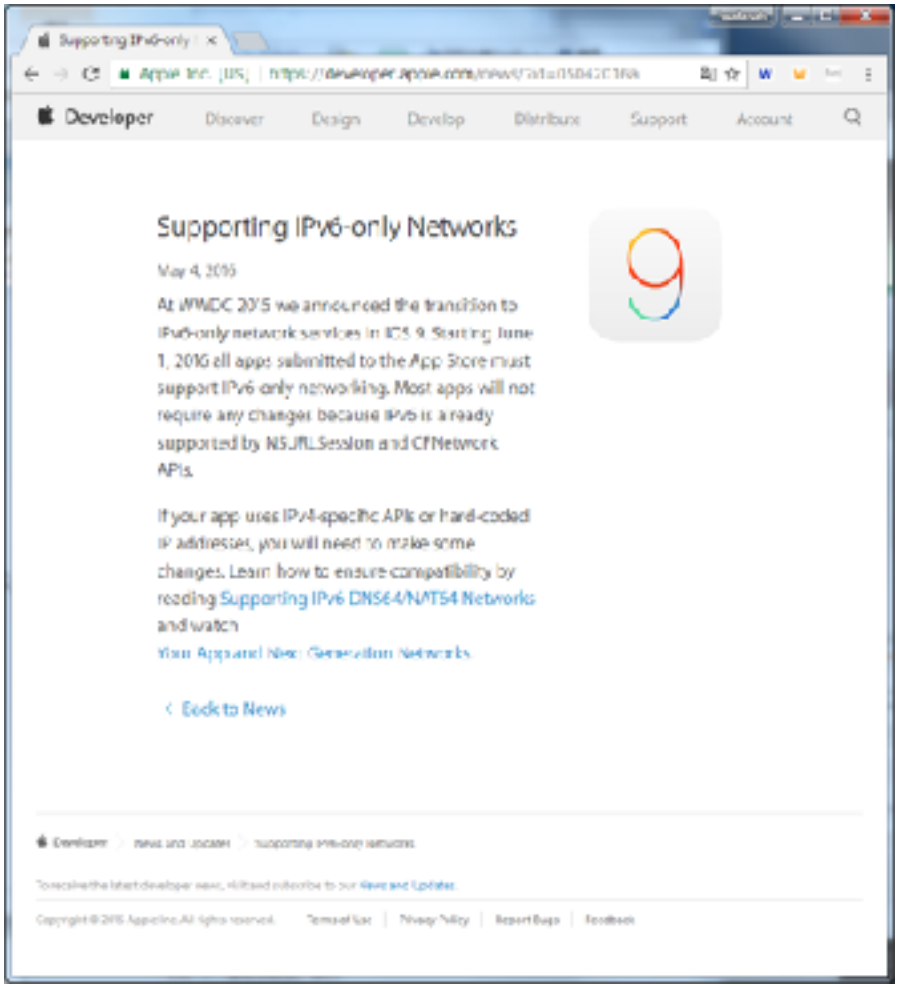
IPv6 ?

Internet Protocol　version 6

インターネットの通信に関する規約
IPネットワークに接続するには1つ以上のIPアドレスが必要
皆さんが馴染んでいるのはIPv4（例：10.1.2.3）

今年5月 Apple のアナウンス

- 2016/6/1 から、App Store に載せるアプリは、IPv6-only ネットワークで動作しないといけない
  - ほとんどのアプリは何も変更しなくて大丈夫なはず
  - もし、IPv4固有のAPIやIPアドレスをハードコードしてたら、（Networking Overview の）「Supporting IPv6 DNS64/NAT64 Networks」を読んで対応してね



https://developer.apple.com/news/?id=05042016a

今年9月 マイクロソフトのアナウンス

- AzureがIPv6に対応！
  - ネイティブでIPv6をサポート
  - 対象
    - ロードバランサー
    - Azure VM
      - デュアルスタック

https://docs.microsoft.com/ja-jp/azure/load-balancer/load-balancer-ipv6-overview

今年8月および10月のAWS のアナウンス

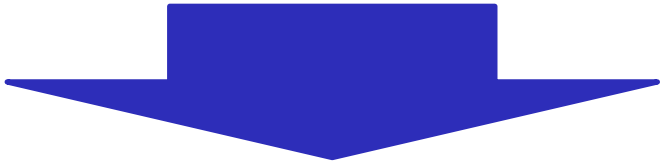
- IPv6サポート開始
  - Amazon S3, S3 Transfer Acceleration
  - CloudFront
  - WAF

https://aws.amazon.com/jp/blogs/news/now-available-ipv6-support-for-amazon-s3/

https://aws.amazon.com/jp/blogs/aws/ipv6-support-update-cloudfront-waf-and-s3-transfer-acceleration/

最近の流れ

- 今年、大御所の IPv6 対応／ IPv6 対応義務化が目立った
  - Apple： iOS App Store 登録アプリのIPv6対応義務化
  - Microsoft： Azure VM およびロードバランサの IPv6 サポート
  - AWS：S3, CloudFront のIPv6サポート



アプリケーションも IPv6 対応で作る時代が到来

Agenda

1. IPv6対応の話をする前に
2. IPv6対応アプリケーションの作り方
3. PHPサンプルコード解説
4. AppleのIPv6対応解説

1. IPv6対応の話をする前に

1. アプリケーションを作る上で知っておくべきIPv6の基礎
2. IPv6対応の前に気を付けるべきこと

実はIPv6を使える環境が増えています（1）

- 最近のOS
  - Windows Vista以降
  - Mac OS X／macOS
  - Linux
  - FreeBSD
  - iOS
  - Android
  - …

いずれも
デフォルトで
利用可能

実はIPv6を使える環境が増えています（2）

- インターネット回線
  - フレッツ光ネクスト
  - au ひかり
  - NURO 光
  - …

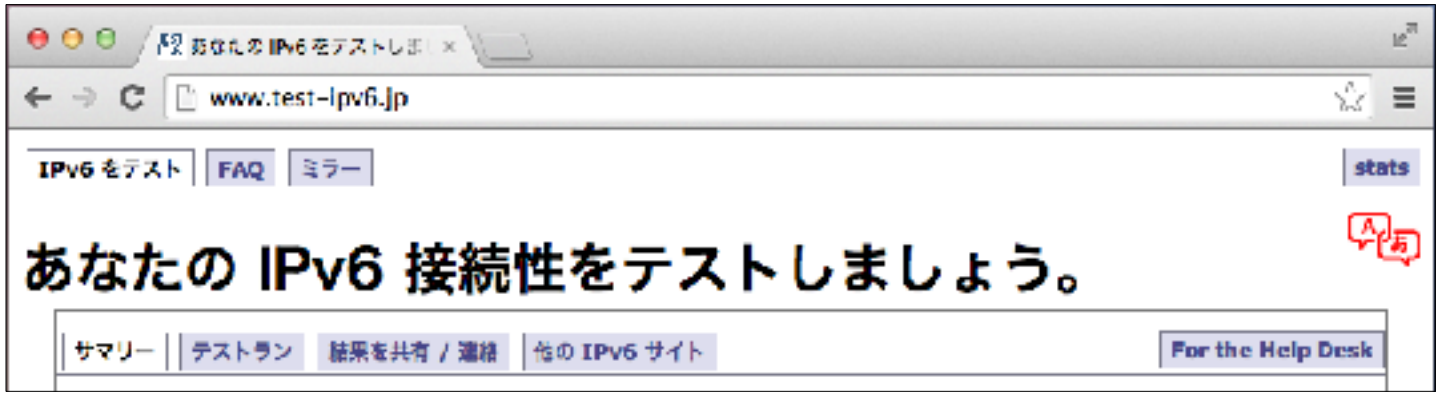
利用可能

既存ユーザへの自動導入も進行中

すでに、ユーザからあなたのサービスにIPv6でアクセスされようとしている

余談：確認してみよう！ IPv6でインターネットにアクセスできるかな？

- Webブラウザで http://www.test-ipv6.jp にアクセス



- Webブラウザで http://www.kame.net にアクセス

IPv6でアクセスすると、亀が踊ります♪

IPv6の背景：IPv4アドレス枯渇

- IPv4アドレスの在庫状況
  （地域インターネットレジストリ）

通信事業者、ISP、データセンター、クラウド事業者等の在庫が残るのみ

世界的に足りなくなってきた

もはやIPv6対応しないと時代遅れ？

- 2015年1月のGHOST騒ぎ…

脚注部分に注目！

http://www.ipa.go.jp/security/announce/20150129-glibc.html

1.1. アプリケーションを作る上で知っておくべきIPv6の基礎

IPv4とIPv6とでは何が違うのか？①

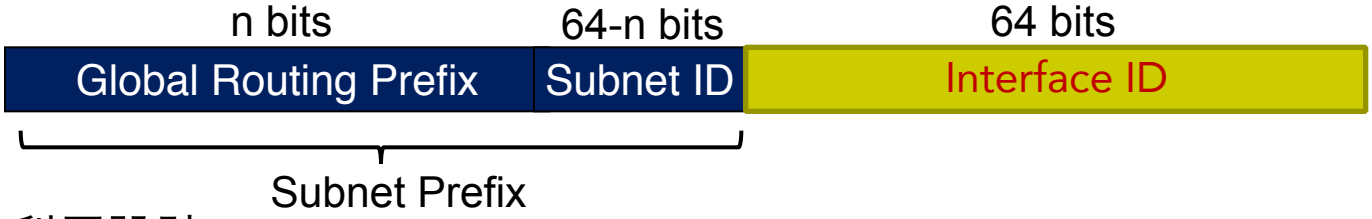
- アドレス体系が異なる（IPv6のアドレス空間は広大）

		IPv4アドレス	IPv6アドレス
アドレス長		32bit	128bit
文字列表記	表記法	8bitずつ区切り、10進数で表記	16bitずつ区切り、16進数で表記
	区切り文字	.（ドット）	:（コロン）
	文字列長	15文字以内	39文字以内

- 例
  - IPv4）192.0.2.1
  - IPv6完全表記）
    2001:0db8:0000:0000:0001:0000:0000:0001
  - IPv6省略表記）2001:db8::1:0:0:1（RFC5952準拠）

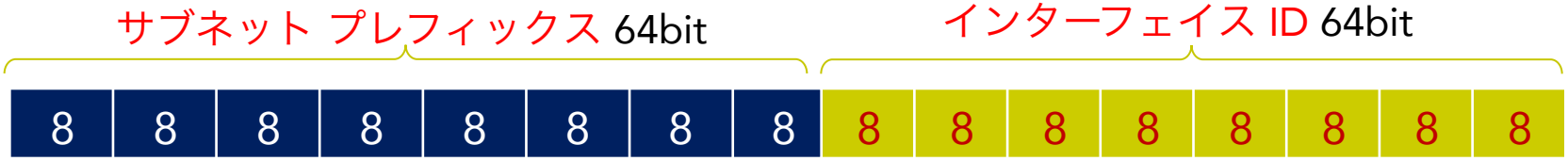
IPv4とIPv6とでは何が違うのか？②

- ユニキャストアドレスの構成



- アドレス利用設計
  - IPv4では、ネットワークアドレス部の長さを調整

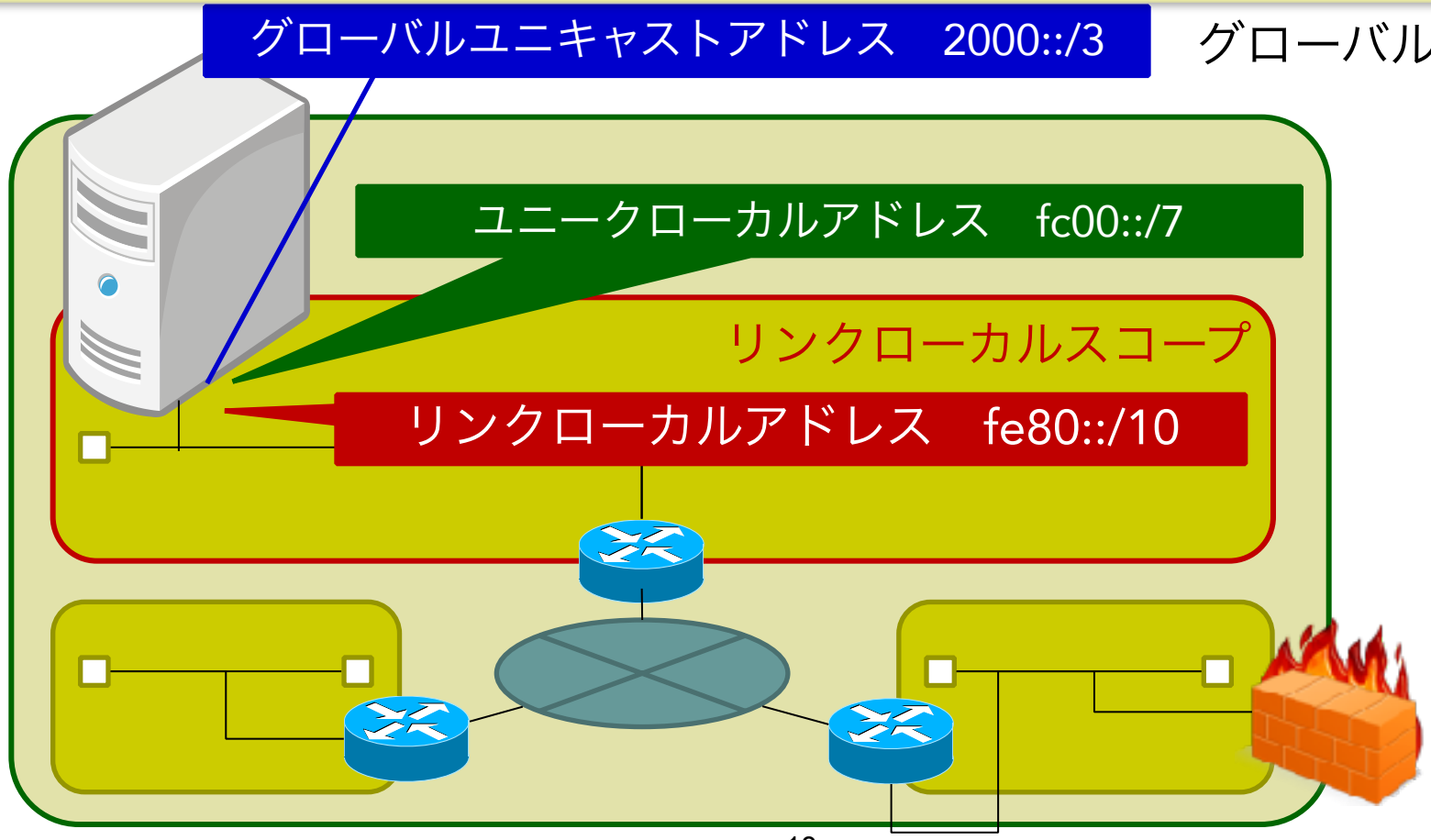
  - IPv6では、原則的にサブネットプレフィックスは固定



IPv4では収容する端末の見込み台数に応じて最適な大きさのサブネットを設計していたが、IPv6では64bit固定

IPv4とIPv6とでは何が違うのか？③

IPv6では1つのNICに複数のアドレスを有効範囲に応じて割当て、使い分ける



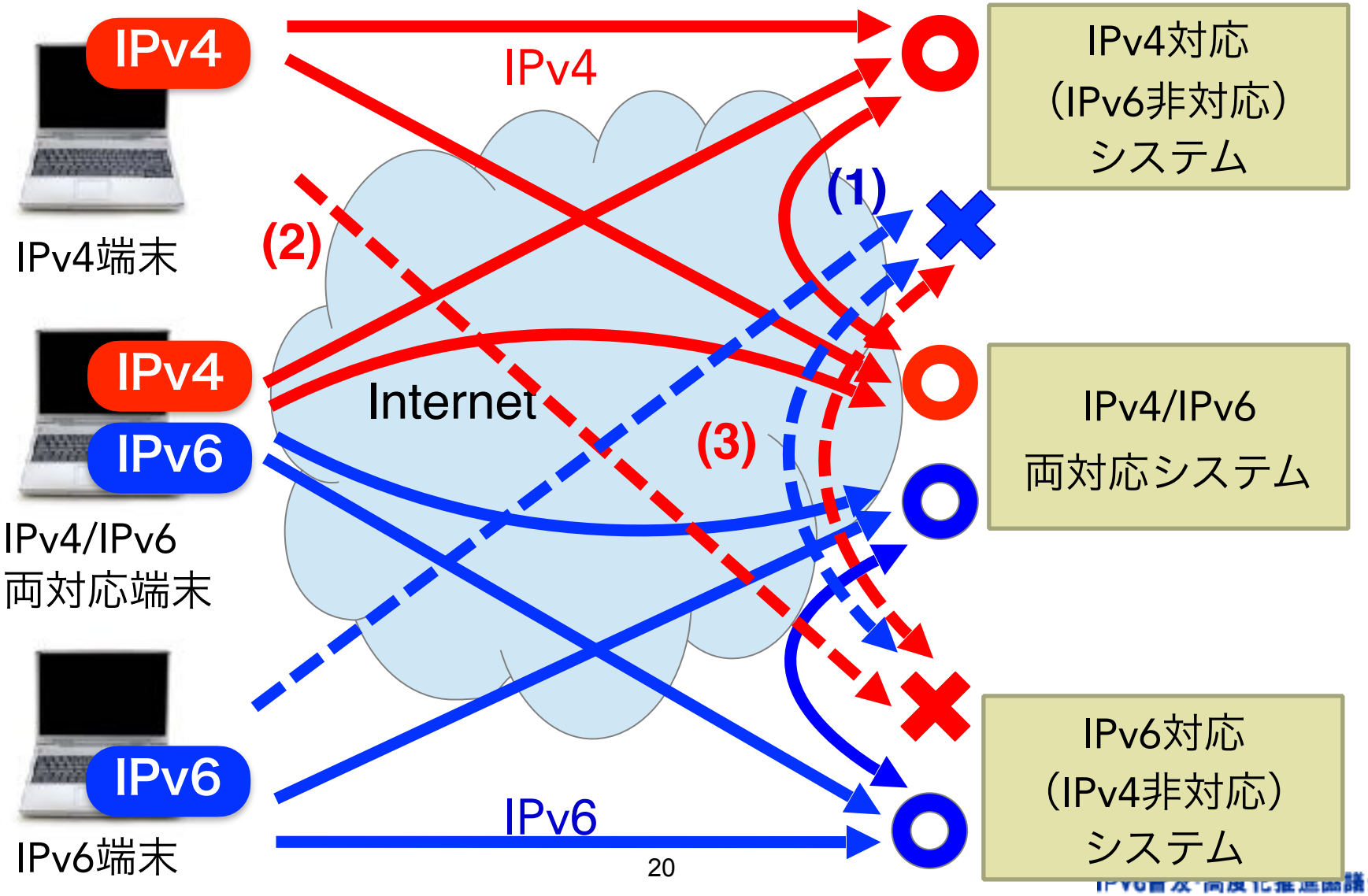
IPv4とIPv6とでは何が違うのか？④

- 他にも機能的にIPv4と異なることがある

- 最も重要なこと：

IPv4とIPv6は直接通信できない

IPv4とIPv6の接続性



IPv6に対応しない場合の影響

1. IPv6のみの環境と通信できない
   - ビジネス機会を損失する
   - システム連携が行えず要件を満たせなくなる

2. 今後、IPv4はサービスレベルが低下していく
   - 通信事業者等によるCGN（Carrier Grade NAT）導入により、遅くなったり、利用できるセッション数が少なくなったりする可能性がある

IPv6に対応しなきゃ！

アプリケーションのIPv6対応の必要性

- ネットワークとサーバがIPv6に対応すれば、IPv6で接続可能
- 接続は可能だが…

サービスが正常に動作しない
かもしれない

例えば

- システム連携がうまくいかない
- 想定外の挙動をする

…

アプリケーションのIPv6対応が不可欠！

1.2. IPv6対応の前に気を付けるべきこと

そのコード、イケてない… (1)

- とある Perl のプログラム

```
use IO::Socket::IP;
$host =  “198.51.100.1” ;
 :
 :
my $sock = IO::Socket::IP->new(
 PeerAddr => $host,
 PeerPort => $port,
 Proto => 'tcp'
) or die  “Error: $!\n” ;
 :
 :
```

このコード、イケてない…(2)

- とある Androidプログラミング書籍におけるソケット通信のサンプルコード

```
public class SocketEx…
…
…
    private final static String IP=“192.168.11.12”;//★変更必須
```

良い子は真似しちゃダメ

どこがイケてない（というかヘン）？

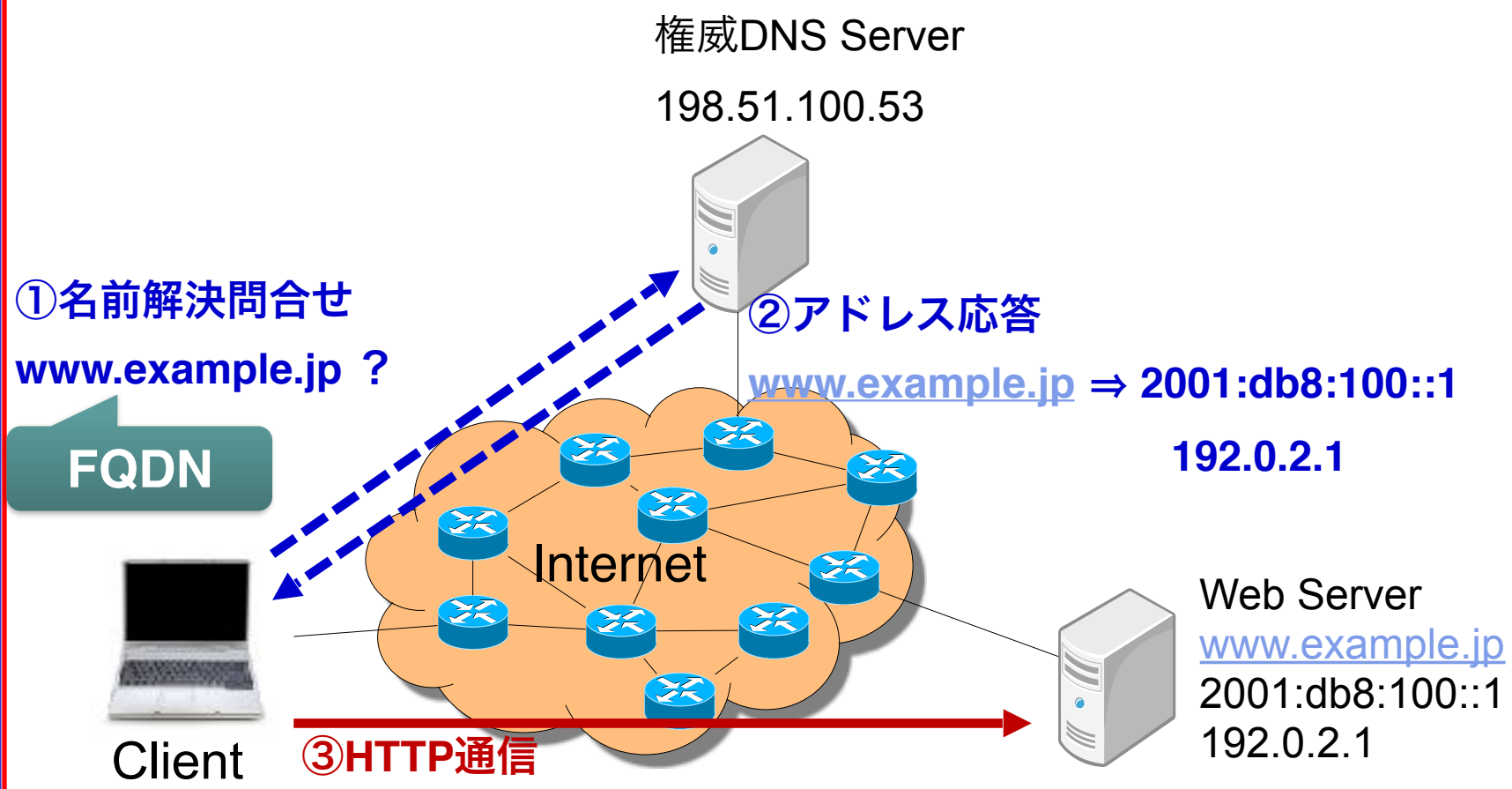
IPアドレスのハードコーディングは NG！

ダメ。ゼッタイ。

$host = “www.example.com”

のようにFQDNで接続先を指定する

ネットワークアクセスの作法＝名前解決を使う



FQDNで接続先を指定し、DNSからアドレス取得

なぜIPアドレス直書きがダメなのか?

	目的	変更・改修の理由
アプリケーション	**機能の提供**	▪ 業務要件の変更 ▪ サービス内容の変更 ▪ ユーザビリティ向上　…,etc.
インフラ	**資源の提供**	▪ 資源管理（**IPアドレス**、サーバラック…） ▪ 性能

同一システムでも変更・改修の理由・時期は異なる

互いに変更の影響を受けるべきではない

アプリケーションは、IPアドレスに依存すべきではない

例）IPアドレスでユーザを識別すべきではない

Cookie内の情報にも気を付ける

- Cookie内に記載される情報の生成がIPv4アドレスを前提としている実装が散見される。このようなアプリケーションに関しては、生成ロジックを変更する必要がある。
- Cookie内の情報としてIPv4アドレスを直接利用している実装がしばしば見られる。特に認証系システムなどでこの種の情報の取扱いがなされている場合が多い。このような実装では、利用者がIPv4/IPv6の両方の空間を利用しており、どちらを利用するかが一位に定まらない場合などに問題が発生する。このような実装の場合、単純にIPv4/IPv6両方に対応させることが困難である。

出典：IPv6普及・高度化推進協議会 セキュリティWG　IPv6対応セキュリティガイドライン（第1.0版）

http://www.v6pc.jp/jp/upload/pdf/swg-IPv6SecurityGuideline_v1.0.pdf

1章まとめ

- アプリケーション開発において重要なIPv4とIPv6の違い
  - アドレス体系が異なる
  - アドレス利用設計が変わる
  - 複数のアドレスを有効範囲に応じて使い分ける
- **IPv4とIPv6は直接通信できない**

- システムのIPv6対応にはアプリケーションの対応が不可欠

- IPアドレスのハードコーディングはダメ。ゼッタイ。

2. IPv6対応アプリケーションの作り方

2.1. プログラミング言語と実行環境

2.2. 通信処理のIPv6対応

2.3. データとしてIPアドレスを扱う箇所の対応

アプリケーションIPv6対応の基本方針①

IPv6対応　＝

IPv4とIPv6の両方で動作する

シングルソースコードで対応

アプリケーションIPv6対応の基本方針②

IPv6対応　＝

IPv4とIPv6の両方で動作する

- IPv6とIPv4の共存期間が長く続く
- これまでIPv4で提供してきたサービスは、今後も継続してIPv4でも動作する必要がある

アプリケーションIPv6対応の基本方針③

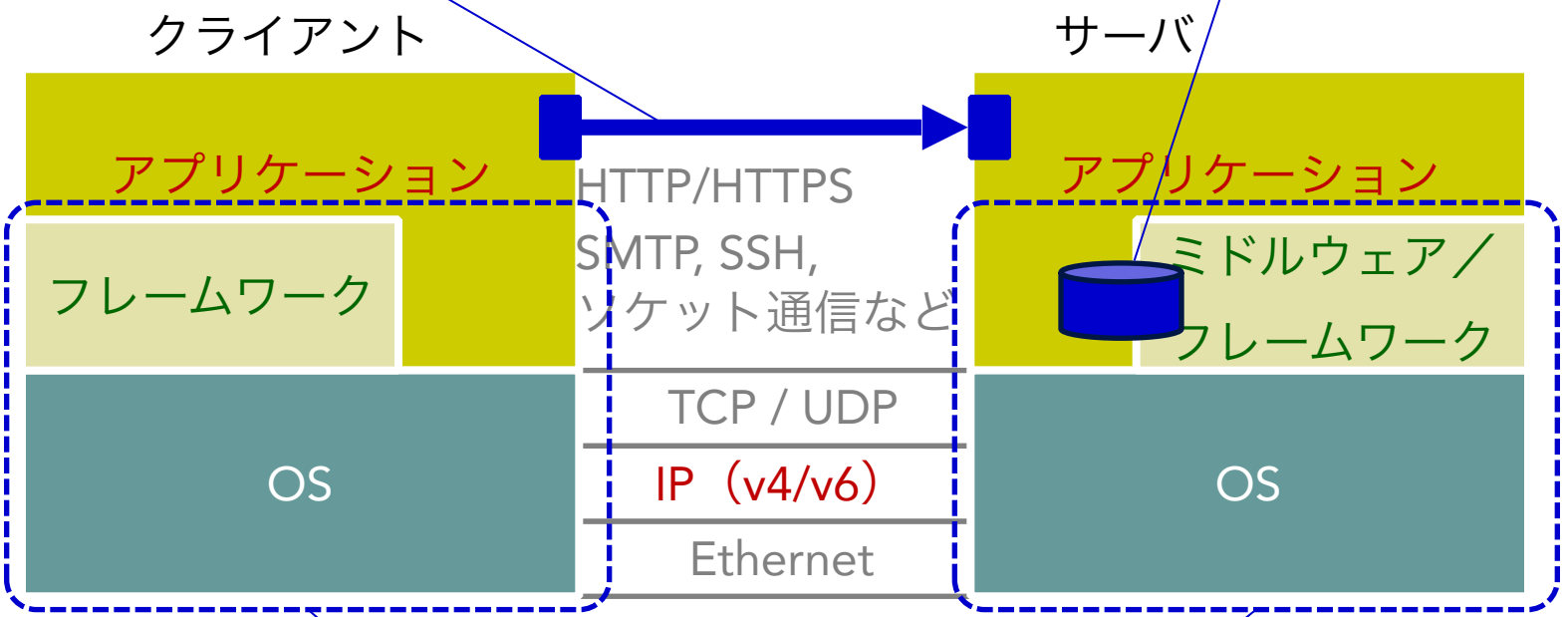
シングルソースコードで対応

- 各開発言語が概ねIPv6に対応しており、プロトコルによって開発言語を分ける必要がなくなった
- アプリケーションの**メンテナンス性を重視**し、プロトコルによって機能差異が生じることを未然に防ぐ

アプリケーションのIPv6対応のポイント

②通信処理をIPv4/IPv6の両方に対応させる

③データとしてIPアドレスを扱う箇所をIPv4/IPv6の両方に対応させる



①IPv4/IPv6両対応のプログラミング言語と実行環境を使う

2.1. プログラミング言語と実行環境

ここでいうIPv4/IPv6両対応とは？

- プログラミング言語と実行環境における IPv4/IPv6両対応とは？

名前解決機構が
IPv4/IPv6両方のアドレスを適切に扱える

IPv4/IPv6両方で通信できる

これらを満たすプログラミング言語、実装環境を利用する

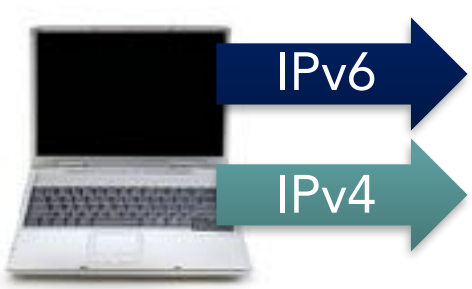
実装上の留意点

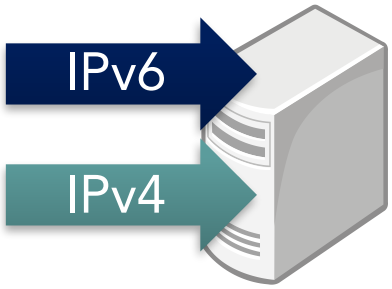
- プログラミング言語、実行環境の選定における留意点
  - 実際には各プロダクトでサポート状況に差異があるため、開発するアプリケーションが提供する機能を考慮し、個別に判断する必要がある
- プログラミングにおける留意点
  - IPv4/IPv6の双方に対応するライブラリ、オブジェクト、関数、データ型を使う
    - 従来（IPv4のみ）のものとは別に用意されていることがある
      - C addrinfo構造体、getaddrinfo()
      - Java InetAddressクラス
      - Perl IO::Socket::IP　　など
  - アドレス検証、変換などはライブラリを有効活用

2.2. 通信処理のIPv6対応

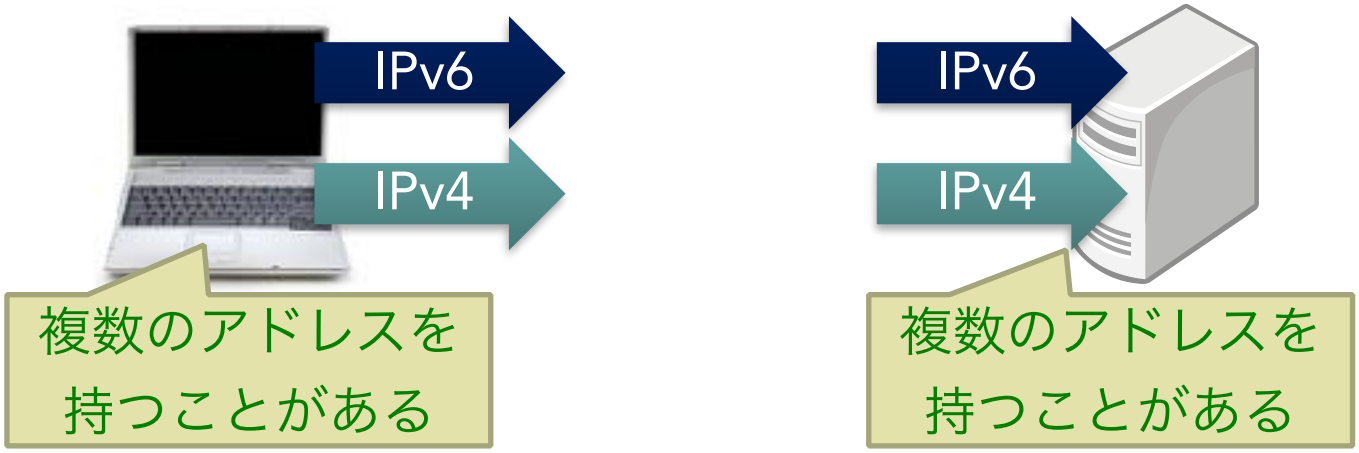
IPv4とIPv6の両方で通信できることとは？

- クライアント
  - IPv4およびIPv6で意図するサーバへ接続できること
- サーバ
  - IPv4およびIPv6で接続を受付けること





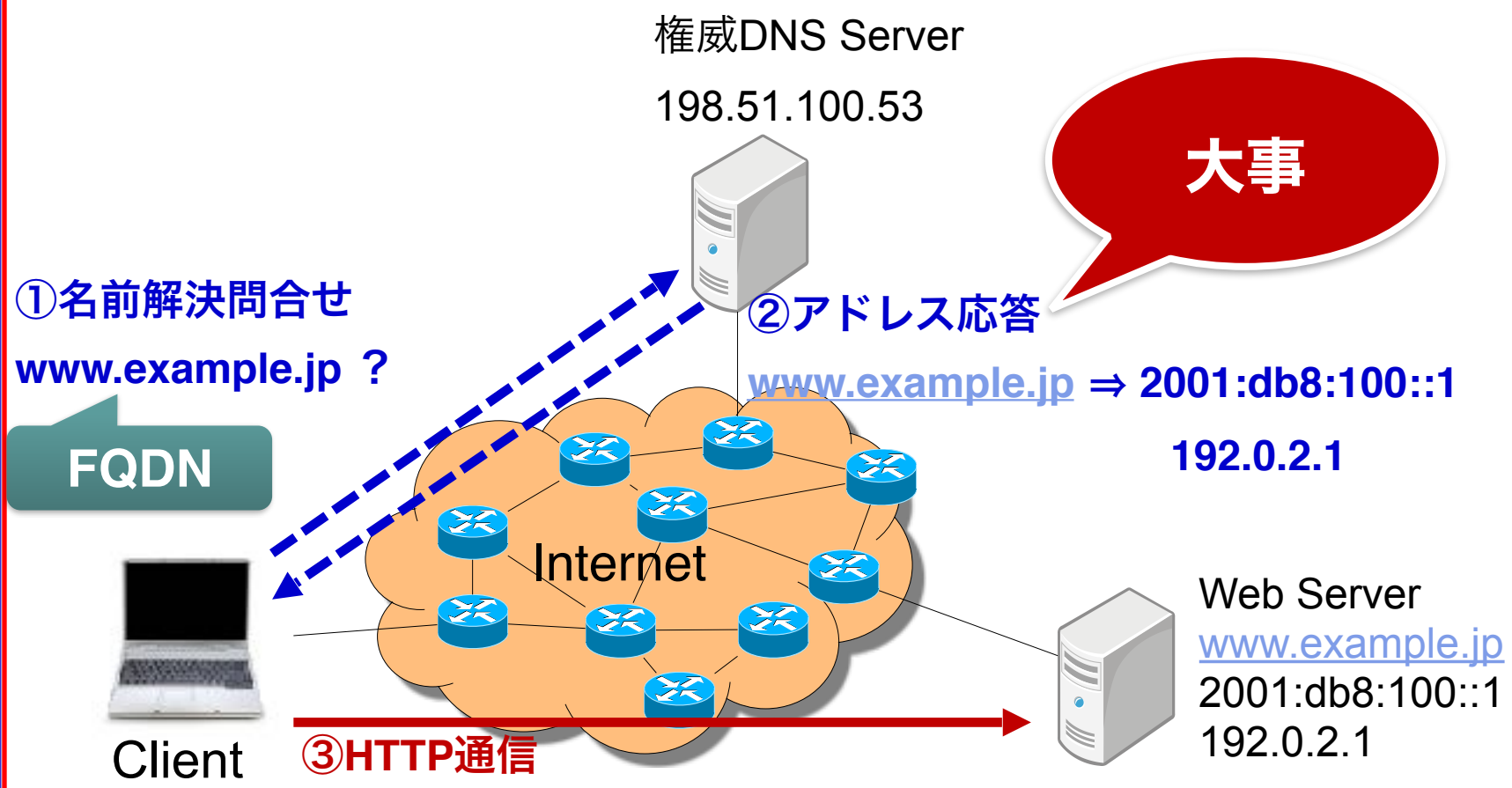
IPアドレスを複数持つことがある



クライアントがどのアドレスにアクセスするかは
サーバ側では予測できない

特定のアドレスに依存したシステムを構成すべきではない

ネットワークアクセスの作法＝名前解決を使う（再掲）



FQDNで接続先を指定し、DNSからアドレス取得

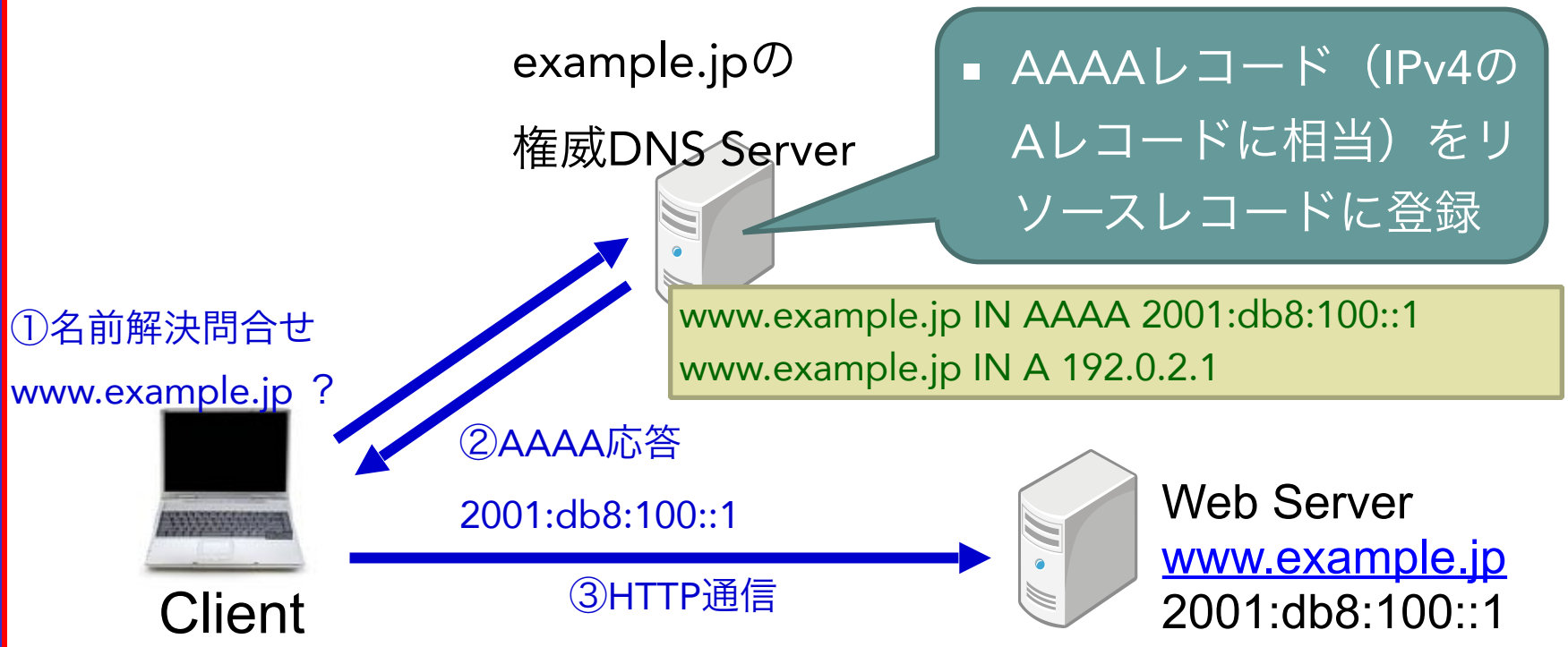
IPv6の名前解決①

- FQDNで接続先を指定してIPv6で通信を行うには、**DNSにてFQDNからIPv6アドレスが名前解決できることが必要不可欠**

- FQDNからIPv6アドレスを名前解決
  - 権威DNSサーバ上で接続先サーバのAAAAレコードにIPv6アドレスが登録されている
  - クライアントから接続先サーバのAAAAレコードが引ける

- アプリケーション開発においては、FQDNのIPv6アドレスが正しく名前解決できることを確認する

IPv6の名前解決②


example.jpの
権威DNS Server
AAAAレコード（IPv4のAレコードに相当）をリソースレコードに登録
www.example.jp IN AAAA 2001:db8:100::1
www.example.jp IN A 192.0.2.1
①名前解決問合せ
www.example.jp ？
②AAAA応答
2001:db8:100::1
③HTTP通信
Client
Web Server
www.example.jp
2001:db8:100::1

通信の試行順序

- RFC6724 Default Address Selection for IPv6

IPv6 > IPv4

- 優先順位が変わるケース
  - デフォルトを変更している環境
  - RFC6724に準拠していない実装

クライアントプログラム

- IPv4/IPv6 両宛先アドレスに接続できるようにする

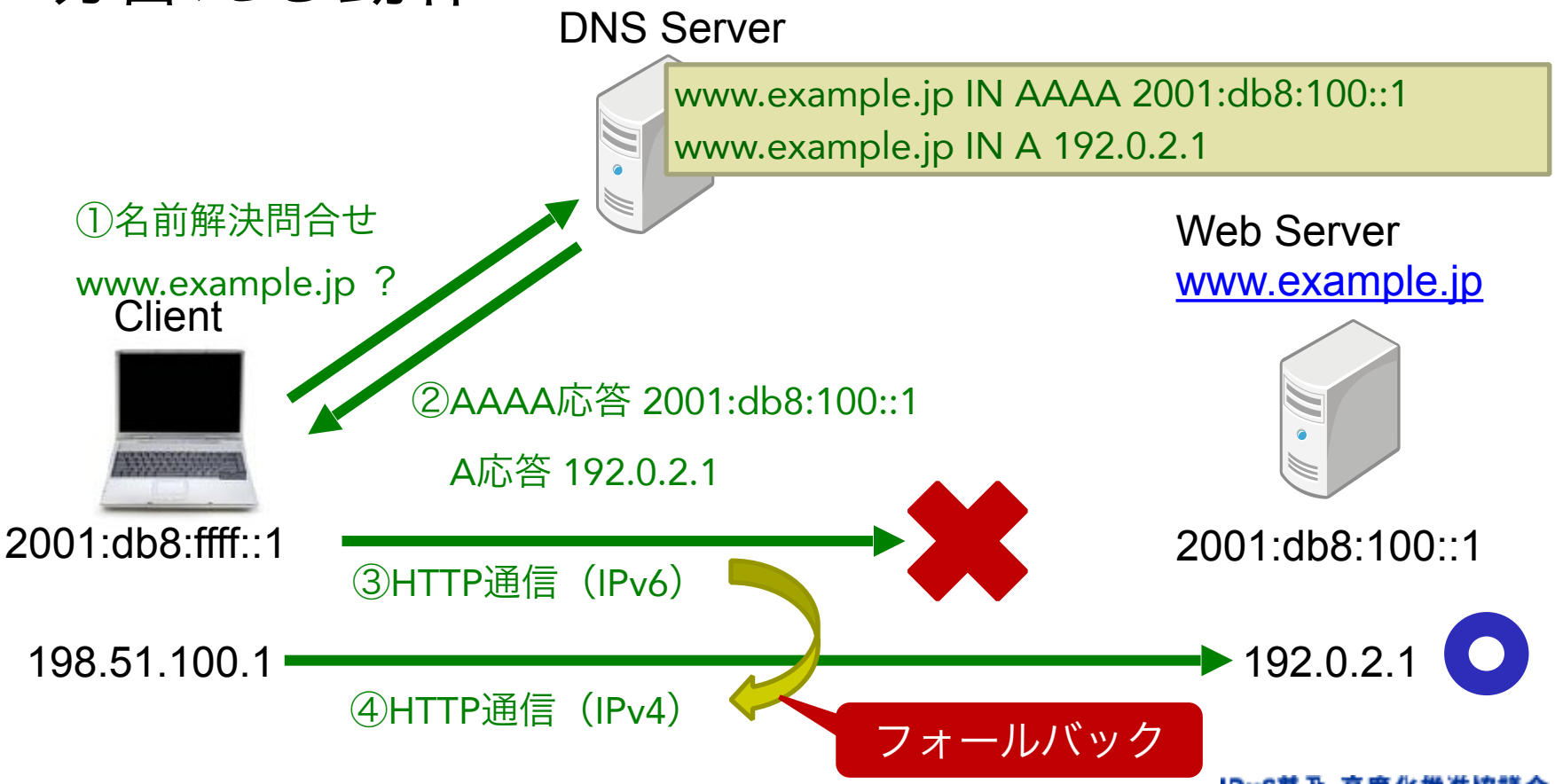
- 接続できない状況も想定し**接続失敗時には別の宛先アドレスに切替えて接続する（フォールバック）**

アプリケーションの作りが悪いと…
- 切替えに時間がかかる
- 正常に切替わらないこともある

ユーザの利便性を損なう

通信処理の補足：フォールバック

- 接続できない場合に別の接続先への接続に切替える動作



想定されるフォールバックの主な原因

サーバ側の問題	サーバが当該のサービスを提供していない - DNS誤登録、障害等
経路の問題	ネットワークの接続性が失われている - ISPの不具合
クライアント側の問題	サーバへの到達性がないアドレスを選択して通信を行おうとしている - グローバルアドレスを使用している閉域網

フォールバックの予防策

サーバ	設定の不備を修正する ・サービスを提供していないIPアドレスをDNSに登録しない ・サービスを適切に提供する
ISP	ネットワークの接続性を健全に保つ
クライアント	IPv6インターネット接続可能なISPと契約する

サーバプログラム

- IPv4/IPv6 両プロトコルでの接続を処理

- 主要なWebサーバプログラムは対応済み
  - Apache HTTP Server
  - Microsoft Internet Information Server (IIS)
  - nginx

iOSアプリ、Androidでは…

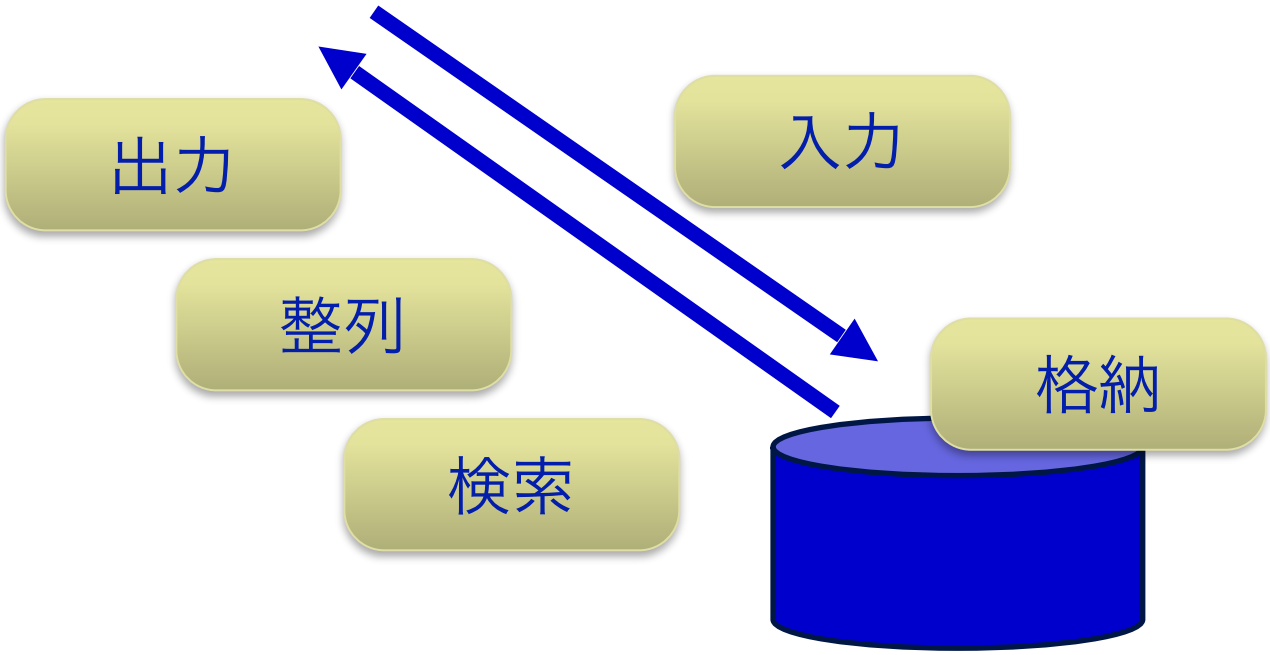
- **プラットフォームが提供するAPIでこの辺を対応**
  ⇒ 開発者に、この辺を意識させない

- **iOSアプリ**
  - 高レベルネットワークフレームワークの使用を推奨
    - WebKit：Webページを読込む複雑なプロセスに対応
    - Cocoa URL：アプリケーションでURLと参照先のリソースを操作
    - CFNetwork.Core Services：さまざまなネットワークタスク

- **Androidアプリ**
  - Web：WebView（Android.webkit.WebView）、
    HttpURLConnection（java.net.HttpURLConnection）
  - Web以外：Socket（java.net.Socket）

2.3. データとしてIPアドレスを扱う箇所の対応

データとしてIPアドレスを扱う箇所



IPv4/IPv6アドレス比較

		IPv4アドレス	IPv6アドレス
アドレス長		32bit	128bit
文字列表記	表記法	8bitずつ区切り、10進数で表記	16bitずつ区切り、16進数で表記（省略表記あり）
	区切り文字	.（ドット）	:（コロン）
	文字列長	15文字以内	39文字以内

サブネットマスク／プレフィックス長を考慮すると、上記＋”/”＋数字3文字

IPv6アドレス表記法

- 特段の事情がない限り RFC5952 の表記ルールに従い表記する（省略表記）

- アドレス表記例

  IPv4）192.0.2.1

  IPv6完全表記）
  2001:0db8:0000:0000:0001:0000:0000:0001
  IPv6省略表記）2001:db8::1:0:0:1

IPv6アドレスの文字列長

- IPv6アドレスの文字列長：39文字
  - プレフィックスを加味すると：43文字

- 例外（39文字を超えることがある）
  - リンクローカルアドレスにゾーンID（スコープID）を付与してインターフェースを識別する場合
    - 例）fe80::1%eth1
  - 一部の特殊アドレス
    - IPv4射影アドレス等
      例）::ffff:192.168.0.1

IPv4射影アドレス
IPv4 アドレスを IPv6 アドレスとして表現するための IPv6 アドレス。上位80ビットに0、81～96ビット目に1、下位32ビットにIPv4アドレスを埋め込む。機器内部での使用に限られ、パケットの始点/終点アドレスには使われない。
出典：
IPv6関連用語集 IPv4-mapped(IPv6) address

IPv6アドレスを扱えない IPアドレス入力・格納

- 15文字までの文字列（varchar(15)）
- 1つの整数として扱う
- 1オクテットずつ4つの整数として扱う

Bad!

そのままではIPv6アドレスを格納できず、エラーとなる

IPv4/IPv6両対応	39文字以内の文字列 [VARCHAR(39)]

Good!

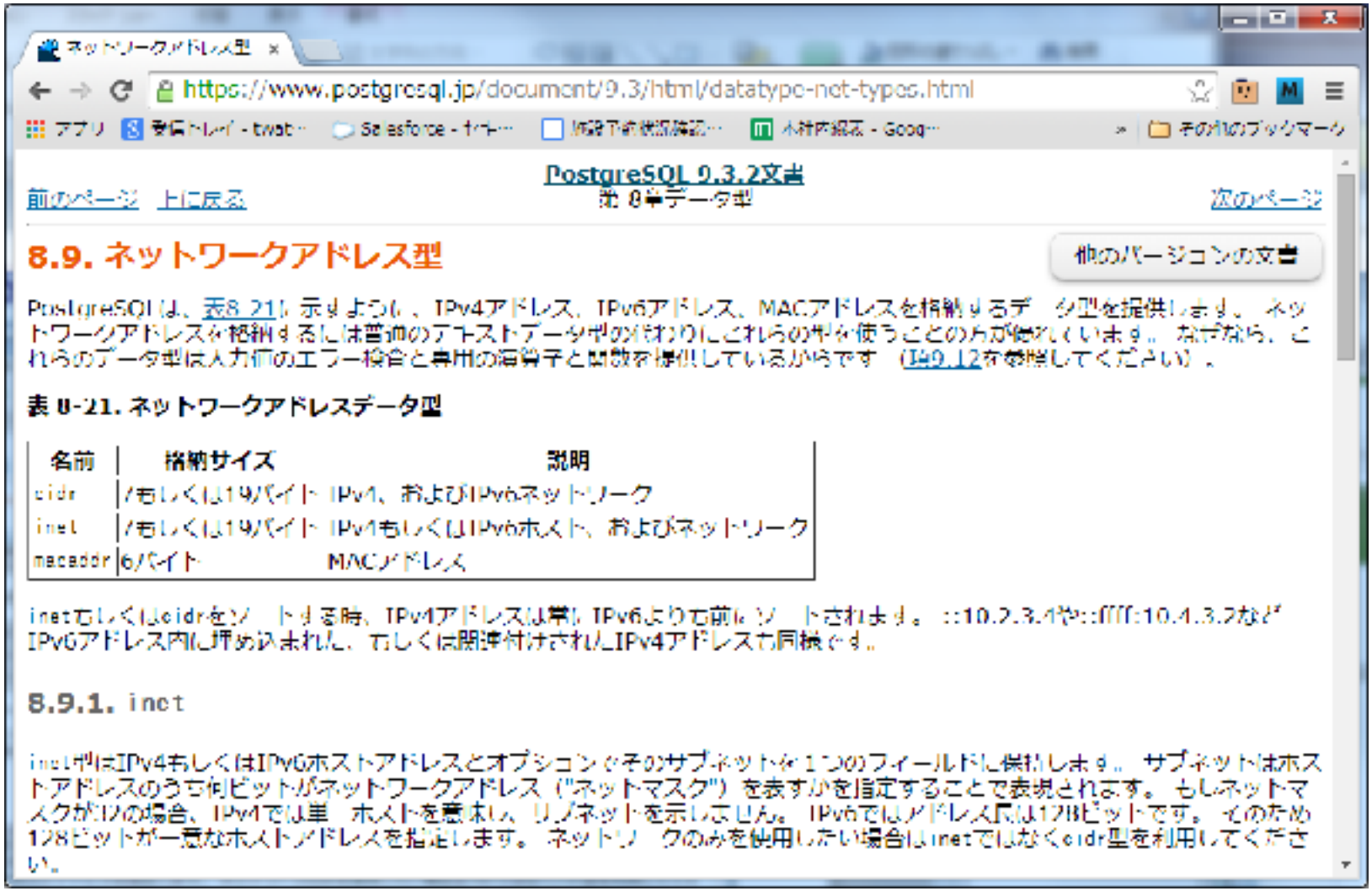
Webフォームからの入力

- 入力値の検証
  - IPアドレスを扱う場合、入力された文字列がIPアドレスとして取りうる値であることを検証
    - IPv4アドレス、IPv6アドレス いずれかとして取りうる値
  - ２箇所で実施可能
    - ブラウザ側（HTML5のForm Validation等）
    - サーバ側
  - アドレス処理ライブラリを利用すると便利
    - 例）PHP
      filter_var($host, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6)

格納、検索、整列、出力

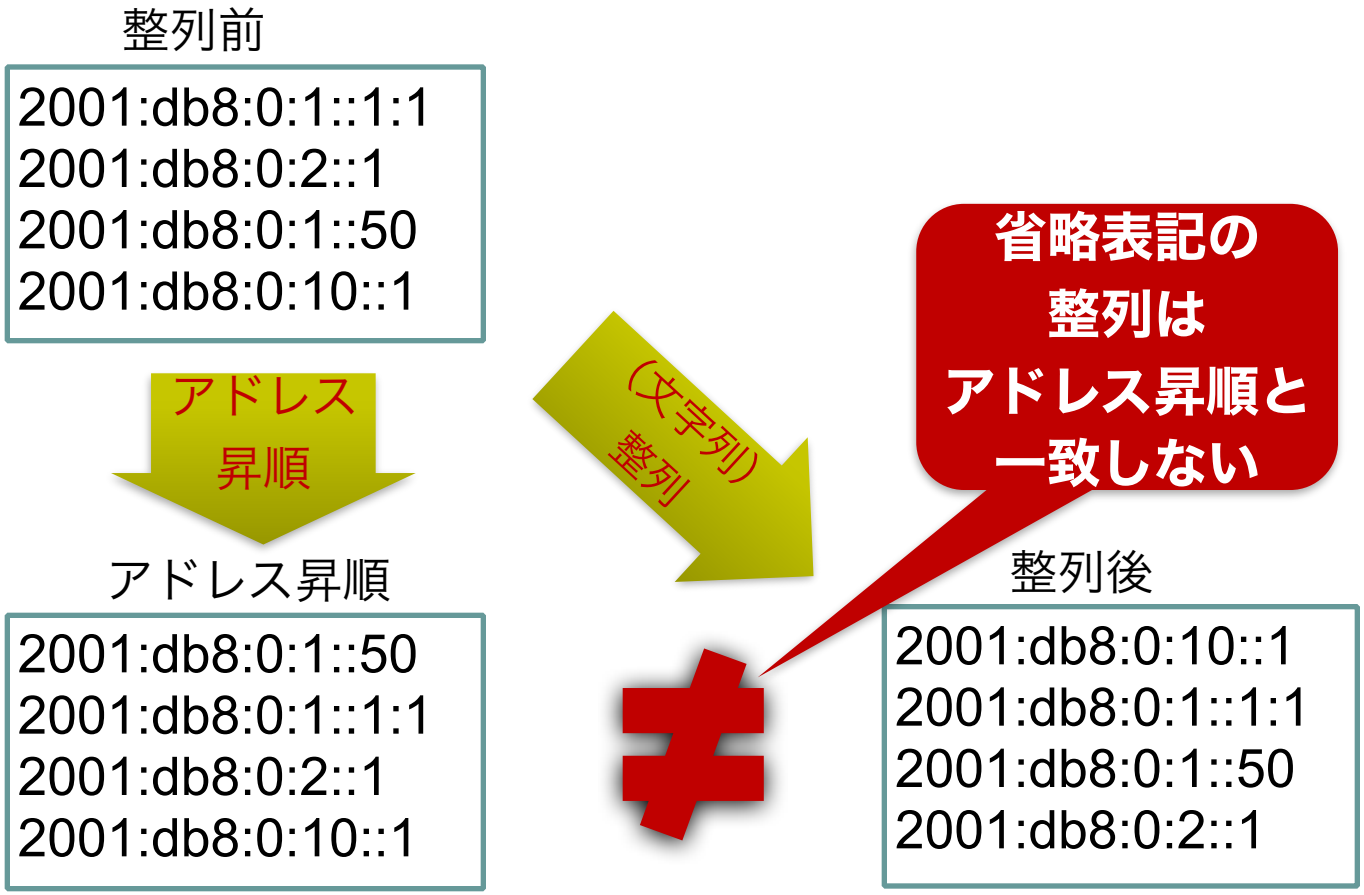
- **IPアドレス型が定義されている場合は、IPアドレス型を使う**
  - 例) PostgreSQLのネットワークアドレス型
- **IPアドレス型が定義されていない場合は、文字列型で完全表記を使う**
  - IPv6完全表記）
    2001:0db8:0000:0000:0001:0000:0000:0001
  - 見やすさを求めるときは、省略表記（RFC5952準拠）で出力
    - 過去に開発されたシステム・ツールでは、RFC5952に準拠しない省略表記が存在しうるので要注意
- **既存システムは、格納領域にIPv6アドレスが収まるかをチェック**

PostgreSQLのネットワークアドレス型



文字列型で扱う場合、なぜ完全表記か？①

- 省略表記のまま整列しても…



文字列型で扱う場合、なぜ完全表記か？②

- 整列は完全表記で行う

整列前

```
2001:db8:0:1::1:1
2001:db8:0:2::1
2001:db8:0:1::50
2001:db8:0:10::1
```

＝

完全表記

```
2001:0db8:0000:0001:0000:0000:0001:0001
2001:0db8:0000:0002:0000:0000:0000:0001
2001:0db8:0000:0001:0000:0000:0000:0050
2001:0db8:0000:0010:0000:0000:0000:0001
```

アドレス昇順

完全表記の整列はアドレス昇順と一致

（文字列）整列

アドレス昇順

```
2001:db8:0:1::50
2001:db8:0:1::1:1
2001:db8:0:2::1
2001:db8:0:10::1
```

＝

整列後

```
2001:0db8:0000:0001:0000:0000:0000:0050
2001:0db8:0000:0001:0000:0000:0001:0001
2001:0db8:0000:0002:0000:0000:0000:0001
2001:0db8:0000:0010:0000:0000:0000:0001
```

文字列型として扱うときの注意点

- 省略表記 ⇔ 完全表記 の変換はライブラリを有効活用する
- 過去に開発されたシステム・ツールでは、RFC5952に準拠しない省略表記が存在しうるので要注意

ログ出力・解析への影響

- 例）Apache HTTP Server ログファイル

```
1 fdb6:5591:2612:10::100 - - [08/Oct/2016:17:52:30 +0900] "GET / HTTP/1.1" 200 144
2 172.16.10.128 - - [08/Oct/2016:18:01:59 +0900] "GET / HTTP/1.1" 200 100
```

  - OSSログ解析プログラムは大抵問題なく処理できる
    - AWStats, Webalizer…
    - 注：アクセス元の国／地域は解析できない場合がある

  - **ログ解析を自作している人は要注意！**
    - アドレス部分の文字列長が長くなる
    - アドレスの区切り文字が変わる

2章まとめ

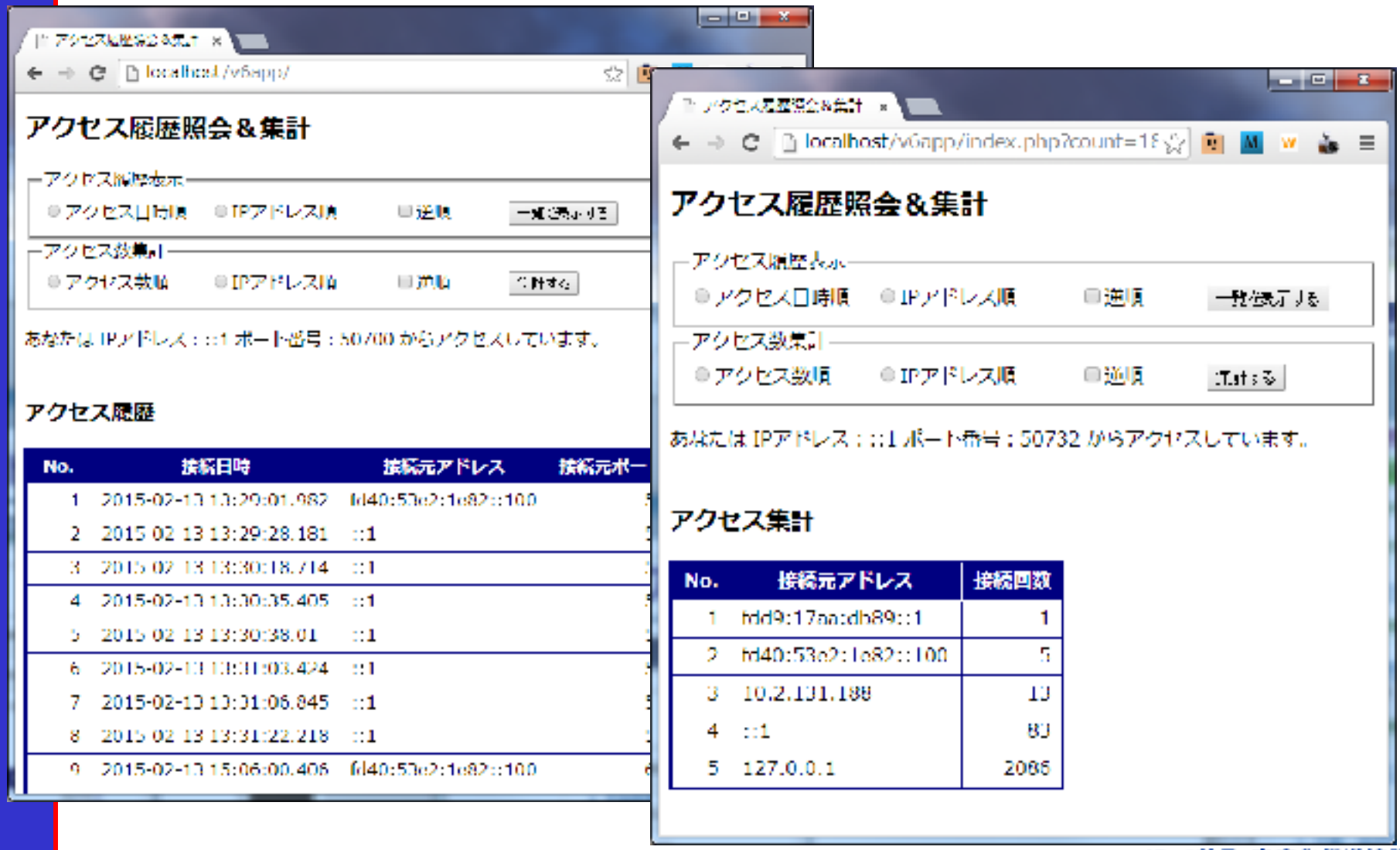
- IPv6対応の基本方針
  - **IPv6対応＝IPv6/IPv4の両方で動作させること**
  - **シングルソースコードで対応する**

- IPv6対応のポイント
  1. **IPv4/IPv6両対応のプログラミング言語と実行環境を使う**
  2. **通信処理をIPv4/IPv6の両方に対応させる**
  3. **データとしてIPアドレスを扱う箇所をIPv4/IPv6の両方に対応させる**

3. PHPサンプルコード

Sample1
アクセス履歴表示・集計Webアプリ（PHP)

サンプルアプリ1：アクセス履歴照会&集計（PHP）



コーディングの留意点

- 関数、データ型はIPv4/IPv6両対応のものを使用する
  - データ型：文字列型
  - 関数：
    - get_dns_record()
    - gethostbyaddr()

gethostbyname()
はIPv6非対応

- ライブラリ、フィルタを用いて入力値検証、変換
  - ライブラリ：Net_IPv6
  - フィルタ：FILTER_VALIDATE_IP

【PHP】名前解決

- 正引き
  - dns_get_record
    - 引数で指定したRRの情報を取得して、配列で返す

```
// www.iajapan.org の IPv6 アドレス (AAAA レコード) を検索
$result = dns_get_record('www.iajapan.org', DNS_ALL);
```

- 逆引き
  - gethostbyaddr
    - 引数で指定したアドレスに対応するホスト名を返す

```
$result = gethostbyaddr('192.168.0.1');
$result = gethostbyaddr('2001:db8:0:1::1:1');
```

【PHP】アドレス処理

- PEAR::Net_IPv6
  - IPv6 アドレスに関する処理を行う
    - http://pear.php.net/package/Net_IPv6
  - 下記のようなメソッドを提供する
    - checkIPv6()　　：IPv6のアドレスか検証
    - compress()　　：IPアドレスの短縮
    - uncompress()　：IPアドレスの伸長
    - isInNetmask()　：IPが指定したアドレス空間にあるかどうかを調べる

Sample1 処理フロー

- 履歴書込み

接続元アドレスを取得

格納先のDBのデータ型が文字列の場合は、省略表記を完全表記に展開

DBにレコード追加（INSERT）

- 履歴一覧表示・集計

ユーザ入力値から整列キー、昇順／降順、集計フラグを取得

ユーザ入力値に従い、DBから履歴を読み込み、昇順／降順、集計して表示

Sample1 コード解説 (1)

index.php

```
<?php
  require_once 'settings.php';
  require_once 'modules.php';

  $now = date('Y/m/d H:i:s');
  $array_access = array (
    'source_addr' => filter_input(INPUT_SERVER, 'REMOTE_ADDR',
                                  FILTER_VALIDATE_IP),
    'source_port' => filter_input(INPUT_SERVER, 'REMOTE_PORT',
                                  FILTER_VALIDATE_INT),
    'server_addr' => filter_input(INPUT_SERVER, 'SERVER_ADDR',
                                  FILTER_VALIDATE_IP),
    'access_time' => $now,
  );

  $logging = write_history($array_access);
```

filter_input() でユーザ入力値がIPアドレス形式であることを検証

関数化したアクセス履歴書込みを呼び出し

Sample1 コード解説 (2)

index.php

```
$sort_mode = array (
   'key' => filter_input(INPUT_GET, 'sort_key', FILTER_VALIDATE_REGEXP,
            array('options' => array('regexp' =>
            '/(^access_date$|^source_addr$|^source_port$|^count$)/'))),
   'desc' => filter_input(INPUT_GET, 'desc', FILTER_VALIDATE_REGEXP,
            array('options' => array('regexp' => '/(^desc$|^asc$)/'))),
   'count' => filter_input(INPUT_GET, 'count', FILTER_VALIDATE_INT,
            array('options' => array('min_range' => 0, 'max_range' => 1))),
 );
 if (!$sort_mode['key']){
   $sort_mode['key'] = 'access_date';
 }
 $history = display_history($sort_mode);
?>
～以下、HTML部分につき割愛～
```

関数化したアクセス履歴表示を呼び出し

Sample1 コード解説 (3)

modules.php

```
<?php
// //////////////////////////
// 履歴書込み関数

function write_history ($array_access) {
  global $DSN;

  // ========================================
  // DBのデータ型がvarcharの場合には、省略表記を完全表記に展開
  // ========================================
  if (constant('STORE_TYPE') !== 'INET') {
    require_once 'Net/IPv6.php';
    if (Net_IPv6::checkIPv6($array_access['source_addr'])) {
      $source_addr = Net_IPv6::uncompress($array_acc
                                          TRUE);
    } else {
      $source_addr = $array_access['source_addr'];
    }
  }
```

Net_IPv6::checkIPv6() で変数がIPv6アドレスであることを検証

STORE_TYPE は、このプログラムで独自に定義

省略表記を完全表記に展開（第2引数をTRUEにすることで完全表記）

Sample1 コード解説 (4)

modules.php

```
$query = 'INSERT INTO access_history ( access_date, source_addr,
                        source_port) VALUES (now(), :ip, :port)';
 if ($dbh = new PDO($DSN)) {
   $sth = $dbh->prepare($query);
   $sth->execute(array(':ip' =>  $array_access['source_addr'],
                                    ':port' => $array_access['source_port']));
   $err_code = $sth->errorCode();
   if ($err_code === '00000'){
     return OK;
   } else {
     return WRITE_ERROR;
   }
 } else {
   echo "DB connection error";
   return OPEN_ERROR;
 }
}
```

Sample1 コード解説 (5)

modules.php

```
// //////////////////////////
// 履歴表示＆アクセス数集計関数

function display_history ($sort_mode) {
  global $DSN;
  if ($sort_mode['count']){
    // 集計時のクエリ作成

    $query = 'SELECT source_addr, count(source_addr) FROM access_history
                          GROUP BY source_addr';
    if ($sort_mode['key'] === 'source_addr'
                                    || $sort_mode['key'] === 'count' ){
      $query .= ' ORDER BY ' . $sort_mode['key'];
      if ($sort_mode['desc']) {
        $query .= ' ' . $sort_mode['desc'];
      }
    }
```

整列時の指定は、
通常のSQL（ORDER BY キー）

Sample1 コード解説 (6)

modules.php

```
} else {
  // 履歴一覧時のクエリー作成

  $query = 'SELECT * FROM access_history';
  if ($sort_mode['key'] && $sort_mode['key'] !== 'count' ){
    $query .= ' ORDER BY ' . $sort_mode['key'];
    if ($sort_mode['desc']) {
      $query .= ' ' . $sort_mode['desc'];
    }
    $query .= ' NULLS LAST';
  }
}
```

整列時の指定は、
通常のSQL（ORDER BY キー）

Sample1 コード解説 (7)

modules.php

```
// ==================================================
 // DB接続＆クエリ実行

 // ==================================================
 $dbh = new PDO($DSN);
 if ($dbh) {
   $sth = $dbh->prepare($query);
   $sth->execute();
   $result = $sth->fetchAll();
   $sth->errorCode();
 } else {
   echo "DB connection error";
 }
```

Sample1 コード解説 (8)

modules.php

```
// ================================================
 // 出力整形
 // ================================================
 if (constant('STORE_TYPE') !== 'INET') {
   // 文字列で格納されている場合は、省略表記にするためにライブラリを呼び出す
   require_once 'Net/IPv6.php';
 }
 if ($sort_mode['count']) {
   $ret_string = '<H2>アクセス集計</H2><TABLE border="1"><TR><TH>No.</
TH><TH>接続元アドレス</TH><TH>接続回数</TH></TR>';
   $size = sizeof($result);
   for ($loopcnt = 0; $loopcnt < $size; $loopcnt++){
```

Sample1 コード解説 (9)

modules.php

```
if (constant('STORE_TYPE') !== 'INET') {
    // 文字列で格納されている場合は、省略表記にする
    if (Net_IPv6::checkIPv6($result[$loopcnt]['source_addr'])){
      $source_addr =
                  Net_IPv6::compress($result[$loopcnt]['source_addr']);
    } else {
      $source_addr = $result[$loopcnt]['source_addr'];
    }
  } else {
    $source_addr = $result[$loopcnt]['source_addr'];
  }

  $ret_string .= "<TR><TD align='right'>" . ($loopcnt +1) . "</TD><TD>"
      . $source_addr . "</TD><TD align='right'>"
      . $result[$loopcnt]['count'] . "</TD></TR>\n";
}
$ret_string .= '</TABLE>';
```

完全表記を省略表記に変換（見やすさ重視）

Sample1 コード解説 (10)

modules.php

```
} else {
   $ret_string = '<H2>アクセス履歴</H2><TABLE border="1"><TR><TH>No.</
TH><TH>接続日時</TH><TH>接続元アドレス</TH><TH>接続元ポート番号</
TH></TR>';
   $size = sizeof($result);
   for ($loopcnt = 0; $loopcnt < $size; $loopcnt++){
     if (constant('STORE_TYPE') !== 'INET') {
       // 文字列で格納されている場合は、省略表記にする
       if (Net_IPv6::checkIPv6($result[$loopcnt]['source_addr'])){
         $source_addr =
                        Net_IPv6::compress($result[$loopcnt]['source_addr']);
       } else {
         $source_addr = $result[$loopcnt]['source_addr'];
       }
     } else {
       $source_addr = $result[$loopcnt]['source_addr'];
     }
```

完全表記を省略表記に変換（見やすさ重視）

Sample1 コード解説 (11)

modules.php

```
    $ret_string .= "<TR><TD align='right'>" . ($loopcnt +1)
        . "</TD><TD>"
        . $result[$loopcnt]['access_date'] . "</TD><TD>"
        . $source_addr . "</TD><TD align='right'>"
        . $result[$loopcnt]['source_port'] . "</TD></TR>\n";
   }
   $ret_string .= '</TABLE>';
 }
 return $ret_string;
}

?>
```

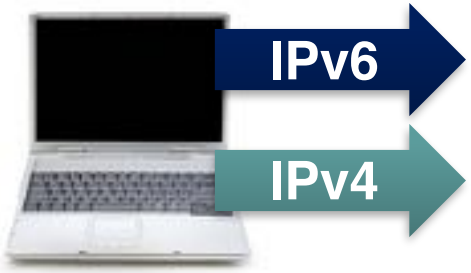
Sample 2
ソケットクライアントWebアプリ（PHP）

Sample2 どんなアプリ？

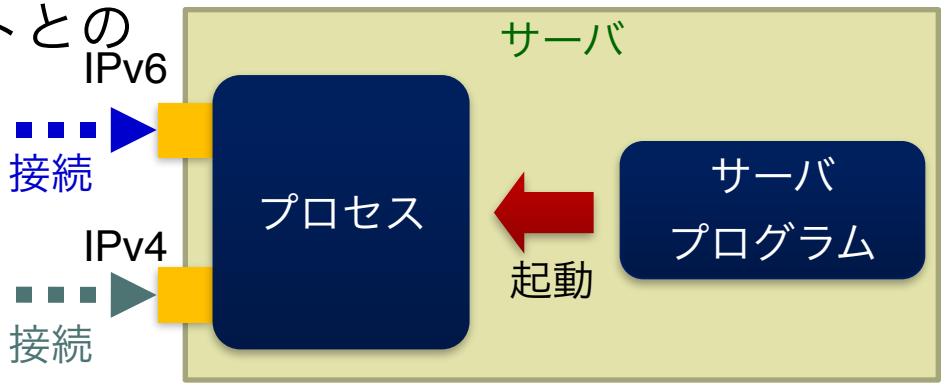
- フォームから入力されたホスト、ポートに対してソケット接続を行い、サーバ側の出力をそのままWebページに出力

ソケット通信？

- クライアント
  - （IPv4/IPv6を問わず）任意のホスト、ポートに対してソケット接続し、サーバとの間でデータを送受信



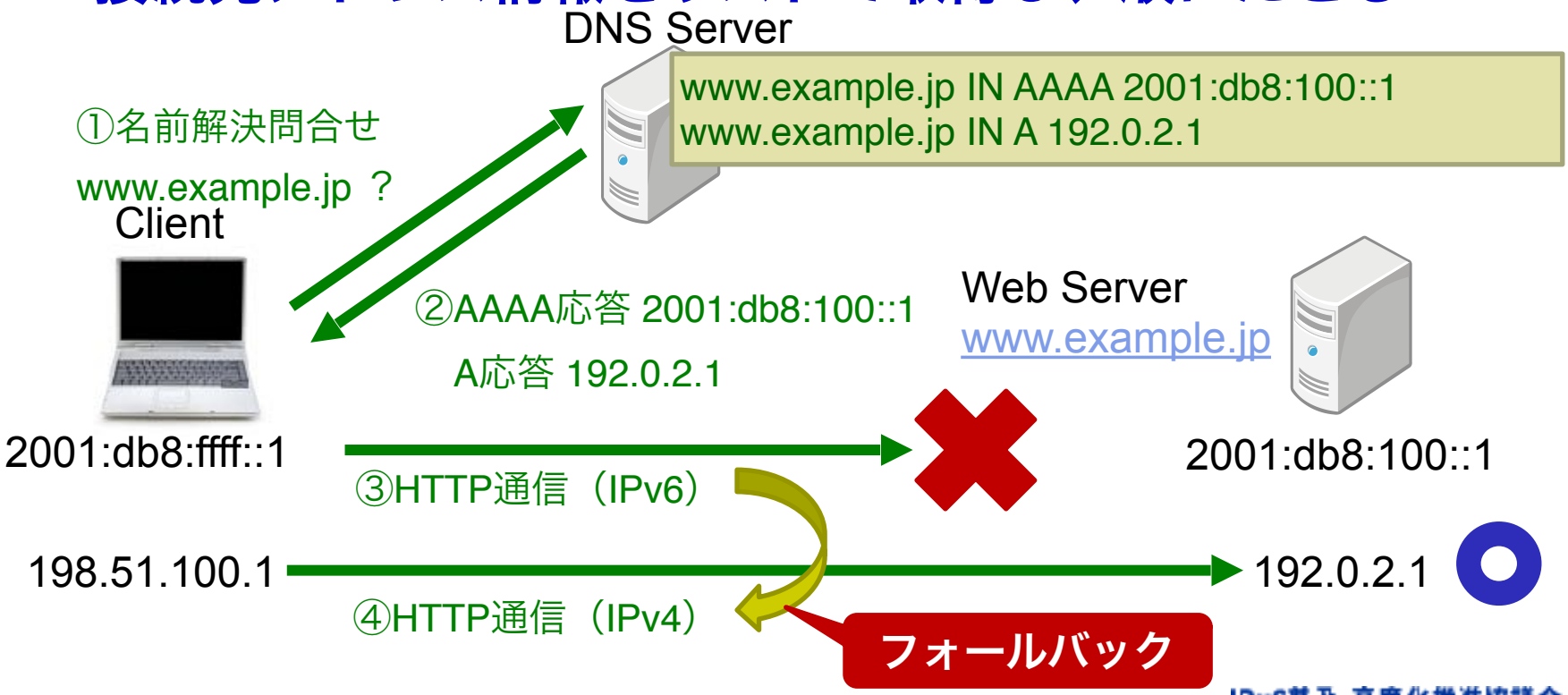
- サーバ
  - （IPv4/IPv6を問わず）任意のポートでソケット接続を待ち受け、接続したクライアントとの間でデータを送受信
  - 複数のソケットを生成するデュアルスタック対応サーバプログラム



サーバ側サンプルコードは割愛

クライアントプログラムのポイント

- フォールバック：接続できない場合に別の接続先への接続に切替える動作
  - **接続先アドレス情報をリストで取得し、順にたどる**



コーディングの留意点

- 関数、データ型はIPv4/IPv6両対応のものを使用する
  - データ型：文字列型
  - 関数：
    - get_dns_record()
    - gethostbyaddr()

gethostbyname()
はIPv6非対応

- ライブラリ、フィルタを用いて入力値検証、変換
  - ライブラリ：Net_IPv6
  - フィルタ：FILTER_VALIDATE_IP

Sample2 処理フロー

ユーザ入力値から
接続先アドレス（リスト）を
取得

↓

接続先アドレス（リスト）に、
順にソケットを生成して接続

Sample2 コード解説 (1)

```
<?php
  $IS_DEBUG = 0;
  $host = filter_input(INPUT_GET, 'host');
  $port = filter_input(INPUT_GET, 'port', FILTER_VALIDATE_INT);

  if ($host && $port){
    $addresses = array();
    if ($host_addr = filter_var($host, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6)){
      $addresses[0]['domain'] = AF_INET6;
      $addresses[0]['address'] = $host_addr;
    } elseif ($host_addr = filter_var($host, FILTER_VALIDATE_IP,
FILTER_FLAG_IPV4)){
      $addresses[0]['domain'] = AF_INET;
      $addresses[0]['address'] = $host_addr;
```

- フィルタを用いて変数がIPv6/IPv4 アドレスか判断
- アドレスからプロトコルに応じたプロトコルファミリを設定

Sample2 コード解説 (2)

```
} else {
  $host_list = dns_get_record($host);
  $size = sizeof($host_list);
  for ($loopcnt = 0; $loopcnt < $size; $loopcnt++){
    if ($host_list[$loopcnt]['type'] === 'AAAA'){
      $addresses[$loopcnt]['domain'] = AF_INET6;
      $addresses[$loopcnt]['address']
                    = $host_list[$loopcnt]['ipv6'];
    } else {
      $addresses[$loopcnt]['domain'] = AF_INET;
      $addresses[$loopcnt]['address']
                    = $host_list[$loopcnt]['ip'];
    }
  }
}
$size = sizeof($addresses);
$message = "接続先ホスト名 " . $host . " ポート番号 " . $port . "<BR>\n";
```

ホスト名の場合にはDNSからアドレスをリストで取得

gethostbyname()は、IPv6非対応

- リストの数だけ、アドレスを取得し接続先候補とする
- IPv6 は AAAA レコード、IPv4 は A レコードに格納

Sample2 コード解説 (3)

```
    $connect_flag = 0;
    for ($loopcnt = 0; $loopcnt < $size && $connect_flag === 0; $loopcnt++){
      if (($socket = socket_create($addresses[$loopcnt]['domain'],
SOCK_STREAM, SOL_TCP)) === FALSE){
        $error_code = socket_last_error();
        $error_msg = socket_strerror($error_code);
        $message .= "connect to " . $addresses[$loopcnt]['address'] . "<BR>\n";
        $message .= 'socket create error: [' . $error_code . '] ' . $error_msg . "<BR>\n";
      } else {
        $message .= 'socket connect (' . ($loopcnt +1) . ') : ' . $addresses[$loopcnt]
['address'] . " port: " . $port . "<BR>\n";
```

ソケット作る

Sample2 コード解説 (4)

```
      if (socket_connect($socket, $addresses[$loopcnt]['address'], $port)){
        $connect_flag = 1;
        $response = socket_read($socket, 1024);
        $message .= "サーバからのメッセージ：" . '<div style="margin: 10px">' .
$response . '</div>' . "<BR>\n";
      } else {
        $error_code = socket_last_error();
        $error_msg = socket_strerror($error_code);
        $message .= 'socket connect error: [' . $error_code . '] ' . $error_msg .
"<BR>\n";
      }
      socket_close($socket);
    }
  }
} else {
  $message = "接続先ホスト名 " . $host . " もしくはポート番号 " . $port . "が入力さ
れていません";
}
```

接続する

切断する

Sample2 コード解説 (5)

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>Socket通信クライアント（デュアルスタック版）</title>
  </head>
  <body>
    <H1>Socket通信クライアント（デュアルスタック版）</H1>
    <form action="<?php echo filter_input(INPUT_SERVER, 'PHP_SELF',
FILTER_SANITIZE_URL)?>" method="GET">
      接続先ホスト <input type='text' name='host' value='<?php echo $host; ?>'>
      ポート番号 <input type='text' name='port'value='<?php echo $port; ?>'>
      <input type="submit" value="実行する">
    </form>
    <HR>
    <?php echo $message; ?>
  </body>
</html>
```

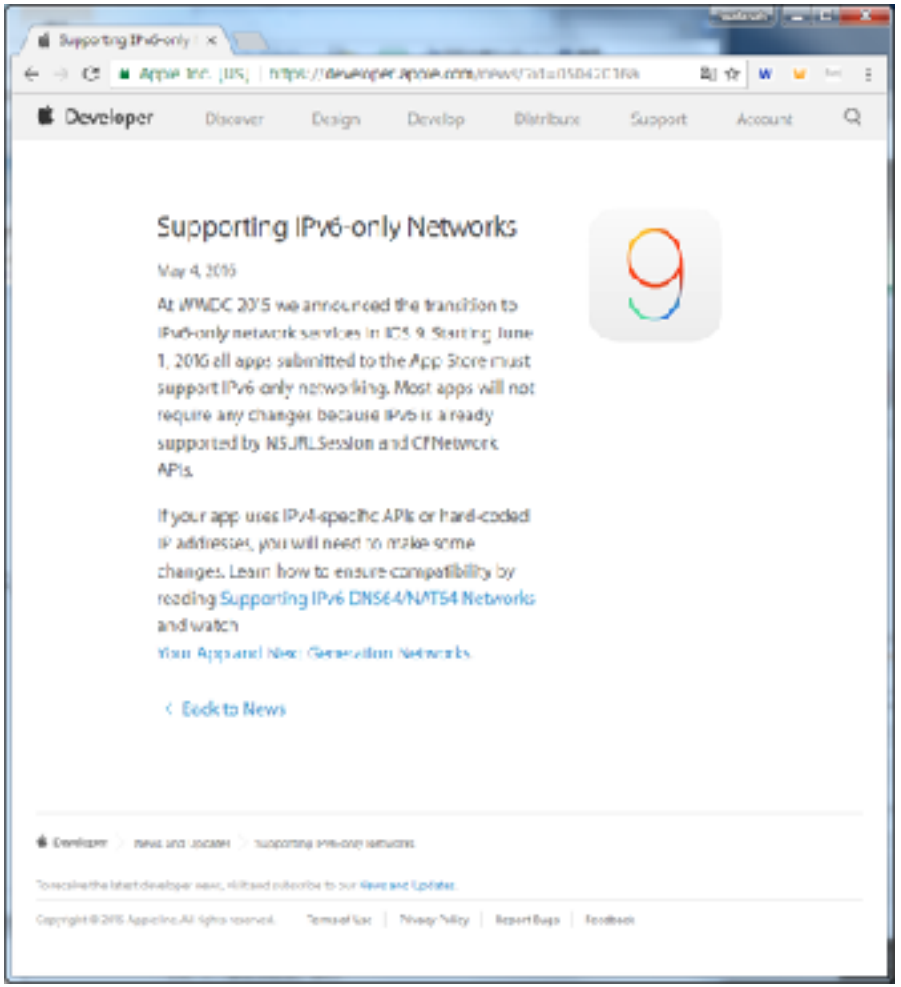
4. Apple の IPv6 対応解説

今年5月のAppleのアナウンス

- 2016/6/1 から、App Store に載せるアプリは、

  IPv6-only ネットワークで動作しないといけない

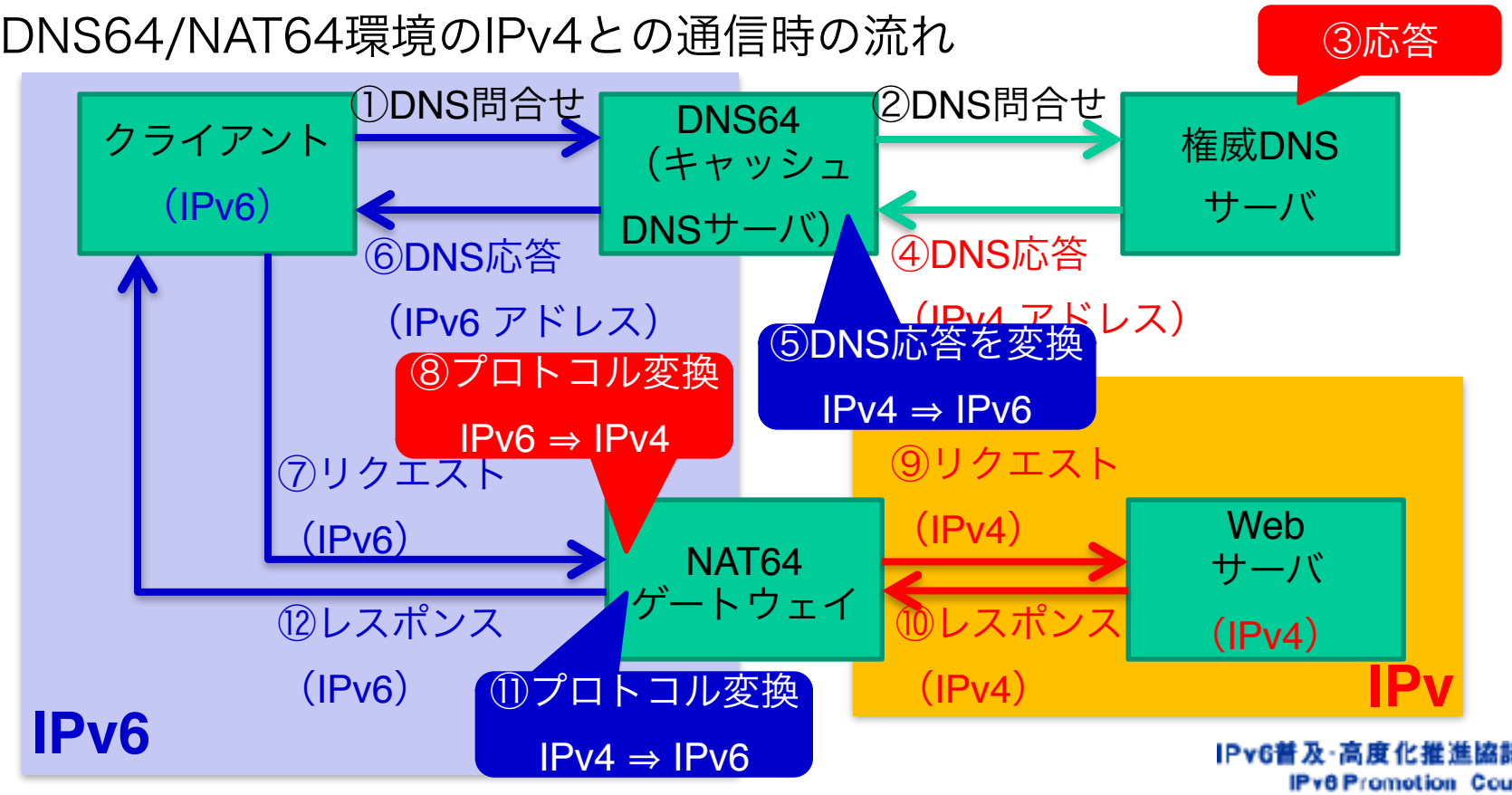
  - **ほとんどのアプリは何も変更しなくて大丈夫なはず**
  - もし、IPv4 固有の API や IPアドレスをハードコードしていたら、**（Networking Overview の）「Supporting IPv6 DNS64/NAT64 Networks」を読んで対応してね**



https://developer.apple.com/news/?id=05042016a

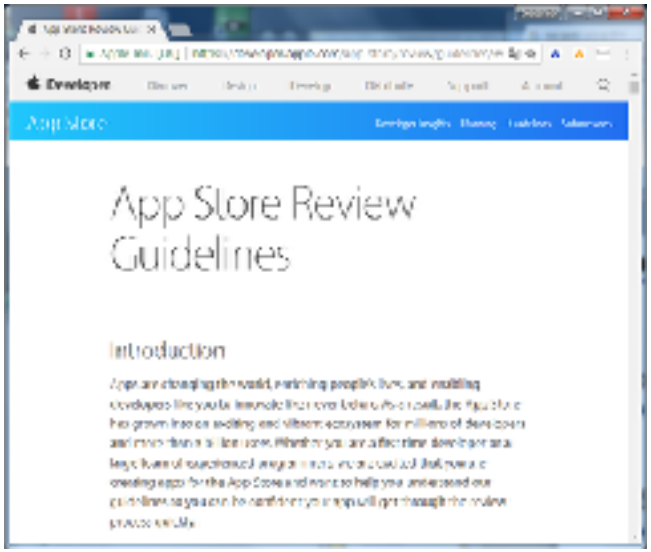
DNS64/NAT64って何？

- IPv6 をベースに IPv4 へのアクセスを変換させる仕組み
  - DNS64 で DNS応答の IPv4 アドレスを IPv6 アドレスに変換
  - NAT64 で通信を変換
- DNS64/NAT64環境のIPv4との通信時の流れ



App Store Review Guidelines でも言及

- App Store Review Guidelines
  - https://developer.apple.com/app-store/review/guidelines/



- 2.5 Software Requirements 内の 2.5.5 にて言及
  - 「IPv6ネットワークでレビューするよ」
  - 「IPv6アドレッシングに対応していないと、レビューで落ちるかもしれない」

本当に確認するらしい

実際にリジェクトされた人も出ている

3.1. Apple の IPv6 対応、解説

～Networking Overviewを読み解く～

Apple Networking Overview って何？

- Apple 社の開発者向けサイトのドキュメントコーナーに掲載されているドキュメント
  - 日本語版あり「ネットワーキング オーバービュー」
  - https://developer.apple.com/jp/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/Introduction/Introduction.html

- ネットワークベースのソフトウェア開発する際に、通信速度、接続性、信頼性など、状況の変化に追随できるようにする方法を解説した文書

Networking Overview の構成

- ネットワーク通信について
- 現実的なネットワークの設計
- ネットワーク処理の要件の評価
- ネットワークサービスの探索と広告
- ウェブやマルチメディア コンテンツの表示
- HTTP/HTTPS要求の処理
- ソケットや ソケットストリームの使い方
- ネットワーク通信の セキュリティ機能
- プラットフォーム特有の ネットワーク技術
- ネットワーク処理において 犯しがちな誤りの回避
- **IPv6 DNS64/NAT64 ネットワークのサポート**

- IPv6 を採用する理由
- DNS64/NAT64 による移行 ワークフロー
- IPv6 および App Store の要件
- IPv6 をサポートする際の よくある障壁
- IPv6 DNS64/NAT64 の互換性の保証
- 関連情報

IPv6を採用する理由

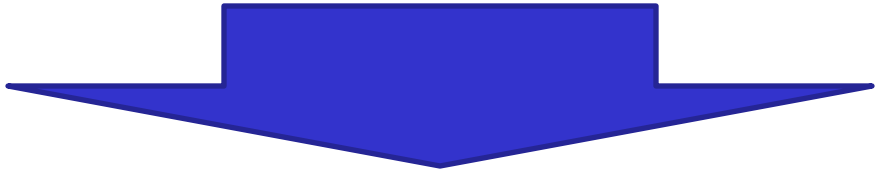
- **IPv4 アドレスの枯渇**
- IPv4 よりも効率的な IPv6
  - **NAT の必要がない**
  - 簡素化されたヘッダを使うことにより、ルーティングの高速化が可能
  - **ネットワークが断片化されない**
  - 近隣アドレス解決のためのブロードキャストを回避
- 4G の導入
  - 4G はパケット交換のみをベース、IPv4アドレス供給は限界
- マルチメディアサービスの互換性
  - 一部のサービスプロバイダが使うIMS（IP Multimedia Core Network Subsystem）は、IPv6としか互換性がない
- 料金
  - **サービスプロバイダーは既存の IPv4 ネットワークのサポートを続けることにより、追加の運用コストと管理コストがかかる**

IPv4ネットワークとIPv6ネットワークを比較して書かれているが、
既存ネットワークの移行は考慮してないように感じる

DNS64/NAT64 による移行ワークフロー

プロバイダーとして理想的なのは、IPv4 ネットワークのサポートを廃止すること
（理由は IPv4 接続の維持にコストがかかるため）

⇒ それをやってしまうと、IPv4 ネットワークに
　クライアントがアクセスできなくなる



主要なネットワークプロバイダーの大半は DNS64/NAT64 による移行ワークフローを実装

と Apple は予測

IPv6 および App Store の要件

- アプリケーションで（IPv6 DNS64/NAT64 ネットワークとの）互換性を保証すること

- 定期的に回帰テストすることが重要

この文書上、App Store での展開の要件は、
DNS64/NAT64環境で動作すればいい
（IPv6のサーバとの接続性は要求されていない）

実はちょっと違う（後述）

IPv6 をサポートする際のよくある障壁

- プロトコルに**埋め込まれたIPアドレスリテラル**
  - プロトコルメッセージにIPアドレスリテラルが含まれたり
  - ヘッダーフィールドの値に表示されたり
- 構成ファイルに**埋め込まれたIPアドレスリテラル**
- ネットワークプリフライト
  - 通信可否の事前チェックを、**IPアドレスリテラルで与えられた接続先**で行っている
- 低レベルネットワークAPIの使用
  - ソケットや、RAWネットワークAPI
  - 誤用されがち、**IPv4 しかサポートしなかったりする**
- 小さなアドレスファミリストレージコンテナの使用
  - **32bit以下のアドレスストレージコンテナが使われている**等

まとめると、直接IPアドレスが使用されてる、IPv6非対応のAPI使われてる、アドレスが格納できない の3点に集約される

IPv6 DNS64/NAT64の互換性の保証

- 高レベルネットワークフレームワークの使用
  - ほとんどの場合、高レベルフレームワークで十分
  - WebKit：Webページを読込む複雑なプロセスに対応
  - Cocoa URL：アプリケーションでURLと参照先のリソースを操作
  - CFNetwork.Core　Services：さまざまなネットワークタスク
- IPアドレスリテラルを使わない

ホスト名・FQDNを使う

- プリフライトなしの接続
- 適性サイズのストレージコンテナの使用

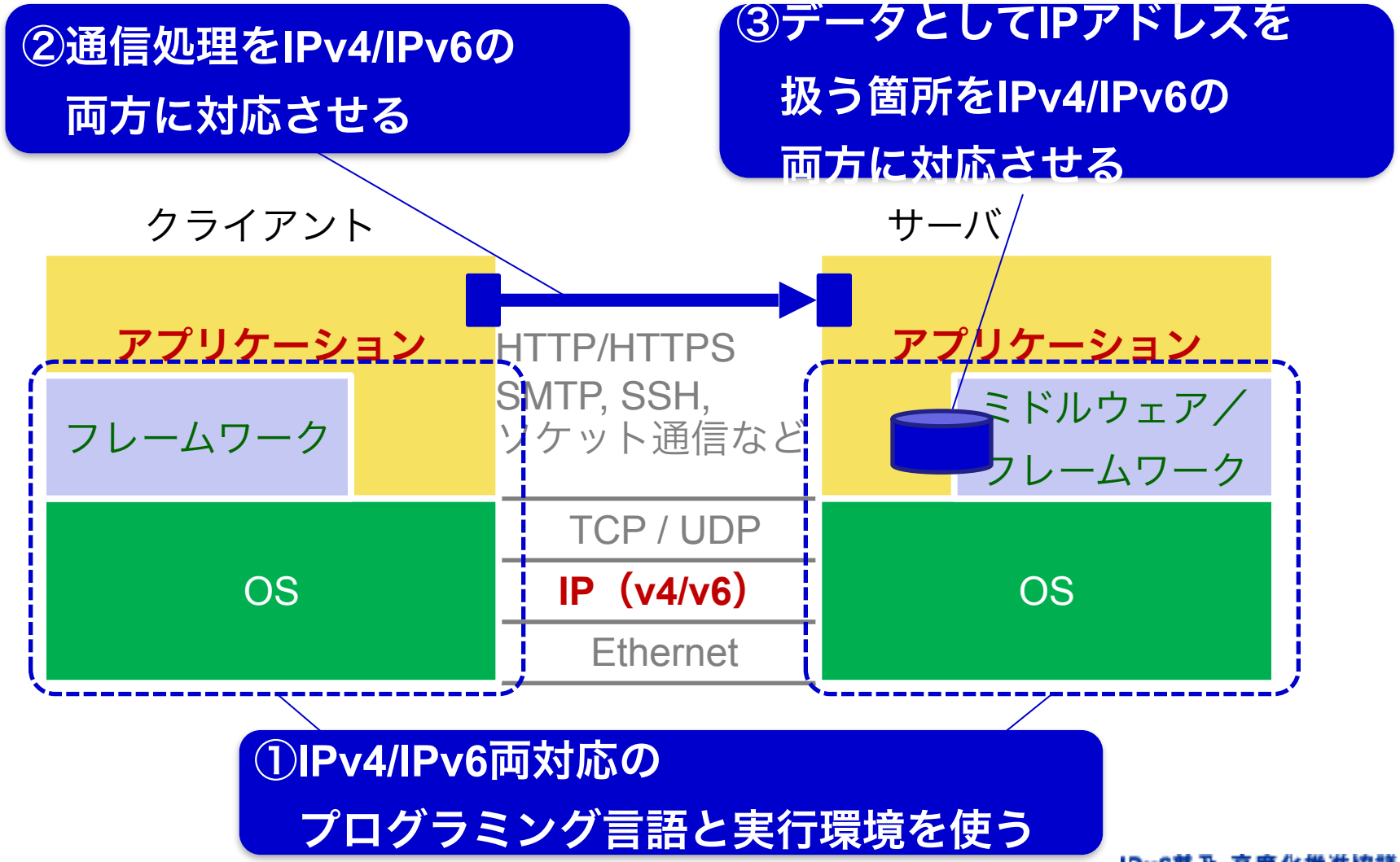
正

- ソースコードをチェックし、IPv6 DNS64/NAT64と非互換性がないか確認
  - IPv4固有のAPIを使用していないか確認

IPv6 DNS64/NAT64 環境下でも動作するためには上記対応が必要

上記対応は、アプリケーションIPv6対応共通のもの と Apple 環境固有のもの が混在

【参考】アプリケーションのIPv6対応のポイント



Networking Overviewを読み解いた結果①

Apple は

コストの視点を重視して、サービスプロバイダーが
IPv6 + DNS64/NAT64 を選択すると予測

DNS64/NAT64 で動けばOK と考えた

⇒ ・DNS64/NAT64 での動作を App Store の要件にした

**しかし、本来、アプリケーションに求められることは、
IPv6 環境でも IPv4 環境でも
IPv6/IPv4混在環境（デュアルスタック）でも動作すること**

Apple のこの考えでは満たせない

Networking Overviewを読み解いた結果②

実際の対応としては

IPv6サポート時のよくある障壁

- 直接IPアドレスが使用されてる
- IPv6非対応のAPI使われてる
- アドレスファミリストレージコンテナの容量が不足

が生じないようにするために

アプリケーションIPv6対応共通の

1. **IPアドレスリテラルではなく、ホスト名・FQDNの使用**
2. **適正サイズのストレージコンテナの使用**
3. **IPv4 固有のAPIを使用していないか確認**

と Apple 固有の

4. **高レベルネットワークフレームワークの使用**
5. **プリフライトなしの接続**

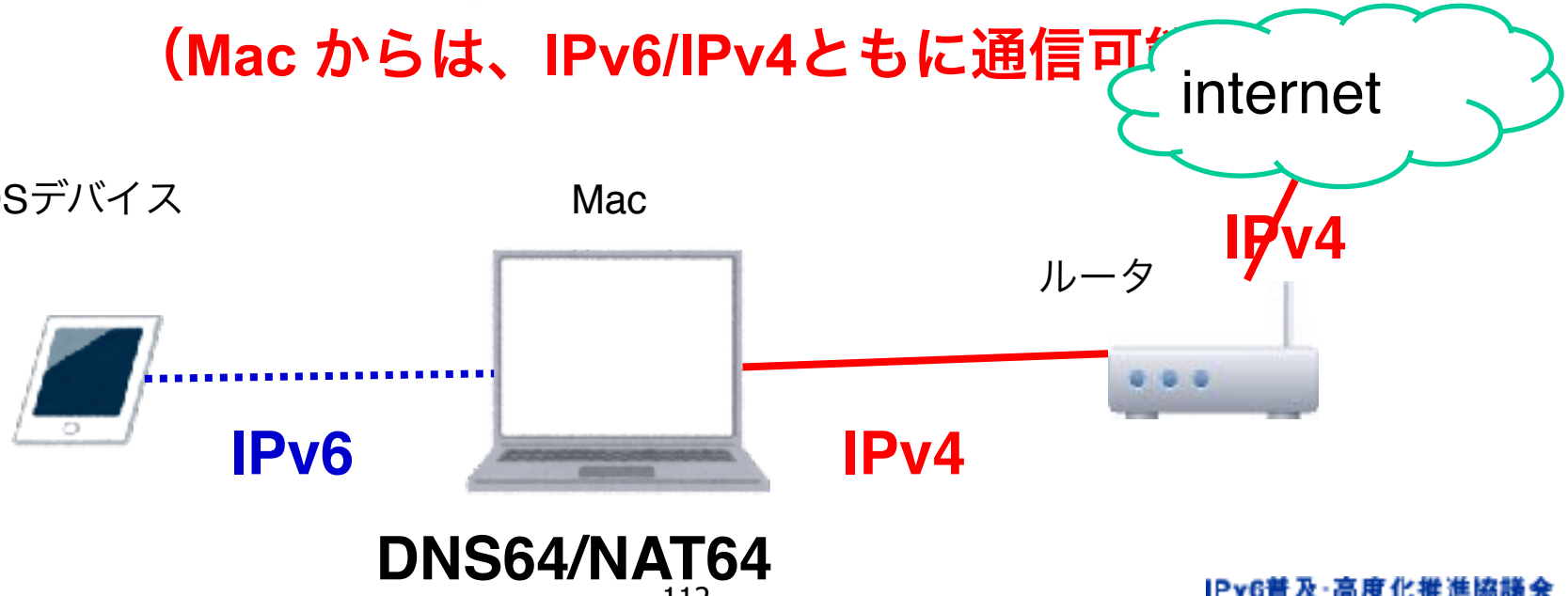
上記 1〜5 に従い実装すれば、DNS64/NAT64環境だけでなく、デュアルスタック環境でも動作する

4.2. Apple の IPv6 対応、検証

～Macによる検証環境の作り方～

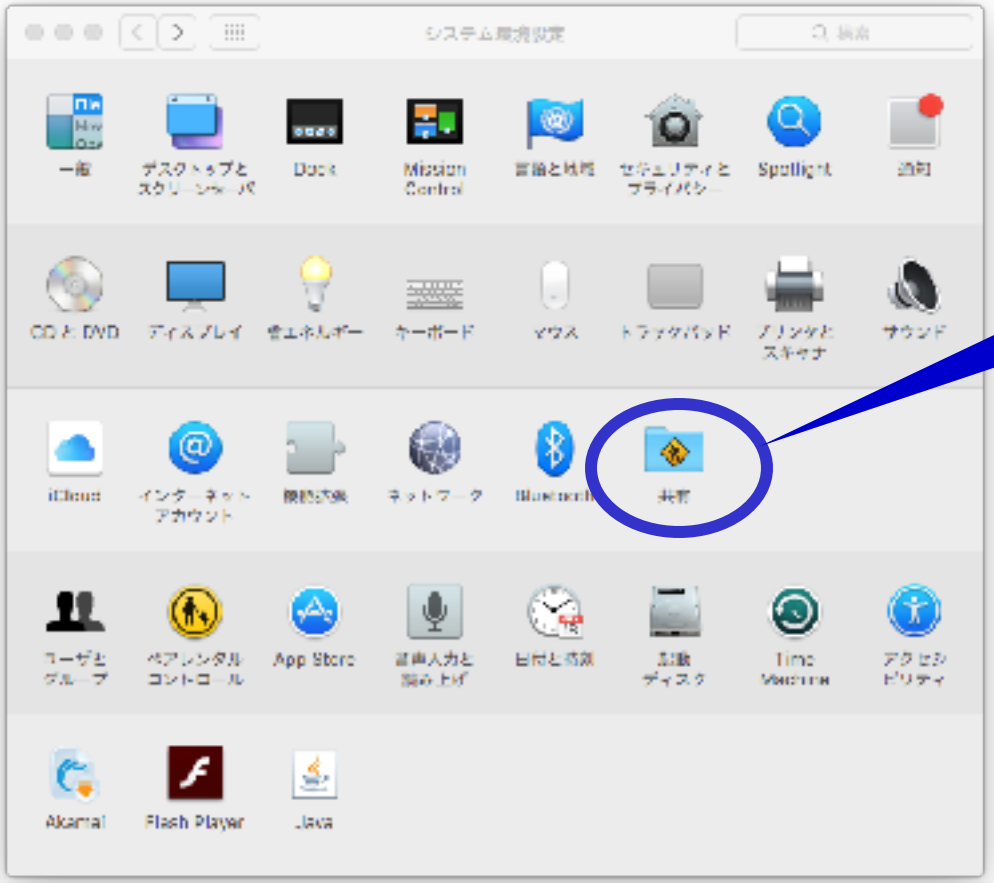
Mac による検証環境

- 先述の「Networking Overview」の中で紹介
- OS X 10.11（El Capitan）以降の OS X / macOS では、**インターネット共有のオプションで DNS64/NAT64 が利用可能**
- **注意：標準では、インターネット側（Uplink）は IPv4 しか通信できない（Mac からは、IPv6/IPv4ともに通信可**

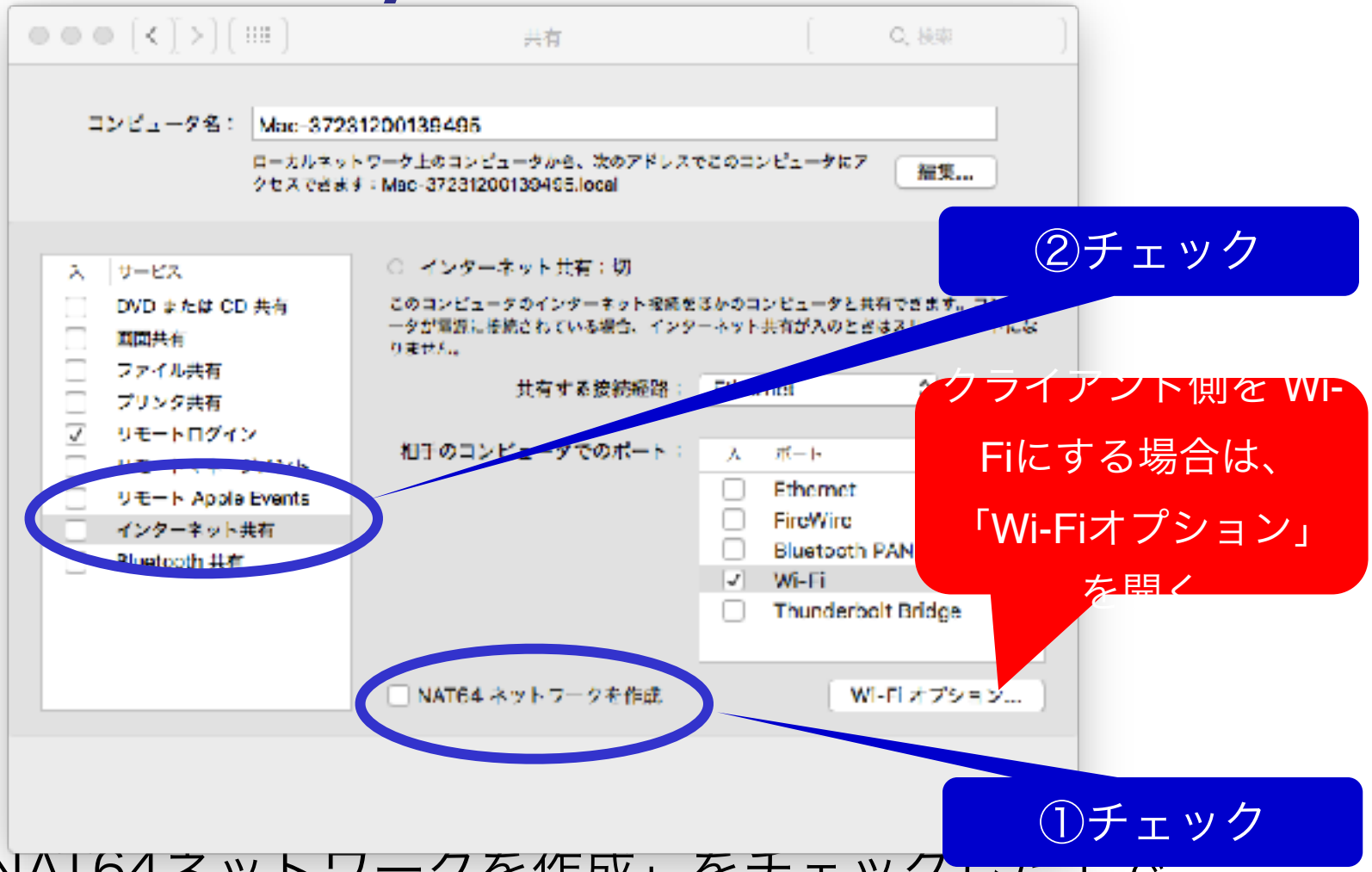


作ってみよう！Mac による DNS64/NAT64 環境①

1. システム環境設定を開く
2. [Option] キーを押しながら、「共有」を開く



作ってみよう！Mac による DNS64/NAT64 環境②



3. 「NAT64ネットワークを作成」をチェックした上で、「インターネット共有」をチェック

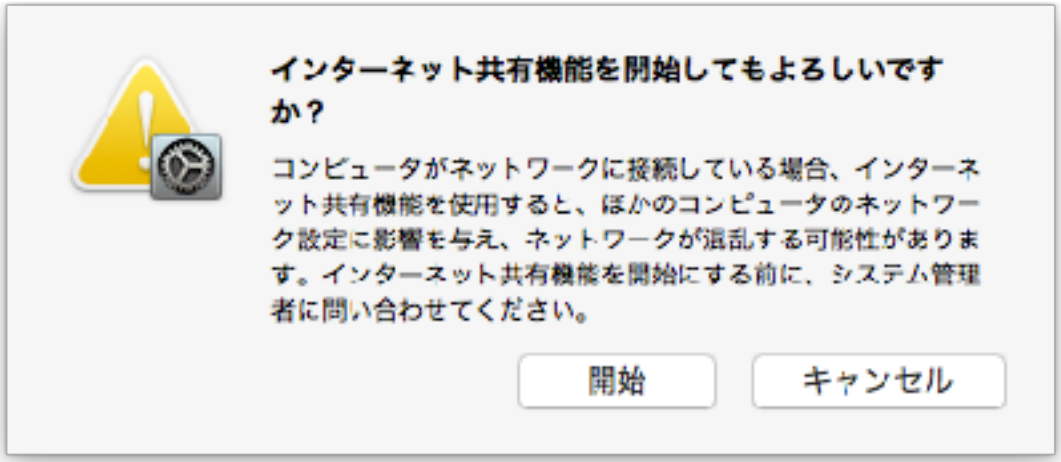
作ってみよう！Mac による DNS64/NAT64 環境③

（参考）Wi-Fi オプションを設定する

- セキュリティで「WPA2パーソナル」を選び、パスワードを設定する

作ってみよう！Mac による DNS64/NAT64 環境④

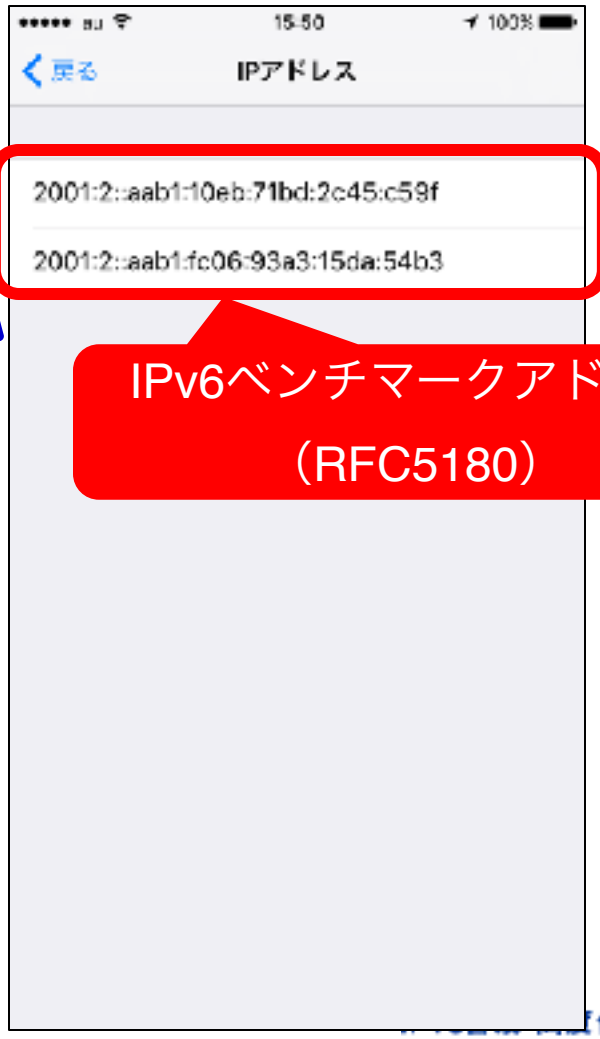
- 確認が入るので、「開始」



作ってみよう！Mac による DNS64/NAT64 環境⑤

- Wi-Fi に接続した iPhone の状況

IPv4 アドレス なし



IPv6ベンチマークアドレス（RFC5180）

Mac による DNS64/NAT64 検証環境は完成したが…

- これで、App Store が要求する IPv6 DNS64/NAT64 での動作は検証できる（はず）

しかし、IPv6 同士の動作は検証できないので、
場合によっては App Store のレビュー要件を満たせない

参考： https://forums.developer.apple.com/message/147579#147579

Let’s Hack（改造してみる）

検証した主な人：

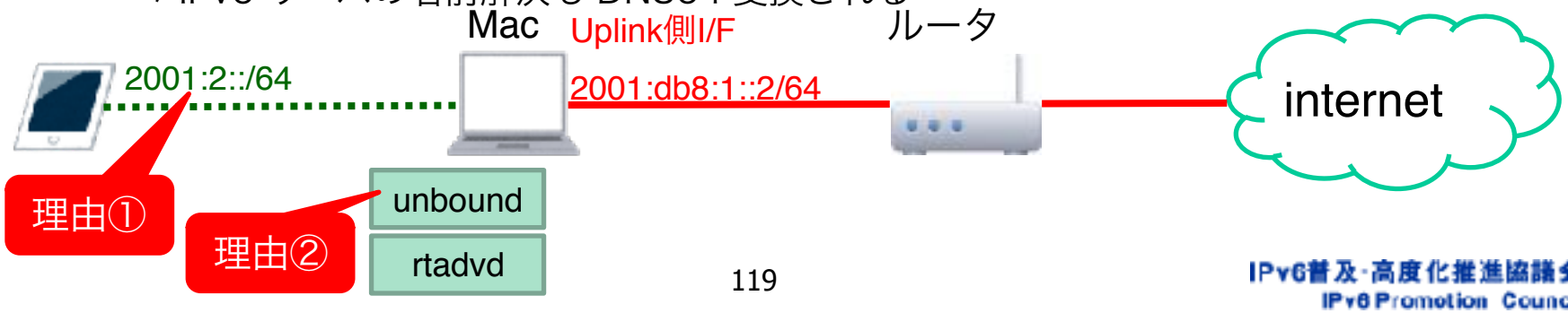
IPv6普及・高度化推進協議会 アプリケーションのIPv6対応検討SWG　藤崎 智宏氏、渡辺 露文

JPNIC　佐藤 秀樹氏

Mac による DNS64/NAT64 検証環境を改造①

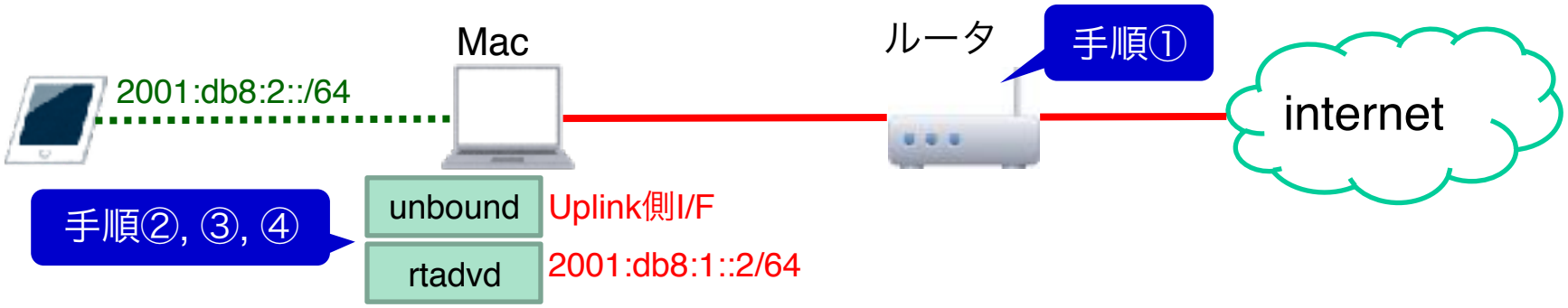
- Mac の DNS64/NAT64 を構成するソフトウェア
  - DNS64：Unbound
    - 設定ファイル：/etc/com.apple.mis.unbound.conf
  - ルータ広告（RA）：rtadvd
    - RDNSSで リゾルバのアドレスを広告
    - 設定ファイル：/etc/com.apple.mis.rtadvd.conf
  - パケットフィルタ：pf

- Uplink が IPv4 しか通信できない理由
  - ① クライアントに割当るアドレスがベンチマークアドレス
    ⇒ グローバルアドレスに変更しないと、外部から IPv6 で到達できない
  - ② unbound の設定で dns64-synthall が有効になっている
    ⇒ IPv6 サーバの名前解決も DNS64 変換される



Mac による DNS64/NAT64 検証環境を改造②

- 改造手順
  1. 上流のルータにて、クライアントに advertise する prefix を、MacのUplink アドレスにルーティング
     - 下図の例では、2001:db8:2::/64 を 2001:db8:1::2 へルーティング
  2. /etc/com.apple.mis.rtadvd.conf を変更
     - クライアントに advertise する prefix, rdnss の値を変更
  3. /etc/com.apple.mis.unbound.conf を変更
     - dns64-synthall の行を削除
     - interface: ::0 -> interface: [rndssサーバのアドレス] に変更
  4. unbound および rtadvd を kill して起動し直し



注意：上記設定ファイルは、NAT64有効時のみ出現する

- NAT64 を無効にした時点で設定が初期化され、設定ファイルも消失

Mac による DNS64/NAT64 検証環境を改造③

- **これで、IPv6同士の通信も行える DNS64/NAT64 環境の構築ができた！**

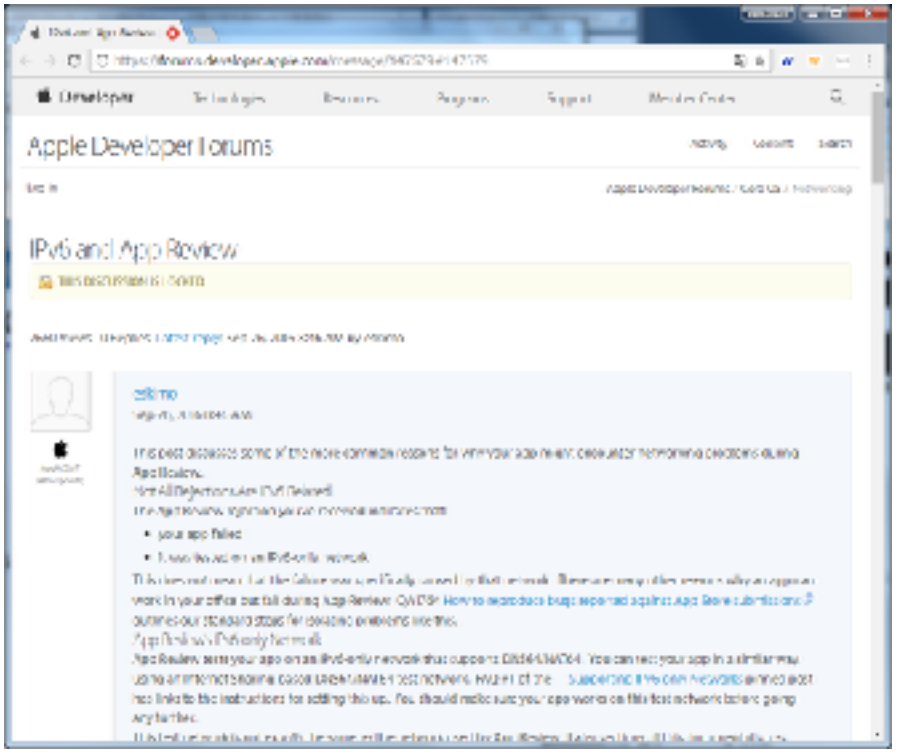
- 構築方法の詳細な手順は、後日、IPv6普及・高度化推進協議会 IPv4/IPv6共存WG アプリケーションのIPv6対応SWGにてドキュメント化し、公開する予定です

Appleスタッフのフォーラムへの投稿を読むと…①

- 実際の投稿

  https://forums.developer.apple.com/message/147579#147579

- リジェクトされるのは IPv6 に関するものだけではない
  - （IPv6対応以前に）App そのものが失敗しているケースもある



Appleスタッフのフォーラムへの投稿を読むと…②

- App レビューで見ているのは DNS64/NAT64 だけではなく、 IPv6-to-IPv6 の接続性も見ている
  - App レビューのネットワーク構成は、厳密には、Networking Overviewに書かれている DNS64/NAT64 と異なる
  - サーバがIPv6に対応しているなら、NAT64で変換させないで直接通信すべし

- **サーバがIPv6対応していても、IPv6で接続できないことがある**
  - **DNS登録名が誤っている**
  - **DNSは正しくても、サーバが IPv6 で listen していない**
  - **サーバが IPv6 で listen していても、IPv6 でリクエストが来ると失敗する**

- **全てのサーバをチェックすること**
  - **ライブラリに隠されているサーバ名**
  - **他のネットワークリクエストの結果として App に返されるサーバ名**
  - **HTTP & HTTPS においては、他のサーバへのリダイレクト**
  - **DNS の CNAMEレコード**

4章まとめ

- Apple は、USのモバイルキャリアの動きから、これからは IPv6 がメインで、IPv4 はNATでアクセスすると予測
- Apple は、iOS App の要件として IPv6-only ネットワークでの動作を要求し、レビュー（審査）で確認している
  - 接続先が IPv4 サーバの場合は、DNS64/NAT64 環境で
  - 接続先が IPv6 サーバの場合は、IPv6の直接通信で
- iOS App の IPv6-only ネットワークでの動作を実現するには Networking Overview を参考にするのが良い
- iOS App の IPv6-only ネットワークでの動作の検証は、Mac による DNS64/NAT64 が簡単
  - 改造して IPv6 サーバとの接続性を確保すれば、検証がさらに簡単になる
- フォーラムの投稿もちゃんと読もう

おわりに

まとめ①

- IPv6を使える環境が増えている
- IPv4とIPv6は直接通信できない
- WebサービスのIPv6対応にはアプリケーションの対応が不可欠
- IPアドレスのハードコーディングはダメ。ゼッタイ。

まとめ②

- アプリケーションのIPv6対応の基本方針
  - **IPv6対応＝IPv6/IPv4の両方で動作させること**
  - **シングルソースコードで対応する**

- アプリケーションのIPv6対応のポイント
  1. **IPv4/IPv6両対応のプログラミング言語と実行環境を使う**
  2. **通信処理をIPv4/IPv6の両方に対応させる**
  3. **データとしてIPアドレスを扱う箇所をIPv4/IPv6の両方に対応させる**

- Apple はIPv6-Onlyネットワークでの動作を App Storeの要件にし、検証環境の構築が容易に行える仕組みを Mac に搭載した

決して難しくない！

今日から開発するWebサービスはIPv6に対応させよう！

つづきはWebで（参考文献）

- 「アプリケーションのIPv6対応ガイドライン 基礎編」／IPv6普及・高度化推進協議会 IPv4/IPv6共存WG アプリケーションのIPv6対応検討SWG
  - http://www.v6pc.jp/jp/entry/wg/2012/12/ipv610.phtml
- 「アプリケーションのIPv6対応ガイドライン Webアプリ編（案）」／IPv6普及・高度化推進協議会 IPv4/IPv6共存WG アプリケーションのIPv6対応検討SWG
  - http://www.v6pc.jp/jp/entry/wg/2014/06/ipv6web.phtml

ご清聴いただき、
ありがとうございました