BSD SocketによるIPv6プログラミングを紐解く

RICOH
imagine. change.

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）

自己紹介

RICOH
imagine. change.

- 1999年頃からIPv6にかかわる
  - IETF行ったり
  - 端末OS触ったり
  - 複合機のネットワークを触ったり
  - IPsecやったり
- プログラミングはあまり得意ではないけど…

今回の説明の概要

RICOH
imagine. change.

- 通信プログラムの基本
  - BSD Socket
    - 元々C言語で作られたネットワークAPI
    - 他の言語でもSocketから派生したAPIを持っていることが多い
    - LLプログラマでも知識として知っておくことをお勧め
- 今回はクライアントに限定して説明
  - 名前解決やフォールバックなど、いろいろな知識が習得できる
    - 別途サーバプログラミングも勉強すると、さらに理解が深まります

IPv6のプログラム解説

RICOH
imagine. change.

IPv6 Promotion Council

- Socketプログラミングについて公開済み
  - 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
- プログラミングの解説は下記のとおり
  - クライアント＆サーバプログラミングの解説
    - 上記協議会Webサイトの解説&サンプルプログラム
      - http://www.v6pc.jp/jp/upload/pdf/socket-20121203.pdf
      - http://www.v6pc.jp/jp/upload/pdf/socket-sample-20121203.pdf
    - 昨年のIW T7セッション
      - https://www.nic.ad.jp/ja/materials/iw/2012/proceedings/t7/

IPv6プログラミングとは

RICOH
imagine. change.

- IPv4対応プログラム（シングルスタック）
  - ひとつのプロトコルに対応していた
- IPv6/IPv4両対応プログラム（デュアルスタック）
  - 複数のプロトコル・複数アドレスの中のどれでサーバに接続するか選ばなければならない

ただ単に関数を変更するだけではだめ
どのプロトコル・アドレスを使うか選択する機構が必要

Socketプリミティブの紹介

シングルスタック・デュアルスタック それぞれの流れ

RICOH
imagine. change.

シングルスタックの流れ

- ホスト名解決
- サービス名解決
- Socket生成
- Connect実行
- デスクリプタによる入出力
- クローズ

デュアルスタックの流れ

- ホスト名解決
- サービス名解決
- 得られた複数のプロトコル・アドレスでループ
  - Socket生成
  - Connect実行
  - 接続失敗したら
    - 次のプロトコル・アドレスを選択
  - 接続成功したら
    - 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

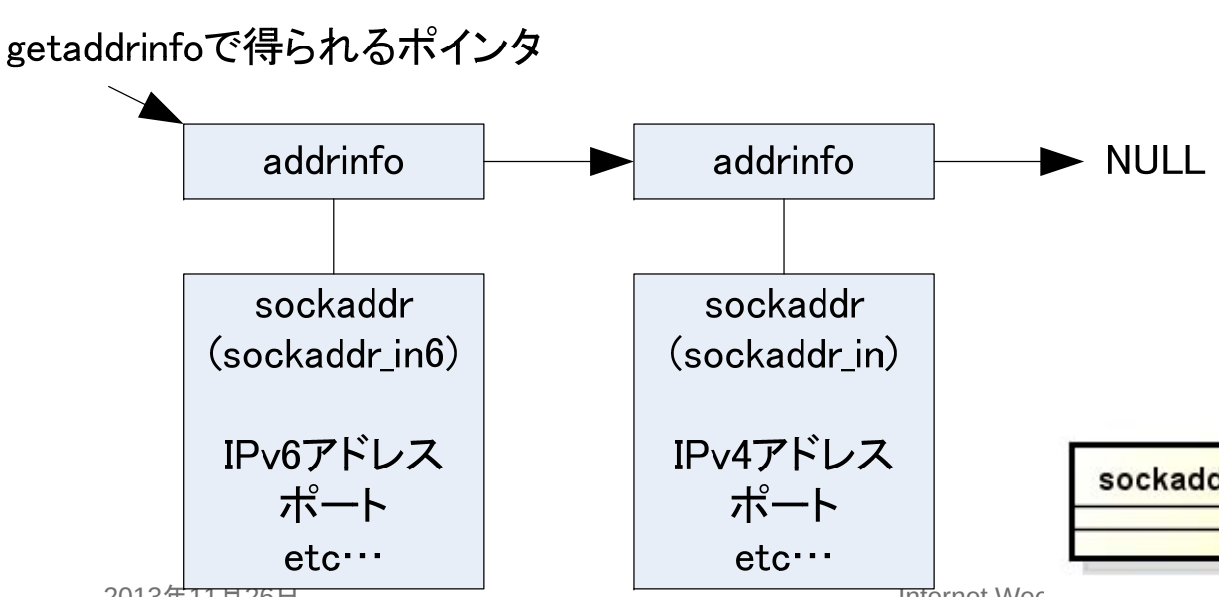
RICOH
imagine. change.

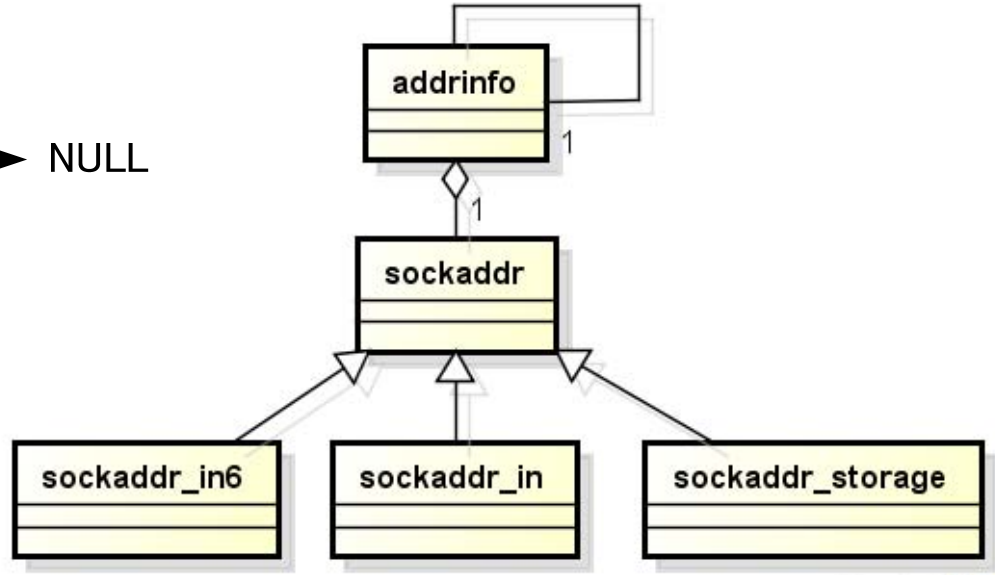
- IPv4
  - ホスト名 : *gethostbyname()*でhostent構造体を得る
  - サービス : *getservbyname()*でservent構造体を得る
- デュアルスタック
  - *getaddrinfo()*でaddrinfo構造体のリストを得る
    - リストの開放も可能で、*freeaddrinfo()* 関数による
- 注意
  - *gethostbyname2()* はIPv6を扱えるが使うべきではない

addrinfo構造体とsockaddr構造体

RICOH
imagine. change.

- addrinfo構造体
  - 複数のアドレス情報をLinked Listで保持する
  - 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
  - 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）





sockaddr構造体の種別

RICOH
imagine. change.

- sockaddr構造体のサブクラス
  - sockaddr構造体はネットワークアドレスを記憶する
  - sockaddr_in
    - IPv4アドレスの構造体（inはInternetの略）
  - sockaddr_in6
    - IPv6アドレスの構造体
  - sockaddr_*
    - いろいろなプロトコルのアドレス
  - sockaddr_storage
    - どのプロトコルのアドレスが書き込まれても記憶できるだけの十分な容量を持った構造体

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()*とaddrinfo構造体からホスト名やサービス名の文字列を取得できる
  - 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);
```

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

■実際のソースコードの紹介

シングルスタック・デュアルスタック それぞれの流れ

RICOH
imagine. change.

シングルスタックの流れ

- ホスト名解決
- サービス名解決
- Socket生成
- Connect実行
- デスクリプタによる入出力
- クローズ

デュアルスタックの流れ

- ホスト名解決
- サービス名解決
- 得られた複数のプロトコル・アドレスでループ
  - Socket生成
  - Connect実行
  - 接続失敗したら
    - 次のプロトコル・アドレスを選択
  - 接続成功したら
    - デスクリプタによる入出力
    - クローズ

実際のソースコード(1)

RICOH
imagine. change.

Includeの指定

```
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <stdlib.h>
#include <netdb.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
```

実際のソースコード(2)

RICOH
imagine. change.

■変数定義～名前解決

```
struct addrinfo hints, *res, *res0;
const char * target_name =″www.v6pc.jp″;
const char * target_port = ″http″;
int s, error; size_t l;char buf[256];

memset(&hints, 0, sizeof(hints));
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;

error = getaddrinfo(target_name, target_port, &hints, &res0);
if(error){
  exit(1);
}
```

実際のソースコード(3)

RICOH
imagine. change.

■接続～フォールバック～通信～終了

```
for (res = res0; res; res = res->ai_next) {
        fprintf(stderr, "trying %s port %s¥n", hbuf, sbuf);
        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;
        }
        write(s,"GET /¥r¥n", 7);
        while ((l = read(s, buf, sizeof(buf))) > 0)
                write(STDOUT_FILENO, buf, l);
        close(s);
        exit(0);
}
exit(1);
```

フォールバック問題

フォールバック問題

RICOH
imagine. change.

- フォールバックとは
  - 前述の名前解決結果のLinked Listを辿ること
- 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
- 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【サーバ側に問題】
  - ネットワークの接続性が失われている【経路に問題】
  - サーバへの到達姓のないアドレスで通信をしようとしている【クライアント側に問題】

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・経路・クライアントと、さまざまな要素が考えられる
- 過渡期なので注意が必要

■最後に

最後に

RICOH
imagine. change.

- C言語によるクライアントプログラミングのまとめ
  - 基本的な流れはわかりやすい
    - 名前解決～接続～通信～切断
  - 名前解決のデータ構造に注意
    - Linked List形式でアドレスを保持
    - 多彩なアドレスを構造体の継承で表現している
- 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
  - kohki@lemegeton.org
  - Twitter：@torawarenoaya
  - facebook：http://www.facebook.com/kohki.ohhira