IPv4 EXHAUSTION

IPv6オペレータ育成プログラム

iDC/ISP/CATV サーバ編

株式会社BeaconNC データセンター事業部

國武 功一

ここで取り上げること

- 目的
  - Linuxを用い、IPv6環境下でのサーバ運用を行う上での必要な知識を身につける。


（注）ディストリビューションに依存する部分については、CentOS(RedHat系）とGNU/Debianを取り上げます。

Overview

- IPv6の主な特徴
- LinuxでIPv6を扱うための基礎知識
  - IPv6のアドレス表記
  - IPv6のアドレス体系
  - ULAについて
  - ヘッダ情報について
  - デュアルスタックとは
  - DNSのIPv6対応とは

- **IPv6の主な特徴**
- LinuxでIPv6を扱うための基礎知識
  - IPv6のアドレス表記
  - IPv6のアドレス体系
  - ULAについて
  - ヘッダ情報について
  - デュアルスタックとは
  - DNSのIPv6対応とは

IPv6の主な特徴

- 広大なアドレス空間(32bitから128bitへ)
- アドレスの自動設定
  - 管理コストの低減を目指す
- IPsec標準装備
- MobileIP
  - IPv4でのMobileIPを、もっと洗練されたものへ

その他：Headerの簡略化、途中経路でのフラグメントの禁止、
拡張ヘッダの導入など

IPv6の広大なアドレス空間

- IPv4のアドレス空間を1とすると、IPv6は

  79228162514264337593543950336

  全世界の人に現行のIPv4アドレスをひとりずつ
  配ってもまだまだ余る！

  - 但し、ネットマスクの考え方の違いから、上の例ほど
    単純には比較できない

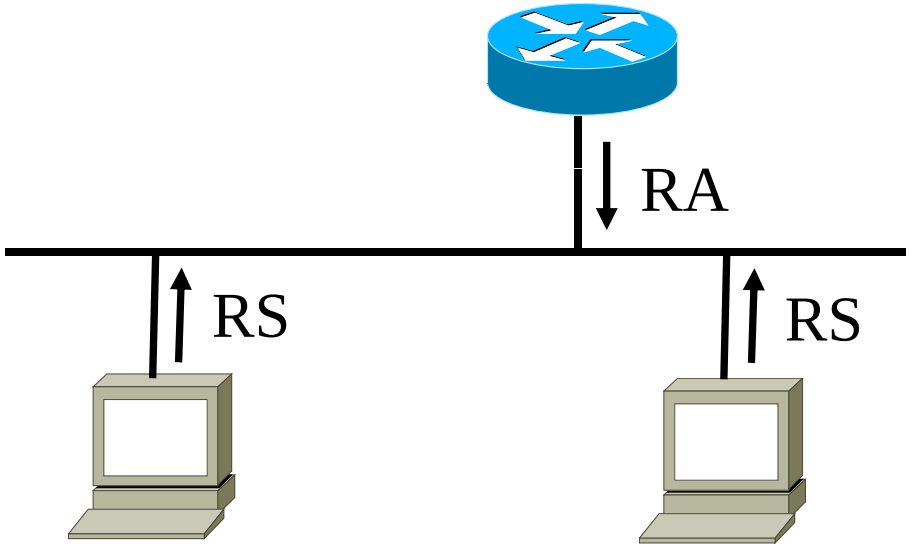
IPv6の広大なアドレス空間(Cont)

- IPv4/IPv6でアドレス個数を比較することは無意味だが、十分に広いと言える。
- 構築可能なセグメント数は？(※)
  - IPv4：1073741824
  - IPv6：18446744073709551616

(※)IPv4は、全アドレス帯を/30で分割した場合。IPv6は、/64で分割した場合。すべてがグローバルアドレスとして利用できるわけではないので、あくまでも目安

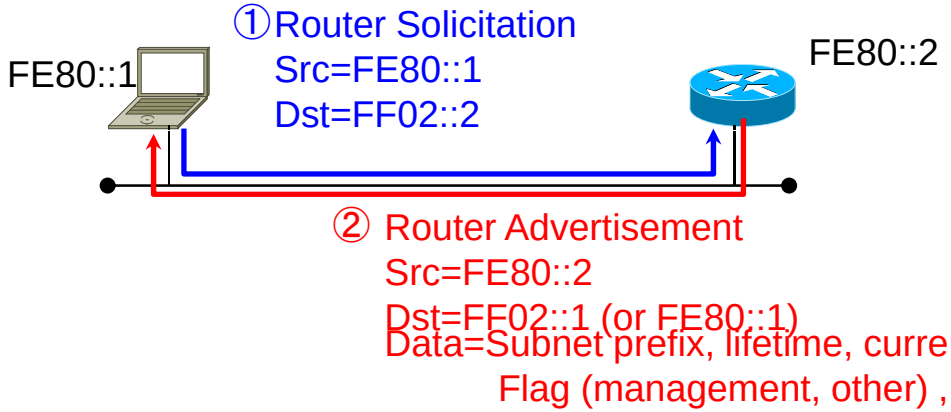
アドレスの自動設定

- IPv6では、予想される莫大な数のデバイスに対応するため、アドレスの自動設定が標準機能として用意されている
  - DHCPV6を利用することも可能



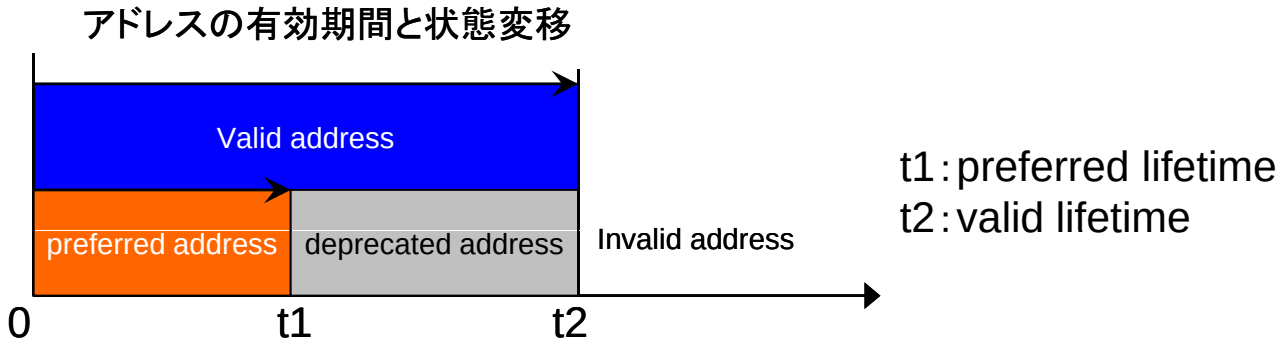
Router Solicitation/Router Advertisement

- RSの宛先アドレスはFF02::2、Hop Limitは255
- RAの宛先アドレスはFF02::1かRS内の始点アドレス、Hop Limitは255
- RA内のCurrent Hop Limitフィールドでノードが用いるホップ制限を設定
- M-flagが0ならステートレスアドレス自動設定、1ならDHCPv6によるアドレス設定
- O-flagが1ならアドレス以外の情報をDHCPv6により取得
- Router Lifetimeはデフォルトルータのみが1以上（65535以下）を指定
- DRP（Default Router Preference：RFC4191）によってデフォルトルータの優先度の通知が可能
  - High（01）、Medium（00）、Low（11）
  - ノード、ルータ双方がサポートしている必要がある

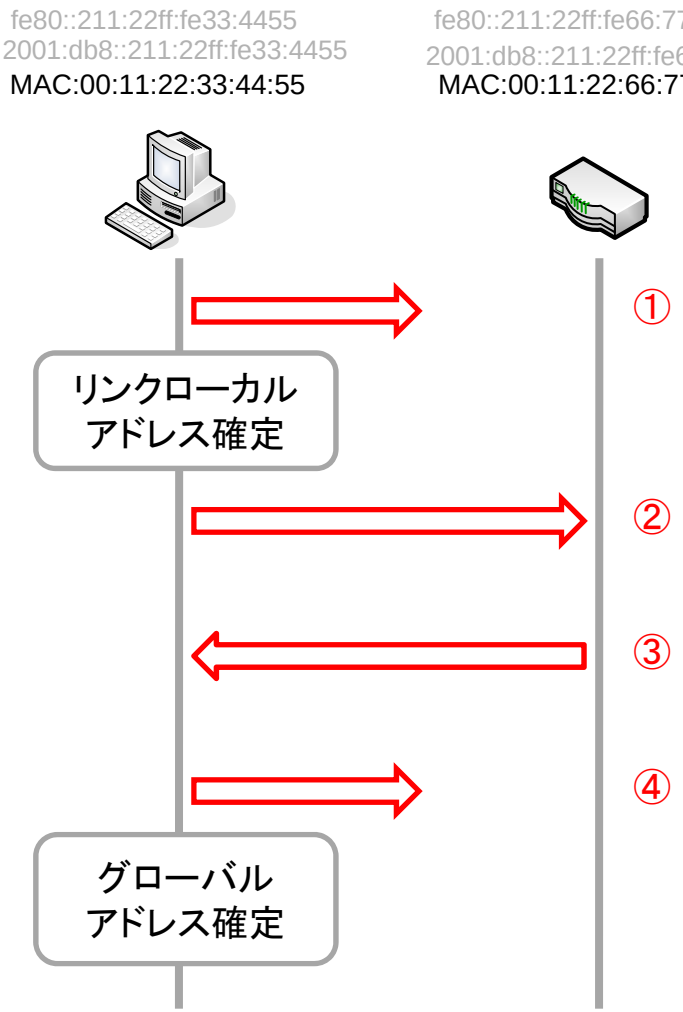


IPv6アドレスの状態、アドレスのlifetime

- tentative address
  - インタフェースに付与されていないアドレスでNDメッセージにしか使用できない。この時点でアドレスの一意性をDADで確認する。
- preferred address
  - インタフェースに付与されたアドレス。アドレスが一意で通信可能な状態
- deprecated address
  - 有効ではあるが、新規通信への使用をしないことが望まれる
- valid address
  - Preferredとdeprecatedのアドレスの双方を指す
- Invalid address
  - 有効アドレスの有効期間が過ぎるとこの無効アドレスになる



ステートレス自動アドレス設定の流れ



①近隣要請(NS)
近隣広告がなければ
ターゲットアドレス
の利用が可能
＜重複アドレス検出＞
要請ノードマルチキャスト

Src MAC 00:11:22:33:44:55
Dst MAC 33:33:FF:33:44:55
Src IPv6 ::(未定義アドレス)
Dst IPv6 ff02::1:ff33:4455
ICMPv6 Type 135 Target fe80::211:22ff:fe33:4455

②ルータ要請(RS)
全ルータマルチキャスト
(ff02::2)宛に送信

Src MAC 00:11:22:33:44:55
Dst MAC 33:33:00:00:00:02
Src IPv6 fe80::211:22ff:fe33:4455
Dst IPv6 ff02::2
ICMPv6 Type 133

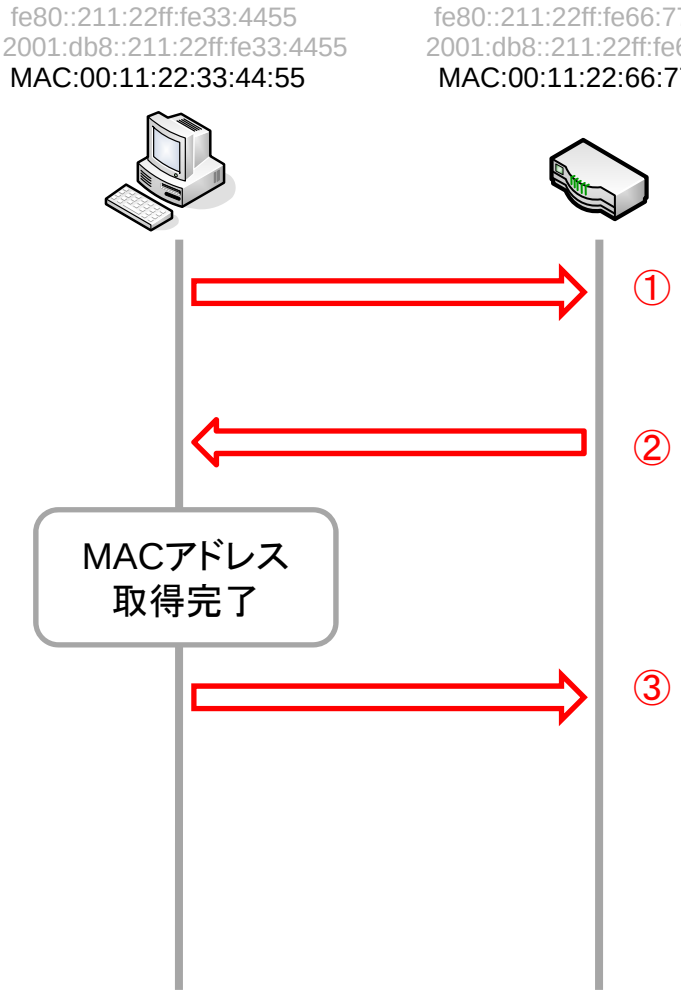
③ルータ広告(RA)
全ノードマルチキャスト
(ff02::1)宛に送信
取得プレフィックス
を用いてグローバル
アドレスを生成

Src MAC 00:11:22:66:77:88
Dst MAC 33:33:00:00:00:01
Src IPv6 fe80::211:22ff:fe66:7788
Dst IPv6 ff02::1
ICMPv6 Type 134 Prefix 2001:db8::

④近隣要請
近隣広告がなければ
ターゲットアドレス
の利用が可能
応答があるとアドレス
を再構成する必要あり
＜重複アドレス検出＞

Src MAC 00:11:22:33:44:55
Dst MAC 33:33:FF:33:44:55
Src IPv6 ::(未定義アドレス)
Dst IPv6 ff02::1:ff33:4455
ICMPv6 Type 135 Target 2001:db8::211:22ff:fe33:4455

リンクレイヤアドレスの解決の流れ



①近隣要請（NS）
通信相手のMACアドレスを探索
近隣広告がない場合は
オンリンクでないと判断

Src MAC 00:11:22:33:44:55
Dst MAC 33:33:FF:66:77:88
Src IPv6 fe80::211:22ff:fe33:4455
Dst IPv6 ff02::1:ff66:7788
ICMPv6 Type 135 Target 2001:db8::211:22ff:fe66:7788

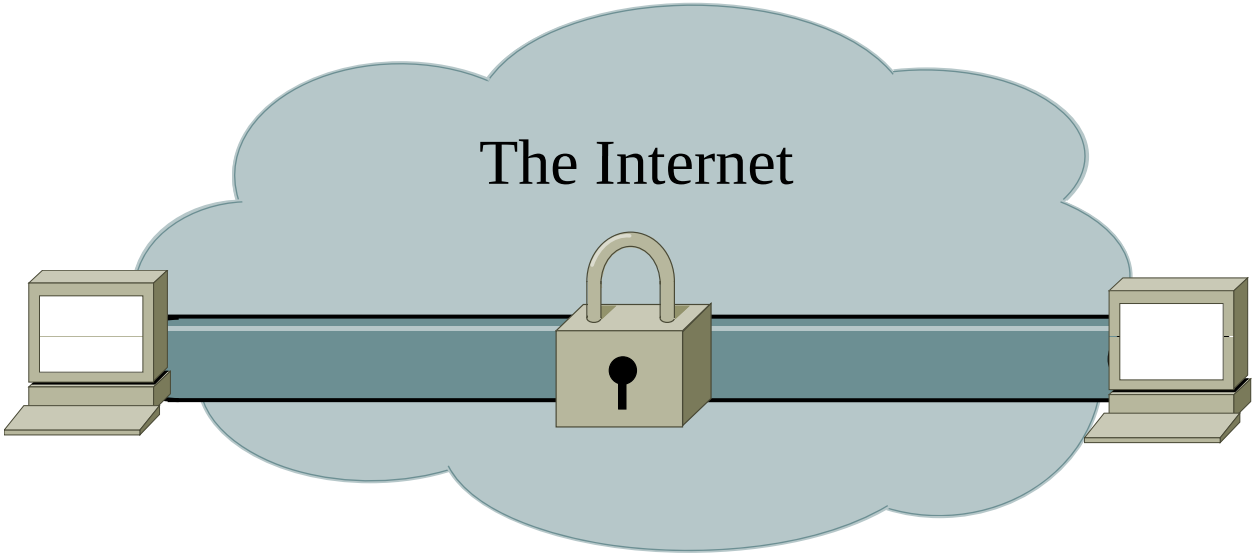
②近隣広告（NA）
ターゲットアドレスを持つノードが回答
ただし誰でもこの応答は可能

Src MAC 00:11:22:66:77:88
Dst MAC 00:11:22:33:44:55
Src IPv6 fe80::211:22ff:fe66:7788
Dst IPv6 fe80::211:22ff:fe33:4455
ICMPv6 Type 136 Target 2001:db8::211:22ff:fe66:7788 Target MAC 00:11:22:66:77:88

③通信開始

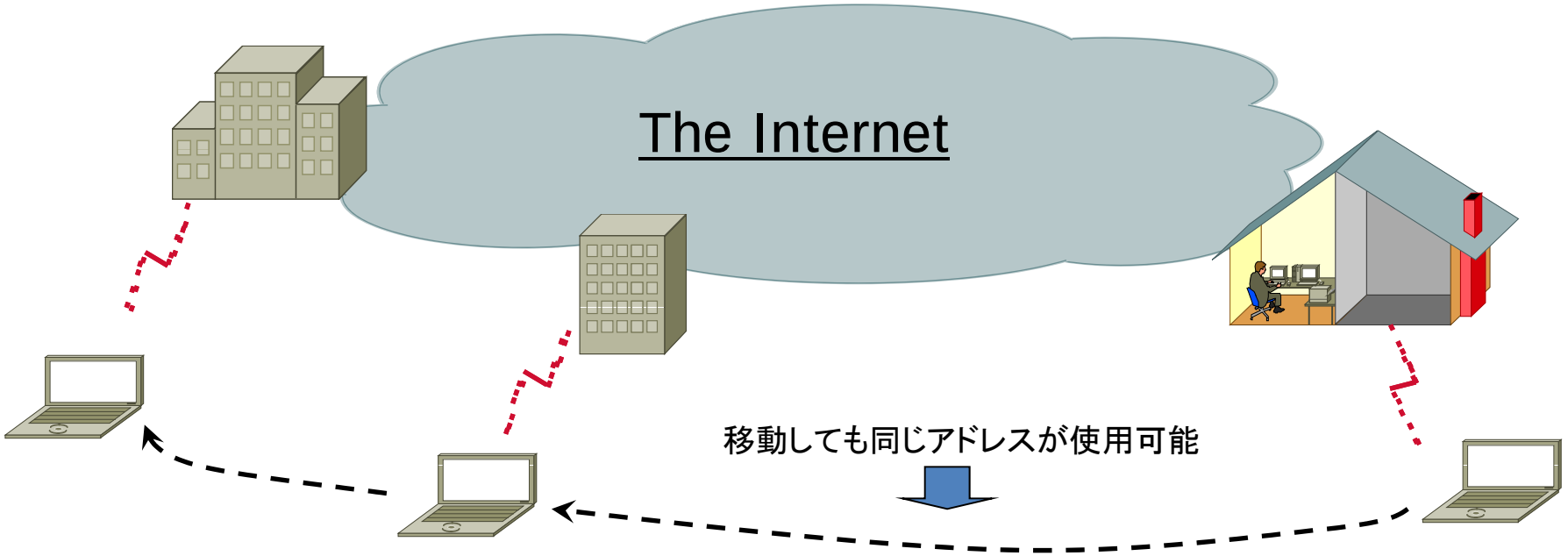
IPsec標準装備

- IPレイヤで通信を保護
  - アプリケーションの改造が必要ない



MobileIP

- ユビキタスとの親和性
  - どこでも同じアドレスが使用可能
  - IPv4でのMobileIPを、より現実的なものへ



IPv4 EXHAUSTION

IPv6オペレータ育成プログラム

LinuxでIPv6を扱うための基礎知識

- IPv6の主な特徴
- LinuxでIPv6を扱うための基礎知識
  - **IPv6のアドレス表記**
  - IPv6のアドレス体系
  - ULAについて
  - ヘッダ情報について
  - デュアルスタックとは

IPv6のアドレス表記

- IPv4のアドレス表記
  - 例）192.0.2.1

    10進数で表した数字を“.”で区切って表記

- IPv6のアドレス表記
  - 例）fe80:0000:0000:0000:02d0:b7ff:fea0:beea

    16進数で表した数字を“:”で区切って表記

IPv6における略記

fe80:0000:0000:0000:02d0:b7ff:fea0:beea

- 各パートの先頭にある0は省略可能
  fe80:0000:0000:0000:**2d0**:b7ff:fea0:beea
- 0000が連続する場合は、一度だけ“::”で省略可能
  fe80**::**02d0:b7ff:fea0:beea

よって、fe80::2d0:b7ff:fea9:beeaと表記できることとなる

では以下の場合は？

- 2001:0db8:0000:0000:fff0:0000:0000:000f

  2001:db8::fff0:0:0:f
  または
  2001:db8:0:0:fff0::f

注）2001:db8::fff0::fとは省略できない

- IPv6の主な特徴
- LinuxでIPv6を扱うための基礎知識
  - IPv6のアドレス表記
  - **IPv6のアドレス体系**
  - ULAについて
  - ヘッダ情報について
  - デュアルスタックとは
  - DNSのIPv6対応とは

IPv6のアドレス体系

IPv6アドレスの分類（一例）

- 挙動
- 適用範囲（スコープ）
- 特殊アドレス

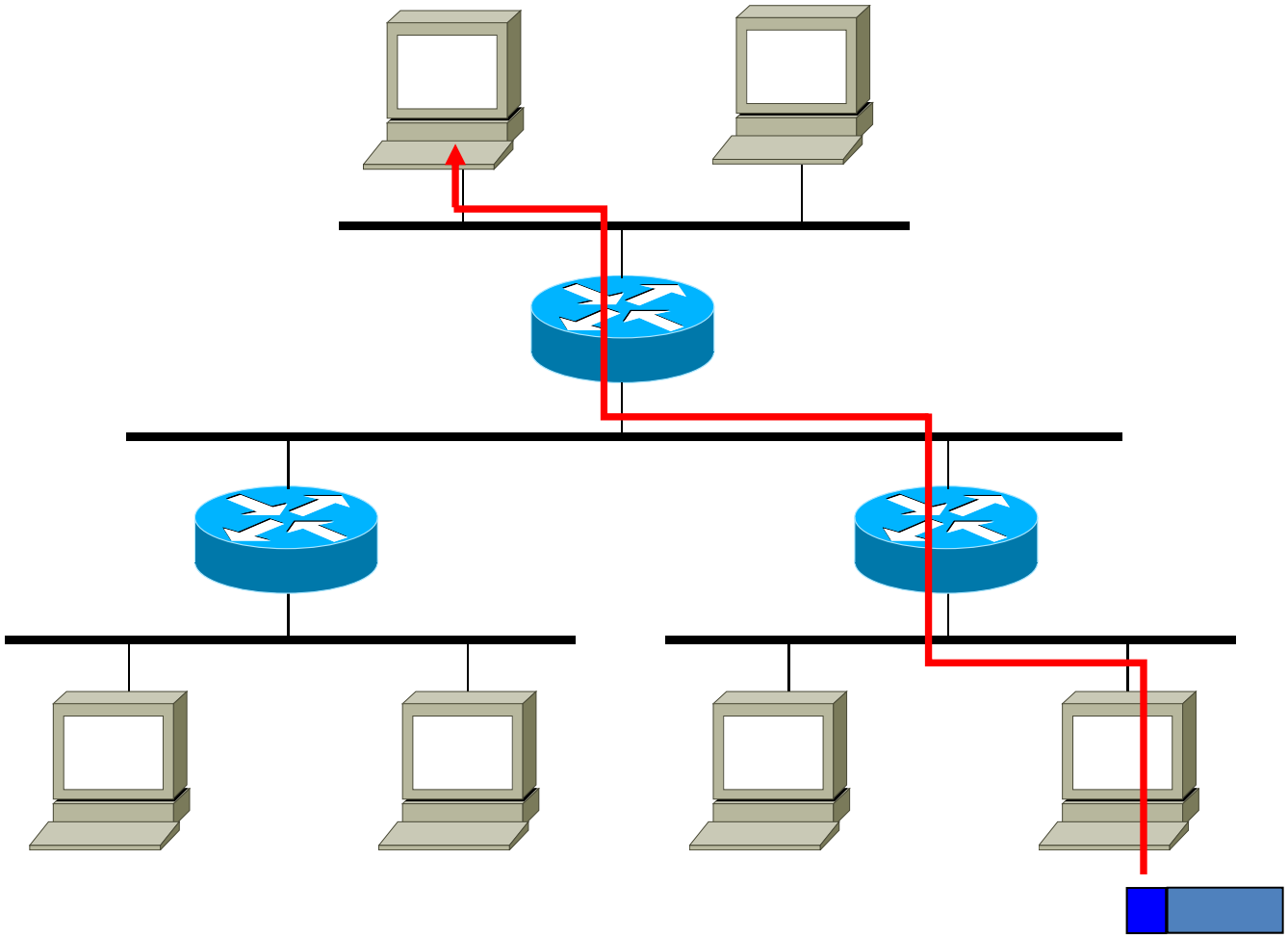
パケットの挙動による分類

- ユニキャストアドレス
- マルチキャストアドレス
- エニーキャストアドレス

ユニキャストアドレス

- 単一のインターフェースに割り当てられるアドレス
- １対１の通信に使用される（普段はこのアドレスが使用される）

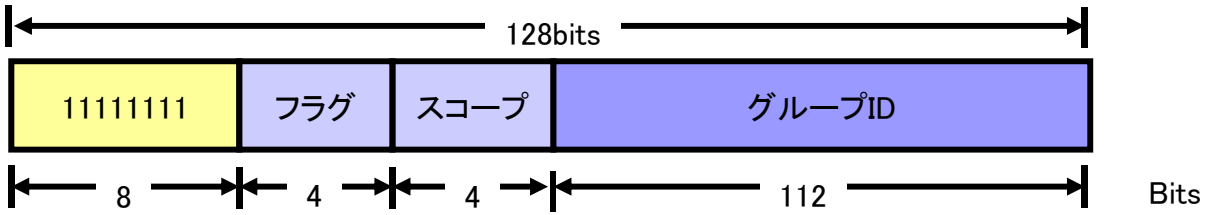
ユニキャストアドレス



マルチキャストアドレス

- あるグループを表すアドレス
  - あるマルチキャストアドレス宛てにパケットをなげると、そのグループに属するすべてのインターフェースに届けられる
  - IPv4におけるブロードキャストは、マルチキャストの１種として取り扱われる。

マルチキャストアドレス

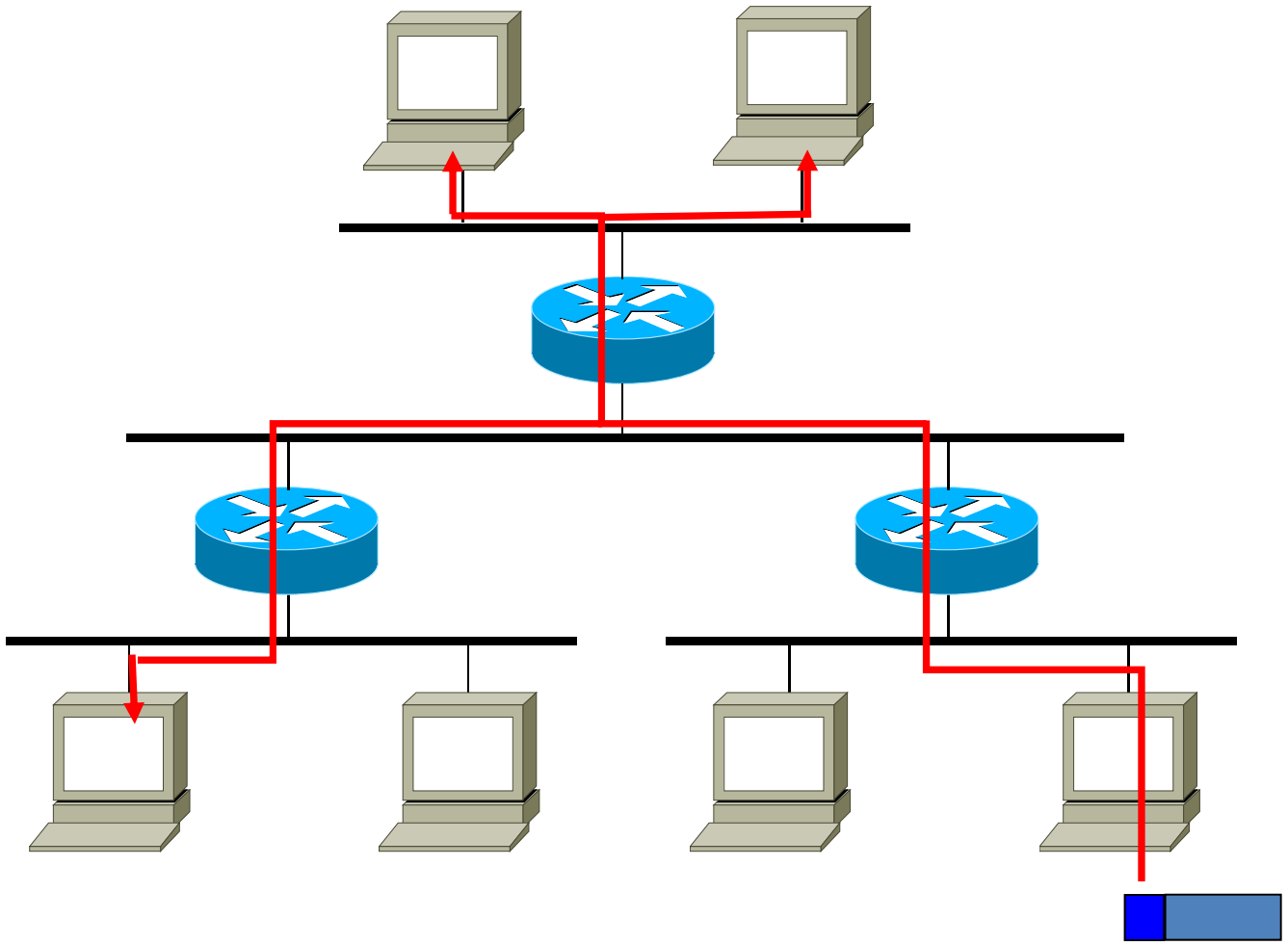


T=0 永久的なマルチキャストを示す
T=1 一時的なマルチキャストを示す

マルチキャストスコープ：4ビットのマルチキャストの力が及ぶ範囲は下記のマルチキャストグループに限られている。

0	予約
1	ノードローカルスコープ
2	リンクローカルスコープ
3	（アンアサインド）
4	（アンアサインド）
5	サイトローカルスコープ
6	（アンアサインド）
7	（アンアサインド）
8	オーガニゼーションローカルスコープ
9	（アンアサインド）
A	（アンアサインド）
B	（アンアサインド）
C	（アンアサインド）
D	（アンアサインド）
E	グローバルスコープ
F	リザーブド

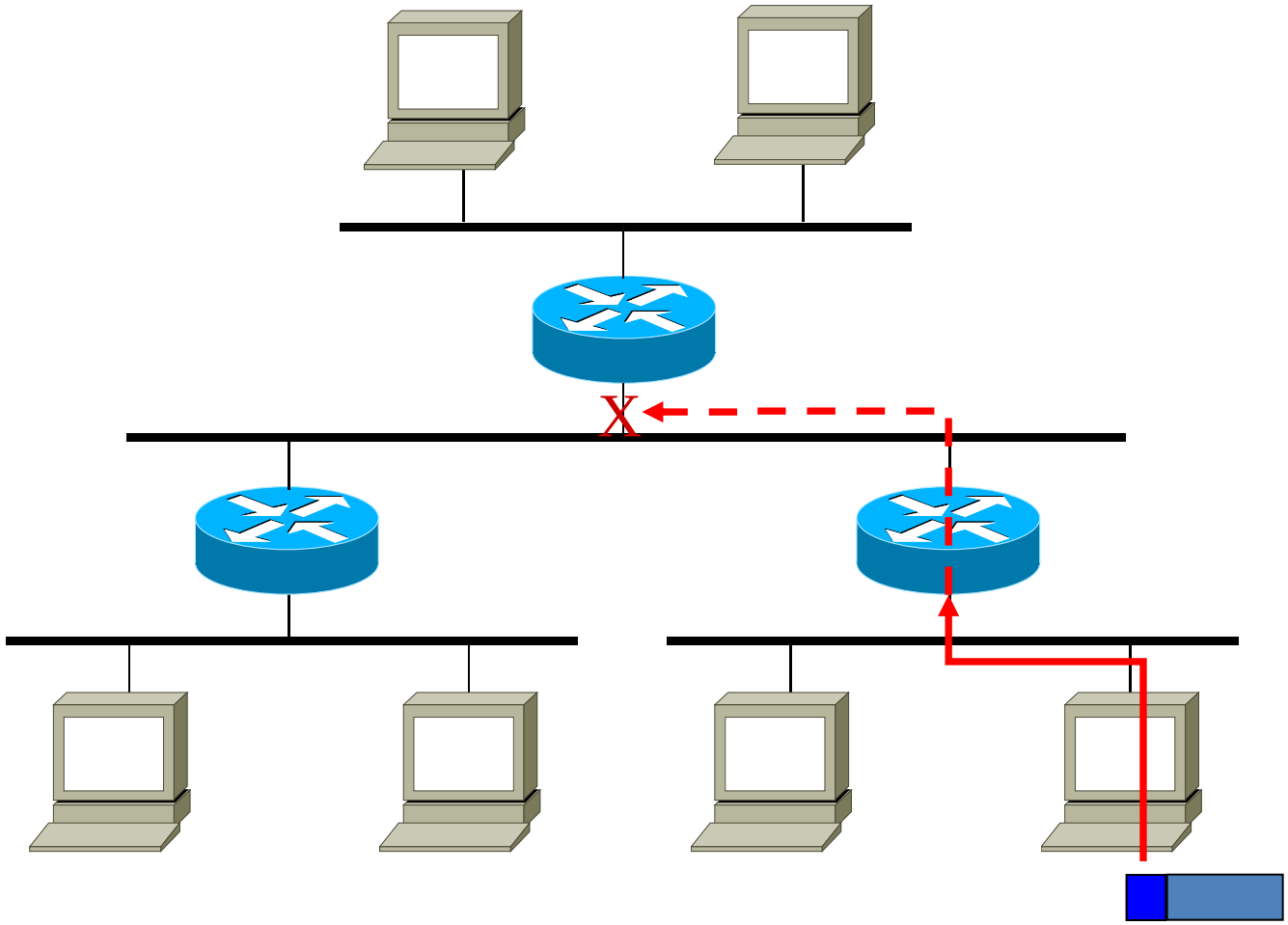
マルチキャストアドレス



エニーキャストアドレス

- マルチキャスト同様、あるグループを表すアドレス
  - マルチキャストとは違い、あるグループに属するインターフェース、すべてに配送されるわけではなく、どれか一つに配送されると、それ以上は配送されない。

エニーキャストアドレス

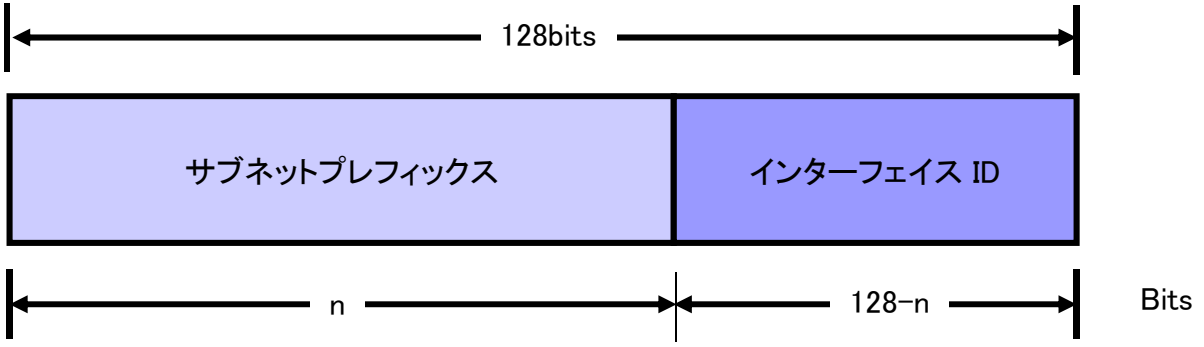


適用範囲による分類

- グローバルアドレス
- サイトローカルアドレス（廃止）
  - ユニークローカルアドレス（ULA)へ
- リンクローカルアドレス

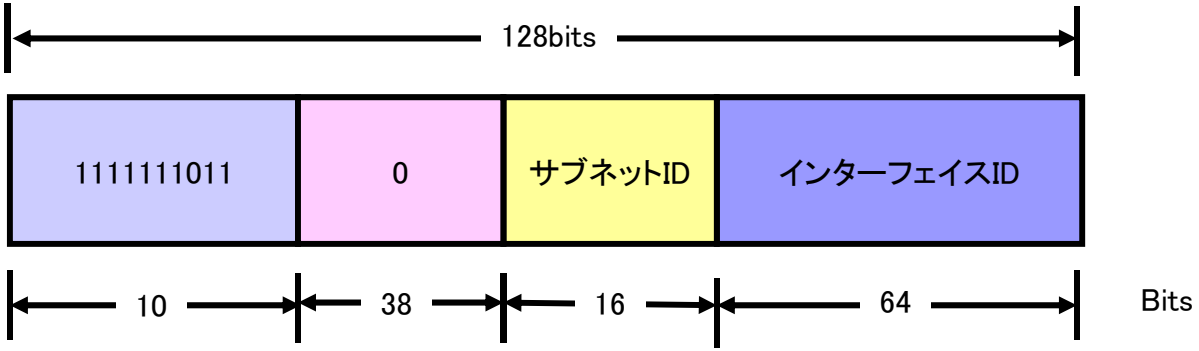
グローバルアドレス

- IPv4でいうところのグローバルアドレスと同義。
- 全世界で一意に決まる識別子



サイトローカルアドレス（廃止に）

- IPv4でいうところのプライベートアドレス
- このアドレスから来るパケットは、他組織には転送されない
- 問題点が多く廃止に（Unique Local IPv6 Unicast Addressへ受け継がれる）

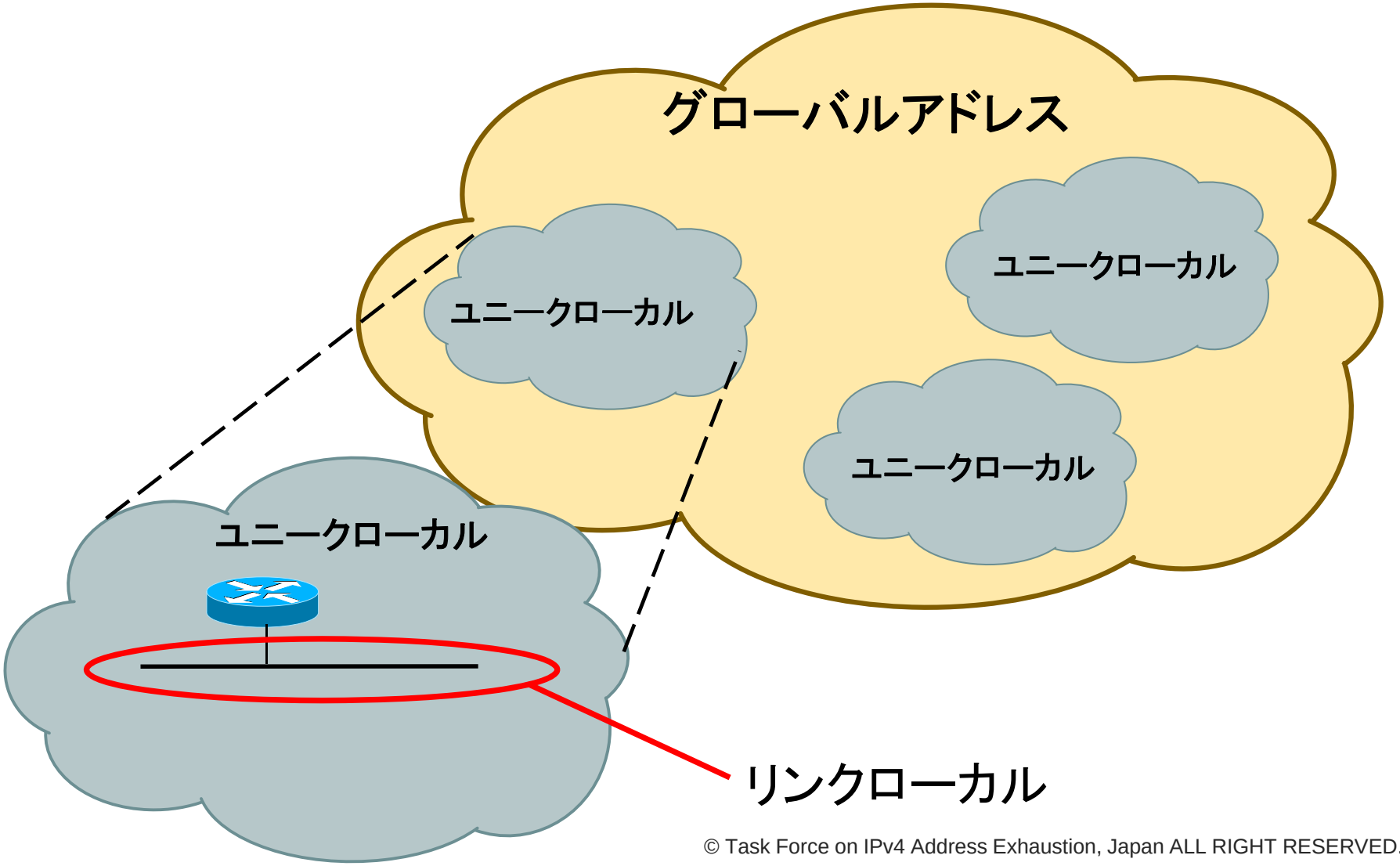


Unique Local IPv6 Unicast(ULA)アドレスでは、サイトローカルアドレスとは異なり、プライベートアドレス的に使えるが、ほぼ他組織と、アドレスが重複することがない

リンクローカルアドレス

- ノードが直接繋がっているリンク内のみで有効なアドレス
- 近隣にRAを投げるルータがなくても自動的に生成される。
- よく知られたリンクローカルアドレス
  - ff02::1 （マルチキャストアドレスでもある）
    - 同一リンク上のすべてのノードが参加している。
  - かならずスコープIDを伴って利用する必要がある。
    - 例： “ping6 –I eth0 ff03::1”, “ssh fe80::dead:beaf%eth0”

各アドレスの適用範囲



特殊なアドレス

- 未指定アドレス
- ループバックアドレス
- IPv4互換アドレス
- IPv4射影アドレス

未指定アドレス

- アドレスが付けられてないことを示します。
- システムの初期化中でまだアドレスがついてないホストがソースアドレスとして使うことがある

0000:0000:0000:0000:0000:0000:0000:0000

⇩

::

ループバックアドレス

- IPv4での127.0.0.1と同じく、ノードがパケットを自分自身に送る場合に用いられる

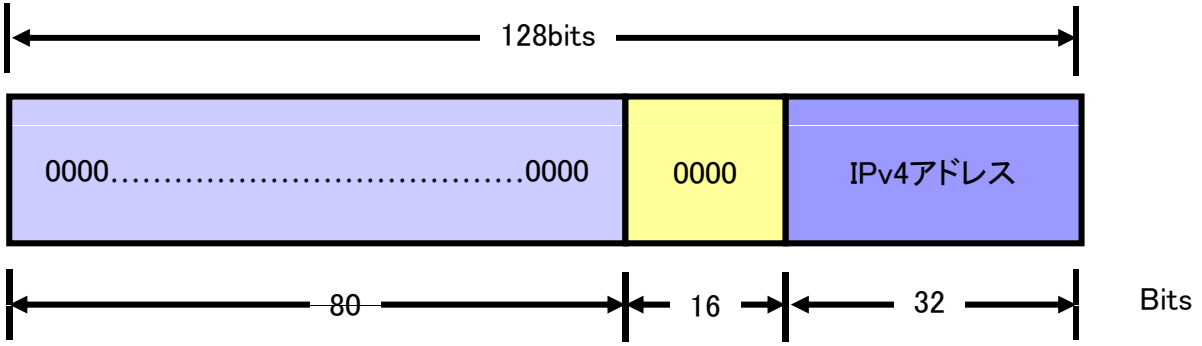
0000:0000:0000:0000:0000:0000:0000:0001

::1

IPv4互換アドレス（廃止へ）

- IPv4ネットワークにしかつながっていないIPv6ノード同士が通信する際に使用されるアドレス
  - 利便性を考え、IPv4アドレス部は10進数表記のままとされている

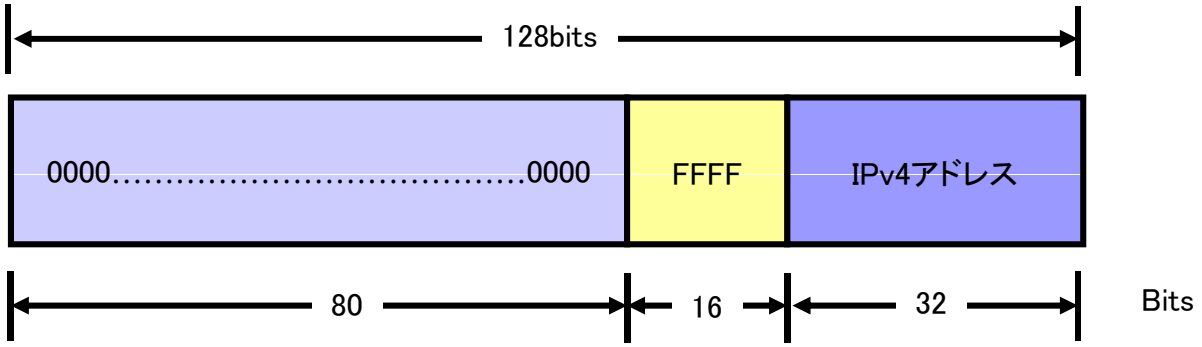
例) “192.168.0.1” => “::192.168.0.1”



IPv4射影アドレス

- IPv6ノードが、IPv4しかサポートしていないノードと通信する際に使用するアドレス
  - 利便性を考え、IPv4アドレス部は10進数表記のままとされている

例）“192.168.0.1” => “::ffff:192.168.0.1”

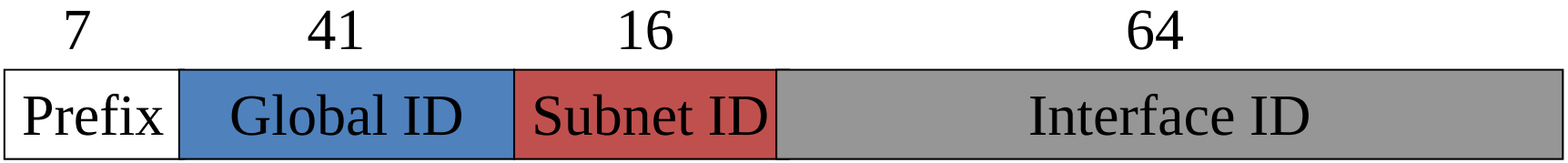


- IPv6の主な特徴
- LinuxでIPv6を扱うための基礎知識
  - IPv6のアドレス表記
  - IPv6のアドレス体系
  - **ULAについて**
  - ヘッダ情報について
  - デュアルスタックとは
  - DNSのIPv6対応とは

Unique Local IPv6 Unicast Addresses

- Globally unique prefix
- Site boundariesで簡単にfilter
- 外部(ISP等)との接続なしに使用できる
- 万が一外部にアドレスを漏らしても、他と重複することがない
- アプリケーションはグローバルアドレスと同様にこのアドレスを取り扱える

Format



- Prefix　　：例(FC00::/7)
- Global ID ：Global Identifier
- Subnet ID ：サイト内部で使用できるsubnet用ID
- Interface ID

Global ID

- MD5 based pseudo-random algorithm使用
- 2つのPrefixを想定
  - FC00::/8　Centrally assigned
  - FD00::/8　Locally assigned
- Centrally assigned
- Locally assigned

Centrally assigned

- 割り当て組織が、アドレスの衝突がないことを保証する
- 定期的な使用料を払うことなく、永続的に使用できる
- 但し1回の割り当てで、10ユーロ(払い戻し不可)の手数料が必要

……というようなことが、検討されています（まだ未決定）

Locally assigned

- Pseudo-random algorithmによってGlobal ID が生成される
- 割り当て組織やISPに申請等の必要無し
  - ULAを自分で作成するには？
    - ULA Generatorが便利

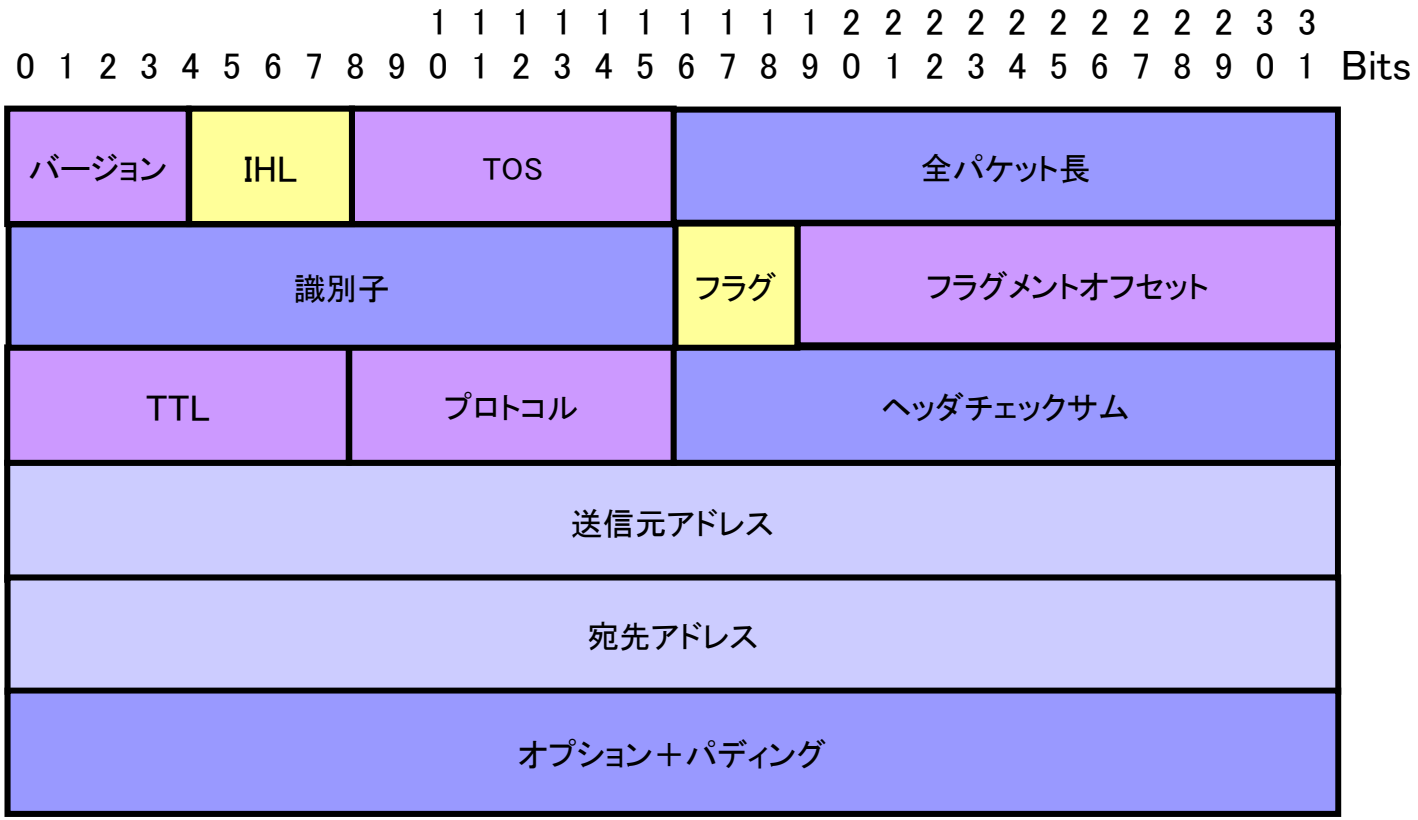
      http://www.kame.net/~suz/gen-ula.html

- IPv6の主な特徴
- LinuxでIPv6を扱うための基礎知識
  - IPv6のアドレス表記
  - IPv6のアドレス体系
  - ULAについて
  - **ヘッダ情報について**
  - デュアルスタックとは
  - DNSのIPv6対応とは

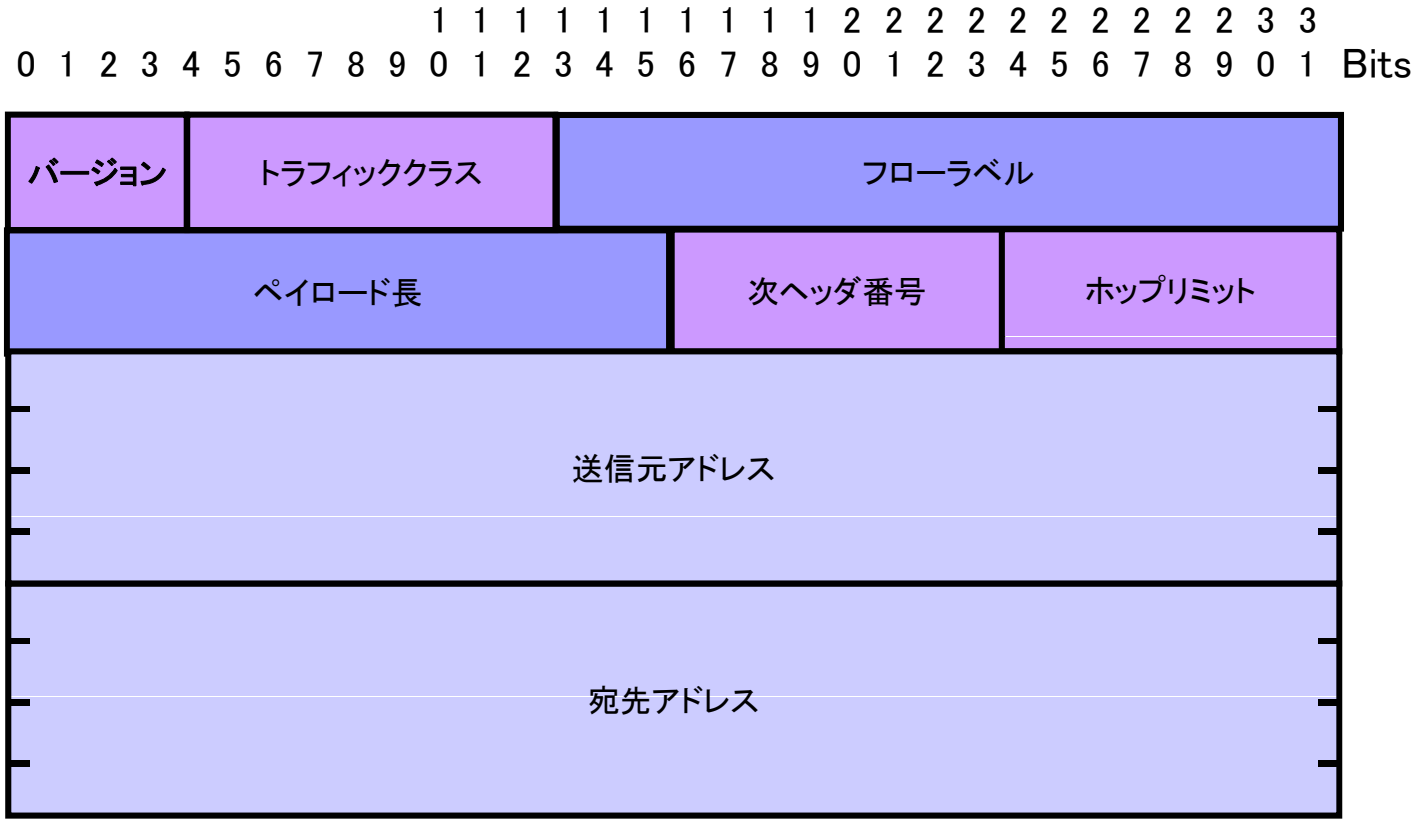
ヘッダ情報について

- ヘッダフォーマットが簡略化
  - 40バイトの固定長に（拡張ヘッダを使用）
- チェックサムの廃止
  - UDPのチェックサムは必須

IPv4ヘッダ



IPv6ヘッダ

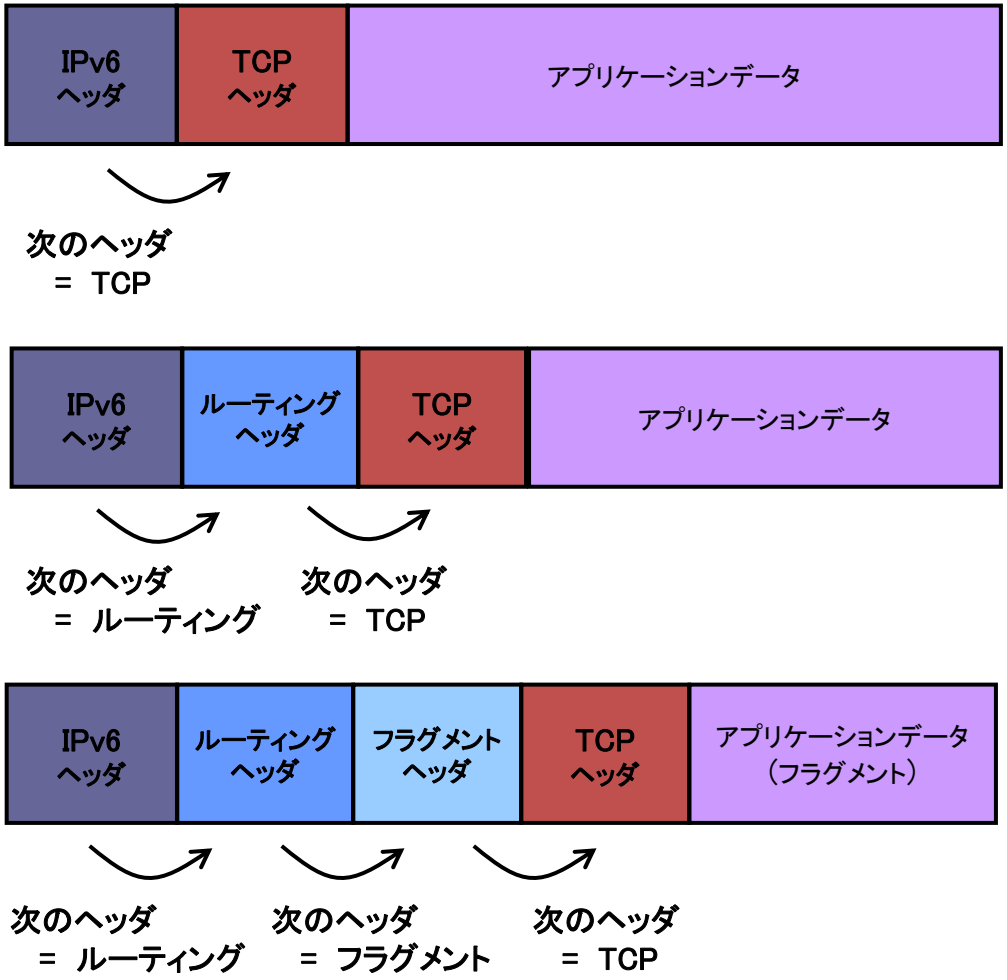


- IPv4に比べ簡略化されている
- オプションが廃止された代わりに拡張ヘッダが用意されている

拡張ヘッダ

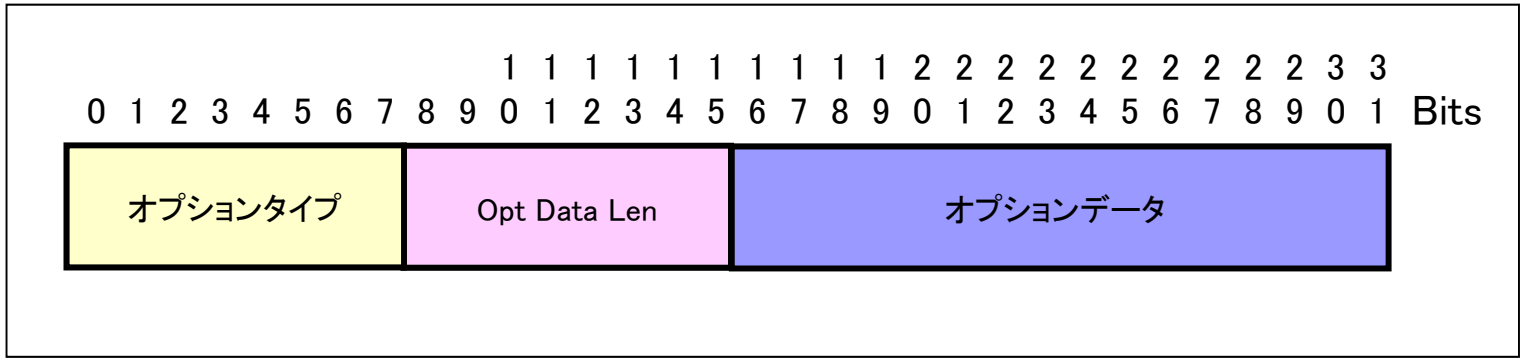
- 中継オプションヘッダ
  - パケットの配送経路上のすべてのノードにおいて、参照される必要がある情報が含まれる
- 終点オプションヘッダ
  - 終点ノードでのみ参照されるオプションが含まれる
- 経路制御ヘッダ
  - 支店から終点へパケットが伝送される際に、訪れるべき中継ノードの一覧が含まれる
- 断片ヘッダ
  - パケットが分割されていることを示す。IPv4とは違い、途中ルータでパケットが分割されることはない。
- 認証ヘッダ
- 暗号ペイロードヘッダ
  - この2つはIPレベルでセキュアな通信を行うためにIPsecにおいて使われる

数珠繋ぎ構造の拡張ヘッダ

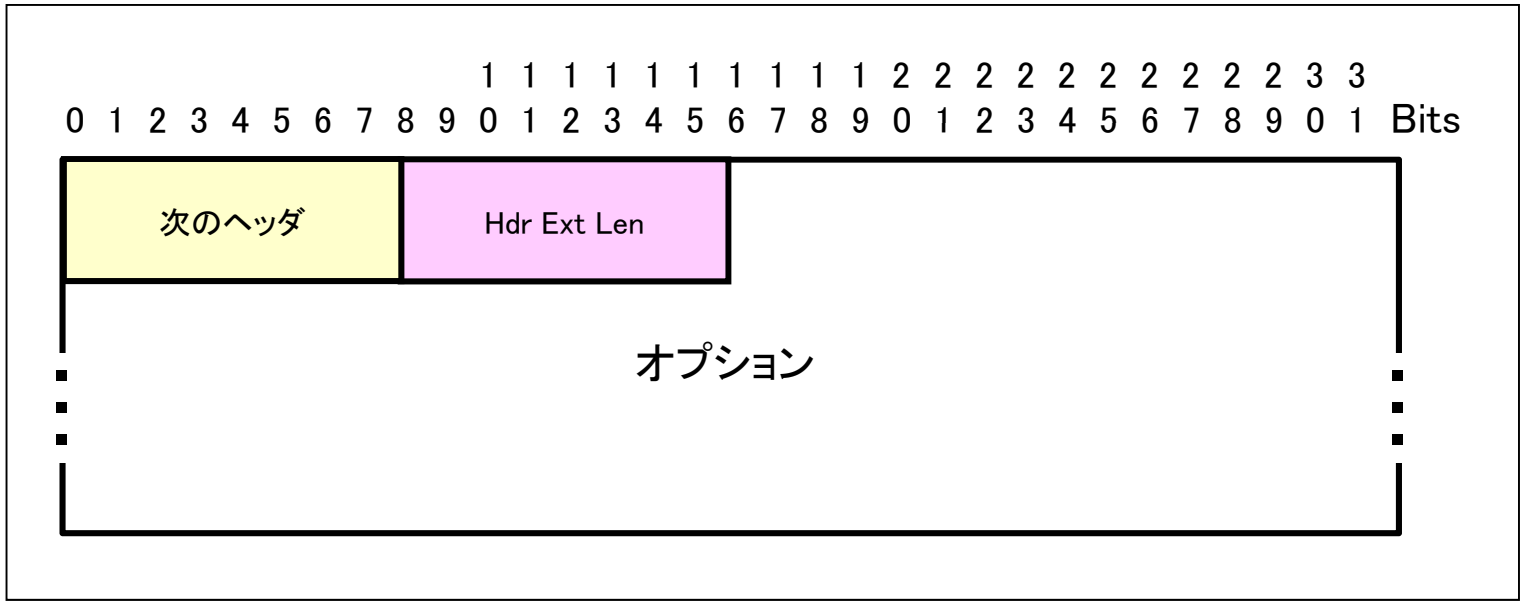


拡張ヘッダ詳細

- 基本型

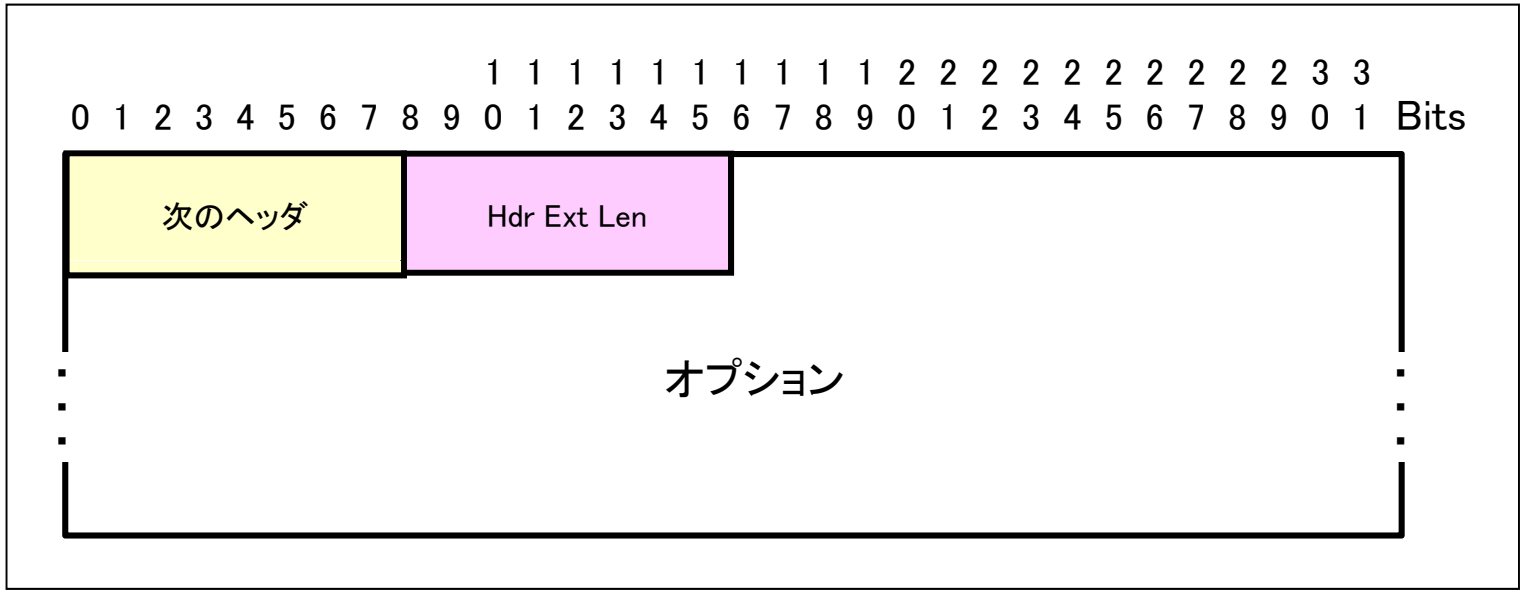


中継点オプションヘッダ



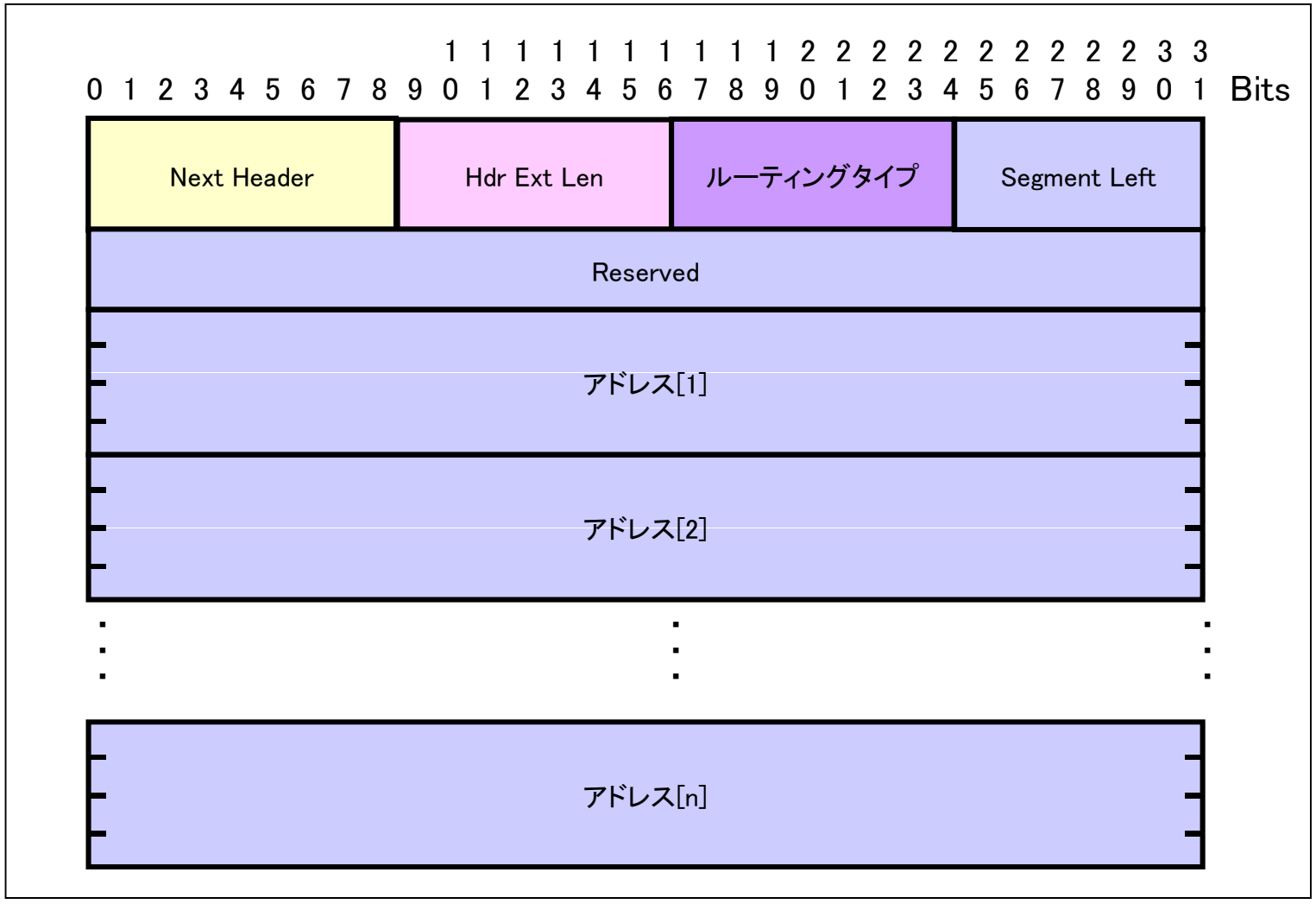
•IPパケットが通過する、すべてのノードにおいて処理されるオプション

終点オプションヘッダ



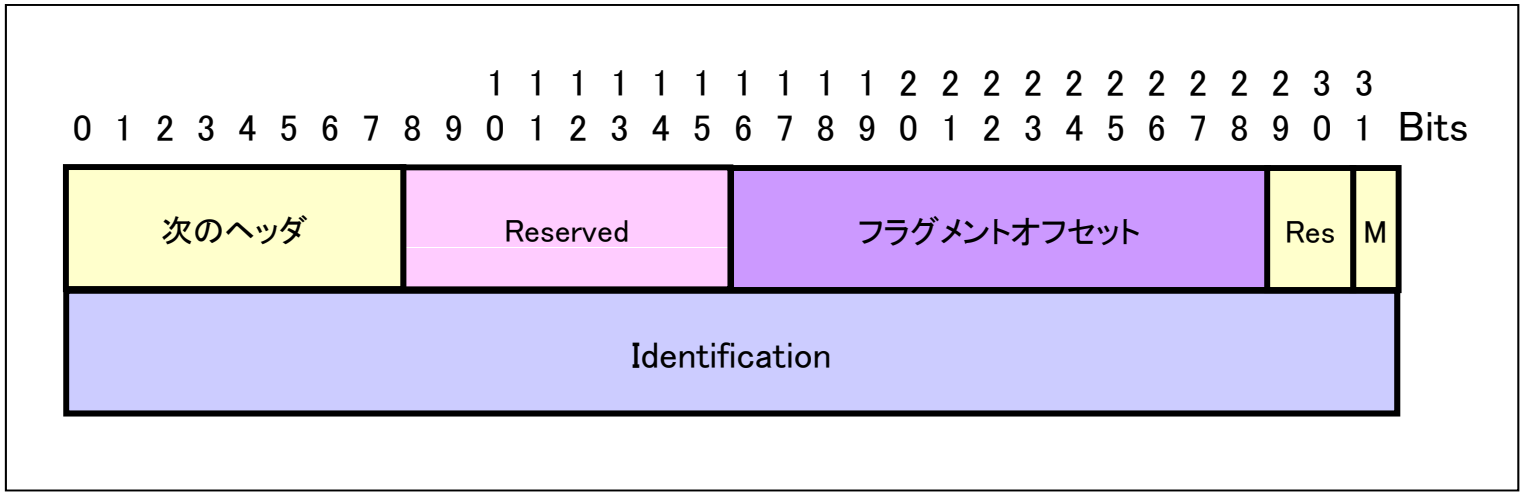
•終点ノードでのみ処理されるオプション

経路制御ヘッダ

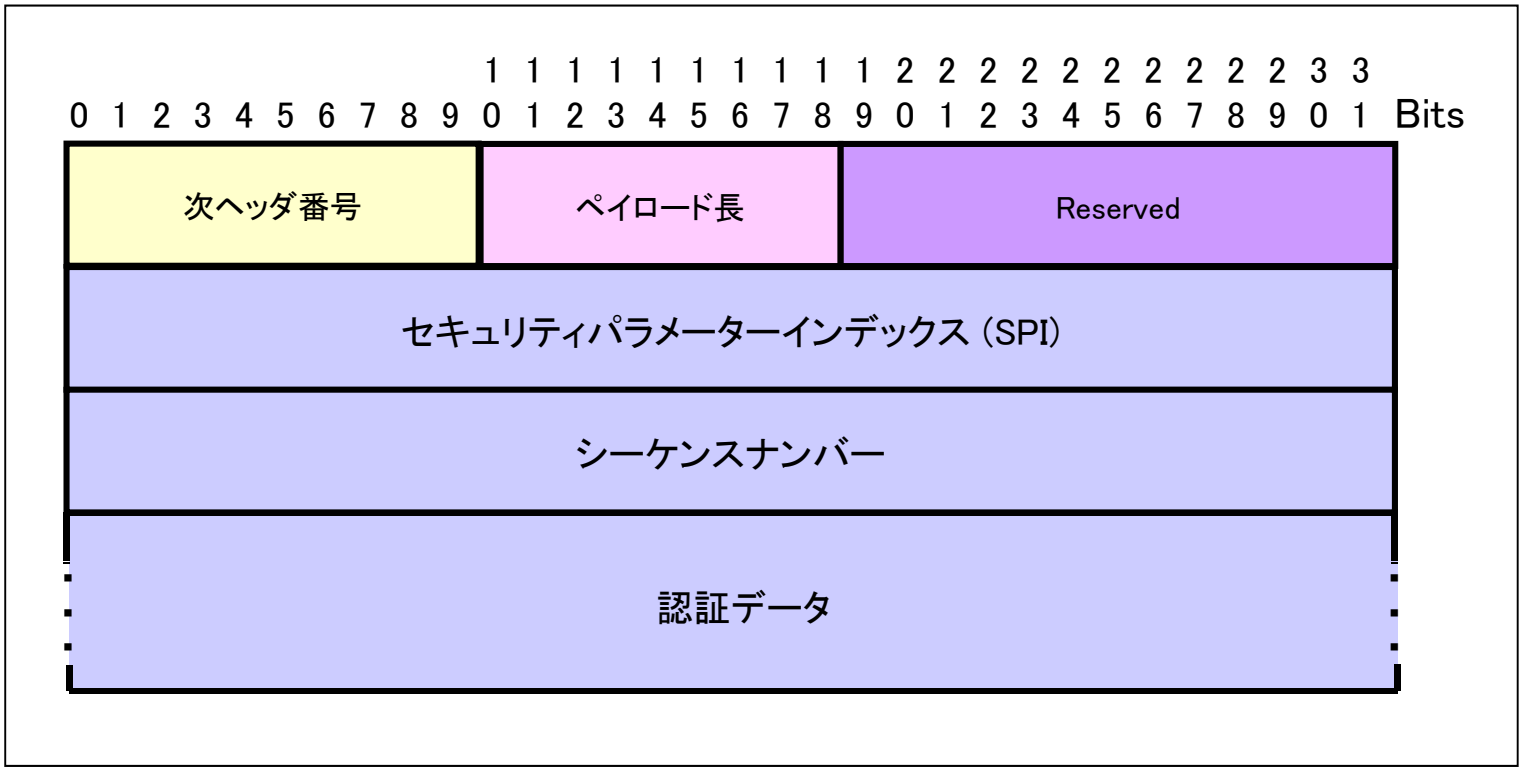


•通過すべきノードを、明示的に指定する場合に使用するオプション

断片ヘッダ

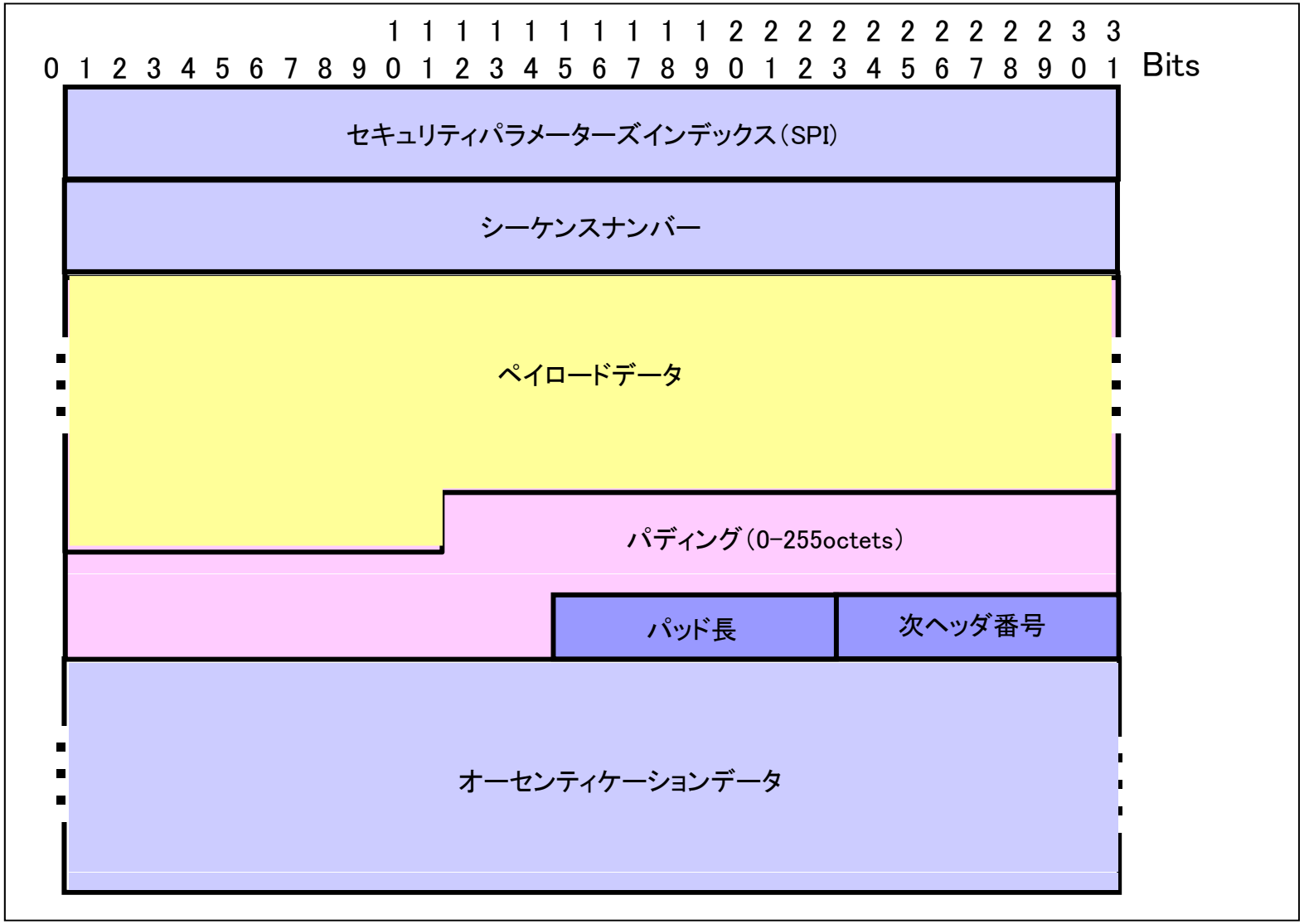


認証ヘッダ



•通信相手を認証するためのオプション（IPsecで使用される）

暗号化ペイロード



•通信相手と、暗号化されたデータ通信を行う際に使用されるオプション（IPsecにおいて使用される）

推奨される拡張ヘッダの順序

- IPv6の主な特徴
- LinuxでIPv6を扱うための基礎知識
  - IPv6のアドレス表記
  - IPv6のアドレス体系
  - ULAについて
  - ヘッダ情報について
  - デュアルスタックとは
  - DNSのIPv6対応とは

デュアルスタックということ

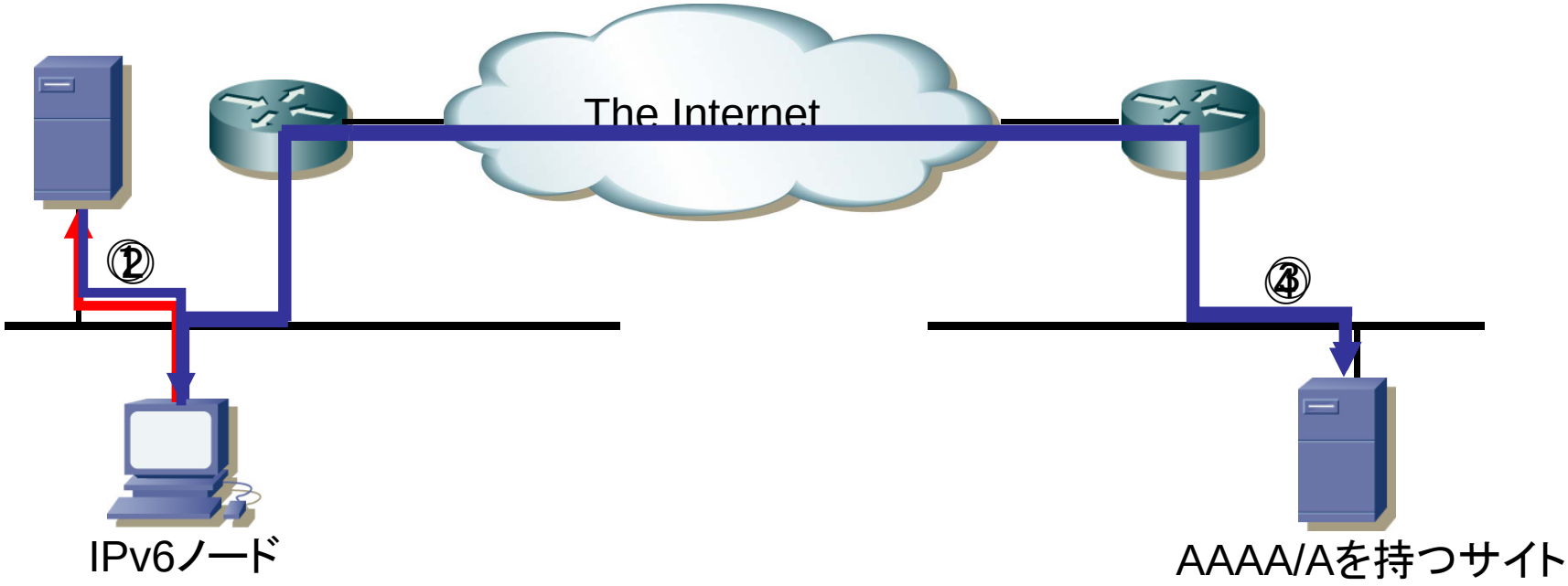
ノードは、IPv6/IPv4の両方のアドレスを持ち、
そのどちらでも通信が可能

```
www.example.com.      IN    A      192.0.2.1
mail.example.com.     IN    A      192.0.2.1
                      IN    AAAA   2001:db8::1
```

- AAAAってなに？
  - DNSのレコードのひとつで、IPv6アドレスが格納される。

TCPコネクション確立

①アドレスを引く
②AAAA RR, A RRが返る
③IPv6で接続
④IPv6で接続できないと、IPv4へフォールバック



プログラムからその挙動を見てみる

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int sock,err;
struct addrinfo hints, *res0, *res;


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


/* getaddrinfo で、AAAAおよびAレコードを取得*/
err = getaddrinfo("www.linux-ipv6.org", "http", &hints, &res0);


if (err) {
   fprintf(stderr, "error : %s", gai_strerror(err));
   freeaddrinfo(res0);
   exit(1);
}


/* getaddrinfoの結果を利用し、接続が成功するまで試行する */
for (res = res0; res; res = res->ai_next) {
       sock = socket (res->ai_family, res->ai_socktype, res->ai_protocol);
       if (sock < 0)
               continue;


       if (connect(sock, res->ai_addr, res->ai_addrlen) < 0) {
               close (sock);
               continue;
       }
       break;
}
freeaddrinfo(res0);
 .
 .
 .
```

- IPv6対応とは、アドレスファミリ独立なネットワークプログラミングをするのと同義

- IPv6の主な特徴
- LinuxでIPv6を扱うための基礎知識
  - IPv6のアドレス表記
  - IPv6のアドレス体系
  - ULAについて
  - ヘッダ情報について
  - デュアルスタックとは
  - **DNSのIPv6対応とは**

DNSのIPv6対応とは

- 保持できるRRの対応
  - AAAAを返す
    - 逆引きは、同じくPTRだが、”in-addr.arpa”に対して、“ipv6.arpa”を利用。ipv6.arpaではなく、ipv6.int が使われていたこともあったが、いまは使われない。
  - bind8.4 以上またはbind9で対応
- IPv6 transportへの対応
  - クエリの送受信をIPv6プロトコルを用いて行う。
  - 自身のFQDNに、AAAAが登録される。

※AAAAは、IPv4ネットワークを通じて、返信することも可能なので、DNSサーバがIPv6に対応しました！というのは上記のどちらの話なのか（もしくは両方なのか）、に留意する必要がある。

DNSクエリに関するOSの対応

- DNSリゾルバの改良（IPv4通信の品質を保つため）
  - Aレコード解決を優先する（FreeBSD、Windows Vista）
    - IPv6が優勢になった時に問題になる可能性あり
  - Aレコード解決時にNXDOMAINならAAAAレコード解決をしない（Windows Vista）
  - Aレコードのレスポンス時間によりAAAAレコードの処理待ち時間を決定（FreeBSD、Windows Vista）
    - AAAAレコードがない場合のタイムアウト時間を小さくするため
- AAAAレコード解決の抑制
  - グローバルIPv6アドレスが付与されない限りAAAAクエリによる名前解決は実施しない（Windows Vista）

DNSクエリの順序

- クエリ順序はOSで異なる
  - FreeBSD-5.5R
    - IPv4の名前解決とIPv6の名前解決を交互に繰り返し名前解決ができた時点で終了
  - Windows XP SP2
    - まずIPv6の名前解決を全て実施し次にIPv4の名前解決を全て実施
  - Windows Vista
    - まずIPv4の名前解決を全て実施し次にNXDOMAINが返されたもの以外の全てに関してIPv6の名前解決を実施

LinuxにおけるIPv6設定およびその確認

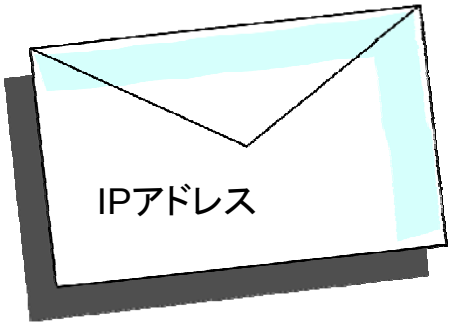
- ネットワーク設定の基礎
  - ルーティングの基礎の基礎
  - IPv4/IPv6アドレスの設定・確認
  - IPv4/IPv6経路の確認
  - Bondingについて
  - 各種設定ファイルについて
  - トンネリングについて
- 稼働状況の確認方法

LinuxにおけるIPv6設定およびその確認

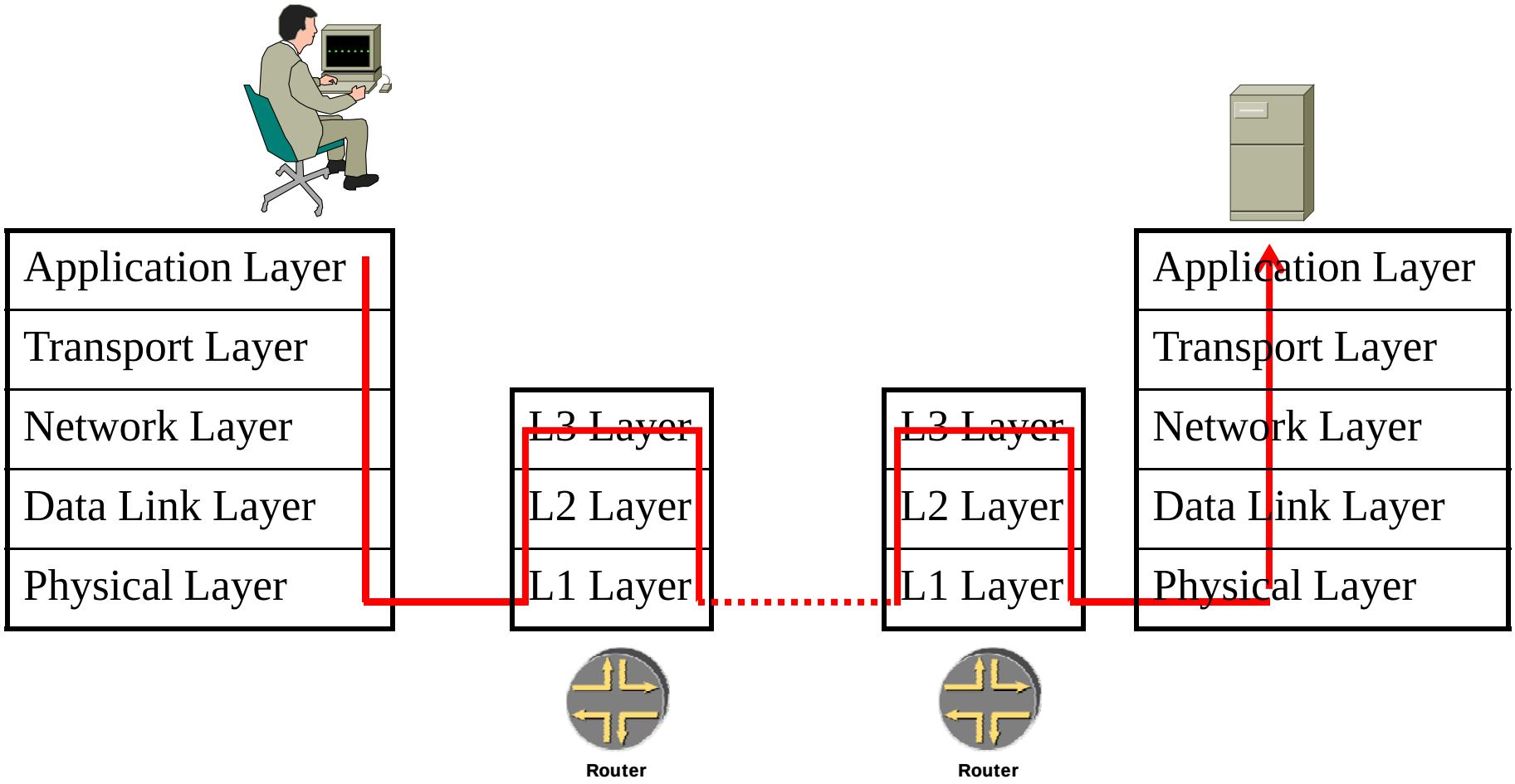
- ネットワーク設定の基礎
  - **ルーティングの基礎の基礎**
  - IPv4/IPv6アドレスの設定・確認
  - IPv4/IPv6経路の確認
  - Bondingについて
  - 各種設定ファイルについて
  - トンネリングについて
- 稼働状況の確認方法

ネットワーク設定の基礎

- インターネットでは、データはIPパケット（小包）に包まれて目的地までバケツリレーで配送されていく
  - 宛先、差出人に書く住所が、IPアドレス



パケットが転送される仕組み



クラスフルな世界

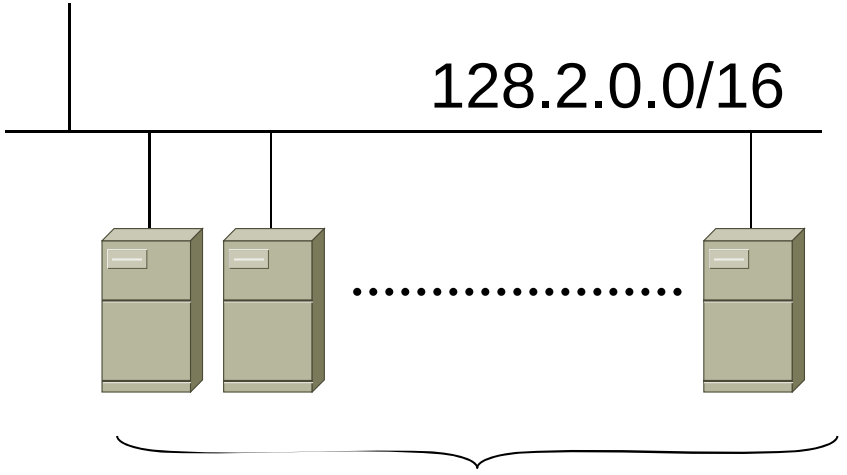
従来はIPアドレスはクラスに分けられていた

	IP Address Range	netmask	length
クラスA	1.0.0.0-126.255.255.255	255.0.0.0	/8
クラスB	128.0.0.0-191.255.255.255.255	255.255.0.0	/16
クラスC	192.0.0.0-223.255.255.255	255.255.255.0	/24

例：クラスB

128.2 . 0. 0

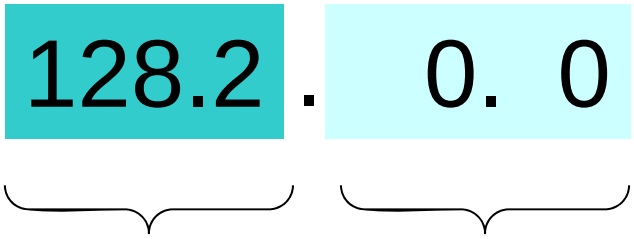
ネットワークアドレス　　ホストアドレス



2^16-2 =65534台分のアドレス

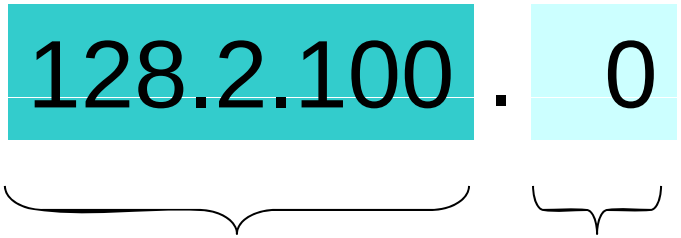
サブネットの登場

例：クラスB



length	Sub netmask	ホスト数
/16	255.255.0.0	65534
/17	255.255.128.0	32766
/18	255.255.192.0	16382
/19	255.255.224.0	8190
/20	255.255.240.0	4094
/21	255.255.248.0	2046
/22	255.255.252.0	1022
/23	255.255.254.0	510
/24	255.255.255.0	254
/25	255.255.255.128	126
/26	255.255.255.192	62
/27	255.255.255.224	30
/28	255.255.255.240	14
/29	255.255.255.248	6
/30	255.255.255.252	2

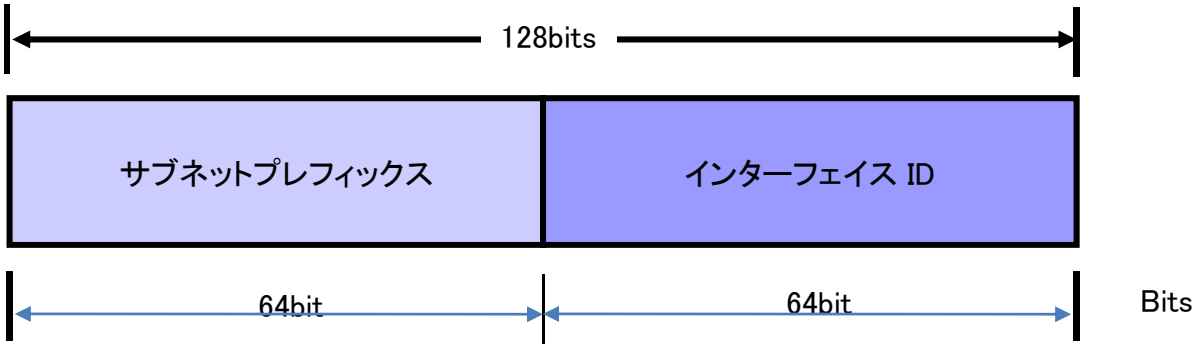
例：サブネットの仕組みを使ってネットワーク部分を拡張



2^8-2=254台分のホストアドレスが利用可能

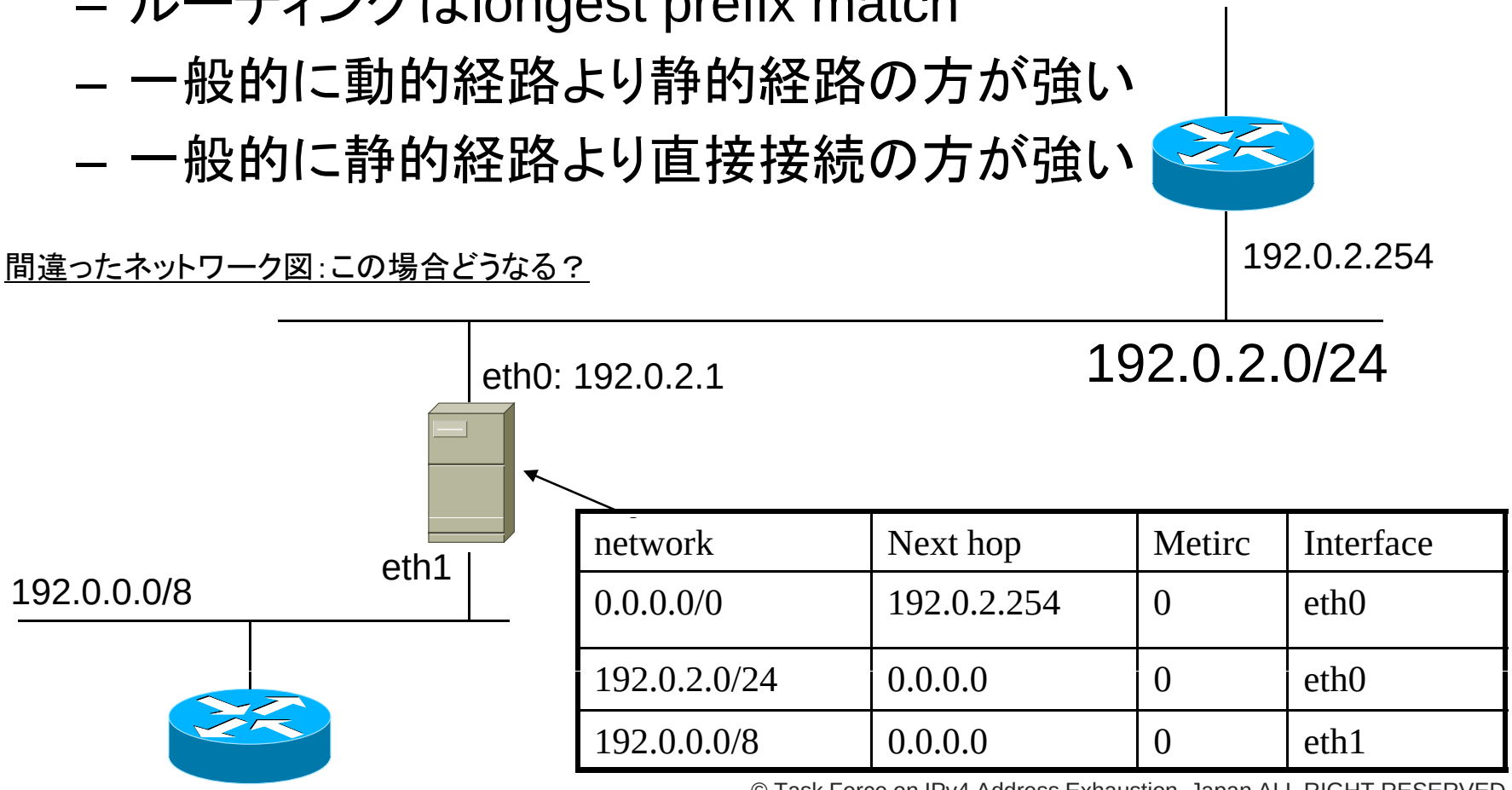
IPv6では？

- もちろんクラスレス
- Netmaskに関しては、サーバとして利用する限りは、/64固定と考えてよい。



ルーティングの基礎の基礎

- ここで取り扱うのは静的経路
  - ルーティングはlongest prefix match
  - 一般的に動的経路より静的経路の方が強い
  - 一般的に静的経路より直接接続の方が強い



network	Next hop	Metirc	Interface
0.0.0.0/0	192.0.2.254	0	eth0
192.0.2.0/24	0.0.0.0	0	eth0
192.0.0.0/8	0.0.0.0	0	eth1

- ネットワーク設定の基礎
  - ルーティングの基礎の基礎
  - **IPv4/IPv6アドレスの設定・確認**
  - IPv4/IPv6経路の確認
  - Bondingについて
  - 各種設定ファイルについて
  - トンネリングについて
- 稼働状況の確認方法

IPアドレスの設定

- IPアドレスを設定するには、以下のコマンドが知られている。
  - ifconfig
    - 書式： ifconfig *interface [aftype] options* | *address …*
  - ip
    - 書式：ip *[OPTIONS] OBJECT { COMMAND* | *help }*

IPv4アドレスの設定 – ifconfig -

- IPv4アドレスの設定例

```
IPv4 アドレスの設定
# ifconfig eth0 192.0.2.1 netmask 255.255.255.0

設定確認
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:00:XX:XX:XX:XX
          inet addr:192.0.2.1  Bcast:192.0.2.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:169

インタフェースを上げる
# ifconfig eth0 up

インタフェースを落とす
# ifconfig eth0 down
```

IPv6アドレスの設定 – ifconfig -

- IPv6アドレスの設定例

```
IPv6 アドレスの設定(事前にinterfaceを upしておく必要がある)
# ifconfig eth0 add 2001:db8::80/64

IPv6アドレスの削除
# ifconfig eth0 del 2001:db8::80/64

設定確認
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:XX:XX:XX:XX:XX
          inet addr:192.0.2.1  Bcast:192.0.2.255  Mask:255.255.255.0
          inet6 addr: 2001:db8::80/64 Scope:Global
          inet6 addr: fe80::2d0:xxxx:xxxx:xxxx/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:169

インタフェースを落とす(IPv6の場合は、落とすとグローバルアドレスが消える)
# ifconfig eth0 down
```

IPv4/IPv6アドレスの設定 – ipコマンド –

```
IPv4/IPv6アドレスの設定
# ip addr add 192.0.2.1/24 dev eth0
# ip addr add 2001:db8::80/64 dev eth0

設定確認
# ip addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global eth0
    inet6 fe80::202:xxx:xxx:xxx:xxx/64 scope link
       valid_lft forever preferred_lft forever
    inet6 2001:db8::80/64 scope global
       valid_lft forever preferred_lft forever

インタフェースを上げる
# ip link set eth0 up

インタフェースを落とす
# ip link set eth0 down
```

IPv4/IPv6アドレスの設定 / lifetime

```
# ip addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global eth0
    inet6 fe80::202:xxx:xxx:xxx:xxx/64 scope link
       valid_lft forever preferred_lft forever
    inet6 2001:db8::80/64 scope global
       valid_lft forever preferred_lft forever
```

IPv4/IPv6　L2アドレスの確認

IPv4の場合：ARPコマンドの代わりに

```
# ip -4 neigh show
192.0.2.1 dev eth0 lladdr 00:00:XX:00:XX:XX REACHABLE
```

IPv6の場合：

```
# ip -6 neigh show
2001:db8:0:1::dead:beaf dev eth0 lladdr 00:00:XX:XX:XX:XX router REACHABLE
fe80::XXX:XXXX:XXXX:XXXX dev eth0 lladdr 00:00:XX:XX:XX:XX router REACHABLE
```

routeコマンドによる経路の設定および確認

Default経路の設定

```
# route add –A inet default gw 192.0.2.254 dev eth0
# route add –A inet6 default gw fe80:x:x:x:x:x dev eth0
```

ネットワーク別経路の設定

```
# route add –net 192.0.2.0 netmask 255.255.255.0 gw 192.168.0.1 dev eth0
# route add –A inet6 2001:db8::/64 gw fe80::2d0:b7ff:fea0:beea dev eth0
```

経路の確認

```
# route -n –A inet6
# route –n –A inet
```

ipコマンドによる経路の設定および確認

```
Default経路の設定
# ip route add default via 10.0.0.1 dev eth0
# ip route add default via fe80::202:b3ff:fe32:faa2 dev eth0

ネットワーク別経路の設定
# ip route add 192.0.2.0/24 via 10.0.0.1 dev eth0
# ip route add 2001:db8::/48 via fe80::202:b3ff:fe32:faa2 dev eth0

経路の確認
# ip -4 route show
# ip -6 route show
```

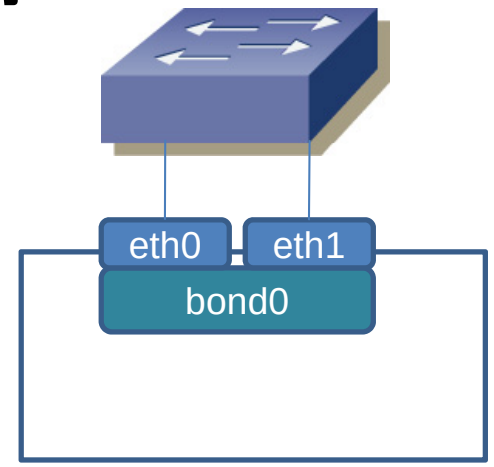
- ネットワーク設定の基礎
  - ルーティングの基礎の基礎
  - IPv4/IPv6アドレスの設定・確認
  - IPv4/IPv6経路の確認
  - **Bondingについて**
  - 各種設定ファイルについて
  - トンネリングについて
- 稼働状況の確認方法

bonding

- ネットワークの冗長化などで利用される。
  - IPv6でも、特に動作に問題はない。
- Active-backupで冗長化した例

/etc/modprobe.d/bonding

```
alias bond0 bondingoptions
bond0 miimon=100 mode=1
```



確認

```
$ cat /proc/net/bonding/bond0
```

- ネットワーク設定の基礎
  - ルーティングの基礎の基礎
  - IPv4/IPv6アドレスの設定・確認
  - IPv4/IPv6経路の確認
  - Bondingについて
  - **各種設定ファイルについて**
  - トンネリングについて
- 稼働状況の確認方法

設定ファイルについて（共通）

名前解決に関するファイル

- /etc
  - hosts - IPアドレスとホスト名の対応表
  - resolv.conf – リゾルバ設定ファイル
  - host.conf – リゾルバ設定ファイル

```
127.0.0.1       hoge.example.jp   localhost hoge

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

```
search example.jp
nameserver 192.0.2.254
```

```
order hosts,bind
multi on
```

設定ファイルについて（共通）

サービス名などの解決に関するファイル

- /etc
  - services : ネットワークサービスリスト
  - protocols : プロトコル定義ファイル

```
tcpmux          1/tcp                                   # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp
msp
chargen
chargen
ftp-data
ftp
fsp
ssh
ssh
～省略～
```

```
ip      0       IP              # internet protocol, pseudo protocol number
#hopopt 0       HOPOPT          # IPv6 Hop-by-Hop Option [RFC1883]
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
```

設定ファイルについて

CentOS

- /etc
  - sysconfig
    - network : ネットワークオプション設定など
    - network-scripts
      - ifcfg-{dev} : ネットワークインターフェース設定

Debian

- /etc
  - network
    - options : ネットワークオプション設定
    - interfaces ：ネットワークインターフェース設定

※1最近のDebian/Ubuntuでは、network-managerパッケージがインストールされていると、設定が反映されない。

CentOSのネットワーク設定

/etc/sysconfig/network

```
HOSTNAME=hoge.example.jp

# IPv4
NETWORKING=yes
GATEWAY=192.0.2.254

#IPv6
NETWORKING_IPV6=yes
IPV6_DEFAULTGW=fe80::xxxx%eth0
#IPV6_AUTOCONF=no
#IPV6FORWARDING=no
```

HOSTNAME
FQDNを記述
NETWORKING
yes: rcスクリプトである network を実行
GATEWAY
IPv4ネットワークのdefault gatewayを指定
NETWORKING_IPV6
yes: IPv6ネットワークを有効化
IPV6_DEFAULTGW
IPv6ネットワークのdefault gatewayを指定

CentOSのネットワーク設定

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETWORK=192.0.2.0
NETMASK=255.255.255.0
IPADDR=192.0.2.1
IPV6INIT=yes
#IPV6_AUTOCONF=no
IPV6ADDR=2001:db8::80/64
ETHTOOL_OPTS="autoneg off speed 100 duplex full"
```

DEVICE
物理インタフェース名
BOOTPROTO
none ： 使用しない
bootp: BOOTPを使用
dhcp ： DHCPを使用
ONBOOT
yes ： 起動時に有効
no ： 起動時には無効

IPADDR
yes ： 起動時に有効
no ： 起動時には無効
NETWORK
ネットワークアドレスを指定
NETMASK
ネットマスクを指定

IPV6INIT
yes: IPv6の初期化を有効
No ： IPv6の初期化を無効
IPV6ADDR
IPv6アドレス/ネットマスク
ETHTOOL_OPTS
ethtoolへのオプションを指定

Debianのネットワーク設定(廃止)

/etc/network/options

```
ip_forward=no
spoofprotect=yes
syncookies=no
```

ip_forward
yes: LinuxをRouterとして利用する際に設定

spoofprotect
yes: Reverse Pathが検証される。Stub networkにつながっている場合は設定を推奨

syncookies
yes: TCP syn flooding attack対策のため、cookieを発行するようになる。

/etc/network/optionsは廃止になったため、これらは、/etc/sysctl.conf に記述

/etc/sysctl.conf

```
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.tcp_syncookies = 1
```

/etc/network/interface

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.0.2.1
        netmask 255.255.255.0
        gateway 192.0.2.254
        # dns-serarch example.jp
        # dns-nameservers 192.0.2.254
        up ethtool -s eth0 autoneg off speed 100 duplex full

iface eth0 inet6 static
        address 2001:db8::80
        netmask 64
        gateway fe80::xxx%eth0
```

pre-up
インタフェースをupさせる前に実行するコマンド

up
インタフェースをupしたときに実行するコマンド

post-up
インタフェースをupした後に実行するコマンド

pre-down
インタフェースをdownさせる前に実行するコマンド

down
インタフェースをdownした時に実行するコマンド

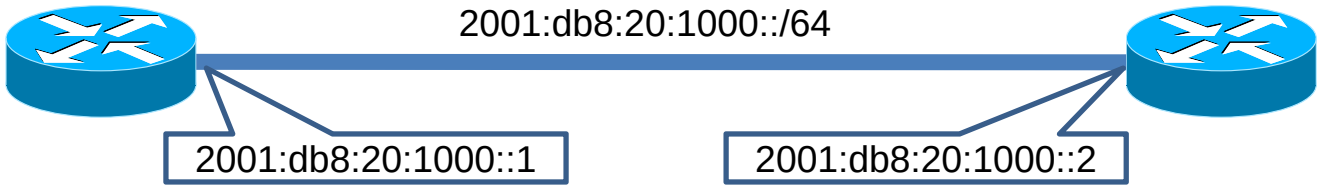
post-down
インタフェースをdownした後に実行するコマンド

Also See man interfaces

- ネットワーク設定の基礎
  - ルーティングの基礎の基礎
  - IPv4/IPv6アドレスの設定・確認
  - IPv4/IPv6経路の確認
  - Bondingについて
  - 各種設定ファイルについて
  - トンネリングについて
- 稼働状況の確認方法

トンネリングについて

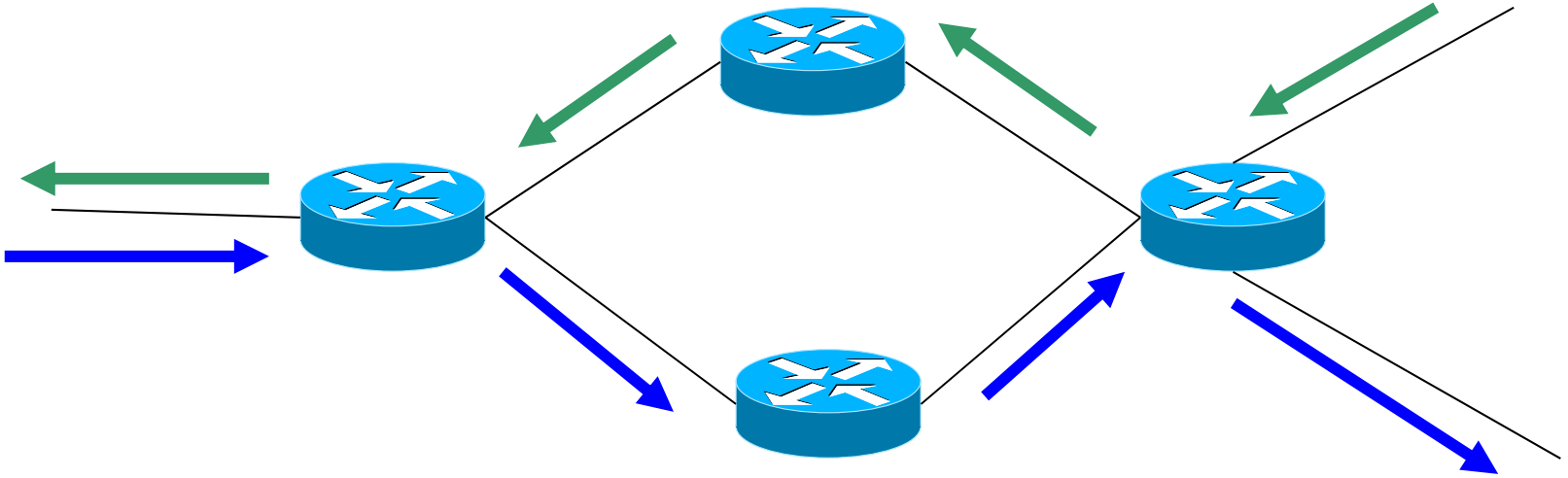
- トンネルはpoint-to-point接続
  - 自分宛てじゃなければ、相手側のアドレスだと、普通は判断する。
  - ネットワークには、/64を割り振ることが基本
    - ということは？下のネットワークに、2001:db8:20:1000::3のパケットが投げられると……



- ネットワーク設定の基礎
  - ルーティングの基礎の基礎
  - IPv4/IPv6アドレスの設定・確認
  - IPv4/IPv6経路の確認
  - Bondingについて
  - 各種設定ファイルについて
  - トンネリングについて
- **稼働状況の確認方法**

接続確認

- 簡単な接続確認方法
  - pingを用いた疎通確認
  - tracerouteを用いた疎通確認
    - tracerouteでわかるのは行きの経路だけ。



行きと帰りで経路が違うことも……

Ping(ICMP)による疎通確認

- Default gatewayに対してpingを打ってみる

```
IPv4経路の疎通性確認
$ ping -c 10 192.0.2.254
PING 192.0.2.254 (192.0.2.254) 56(84) bytes of data.
64 bytes from 192.0.2.254: icmp_seq=1 ttl=255 time=2.09 ms
64 bytes from 192.0.2.254: icmp_seq=2 ttl=255 time=2.04 ms
64 bytes from 192.0.2.254: icmp_seq=3 ttl=255 time=4.30 ms
64 bytes from 192.0.2.254: icmp_seq=4 ttl=255 time=2.00 ms
64 bytes from 192.0.2.254: icmp_seq=5 ttl=255 time=2.01 ms
64 bytes from 192.0.2.254: icmp_seq=6 ttl=255 time=2.02 ms
64 bytes from 192.0.2.254: icmp_seq=7 ttl=255 time=2.03 ms
64 bytes from 192.0.2.254: icmp_seq=8 ttl=255 time=2.62 ms
64 bytes from 192.0.2.254: icmp_seq=9 ttl=255 time=3.84 ms
64 bytes from 192.0.2.254: icmp_seq=10 ttl=255 time=4.77 ms

--- 192.0.2.254 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9036ms
rtt min/avg/max/mdev = 2.009/2.776/4.774/1.038 ms
```

Ping(ICMP)による疎通確認

- Default gatewayに対してpingを打ってみる

```
IPｖ6経路の疎通性を確認
$ ping6 -c 10 fe80::2000:1 -I eth0
PING fe80::2000:1(fe80::2000:1) from fe80::2d0:b7ff:fea0:beea
eth1: 56 data bytes
64 bytes from fe80::2000:1: icmp_seq=1 ttl=64 time=2.38 ms
64 bytes from fe80::2000:1: icmp_seq=2 ttl=64 time=7.71 ms
64 bytes from fe80::2000:1: icmp_seq=3 ttl=64 time=7.47 ms
64 bytes from fe80::2000:1: icmp_seq=4 ttl=64 time=2.41 ms
64 bytes from fe80::2000:1: icmp_seq=5 ttl=64 time=2.39 ms
64 bytes from fe80::2000:1: icmp_seq=6 ttl=64 time=3.91 ms
64 bytes from fe80::2000:1: icmp_seq=7 ttl=64 time=5.00 ms
64 bytes from fe80::2000:1: icmp_seq=8 ttl=64 time=2.29 ms
64 bytes from fe80::2000:1: icmp_seq=9 ttl=64 time=2.30 ms
64 bytes from fe80::2000:1: icmp_seq=10 ttl=64 time=2.32 ms

--- fe80::2000:1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9036ms
rtt min/avg/max/mdev = 2.292/3.822/7.711/2.069 ms
```

速度がでない？

- そんなときには、Ethernet Cardがちゃんと Full Duplex(全二重通信)になっているかを確認

```
現在の設定確認
# ethtool eth0

Full-duplex固定に設定
# ethtool -s eth0 autoneg off speed 100 duplex full
```

```
現在の設定確認
# mii-tool eth0

Full-duplex固定に設定
# mii-tool -F 100baseTx-FD eth1
```

sysctlによる制御

- IPv6の自動設定については、インターフェースがUPした時に実行されるので、不要な場合は事前に無効化しておく

```
# sysctl –w net.ipv6.conf.eth0.accept_ra=0
```

- TCP Syn flooding attack対策

```
# sysctl –w net.ipv4.tcp_syncookies=1
```

- Broadcast宛のICMPを無視

```
# sysctl –w net.ipv4.icmp_echo_ignore_broadcasts
```

稼動状況確認

サーバの稼動状況確認

- どんなプログラムが起動しているのか？
  - psコマンド

```
# ps aux | less
```

- どんなポートをListenしているのか？
  - netstat
  - fuser

サーバの稼働状況確認(netstat)

- Netstatを用いて、サーバのListenポートを表示させる

オプション例

-l, --listening
  接続待ち状態にあるソケットのみを表示する
-p, --program
  各ソケットが属しているプログラムのPIDと名前が表示される
-n, --numeric
  ホスト、ポート、ユーザなどの名前を解決せずに、数字のアドレスで表示する。
-t,--tcp
  tcpに関する情報を表示
-u,--udp
  udpに関する情報を表示

サーバの稼働状況確認(netstat)

実行例

```
# netstat -ltupn
Proto Recv-Q Send-Q Local Address       Foreign Address     State       PID/Program name
tcp        0      0 127.0.0.1:993       0.0.0.0:*           LISTEN      3785/famd
tcp        0      0 127.0.0.1:111       0.0.0.0:*           LISTEN      3175/portmap
tcp        0      0 192.0.2.1:53        0.0.0.0:*           LISTEN      3380/named
tcp        0      0 127.0.0.1:53        0.0.0.0:*           LISTEN      3380/named
tcp        0      0 0.0.0.0:5432        0.0.0.0:*           LISTEN      3619/postmaster
tcp        0      0 0.0.0.0:25          0.0.0.0:*           LISTEN      3596/master
tcp        0      0 127.0.0.1:953       0.0.0.0:*           LISTEN      3380/named
tcp6       0      0 :::80               :::*                LISTEN      11254/apache2
tcp6       0      0 :::53               :::*                LISTEN      3380/named
tcp6       0      0 :::22               :::*                LISTEN      3659/sshd
tcp6       0      0 :::5432             :::*                LISTEN      3619/postmaster
udp        0      0 0.0.0.0:32768       0.0.0.0:*                       3380/named
udp        0      0 127.0.0.1:161       0.0.0.0:*                       3653/snmpd
udp        0      0 192.0.2.1:53        0.0.0.0:*                       3380/named
udp        0      0 127.0.0.1:53        0.0.0.0:*                       3380/named
udp        0      0 127.0.0.1:111       0.0.0.0:*                       3175/portmap
udp6       0      0 :::32769            :::*                            3380/named
udp6       0      0 :::53               :::*                            3380/named
```

※”-A”でアドレスファミリーを指定することも可能(inet or inet6)

netstatによる注意点

- netstat で表示される文字幅の制限から、IPv6アドレスが一定長を超えると、表示されなくなる（丸められる）
- Debianパッケージの net-tool_1.60-22以降であれば、”-W/--wide”オプションで、対応可能

ssで稼働状況確認

- ほぼ引数は netstatと同様だが、IPv6アドレスが、丸められることはない

```
$ ss -ltun
Netid  Recv-Q Send-Q          Local Address:Port            Peer Address:Port
tcp    0      50                 127.0.0.1:3306                 *:*
tcp    0      128                      *:111                    *:*
tcp    0      128                     :::80                    :::*
tcp    0      5                  127.0.0.1:33843                *:*
tcp    0      3                 192.0.2.136:53                  *:*
tcp    0      3                  127.0.0.1:53                   *:*
tcp    0      3                       :::53                    :::*
tcp    0      128                127.0.0.1:631                  *:*
tcp    0      128                127.0.0.1:5432                 *:*
tcp    0      128                     ::1:953                   :::*
tcp    0      128                127.0.0.1:953                  *:*
tcp    0      100                     :::25                    :::*
tcp    0      100                      *:25                     *:*
tcp    0      3                        *:1723                   *:*
```

サーバの稼働状況確認(fuser)

- fuserを用いてサーバのListenポートから Listenしているプロセスを特定する。

```
# fuser -vn tcp 80

                     USER        PID ACCESS COMMAND
80/tcp:              root       4699 F.... apache
                     www-data   4706 F.... apache
                     www-data   4707 F.... apache
                     www-data   4708 F.... apache
                     www-data   4709 F.... apache
                     www-data   4710 F.... apache
                     www-data   8407 F.... apache
                     www-data   8408 F.... apache
                     www-data   8409 F.... apache
```

基礎サービスの設定について

- DNS(bind9)
- SMTP(postfix)
- POPサーバ（dovecot)
- Apache
- NTP

- **DNS(bind9)**
- SMTP(postfix)
- POPサーバ（dovecot)
- Apache
- NTP

DNS

- BIND9のIPv6 transportを有効にするには、listen-on-v6を指定する。

```
options {
        directory "/var/named";
        listen-on-v6 { any; };
…
..
.
```

DNS

- ACL設定例
  - IPv4と同様に、生アドレスが記載可能

```
acl “slaves” {
        192.0.2.1;      // slave server
        2001:db8::53;   // slave server
        127.0.0.1;      // for debug
        ::1;            // for debug
};
```

DNS

- AAAA　RR登録例

```
;; Server
;;   example.jp
www     IN      A       192.0.2.1
www     IN      AAAA    2001:db8::1
```

- AAAA　RR確認

```
$ dig www.example.jp AAAA
```

DNS

- IPv6の逆引き設定

```
;; IPv6 PTR for 2001:db8::/64
zone “0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ipv6.arpa” {
        type master;
        file “2001.0db8.0000.000.reverse”;
        allow-transfer { slaves; };
        allow-query { any };
}
```

```
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   IN   PTR   www.example.jp.
```

- IPv6の逆引き設定確認

```
$ dig –x 2001:db8::1
```

DNS:リゾルバでの指定

- IPv4と同様、/etc/resolv.conに直接、IPv6アドレスを記述する。

```
search example.jp
nameserver 192.0.2.254
nameserver 2001:db8::53
```

- DNS(bind9)
- **SMTP(postfix)**
- POPサーバ（dovecot)
- Apache
- NTP

postfixをIPv6対応に

- /etc/postfix/main.cf

```
# inet_protocols = ipv4
# inet_protocols = ipv4, ipv6 # allと等価です
# inet_protocols = ipv6
inet_protocols = all
```

これだけ……

Listenするアドレスを制限

- /etc/postfix/main.cf

```
# inet_interfaces = alll
# inet_interfaces = loopback-only
inet_interfaces = 127.0.0.1, [::1], [2001:db8::25]
```

送信時のアドレスを固定

- /etc/postfix/main.cf※

```
smtp_bind_address6 = 2001:db8::25
```

※master.cfでも利用可能です。

[]で囲む必要のあるもの

- mynetworksやdebug_peer_listのように、Postfixマッチリストを設定する場合、“type:table”形式と混乱しないためにも、IPv6アドレスは、[]で囲う必要があります

```
# mynetworks = hash:/etc/postfix/network_table
mynetworks = 127.0.0.0/8 [::1]/128
```

- DNS(bind9)
- SMTP(postfix)
- **POPサーバ（dovecot)**
- Apache
- NTP

DovecotのIPv6化

- 特別な設定は不要。CentOS5系に付属のDovecotでは、Listenアドレスの指定に関して、あまり複雑な指定はできない。
- /etc/dovecot.conf

```
# “*”を指定すると、すべてのインタフェースのIPv4アドレスを
# Listenする
#Listen = *
#
# “[::]”を指定すると、すべてのインタフェースのIPv6アドレスを
# Listenする。OSによっては、すべてのインタふぇすのIPv4も
# Listenする。
#Listen = [::]
```

- DNS(bind9)
- SMTP(postfix)
- POPサーバ（dovecot)
- **Apache**
- NTP

Apacheの設定

- 実は、特別なことはなにもない
  - Apache2.0系からIPv6に対応
  - 以降は、IPv6依存部分について言及

Listen

アドレスは、IPv4の場合と異なり、[] で括る必要がある

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

ACL / アドレスによるアクセス制御

- 2001:db8:0:1000/64とはできないので注意。きちんとネットワークアドレスを指定してやる必要がある

```
AuthName “Staff Only”
AuthType Basic
AuthUserFile “/var/www/www.example.jp/.htpasswd”
Require valid-user
Order Deny,Allow
Deny from all
Allow from 192.168.1.1
Allow from 2001:db8:0:1000::/64
Satisfy Any
```

アドレスベースのVirtualHost

Listenと同様に、アドレスは、IPv4の場合と異なり、[] で括る必要がある

```
#<VirtualHost *:80>
<VirtualHost [2001:db8:0:1000::80]:80>
    ServerName www.example.co.jp
  …
  ..
  .
</VirtualHost>
```

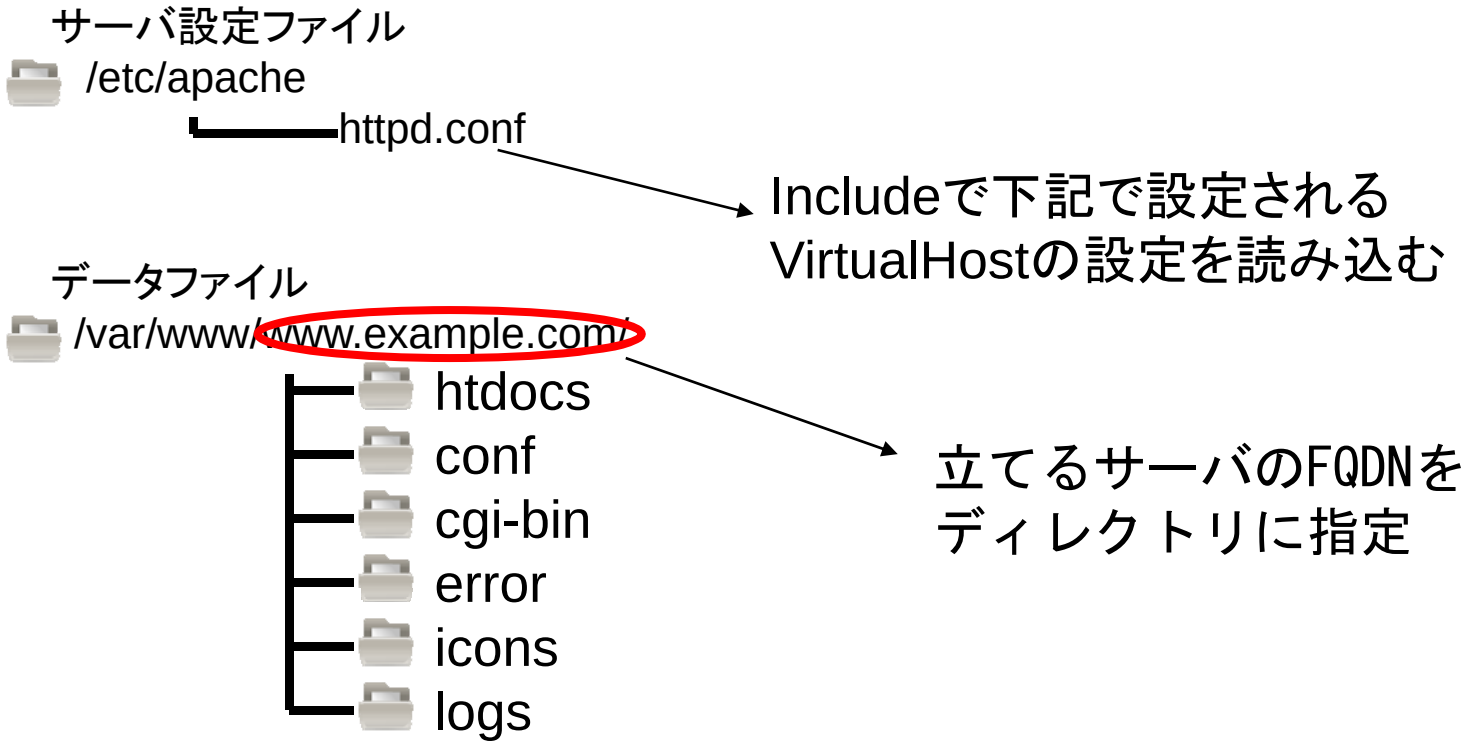
アクセスログ

下記は、一般的な出力ログの例

2001:db8:0:1000:211:24ff:dead:beaf - - [04/Nov/2008:09:30:59 +0900] "GET /favicon.ico HTTP/1.1" 404 272
192.0.2.1 - - [04/Nov/2008:09:35:59 +0900] "GET /favicon.ico HTTP/1.1" 404 272

※LinuxでApacheを動かすと、IPv6のsocketが、IPv4接続も処理しているが、ログにでるIPv4アドレスは、::ffff:192.0.2.1ではなく、192.0.2.1となっている

設定ファイルなどのレイアウト例



- メリット
  たとえ運用するVirtualHostが増減してもディレクトリレイアウトが崩れない
  VirtualHost毎のフルバックアップもディレクトリ指定でOK。
- デメリット
  標準構成から崩れるため lograterateなどを使う場合は別途見直しも必要

- DNS(bind9)
- SMTP(postfix)
- POPサーバ（dovecot)
- Apache
- **NTP**

NTP

- 上位NTPサーバの指定
  - FQDNおよびIPv6アドレスでの指定が可能

```
server     ntp1.v6.mfeed.ad.jp
server     2001:3a0:0:2005::57:123
```

NTP

- クエリの制限
  - restrictコマンドでは、IPv4/IPv6アドレスが扱えるが、”-4”, “-6” で明示的に指定することが推奨されている。

```
#restrict -4 default kod notrap nomodify nopeer noquery
#restrict -6 default kod notrap nomodify nopeer noquery
#restrict 127.0.0.1
#restrict ::1
restrict -4 192.0.2.0 mask 255.255.255.0 knod notrap nomodify nopeer noquery
restrict -6 2001:db8:: mask ffff:ffff:ffff:ffff:: knod notrap nomodify nopeer
#restrict 192.168.123.0 mask 255.255.255.0 notrust
```

NTP

- 同期確認は、IPv4と同様、ntpqコマンドを用いる(ただし、表示が丸められる）

```
$ ntp -pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 2001:3a0:0:2001 210.173.160.86   2 u   57   64    1    3.195    9.845   0.002
 2001:3a0:0:2005 210.173.160.56   2 u   56   64    1    3.173    9.871   0.002
```

```
$ ntp –pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 2001:3a0:0:2001 210.173.160.56   2 u   29   64   77    3.130    5.788   2.341
*2001:3a0:0:2005 210.173.160.86   2 u   33   64   77    3.173    9.871   4.693
```

- **IPv6トラフィックの測定(MRTG)**
- IPv6対応ツールの紹介
  - Smokeping
  - Nagios

MRTGによるトラフィック取得

- 下記のPerlモジュールがインストール済みなら、MRTGは、IPv6でデータの取得が可能（cfgmakerもIPv6対応）
  - Socket6
  - IO::Socket::INET6
- IPv6でのデータ取得の際には、下記の設定をわすれないこと

```
EnableIPv6: yes
```

IPv6トラフィックの計測

- 通常のMIBでは、IPv6のみのトラフィックは取れないため、工夫が必要
- ip6tablesを利用し、パケットを測定する方法で、取得は可能

MRTGの設定

- MRTGでは、SNMP以外に、外部スクリプトを用いてもデータをグラフ化できる。

```
Target[Linux]: `/usr/local/bin/v6counter.pl`
```

ip6tablesを用いる方法

```
#!/usr/bin/perl
# for IPv6 traffic counting
#
use strict;
use warnings;

########################################
# Edit following values.
########################################
my $ip6tables_cmd = '/sbin/ip6tables';
my $head_cmd      = '/usr/bin/head';

########################################
# Main Routine
#
my $input;
my $output;

{
  foreach my $TARGET qw/INPUT OUTPUT/ {
    chomp(my $data = `$ip6tables_cmd -vnx -L $TARGET --line-number | $head_cmd -1`);
    my @counter = split(/¥s+/, $data);
    if ($data =~ /INPUT/) {
        $input = $counter[6];
    } else {
        $output = $counter[6];
    }
  }
  print "$input¥n$output¥n";
}
```

- IPv6トラフィックの測定(MRTG)
- **IPv6対応ツールの紹介**
  - Smokeping
  - Nagios

smokeping

- ネットワークのlatencyを計測できる
  - Probeの追加で、ICMPだけでなく、Port監視も可能
- Site
  - http://oss.oetiker.ch/smokeping/

SmokepingのIPv6対応

- Probeにfping6を指定

```
+ FPing6
binary = /usr/sbin/fping6
```

- Probeに、設定したFping6を指定する。

標準のProbeにFping6を利用する場合

```
probe = FPing6

menu = Top
title = Network Latency Grapher

+ Servers
menu= Server
title = Server
++ server1
menu = server1
title= server1
host = www.example.jp
```

標準のProbeはFpingの場合

```
++ server1
probe=FPing6
menu = server1
title= server1
host = www.example.jp
```

Nagios

- 統合監視ツール
  - Pluginにより、ICMPだけでけでなく、柔軟な監視が可能。
- Site
  - http://www.nagios.org/

NagiosのIPv6対応

- ほとんどのpluginは既にIPv6に対応しており、IPv6アドレスを$HOSTADDRESS$に指定してやればよい。但しFQDNを利用する場合には、別途commandを定義する。

```
define command {
        command_name    check_ping6
        command_line    $USER1$/check_ping –H $HOSTADDRESS$ -w
$ARG1$ -c $ARG2$ -p 5 -6
}

define command {
        command_name    check_ping4
        command_line    $USER1$/check_ping –H $HOSTADDRESS$ -w
$ARG1$ -c $ARG2$ -p 5 -4
}
```

セキュリティ

- パケットフィルタ
  - Path MTU Discovery
  - 拡張ヘッダについて
  - ip6tables概要
  - ip6tables設定
- ACL
  - tcp_wrappers
  - Mapped_address
  - FQDN

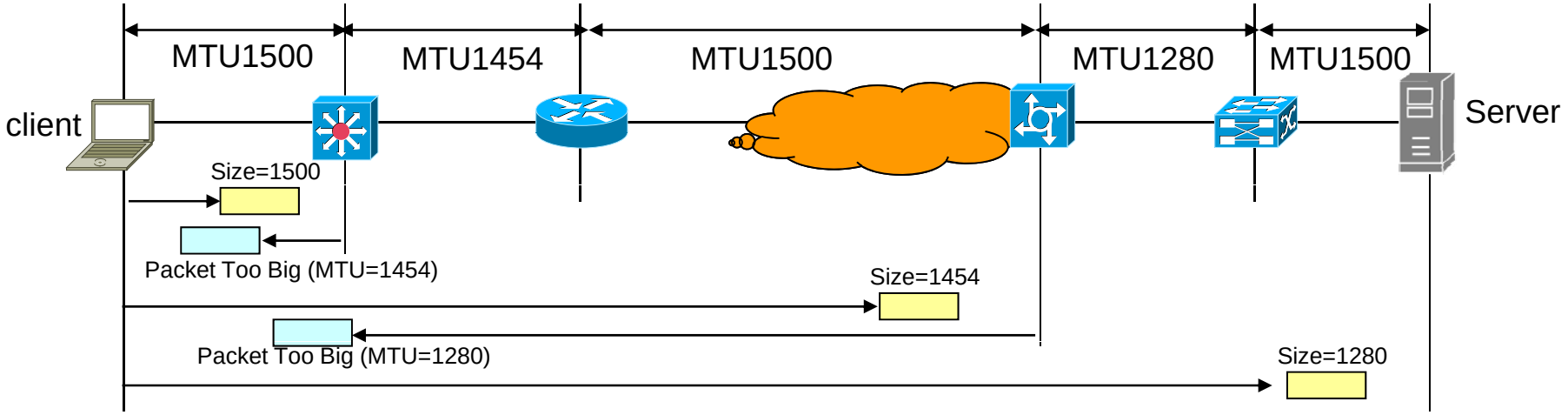
- **パケットフィルタ**
  - Path MTU Discovery
  - 拡張ヘッダについて
  - ip6tables概要
  - ip6tables設定
- ACL
  - tcp_wrappers
  - mapped_address
  - FQDN

パケットフィルタについて

- IPv4と違い、IPv6ではICMPをすべて止めてしまうと、通信ができない場合があるので注意が必要。
- ICMPv6が必須の理由
  - IPv6では、途中ルータの負荷低減のために、途中経路でのフラグメントが禁止されている。このため、Path MTU Discoveryの動作が必要となる（この動作に、ICMPv6が必須）

Path MTU Discovery

- 送信元ホストは送出先リンクのMTUをパスMTUと仮定
- 経由するルータ上でパケットを転送できない場合、ルータはそのパケットを破棄してPacket Too Big（ICMPv6 type2）を送信元に返信する（次ホップへのリンクのMTU情報を盛り込む）
- IPv6の最小MTUは1280バイト
- マルチキャストでも同様
  - 宛先全体の最小MTUとなる
- L2SWのMTUに引っかかった場合には破棄される



パケットフィルタの基本

- end-to-endの通信を想定しているため端末側でしっかり守る必要がある
- IPv6での注意点
  - ICMPv6はとめない
    - 特にtype2(Packet Too Big)
  - EDNS0やTCP53も通す
    - IPv6ではDNS回答パケットが大きくなりがちなためほぼ必須
- 拡張ヘッダへの対応
  - 単純なパケットフィルタでは対応が難しいものもある
    - RH0、フラグメントヘッダ等
  - ファイアウォールでの検討も必要

パケットフィルタ１（参考）

	Ingress	Egress
必須	[1]全ICMPv6をaccept [2]以下がSourceアドレスとなっているパケットをreject ・予約済みアドレス ::/8 ・元サイトローカルアドレス fec0::/10 ・ユニークローカルアドレス fc00::/7 ・マルチキャストアドレス ff00::/8 ・ドキュメントアドレス 2001:db8::/32 [3]自ASで持っているprefixがSourceアドレスに なっているパケットをreject（トランジット接続）	特になし
オプション	[1]境界インタフェース宛となっているICMPv6パケットの制限をする - 前提条件 1. Neighbor Discovery で使われる ICMPv6 TYPEはaccept をする 2. Path MTU Discovery で使われる ICMPv6 TYPE =2 (Packet Too Big) は accept をする 3.速やかな IPv6/IPv4 フォールバック の為に、ICMPv6 TYPE = 1 (Destination Unreachable)は accept をする [2]境界インタフェース宛となっている上記以外のICMPv6をreject ※traceroute, pingの確認ができなくなる [3]6bone用アドレス（廃止）をreject 3FFE::/16	[1]全てのICMPv6をacceptする [2]以下がSourceアドレスになっているパケットをreject ・予約済みアドレス ::/8 ・元サイトローカルアドレス fec0::/10 ・ユニークローカルアドレス fc00::/7 ・マルチキャストアドレス ff00::/8 ・ドキュメントアドレス 2001:db8::/32 ・ 6bone用アドレス 3FFE::/16

参考 ： http://www.janog.gr.jp/doc/janog-comment/jc1006.txt

パケットフィルタ２．１（参考）

Recommendations for ICMPv6 Transit Traffic

Traffic that Must Not be Dropped	Destination Unreachable (Type 1) - All codes Packet Too Big (Type 2) Time Exceeded (Type 3) - Code 0 only Parameter Problem (Type 4) - Codes 1 and 2 only Echo Request (Type 128) Echo Response (Type 129)
Traffic that Normally Should Not be Dropped	Time Exceeded (Type 3) - Code 1 Parameter Problem (Type 4) - Code 0 Home Agent Address Discovery Request (Type 144) Home Agent Address Discovery Reply (Type 145) Mobile Prefix Solicitation (Type 146) Mobile Prefix Advertisement (Type 147)
Traffic That Will Be Dropped Anyway (All these messages should never be propagated beyond the link which they were initially transmitted)	Router Solicitation (Type 133) Router Advertisement (Type 134) Neighbor Solicitation (Type 135) Neighbor Advertisement (Type 136) Redirect (Type 137) Inverse Neighbor Discovery Solicitation (Type 141) Inverse Neighbor Discovery Advertisement (Type 142) Listener Query (Type 130) Listener Report (Type 131) Listener Done (Type 132) Listener Report v2 (Type 143) Certificate Path Solicitation (Type 148) Certificate Path Advertisement (Type 149) Multicast Router Advertisement (Type 151) Multicast Router Solicitation (Type 152) Multicast Router Termination (Type 153)
Traffic for Which a Policy Should Be Defined	Seamoby Experimental (Type 150) Unallocated Error messages (Types 5-99 inclusive and 102-126 inclusive) Unallocated Informational messages (Types 154-199 inclusive and 202-254 inclusive)

参考：http://www.ietf.org/rfc/rfc4890.txt?number=4890

パケットフィルタ２．２（参考）

Recommendations for ICMPv6 Local Configuration Traffic

Traffic that Must Not be Dropped	Destination Unreachable (Type 1) - All codes Packet Too Big (Type 2) Time Exceeded (Type 3) - Code 0 only Parameter Problem (Type 4) - Codes 1 and 2 only Echo Request (Type 128) Echo Response (Type 129) Router Solicitation (Type 133) Router Advertisement (Type 134) Neighbor Solicitation (Type 135) Neighbor Advertisement (Type 136) Inverse Neighbor Discovery Solicitation (Type 141) Inverse Neighbor Discovery Advertisement (Type 142) Listener Query (Type 130) Listener Report (Type 131) Listener Done (Type 132) Listener Report v2 (Type 143) Certificate Path Solicitation (Type 148) Certificate Path Advertisement (Type 149) Multicast Router Advertisement (Type 151) Multicast Router Solicitation (Type 152) Multicast Router Termination (Type 153)
Traffic that Normally Should Not be Dropped	Time Exceeded (Type 3) - Code 1 Parameter Problem (Type 4) - Code 0
Traffic That Will Be Dropped Anyway (if the service is not implemented)	Router Renumbering (Type 138) Home Agent Address Discovery Request (Type 144) Home Agent Address Discovery Reply (Type 145) Mobile Prefix Solicitation (Type 146) Mobile Prefix Advertisement (Type 147) Seamoby Experimental (Type 150)
Traffic for Which a Policy Should Be Defined	Redirect (Type 137) Node Information Query (Type 139) Node Information Response (Type 140) Unallocated Error messages (Types 5-99 inclusive and 102-126 inclusive)

参考：http://www.ietf.org/rfc/rfc4890.txt?number=4890

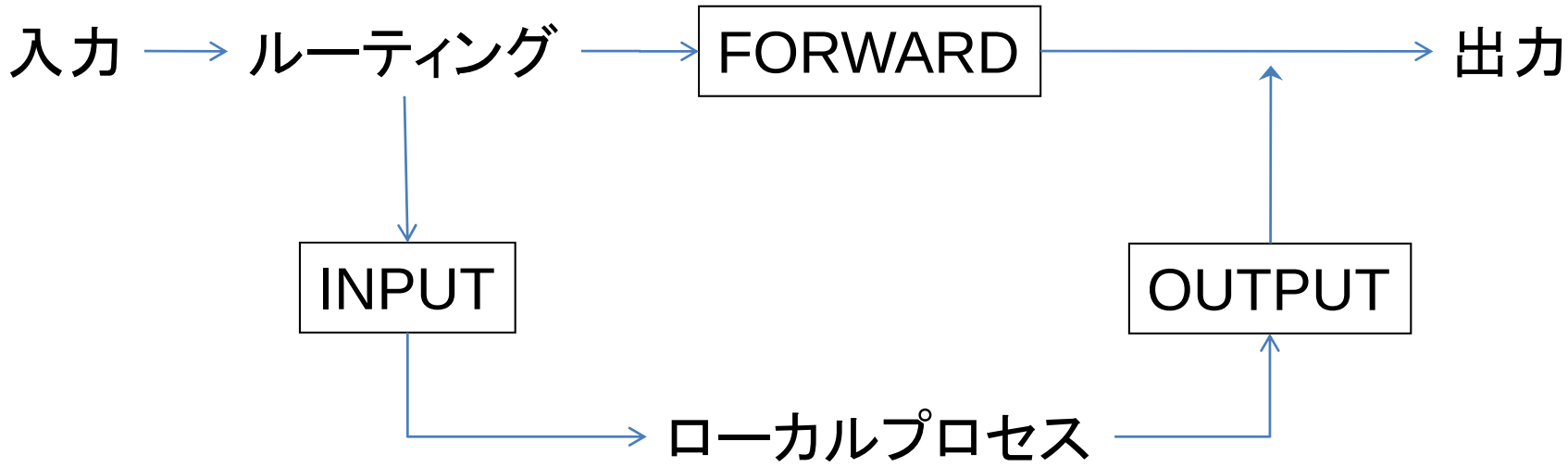
ip6tables概要

- iptablesのIPv6版
  - カーネルのパケットフィルタルールの設定などを行うことができる。
  - NATに関連するもの以外はほぼIPv4の時と同じ。ただし、ICMPとICMPv6に互換性はないので、注意が必要
  - RHEL5/CentOS5ではConnection tracking がサポートされないため、設定は投入できるものの、期待した挙動にはならない。

例:
ip6tables -A TEST -m state --state RELATED,ESTABLISHED -j ACCEPT

ip6tables設定

- 各ターゲットにルールを定義する



ICMPv6設定例

```
# Allow some ICMPv6 types in the INPUT chain
# Using ICMPv6 type names to be clear.

ip6tables -A INPUT -p icmpv6 --icmpv6-type destination-unreachable -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type time-exceeded -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type parameter-problem -j ACCEPT


# Allow some other types in the INPUT chain, but rate limit.
ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-request -m limit --limit 900/min -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-reply -m limit --limit 900/min -j ACCEPT

# Allow others ICMPv6 types but only if the hop limit field is 255.

ip6tables -A INPUT -p icmpv6 --icmpv6-type router-advertisement -m hl --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-solicitation -m hl --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-advertisement -m hl --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type redirect -m hl --hl-eq 255 -j ACCEPT


# When there isn't a match, the default policy (DROP) will be applied.
# To be sure, drop all other ICMPv6 types.
# We're dropping enough icmpv6 types to break RFC compliance.

ip6tables -A INPUT -p icmpv6 -j LOG --log-prefix "dropped ICMPv6"
ip6tables -A INPUT -p icmpv6 -j DROP
```

ICMPv6設定例（Cont)

```
# Allow ICMPv6 types that should be sent through the Internet.

ip6tables -A OUTPUT -p icmpv6 --icmpv6-type destination-unreachable -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type time-exceeded -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type parameter-problem -j ACCEPT


# Limit most NDP messages to the local network.

ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbour-solicitation -m hl --hl-eq 255 -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbour-advertisement -m hl --hl-eq 255 -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type router-solicitation -m hl --hl-eq 255 -j ACCEPT


# If we're acting like a router, this could be a sign of problems.

ip6tables -A OUTPUT -p icmpv6 --icmpv6-type router-advertisement -j LOG --log-prefix "ra ICMPv6 type"
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type redirect -j LOG --log-prefix "redirect ICMPv6 type"
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type router-advertisement -j REJECT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type redirect -j REJECT


# Accept all other ICMPv6 types in the OUTPUT chain.

ip6tables -A OUTPUT -p icmpv6 -j ACCEPT
```

参考：　http://www.cert.org/downloads/IPv6/ip6tables_rules.txt

セキュリティ

- パケットフィルタ
  - Path MTU Discovery
  - 拡張ヘッダについて
  - Ip6tables概要
  - Ip6tables設定
- **ACL**
  - tcp_wrappers
  - Mapped_address
  - FQDN

tcp_wrappers

- tcp_wrappersと複数形であることの理由は、ライブラリ形式でも提供されているから。多くのディストリビューションのsshdが、tcpdと組み合わせなくても、
  - /etc/hosts.allow
  - /etc/hosts.deny

  でコントロールできるのはそのため。

tcp_wrappers書式

- IPv6アドレス、ネットワークアドレスは []で囲う。

```
ALL:  [2001:db8:1000:2000::]/64
```

※）個別にパッチが出ていた時と、書式が異なっていることもあるので、古い文献を参照する際は注意が必要

mapped address

- mapped addressを用いて、IPv4ノードと通信を行うIPv6ノードの場合、リモートアドレスが、xx.xx.xx.xx という形式にならないので、不具合が生じる可能性がある。

FQDN

- Apacheなど、アクセス元のユーザのドメイン名で、許可・不許可を決めているものがあるが、IPv6では、クライアントの逆引きが設定されないことが多いので、既存のサーバをIPv6に対応させる際には、注意が必要

運用

- **DNS逆引きについて**
- SMTP問題
- ログ形式について
- アドレス自動設定について
- 障害の切り分け(FQDN)
- VLAN

DNS逆引きについて

- クライアントのアドレスは、プライバシー拡張などを用いてアクセスされることなどを考えると、DNSの逆引きによるACLは、事実上使えないと思ったほうがよい。

- DNS逆引きについて
- **SMTP問題**
- ログ形式について
- アドレス自動設定について
- 障害の切り分け(FQDN)
- VLAN

SMTP問題

- MXにA RRおよびAAAA RRを持つサーバを指定していると、うまくフォールバックできない実装がある（らしい）

SMTP問題

- 対処方法

```
$ORIGIN example.jp.
;
@           IN          MX    10    mail
;
mail        IN          A           192.0.2.1
            IN          AAAA        2001:db8::25
```

これを……

```
$ORIGIN example.jp.
;
@           IN          MX    10    mail
                        MX    20    mail4
;
mail        IN          A           192.0.2.1
            IN          AAAA        2001:db8::25
;
Mail4       IN          A           192.0.2.1
```

- DNS逆引きについて
- SMTP問題
- **ログ形式について**
- アドレス自動設定について
- 障害の切り分け(FQDN)
- VLAN

ログ形式について

- syslogなどで記録されるアドレスは、決まった短縮法が適用されるわけではない。吐き出すアプリケーションに依存する。このため、grepなどで、単純にアドレスをマッチさせることはできないことがある（現在、IETFで、この問題が議論されている）

- DNS逆引きについて
- SMTP問題
- ログ形式について
- **アドレス自動設定について**
- 障害の切り分け(FQDN)
- VLAN

アドレスの自動設定について

- 多くのクライアントノードを管理する管理者の作業低減の目的があった。
  - しかしながら、サーバ用途では、NICの交換によって、アドレスが変わってしまうこともあるので、自動設定は避けたほうがよい。
    - アドレス変更にともなう、フィルタのルール変更なども必要なため。

- DNS逆引きについて
- SMTP問題
- ログ形式について
- アドレス自動設定について
- **障害の切り分け(FQDN)**
- VLAN

障害の切り分け（FQDN）

- 監視ツールで、FQDNでノードを登録していた場合、IPv6/IPv4のフォールバックの影響を受けないか注意が必要。

運用における注意点

- DNS逆引きについて
- SMTP問題
- ログ形式について
- アドレス自動設定について
- 障害の切り分け(FQDN)
- **VLAN**

VLAN

- ポートVLAN, tagged VLANなど、特に問題なし。
- MACアドレスベースVLAN
  - ISPなどではほとんど使われないだろうが、現状まともに動く実装はない。
    - RAなど、multicast宛てに投げられると、全ユーザにアドレスが割り当てられてしまう
    - 利用しないほうが無難

- Q&A?

付録

- IPv6の接続性を得るには？
- IPv6用語

IPv6の接続性を得るには？

- 上流ISPからIPv6のトランジットを買う
  - 意外とあります。
- IPv6対応のiDCに入る
  - 意外とあります。
- Tunnel Brokerを探す
  - Feel6(dtcp)
  - OCN IPv6(L2TP)

http://ipv6.blog.ocn.ne.jp/ipv6/2006/04/linuxocn_ipv62_5915.html

IPv6用語

- ノード
  - IPv6が実装されている機器
- ルータ
  - 他へIPv6パケットを転送するノード
- ホスト
  - ルータではないノード

IPv6用語　cont(2)

- 上位層(upper layer)
  - IPv6の直上のプロトコル層(TCP,UDPなど）
- リンク
  - IPv6の直下の層を指す(Ethernet,PPPなど）
- 近隣（neighbors)
  - 同じリンクに接続しているノード
- インターフェース
  - ノードがリンクへ接続するためのアタッチメント

IPv6用語 cont(3)

- パケット
  - IPv6ヘッダを含むペイロード（データ部）
- リンクMTU
  - そのリンク上で伝送させることのできるパケットの最大の伝送単位
- パスMTU
  - 送信元と送信先のノード間に存在するすべてのリンクにおいて、もっとも小さなリンクMTU