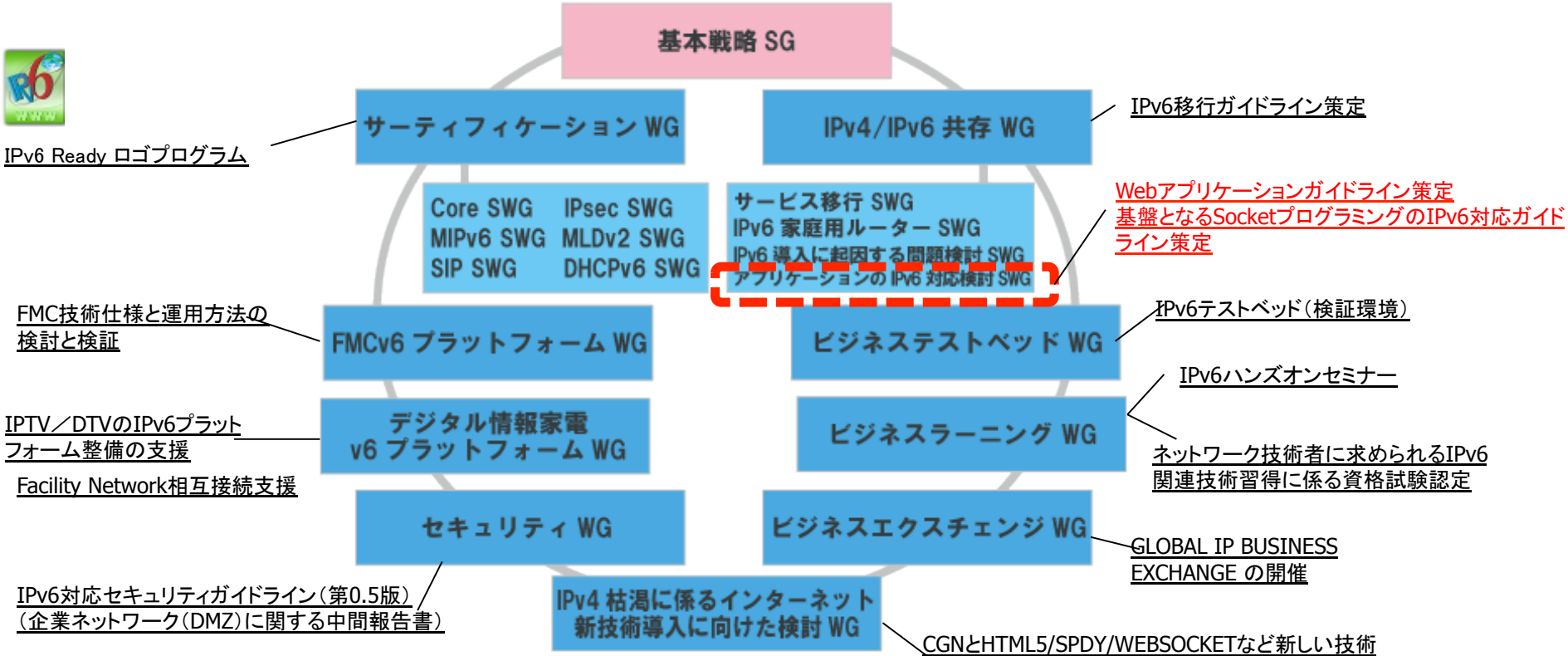
IPv6アプリケーション/プログラムの注意点

～Socketを使用した IPv6プログラミングの基礎から～

IPv6普及･高度化推進協議会
IPv6/IPv4共存WG
アプリIPv6化検討SWGメンバー
Ruri Hiromi

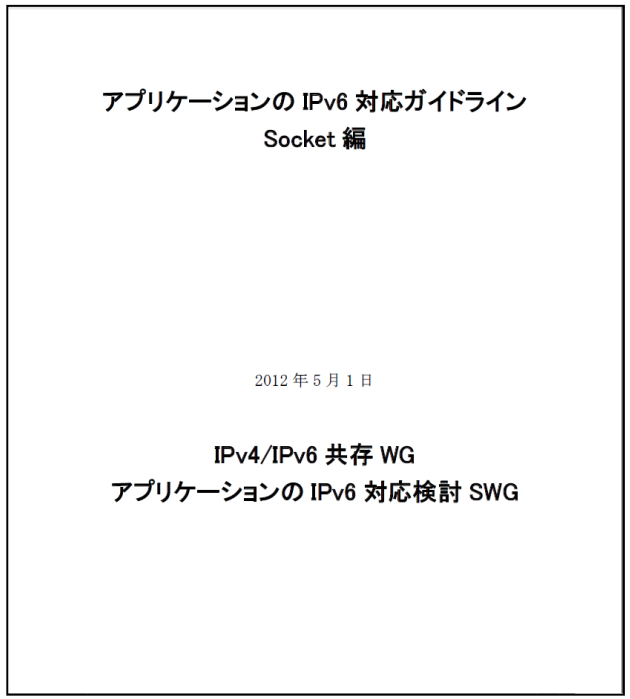
IPv6普及・高度化推進協議会
IPv6/IPv4共存WG
アプリケーションのIPv6対応検討SWG

- 2011年9月に発足



アプリケーションのIPv6対応ガイドライン＜基礎編＞

- 2012年12月公開

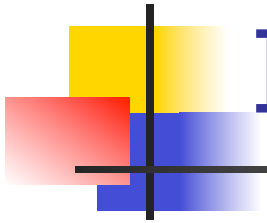


アプリケーションの IPv6 対応ガイドライン
Socket 編

2012 年 5 月 1 日

IPv4/IPv6 共存 WG
アプリケーションの IPv6 対応検討 SWG

目次



IPv6とその必要性

- 1990年代よりインターネットが流行した
  - IPが多数使われるようになった
  - IPを使う端末が増えた
- IPアドレスの枯渇
  - インターネットで通信する端末の識別番号（端末の住所：IPアドレス）が不足するようになった
- 新しいIPアドレスを持つIPにしよう
  - IPv4（従来型）からIPv6（次世代型）へ
  - IPv6はIPv4を参考にしているが相互運用性はない

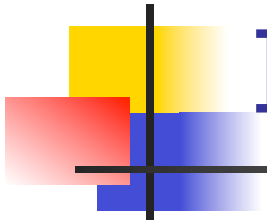
IPv6対応機材の最近

- IPv6 Ready Logo認証済みの国内機材
  1. ネットワーク機器：245
  2. プリンタ・複合機：110
  3. その他（ストレージ、カメラ、IP電話機等）：60
  4. OS・ソフトウェア：91

種別	分類	会社名	製品名／型番	概要
ネットワーク機器	スイッチ	Avaya	イーサネットルーティングスイッチ / ERS 8800	10Gイーサネットスイッチ
ネットワーク機器	スイッチ	D-Link	DES-1210-08P	10/100M Webスマート レイヤ2スイッチ
ネットワーク機器	ルータ	シスコシステムズ	Cisco 893 シリーズ ルータ	小規模オフィス向けサービス統合型 ルータ
ネットワーク機器	スイッチ	Avaya	仮想サービスプラットフォーム / VSP9000	仮想サービスプラットフォーム
ネットワーク機器	サーバ	Dell	PS6000, PS4000 シリーズ ストレージ アレイ	iSCSI ストレージサーバ
プリンタ・複合機	通信ソフトウェア	キヤノン	キヤノン IPv4/IPv6 組込みシステム用スタック	組込システム用 IPv4/IPv6 デュアルスタック
プリンタ・複合機	プリンタ・複合機	ヒューレットパッカード	HP Designjet プリンタ / T520, T120	デザインジェットプリンタ
プリンタ・複合機	プリンタ・複合機	コニカミノルタ	Konica IPv4/v6 プロトコル スタック	プリンタ複合機用デュアルスタック
プリンタ・複合機	プリンタ・複合機	ブラザー工業	SD9 series	ネットワークプリンタ
その他	ストレージ	ヒューレットパッカード	HP EVA P6000 ABM	仮想化ストレージ集約装置 アレイベース管理モジュール
その他	テープライブラリ	Dell	Tape Library / PowerVault TL2000, TL4000	データストレージ用テープライブラリ
OS・ソフトウェア	OS	Vmware	vSphere	クラウド構築用仮想化プラットフォーム
OS・ソフトウェア	OS	Microsoft	Windows 8, Windows Server 2012	ウィンドウズOS

2012年にIPv6対応済みとして発表された機材

http://www.jate.jp/approved_list/ より抽出



IPv6の対応状況

- iDC/ホスティング→続々対応中
- ISP→続々対応中
- 端末OS→PCは対応済み（随時機能向上中）

- …じゃぁ、アプリケーションソフトウェアは？
  - ApacheやBINDなど、公共性の高いものは対応済み
- …じゃぁ、私たちが作るアプリは？
  - 私たち自身がこれからがんばらないと！

- アプリケーションソフトウェアがIPv6対応するにはどうしたらいいか

IPアドレスとアプリケーションの関係

- ネットワークを利用するアプリケーションには、IPアドレスを扱うコードが潜んでいます

	ネットワークとの依存関係	代表例
1	ネットワークとは完全に無関係なもの	人事管理ソフト、名刺ソフト
2	アプリケーションとネットワークは完全に独立し、アプリケーションの動作に影響しないもの	文書作成ソフトの公開テンプレート ソフトウェアアップデート
3	アプリケーションとは独立にネットワークを利用するが、両立していないと動作しないもの	セキュリティ対策ソフト
4	ネットワークの利用を前提とするもの	Webブラウザ、Webサービス、skypeなどのコミュニケーションツール

今どきのアプリケーションは、ネットワーク依存

アプリケーションのIPv6対応

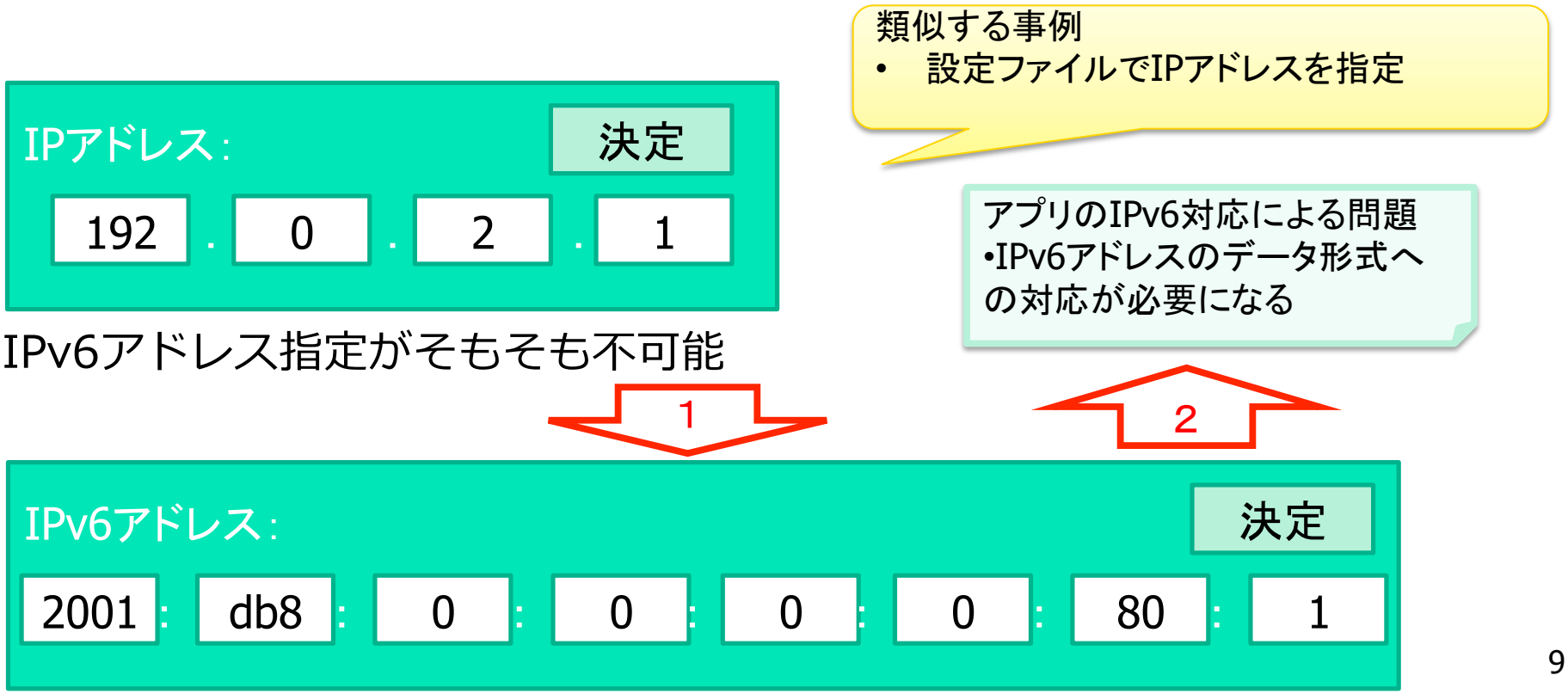
- IPv4との共存期に入る。（長期間続く見込み）
- IPv4とIPv6の互換性はなく、共存を意識したプログラムが必要となる。
- 既存のプログラムの多くは、IPv4環境を前提としているので動かなくなる可能性がある。

IPv4とIPv6が共存するシステム構築、プログラム開発が求められる

既存の改修が必要かどうかの診断の需要が高まる

具体的には？（その1）

- 例えば、GUIのIPアドレス入力画面



具体的には？（その2）

- IPv4依存コードからの脱却

IPv4依存のプログラムコードの対応例

開発言語：C/C++の例

IPv4アドレスを前提とした変数宣言や判定、関数呼び出しや、Socket作成時にIPv4アドレスのみを前提で作成してしまうなど、アドレスファミリーに依存したプログラミングを行っている場合、IPv6ネットワークに移行するとアプリケーション例外が発生する可能性がある。

以下のIPv4依存関数・構造体を使用していた場合、推奨関数・構造体を利用するように変更し、アドレスファミリーに依存しないプログラミングを行う。通信を行う場合にはまずDNSによりFQDNの名前解決を行った上で、取得できたIPアドレスにて接続を行うようプログラミングする。

● IPv4依存関数・構造体

inet_addr(), inet_aton(), inet_lnaof(), inet_makeaddr(), inet_netof(), inet_network(), inet_ntoa(), inet_ntop(), inet_pton(), addr_ntoa(), network(), getservbyport(), gethostbyname(), gethostbyname2(), gethostbyaddr(), getservbyname(), in_addr

●推奨関数・構造体

sockaddr_storage, getaddrinfo(), getnameinfo()

どうすればいいのか？ 知識武装しよう

- IPv6のアドレスの長さや表記方法
  - 入出力は万全か
  - 格納領域は万全か
- 処理順序
  - IPv6を優先、だめだったらIPv4へ
  - （IPv6がない環境だったら、ポリシー設定で逃げるなど対策もある）
- DNSにはIPv6専用のレコード（AAAA）がある
  - IPv4しかない環境でDNSを参照してもAAAAが返答されることがある
  - IPv6のレコードを返答させるには、EDNS0という対応が必要。この運用がされていない環境ではエラーになることもある
- 確認方法もIPv6対応版で
  - traceroute,ping等はIPv6オプションを使って確認
  - ICMPv6ではエラーの種類や数も豊富になっている

知識武装の続き

- 開発言語の対応状況を調べる

言語	IPv6対応状況	プロトコル依存(関数やマクロ)	プロトコル非依存
C C++ C#	OSのsocketAPIの対応状況による	inet_addr, inet_aton, in_inaof, in_makeaddr, inet_netof, inet_network, inet_ntoa, inet_ntop, inet_pton, addr, ntoa, network, getservbyport, gethostbyname, gethostbyname2, gethostbyaddr, getservbyname, sockaddr_in, struct sockaddr, struct in_addr, INADDR_LOOPBACK, INADDR_ANY, IP_TTL, rresvport, rcmd, AF_INET, PF_INET	sockaddr_strage, getaddrinfo, getnameinfo
Java	SolarisとLinux :J2SE 1.4以降 Windows:J2SE 5.0 以降	Inet4Address, Inet6Address (IPv6対応版から新設)	InetAddress
Perl	5.10.0以降 5.14以降:IPv6フルセット	IO::Socket::INET	IO::Socket::IP
Ruby	1.9.2以降	UDPSocket,TCPServer	Socket.udp_server_loop、 Socket.tcp_server_loop
PHP	5.0.0以降	gethostbyname,gethostbynamel	checkdnsrr,PEAR::Net_DNS

知識武装の続きの続き

- 利用するミドルウェアにも注意して調べる

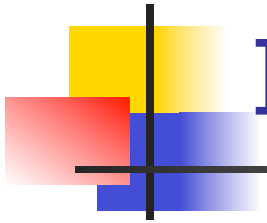
カテゴリ	製品名	IPv6対応状況※(バージョン)
開発・実行環境	.NET	**×:1.0** ○:1.1以降
サーブレットエンジン	Tomcat	**×:JavaVM1.41** ○:JavaVM1.4.2以降
データベース	Oracle	**×:9i △:10g** ○:11gR2以降
データベース	SQL	**×:2000** ○:2005以降
ミドルウェア	MQ	**×:5.3** ○:6.0以降
ミドルウェア	Tuxedo	**×:10.0** ○:10.0gR3以降
ERPパッケージ	SAP	**×:4.7 ×:6.0** ○:7.10以降
webサーバ	apache	**×:1.2.x** ○:1.3.x以降
webサーバ	IIS	**×:5.2** ○:6.0以降

バージョンまで調べましょう

全体を見渡してみよう

- 新規はどんどんやるべし
- 改修は金銭面がネック？

1	アプリケーションの仕様確認	データテーブル/データファイル/外部システムのデータ取り扱い、ログ出力、アクセス制御、認証、名前解決、タイムアウト制御、GUI、帳票などでのIPアドレスの扱い方を確認
2	システム構成確認	ハードウェア、ソフトウェア、外部システムとのインタフェースなどにおけるIPアドレスの扱い方を確認
3	実装方法の確認	ロジック/アルゴリズム、記述方法、API/オブジェクト、設定、IPv4依存コードの有無などの確認
4	運用方法の確認	運用保守ツール、監視ツール、設定、利用内容におけるIPアドレスの扱い方を確認
5	開発環境の確認	開発言語、実装開発環境、テスト環境などの確認
6	IPv6対応方針、内容決定	IPv6導入シナリオ、スケジュール、問題発見時の対応方針、マイルストンなどの確認



IPv6プログラミングの情報について

- 今回の解説で参考にしている書籍
  - IPv6ネットワークプログラミング ASCII社刊
    - http://ascii.asciimw.jp/books/books/detail/4-7561-4236-2.shtml
  - 著者は萩野純一郎（itojun）氏
  - 今回参考にしたプログラムもこの本に掲載されているもの
    - itojun氏が製作し、パブリックドメインで公開
- IW2012のT7「IPv6実践講座～トラブルシューティング、セキュリティ、アプリ構築まで～」セッション
  - https://www.nic.ad.jp/ja/materials/iw/2012/proceedings/t7/

IW2013でも最新情報の発表あり☑

IPv6普及･高度化推進協議会での活動

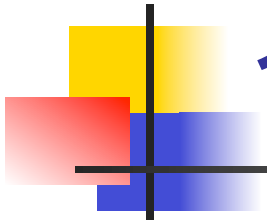
- IPv6/IPv4共存WG アプリケーションIPv6化検討SWG
  - http://www.v6pc.jp/jp/wg/coexistenceWG/v6app-swg.phtml

- SocketアプリケーションのIPv6化
  - http://www.v6pc.jp/jp/entry/wg/2012/12/ipv610.phtml
  - 手法本文/サンプルプログラム/ソリューションサンプル
    - http://www.v6pc.jp/jp/upload/pdf/socket-20121203.pdf
    - http://www.v6pc.jp/jp/upload/pdf/socket-sample-20121203.pdf
    - http://www.v6pc.jp/jp/upload/pdf/about_asterisk_ipv6v5-9.pdf
- WebアプリのIPv6化
  - 現在調査･検討中

IPv6 Summitも北海道で開催

- IPv6 Summit 2014 in Hokkaido
  - IPv6に関する最新事情の講演会
  - 1月ないしは2月に実施
- 社団法人日本インターネット協会（ IAjapan）が主催
  - http://www.iajapan.org/
  - IPv6ディプロイメント委員会による
  - http://www.iajapan.org/ipv6/

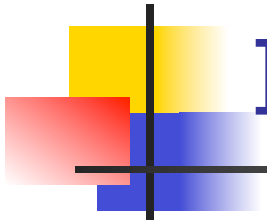
- みなさまお誘い合わせの上、ご参加をよろしくお願いいたします



今回の説明の概要

- BSD Socket APIを使用したアプリケーションソフトウェアのIPv6化を説明
- クライアントプログラムのIPv6対応
  - 具体的な手順
- サーバプログラムのIPv6対応
  - 手法の分類
  - 具体的な手順は割愛（すみません）
- 名前解決の問題と解決案
- 組み込みの話

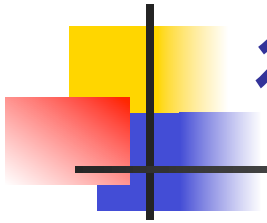
BSD Socket による クライアントアプリケーションの IPv6化



IPv6対応・デュアルスタック対応とは

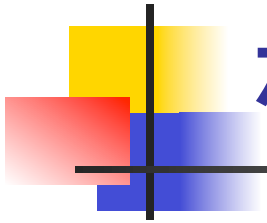
- IPv6だけでなく従来のIPv4にも対応しなければならない
- 従来：シングルスタック
  - ひとつのプロトコル（IPv4）に対応していた
- 今後：デュアルスタック（ IPv6/IPv4両対応プログラム）
  - クライアントは複数のプロトコル・複数アドレスの中のどれで送信を行うか選ばなければならない
  - サーバは複数のプロトコル・アドレスで同時に受信しなければならない

ただ単に関数を変更するだけではだめ
選択機構や、並列受信機構などが新たに必要となる



従来のクライアントプログラミング

- IPv4対応シングルスタックによるクライアントアプリの大まかな流れ
  - ホスト名解決
  - サービス名解決
  - Socket生成（ファイルデスクリプタ生成）
  - Connect実行（通信相手指定・接続を確立）
  - デスクリプタによる入出力
  - クローズ

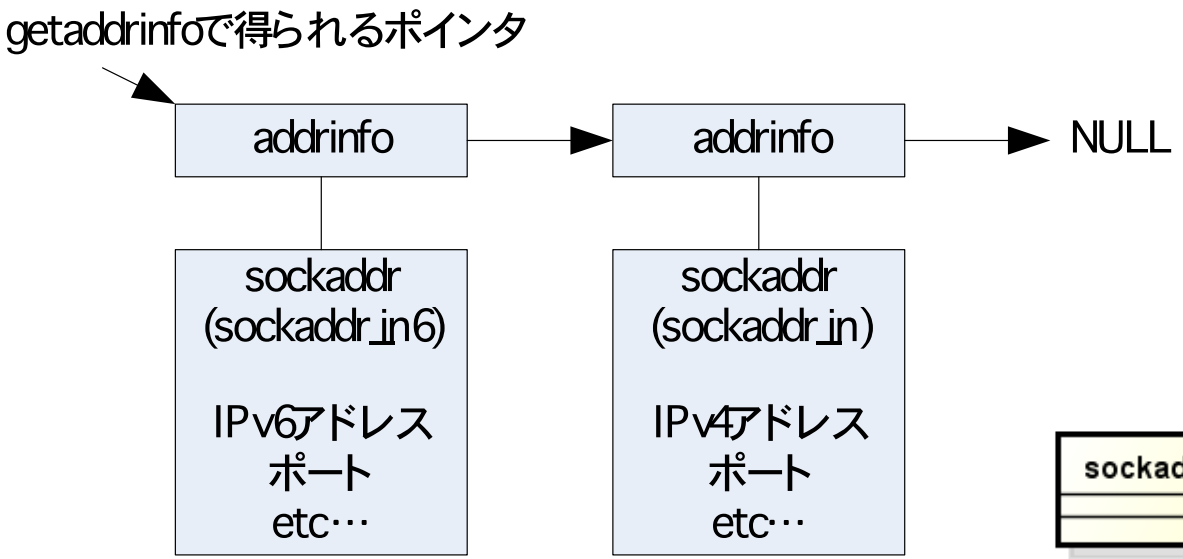


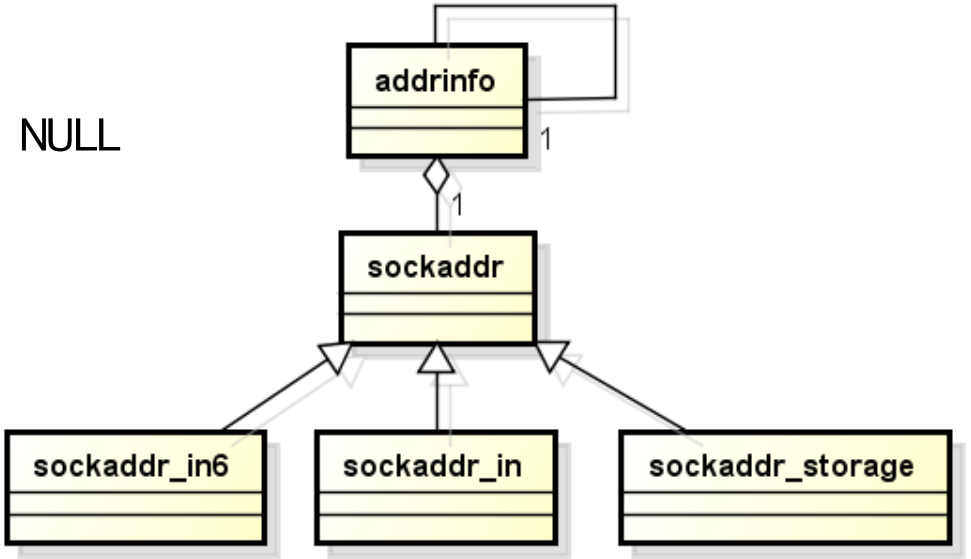
ホスト名・サービス名解決

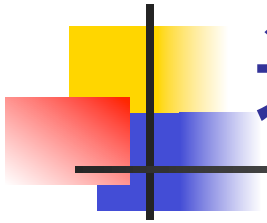
- IPv4
  - ホスト名 : *gethostbyname()*でhostent構造体を得る
  - サービス: *getservbyname()*でservent構造体を得る
- デュアルスタック
  - *getaddrinfo()を呼ぶと*addrinfo構造体のリストが得られる
    - リストの開放は*freeaddrinfo()* 関数に引数でそのリストを与える
- 注意
  - *gethostbyname2()* はIPv6を扱えるが使うべきではない

addrinfo構造体とsockaddr構造体

- addrinfo構造体
  - 1インスタンスで1アドレス情報を持ち、リストを構築している
  - 内部でアドレスを保持するsockaddr構造体へのリンクを持つ
- sockaddr構造体
  - IPv4やIPv6など各種アドレス情報を汎化した構造体
  - 実体はsockaddr_in6（v6）やsockaddr_in（v4）
  - どのアドレスが入るかわからない場合はsockaddr_storageで定義







逆引き

- IPv4
  - アドレス：*gethostbyaddr()*でhostent型構造体を得る
  - サービス：*getservbyport()*でservent構造体を得る
- デュアルスタック
  - *getnameinfo()*にsockaddr構造体を与えるとホスト名やサービス名の文字列を取得できる
    - 文字列領域は呼び出し元が引数で与える
  - いろいろなオプションが指定可能
    - NI_NOFQDN …FQDNではなくホスト名だけ
    - NI_DGRAM …UDPのポート情報を得る
    - NI_NUMERICHOST…逆引きせずアドレスの文字列表現を返す
    - etc…

ソケット生成・コネクト

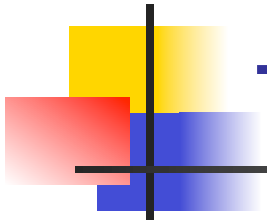
- デュアルスタックの場合addrinfo構造体を参照する
  - 例：addrinfo ai;
  - プロトコルファミリ：　ai->ai_family
  - ソケットタイプ：　ai->ai_socktype
  - プロトコル：　ai->ai_protocol
  - アドレス：　ai->ai_addr
  - アドレス長：　ai->ai_addrlen
- 実例

```
s=socket(ai->ai_family, ai->ai_socktype, ai->ai_protocol);
connect(s, ai->ai_addr, ai->ai_addrlen);
```

クライアントのコード概要

```
struct addrinfo hints, *res, *resall;
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
getaddrinfo(“www.v6pc.jp”,”http”, &hints, &resall);

for (res = resall; res; res = res->ai_next) {
        s=socket(res->ai_family, res->ai_socktype, res->ai_protocol);
        if (s<0) continue;
        if (connect(s, res->ai_addr, res->ai_addrlen) < 0){
                close(s);
                continue;
        }
        /*読み書き*/
        close(s);
        break;
}
```

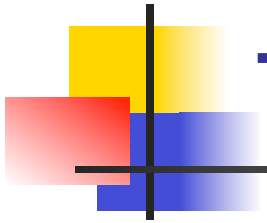


デュアルスタッククライアントのまとめ

- *getaddrinfo()*で接続先IPアドレスのリストを得る
  - addrinfo構造体のリスト
- リストの順にソケット生成・接続を行い、成功したら通信して終了する

- サンプルプログラム
  - **http://www.v6pc.jp/jp/upload/pdf/socket-sample-20121203.pdf**

BSD Socket による サーバアプリケーションのIPv6 対応



サーバのデュアルスタック化

- いくつか手法がある
  - inetdを使用する
  - 自身のプロトコル・アドレスの数だけ*socket()*を生成して、全てのFDに対して応答処理をする
  - シングルスタック仕様のプログラムを複数プロセス走行させる
  - IPv4 マップドアドレス（IPv4 Mapped IPv6 address）を使用して、v6ソケットで通信する

サーバアプリ実現手法の分類

手法	利点	欠点
【手法１】 inetdを使用する	通信部分をinetdが代行するため、通信のIPv6化を意識しなくてよい	inetdを必要とする
【手法2】 複数のsocketを生成する	ひとつのプロセスでマルチプロトコルに対応できる	複数ソケットを生成し、それらを同時に待つため、プログラムが複雑になる
【手法3】 シングルスタックプログラムを複数プロセス走行させる	プログラム構成の変更なしにIPv6に対応できる	共有リソースを扱う場合、プロセス間で排他制御する必要がある
【手法4】 IPv4マップドアドレスを使用する	ひとつのソケットでIPv4/IPv6両方を処理でき、プログラム構成の変更が必要ない	IPv4とIPv6の処理が混在する。アドレスを扱う際にはIPv4マップドアドレスかどうかの判定が必要となる場合もある

inetdによるデュアルスタックサーバ

手法1

- 通信部分は変わらない
- 通信相手アドレスを取得する部分で注意が必要
  - *getpeername()* FDとsockaddr構造体を引数で渡すとsockaddr 構造体に相手ピアアドレスを書く
  - 引数は sockaddr構造体ではなく、sockaddr_storage構造体を使用する
    - sockaddr_storage構造体はどんなプロトコルのアドレスでも記憶できる領域を持つ

```
sockaddr_storage from;
getpeername(0,(sockaddr*)&from,sizeof(from));
```

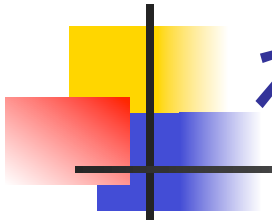
  - あとは*getnameinfo()*で文字列化

複数socketで待ち受けるサーバ

手法2

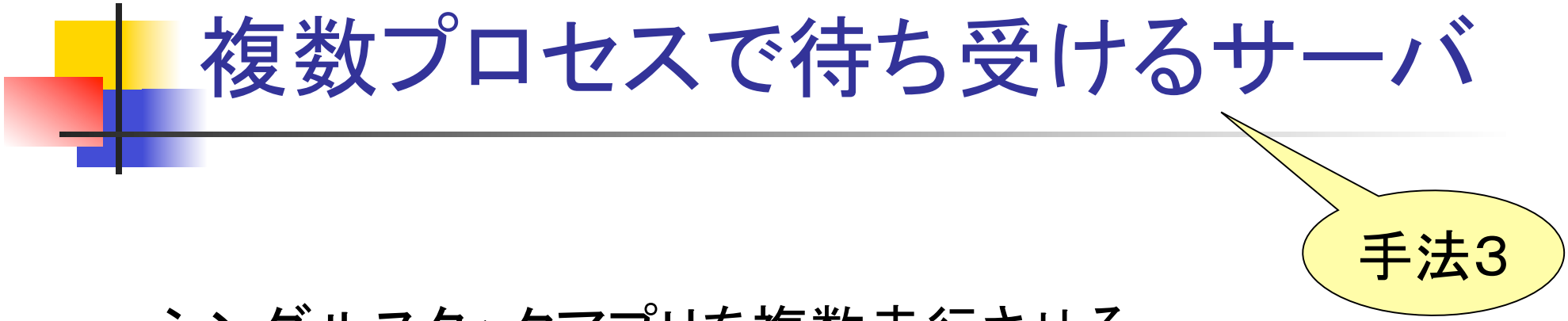
- もっとも典型的で理想的な対応
- プログラム構成が変化する
  - 複数のデスクリプタを同時待ち受けする機構
- 完全な新規で設計する通信プログラムはこの構成が望ましい

複数ソケットを処理するサーバの流れ



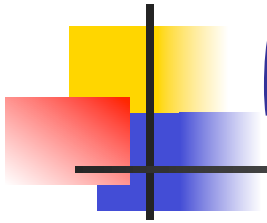
- *getaddrinfo()* で自身のプロトコル・アドレスを*addrinfo*構造体のリストで得る
  - hintsパラメータでAI_PASSIVEを指定するとIN_ADDR_ANYとIN6ADDR_ANY_INITが得られる
- リストで得られたプロトコル・アドレス個別に下記を実施
  - *socket()*、*bind()*、*listen()*
- 得られた複数のFDをfd_set構造体に保存
- 以降はループ
  - fd_set構造体を引数に*select()*で接続の待機
  - *select()*を抜けてきたfdに対して*accept()*
  - 読み書き処理
  - クローズ

複数プロセスで待ち受けるサーバ

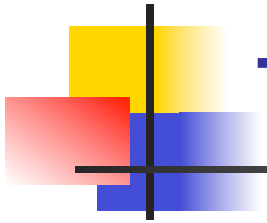


- シングルスタックアプリを複数走行させる
  - *getaddrinfo()* で AF_INETとAF_INET6のどちらかを設定する
- forkでひとつのプログラムがv4/v6に分離するようにすればリソースの節約も可能
  - Copy on Write機能による

自分自身のIPアドレスを得るには？



- 環境依存
  - UNIXライクOSならioctl関数を利用するのが一般的
- オープンソースOSの場合はifconfigのソースを参照するとよい
  - FreeBSDの場合、/usr/src/sbin/ifconfig/*
  - ubuntuの場合、net-toolsパッケージ

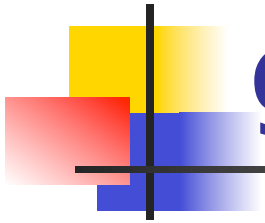


デュアルスタックサーバのまとめ

- いくつか手法があるので、メリットとコスト・リスクを比較して適切な選択をしましょう

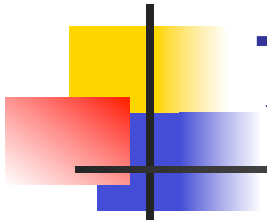
- サンプルプログラム
  - **http://www.v6pc.jp/jp/upload/pdf/socket-sample-20121203.pdf**

名前解決の問題と最近のテクニック



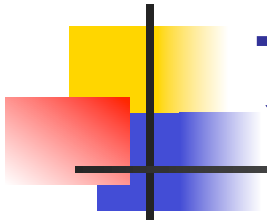
getaddrinfoの並びは？

- *getaddrinfo()* は名前解決
  - 出力されるaddrinfo構造体のリストはどういう順序になるのか？
- 長らく RFC3484で定義されていた
- RFC6724 がRFC3484をObsoleteした
  - デフォルトポリシーテーブルの修正
  - アドレス選択ルールの修正
  - フォールバック問題の記述
  - etc...



フォールバック問題

- フォールバックとは
  - クライアントが接続先IPアドレスのリストを得る
  - リストの先頭にあるIPアドレスに接続しようとして、失敗すると次のリスト要素のIPアドレスを試す
- 原因
  - サーバがそのプロトコル・IPアドレスでアプリサービスをしていない
  - ネットワークの接続性が失われている
- 問題
  - タイムアウトを繰り返すので、接続まで時間がかかる
  - 環境によっては数十秒かかる場合もありうる

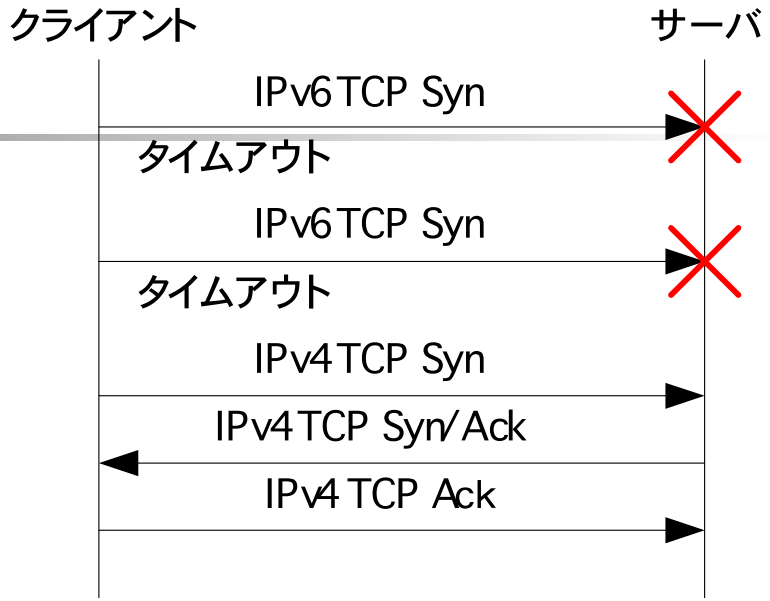


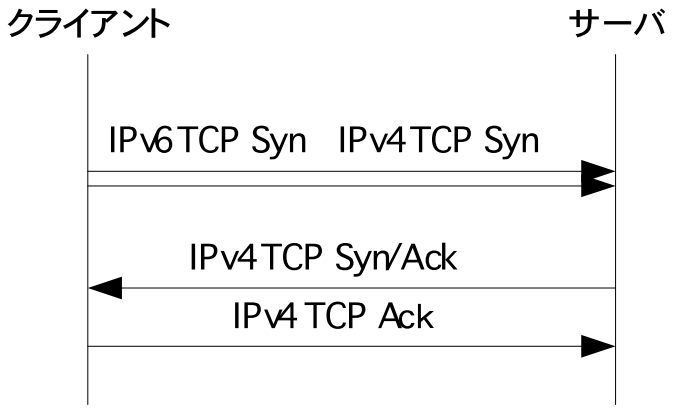
フォールバックの回避

- サーバがサービスしていないIPアドレスはDNSに登録しない
- IPの接続性を健全に保つ
- ポリシーテーブルを変更する
  - ポリシーテーブルの参照法は下記のとおり
    - Windows: netsh interface ipv6 show prefixpolicies
    - Linux: ip addrlavel show
    - FreeSBD: ip6addrctl show
- サーバサイドだけでの対応では完全には解決できない

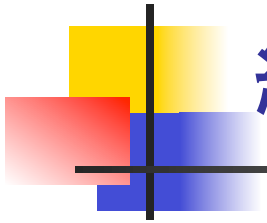
Happy Eyeballs

- RFC6555、RFC6556で定義
- 従来はTCP Synの接続失敗を受けて次のTCP Synを送っていた
- Happy EyeballsはIPv6 / IPv4アドレスに両方に対して一気にTCP Synを送る
  - Syn/Ack応答が帰ってきたIPにTCP Ackを送って接続
- 詳細の動きが不明瞭で実装依存性が高い
  - 複数I/Fの場合やIPv6アドレスが複数ある場合は？
  - IPv6 Syn/Ackが遅れて届いたら？
  - 今後の展開や運用を注視すべき



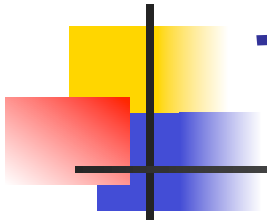


組み込み用途でのIPv6対応



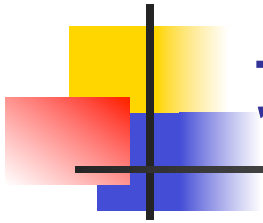
組み込み機器へのアドレス埋め込み

- 組み込みではSocketを使うことが多い
- 組み込み機器はいろいろ大変
  - お客様の環境でDNSが使えない
  - 名前解決処理に必要なリソースが不足している
- 組み込み機器でIPアドレスをハードコーディングしたいこともある
  - しかしやめたほうがいい
  - RFC4085でこの問題が記述されている



アドレスハードコーディングの問題

- そもそもIPアドレスは借り物
  - リナンバリングのリスクが伴う
- ホスト名は名前指定して、名前解決には*getaddrinfo()*を使うべき
  - RFC6724やRFC3484 に従った適切な処理が約束されている
  - これを自作するということはRFCに従うコストやそれから外れるリスクを自己負担するということ
- これらのリスク・コスト・インターネットの道義的責任がハードコーディングしないリスク・コストを上回ったとき
  - 十分な注意・サポートとともにやむをえずハードコーディングすることは考えられる



最後に

- IPv6はどんどん浸透してきている
  - アプリでIPv6を先取りして、時代もお客様も先取りしよう
- 何かありましたらいつでもこちらまで
  - IPv6普及・高度化推進協議会の連絡先
    - https://www.v6pc.jp/jp/info/inquiry_web.phtml