NTTソフトウェア

Asterisk の IPv6 対応について

エヌ・ティ・ティ・ソフトウェア株式会社
高宮紀明

Asteriskは米国Digium社の登録商標または商標です。
そのほかの記載の会社名、製品名は、それぞれの会社の
商標もしくは登録商標です。

自己紹介

- 1999 年より IPv6 にかかわり始める。
- 2000 年 IPv6 対応ルータを販売
  - 第一回 TAHI プロジェクト相互接続試験に参加
- USAGI プロジェクト参加（～2008）
  - 主に MIPv6 スタック開発担当
  - 成果は 2.6.21 以降の Linux カーネルにマージ
- IPv6 普及・高度化推進協議会参加
  - アプリケーションのIPv6対応検討SWG に参加
- NGN対応ソフトウェア開発・販売（2008～）

目次

- はじめに
- SIPについて
- Asterisk について
- 実際の通信について
- まとめ

はじめに

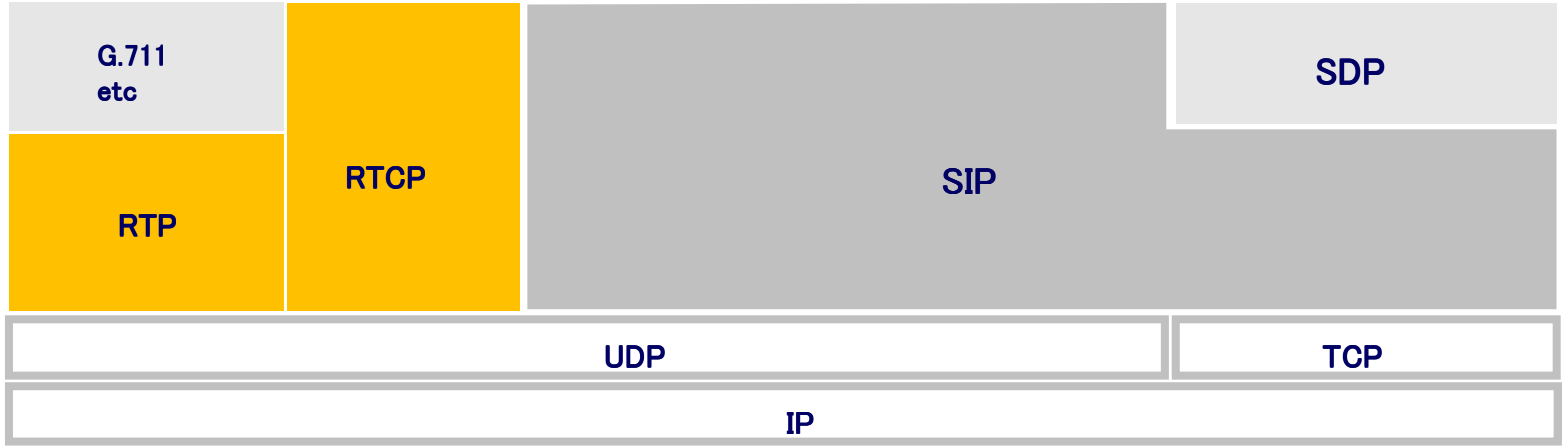
- IPv6に対応したアプリケーション開発に必要な要素として、開発環境、ツール、そしてコーディングなどがあげられるが、これらに関し、必要な情報が不足している
- 本セッションでは前半の内容をもとに、オープンソースにおけるIPv6 対応の実例として、Asterisk(IP-PBXソフトウェア) のIPv6対応内容を紹介する。

SIP について

- SIP プロトコル概要
- 接続シーケンス

SIP プロトコル概要

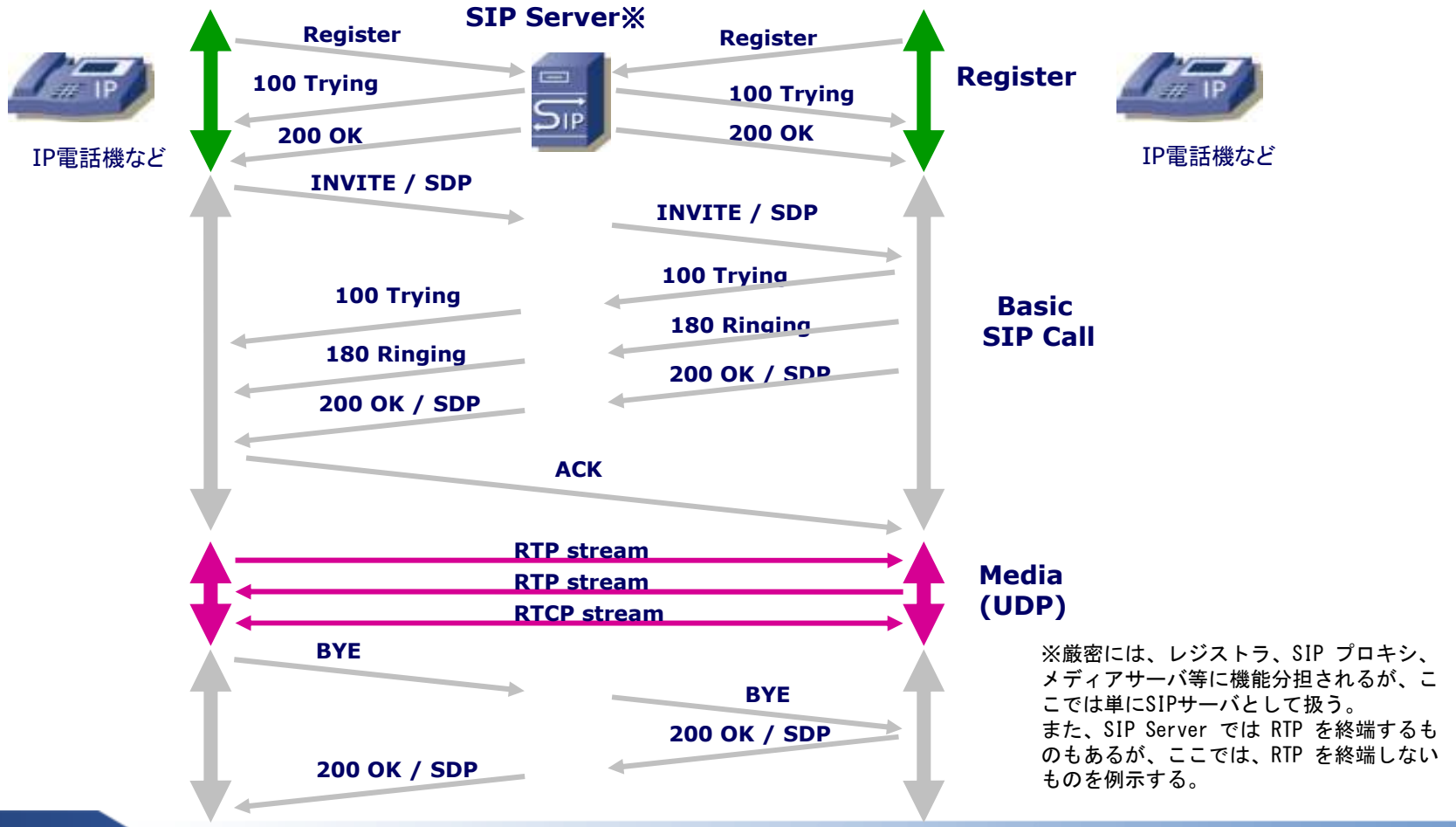
- SIP(Session Initiation Protocol)とは、2つ以上のクライアント間でセッションを確立するためのIETF標準の通信プロトコル
- IETFにて、汎用のセッション制御プロトコルとして開発された
- 特徴として、HTTPに似た、テキストベースのリクエストとレスポンスによって通信を行い、相手先(通話先) はURI(Uniform Resource Identifier)を指定する。



G.711: ITU-T standard for audio companding.
SDP: Session Descripton Protocol
RTCP: RTP Control Protocol
SIP: Session Initiation Protocol
RTP: Realtime Transport Protocol
TCP: Transmisson Control Protocol
UDP: User Datagram Protocol
IP: Internet Protocol

SIP の一般的な接続シーケンス

- IP電話機などのSIP端末は、SIPサーバにREGISTERを送信することで発着可能な状態となる。この時に、IP電話機とSIPサーバ間で使用するプロトコルファミリが決まる。



※厳密には、レジストラ、SIP プロキシ、メディアサーバ等に機能分担されるが、ここでは単にSIPサーバとして扱う。
また、SIP Server では RTP を終端するものもあるが、ここでは、RTP を終端しないものを例示する。

Asterisk について

- Asterisk とは
  - オープンソースの IP-PBX ソフトウェア
    （IPネットワーク内で、IP電話端末の回線交換を行う装置およびソフトウェア）
  - http://www.asterisk.org/
  - 複数のバージョン（1.4.x、1.8.x、10.x、11.x、12.x）
    - 1.8.x、11.x はLTS（Long Term Support）としてリリース
- Asterisk の IPv6 対応について
  - バージョン 1.8 系より対応（最新版は、11.0.1（2012/11/19現在））
  - IPv6 対応箇所
    - 呼制御（SIP/IAX）
      SIP は、UDP/TCP/TLS に対応
    - 管理機能（設定用 Web インタフェース、AMI:Asterisk Management Interface）
    - メディアトランスポート（RTP/SRTP）
    - IPv4/IPv6 の相互接続について
      - B2BUA （ゲートウェイ）の接続形式で、IPv4 端末と IPv6 端末との相互接続が可能（この場合、Asterisk が動作している計算機の OS がデュアルスタックで動作していることが前提）

Asterisk の主要機能と IPv6対応

大項目	中項目	概要	IPv6 対応
呼制御	SIP	SIP による呼制御機能	○
	IAX	IAX による呼制御機能	○
	H.323	H.323 による呼制御機能	×
	Websocket	Websocket による呼制御連携	○
メディア処理	RTP	音声/映像ストリーム	○
暗号化	SIPS 対応	SIP over TLS	○
	SRTP	暗号化 RTP	○
管理機能	AMI	Asterisk Management Interface	○
	Web インタフェース	ブラウザからの設定機能	○
PBX 間連携	DUNDi	Asterisk 間の相互接続機能	×

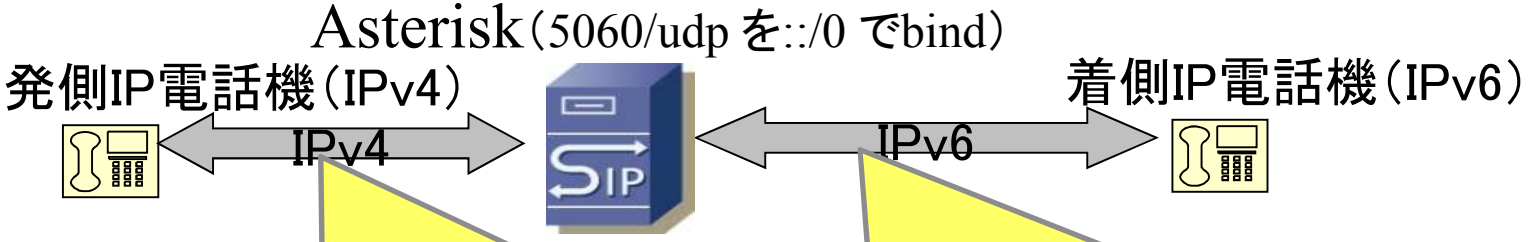
2013/11月 現在（Asterisk 11）

Asterisk の動作環境

- ここでは、Asterisk のデュアルスタック環境での動作について説明します。
  - Asterisk の接続形式
  - Asterisk の設定
  - Asterisk が出力するログファイル

Asterisk の接続形式

- 発信側 IP 電話機は REGISTER 送信時に指定したプロトコルファミリに従い、発信を行う（INVITE の送信）。
- Asterisk は、着側が REGISTER 送信時に指定したプロトコルファミリにより、INVITE を転送する。




```
INVITE sip:6001@192.168.1.212:5060;...
From:
<sip:6000@192.168.1.214>;tag=as3350c74c
To: <sip:6001@192.168.1.212:51784>
Contact: <sip:6000@192.168.1.214:5060>
Call-ID: 4c9d4cb63@192.168.1.214:5060
CSeq: 102 INVITE
Content-Type: application/sdp
Content-Length: 442

v=0
c=IN IP4 192.168.1.214
:
```

```
INVITE sip:6001@[2001:db8:cafe:babe::2] SIP/2.0
From: <sip:6000@[2001:db8:cafe::babe::1]>;tag=623328519
To: <sip:6001@[2001:db8:cafe:babe::2]>
Contact: <sip:6000@[2001:db8:cafe:babe::1]>
Call-ID: 134945149
CSeq: 20 INVITE
Content-Type: application/sdp
Content-Length: 384

v=0
c=IN IP6 2001:db8:cafe:babe::1
:
```

設定ファイル

- 待ち受けに使用される IP アドレスは IPv4/IPv6 の両方が記述可能（設定ファイルは sip.conf）
- IPv4 example: bindaddr=0.0.0.0:5060
  bindaddr=0.0.0.0
- IPv6 example: bindaddr=[::]:5060
  bindaddr=::
- bindaddr=0.0.0.0 とした場合、IPv4 のみ受信可能
- bindaddr=<特定の IPv4/IPv6 アドレス>と指定した場合は、そのプロトコルファミリのみ受信可能
- デュアルスタックにする場合は、bindaddr=:: とするが、挙動は OS 依存である
  - Linux の場合、デュアルスタックで運用するためには /proc/sys/net/ipv6/bindv6only=0 であることを確認すること。

ログファイル

- ログファイルでは、IPv6 アドレスが出力される。
  - アドレス文字列の実体は getnameinfo() により得られるアドレス文字列が出力される。
  - RFC5952（A Recommendation for IPv6 Address Text Representation）に従うかどうかは実行環境での getnameinfo() の実装に依存する。

```
[Nov 16 22:50:02] VERBOSE[6103] chan_sip.c: Peer audio
RTP is at port [2001:db8:cafe:babe::2]:7078
[Nov 16 22:50:02] VERBOSE[6103] chan_sip.c: Peer video
RTP is at port [2001:db8:cafe:babe::2]:9078
```

ソケット関連の構造体・関数の扱いについて

- BSD ソケットの扱いは、Asterisk の内部でライブラリ化されており、SIP の処理の中で直接呼び出すことはほとんどない。
- ここでは、Asterisk が保持しているアドレス情報や内部関数の中で、BSD ソケットで使用される構造体・関数がどのように使用されているかを解説する。
  - アドレス情報の保持
  - IPv6 汎用関数
  - 実際の接続について

アドレス情報の保持(1)

- IPv6対応前では *struct sockaddr_in* が使用されていたが、IPv6対応後は *struct sockaddr_storage* が使用される

IPv6 対応前
(asterisk 1.4.40)

channel/chan_sip.c より

```
struct sip_pvt {
:
struct sockaddr_in sa;
};
struct sip_peer {
:
struct sockaddr_in addr;
};
```

IPv6対応後
(asterisk 11.6)

include/asterisk/netsock2.h より

```
struct ast_sockaddr {
struct sockaddr_storage ss;
socklen_t len;
};
```

channel/sip/include/sip.h より

```
struct sip_pvt {
:
struct ast_sockaddr sa;
};
struct sip_peer {
:
struct ast_sockaddr addr;
};
```

一部のOS(MacOS 等)では構造体の長さが必要なため、移植性を考慮して追加されている

アドレス情報の保持(2)

- アドレスの文字列を格納する文字配列の要素数を *INET_ADDRSTRLEN* (16)から *INET6_ADDRSTRLEN* (46)へ変更している

IPv6 対応前
(asterisk 1.4.40)

channel/chan_sip.c より

```
static void realtime_update_peer(…)
{
:
char ipaddr[INET_ADDRSTRLEN]

};
```

IPv6対応後
(asterisk 11.6)

channel/sip/include/sip.h より

```
static void realtime_update_peer(…)
{
:
char ipaddr[INET6_ADDRSTRLEN]

};
```

IPv6 対応汎用関数(1)

【main/netsock2.c】

- int ast_sockaddr_split_hostport(char *str, char **host, char **port, int flags)
  - IP アドレスとポートを分離する
  - ex) [2001:db8:cafe:babe::1]:5060 => 2001:db8:cafe:babe::1 と 5060 に分離する

```
if (*s == '[') {
        *host = ++s;
        for (; *s && *s != ']'; ++s) {
        }
        if (*s == ']') {
                host_end = s;
                ++s;
        }
        if (*s == ':') {
                *port = s + 1;
        }
} else {
    IPv4 文字列を想定した処理(省略)
}
```

IPv6アドレスは"["と"]"でアドレス文字列がくくられているため、その文字列チェックを行っている(rfc3261)

IPv6 対応汎用関数(2)

【main/netsock2.c】

- int ast_sockaddr_resolve(struct ast_sockaddr **addrs,
  const char *str, int flags, int family)
  - アドレス解決を行う。引数 family にプロトコルファミリーが入る
  - 内部で呼ばれる getaddrinfo() の結果が、引数 addrs に保存される
  - 名前解決のみで使用され、getaddrinfo() の結果はチェックされない。

```
memset(&hints, 0, sizeof(hints));
hints.ai_family = family;
hints.ai_socktype = SOCK_DGRAM;
if ((e = getaddrinfo(host, port, &hints, &res))) {
    (エラー処理、省略)
}
res_cnt = 0;
for (ai = res; ai; ai = ai->ai_next) { res_cnt++; }
i = 0;
for (ai = res; ai; ai = ai->ai_next) {
    (*addrs)[i].len = ai->ai_addrlen;
    memcpy(&(*addrs)[i].ss, ai->ai_addr, ai->ai_addrlen);
    ++i;
}
```

getaddrinfo()の検索条件を設定する

getaddrinfo()の検索結果の数を計上し、関数の引数にコピーしている。

IPv6 対応汎用関数(3)

【channel/chan_sip.c】

- static int ast_sockaddr_resolve_first_af(struct ast_sockaddr *addr, const char* name, int flag, int family)
  - 指定したプロトコルファミリでホスト名の解決を行い、先頭の1つだけを返却する
  - 内容は前ページの ast_sockaddr_resolve() を呼び、その一つだけを返す実装となっている。
- static int ast_sockaddr_resolve_first(struct ast_sockaddr *addr, const char* name, int flag)
  - 設定ファイルで指定されたデフォルトのプロトコルファミリでホスト名の解決を行い、先頭の1つだけ返却する
  - 上記の ast_sockaddr_resolve_first_af() の最後の引数 family を、設定ファイルで記述されているサーバの受信アドレスに従って設定される。
    →[::] となっていれば、PF_UNSPEC が指定される。この場合は DNS の返却結果と getaddrinfo() が struct addrinfo 構造体に結果を返す実装に依存する。

IPv6 対応汎用関数(4)

【main/netsock2.c】

- int ast_sockaddr_is_ipv4_mapped(const struct ast_sockaddr *addr)
  - 引数のソケットアドレス構造体が IPv4 mapped address かどうかをチェックする。

```
int ast_sockaddr_is_ipv4_mapped(const struct ast_sockaddr *addr)
{
        const struct sockaddr_in6 *sin6 =
                (struct sockaddr_in6 *)&addr->ss;
        return addr->len && IN6_IS_ADDR_V4MAPPED(&sin6->sin6_addr);
}
```

<netinet/in.h> で定義されている、アドレスの種類を判定するマクロ

IPv6 対応汎用関数(5)

【main/netsock2.c】

- int ast_sockaddr_ipv4_mapped(const struct ast_sockaddr *addr, struct ast_sockaddr *ast_mapped)
  - IPv4 mapped address(実体は struct sockaddr_in6)が格納されたソケットアドレス構造体を、IPv4 アドレスが格納された形式(実体は struct sockaddr_in)に変換する

```
if (!ast_sockaddr_is_ipv4_mapped(addr)) {
    return 0;
}
struct sockaddr_in6 *sin6 = (const struct sockaddr_in6*)&addr->ss;
struct sockaddr_in sin4;
memset(&sin4, 0, sizeof(sin4));
sin4.sin_family = AF_INET;
sin4.sin_port = sin6->sin6_port;
sin4.sin_addr.s_addr = ((uint32_t *)&sin6->sin6_addr)[3];
ast_sockaddr_from_sin(ast_mapped, &sin4);
```

ast_sockaddr_from_sin() 内で、ast_mapped に struct sockaddr_in 構造体がコピーされる。
ast_mapped->len は、sizeof(struct sockaddr_in) が設定される

実際の通信について

- Asterisk 内部では、これまでに説明した汎用関数を使用して名前解決を実施するが、たいていの場合はgetaddrinfo() の最初のエントリのみを参照する仕様となっている。プロトコルファミリについては、REGISTER 受信時で使用されるプロトコルファミリをもとに決定しているため、Asterisk と IP 電話機間では使用するプロトコルはあらかじめ決定されている。

まとめ

- アドレス操作関数は、IPv4/IPv6 を意識しているものの、局所化できるように内部関数を作成して使用している
- getaddrinfo() の最初のエントリを使用する仕様になっているが、SIP による通話の開始前の REGISTER 処理によりプロトコルファミリが解決されている前提となっている。

参考文献(Webサイト)

- SIP IPv6 関連
  - SIP FORUM IPv6 task group
    http://www.sipforum.org/content/view/398/286/
- Asterisk 関連
  - Asterisk wiki
    http://wiki.asterisk.org
- IPv6 普及・高度化推進協議会 IPv4/IPv6共存WG
  - http://www.v6pc.jp/jp/entry/wg/2012/12/ipv610.phtml
- IPv6 関連
  - Internet Week 2011より 事例から学ぶIPv6トラブルシューティング
  - http://www.nic.ad.jp/ja/materials/iw/2011/proceedings/t2/t2-02.pdf