Socketを使用したIPv6プログラミングの基礎

RICOH
imagine. change.

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

株式会社リコー
研究開発本部
基盤技術開発センター
大平浩貴（おおひら こうき）

IPv6とその必要性

RICOH
imagine. change.

- 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の対応状況

RICOH
imagine. change.

- 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

IPv6プログラミングの情報について

RICOH
imagine. change.

- 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/

IPv6普及・高度化推進協議会での活動

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

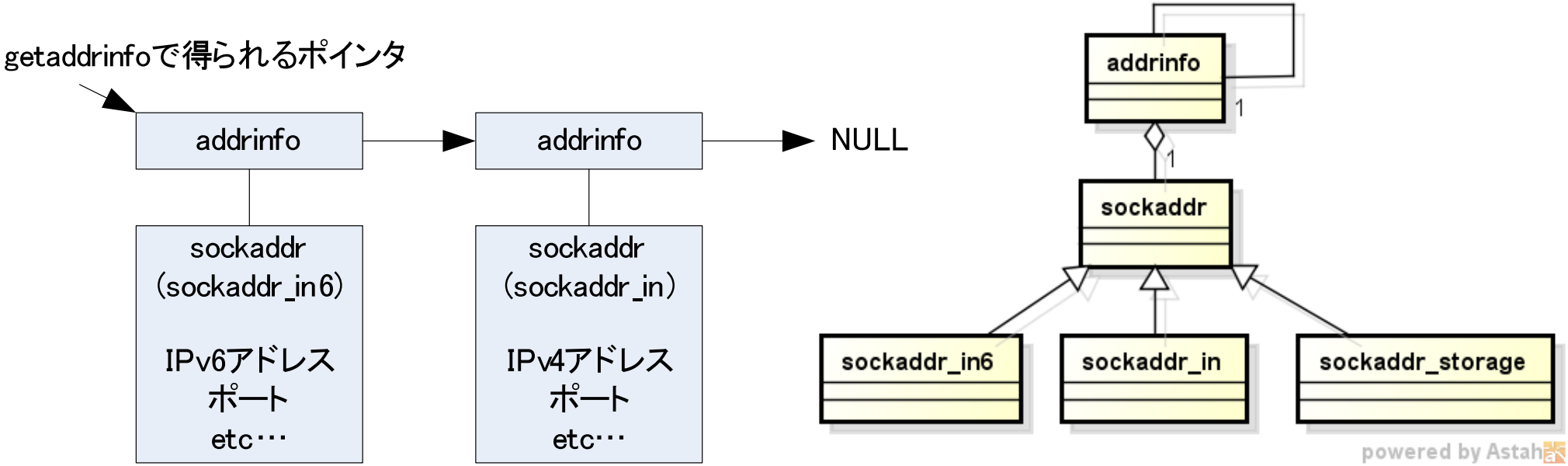
RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

```
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

RICOH
imagine. change.

- *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

RICOH
imagine. change.

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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

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

RICOH
imagine. change.

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

RICOH
imagine. change.

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

RICOH
imagine. change.

- *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

RICOH
imagine. change.

手法3

- 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？

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

名前解決の問題と最近のテクニック

getaddrinfoの並びは？

RICOH
imagine. change.

- *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

RICOH
imagine. change.

- 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

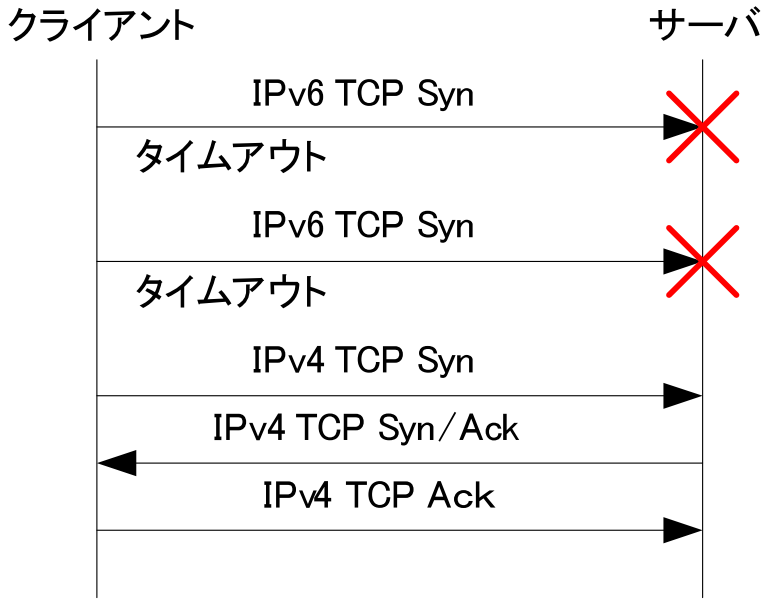
RICOH
imagine. change.

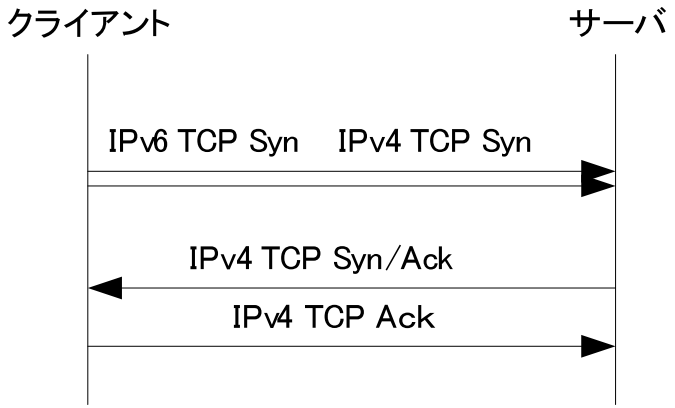
- サーバがサービスしていないIPアドレスはDNSに登録しない
- IPの接続性を健全に保つ
- ポリシーテーブルを変更する
  - ポリシーテーブルの参照法は下記のとおり
    - Windows: netsh interface ipv6 show prefixpolicies
    - Linux: ip addrlavel show
    - FreeSBD: ip6addrctl show
- サーバサイドだけでの対応では完全には解決できない

Happy Eyeballs

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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

RICOH
imagine. change.

- 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
  - 大平の連絡先
    - ohhira@src.ricoh.co.jp
    - kohki@lemegeton.org
    - Twitter： @torawarenoaya
    - facebook： http://www.facebook.com/kohki.ohhira
    - LinkedIn：
      http://jp.linkedin.com/pub/kohki-ohhira/10/7ba/6a2