実践！IPv6 Webサービス構築
IPv6環境構築の基礎

株式会社BeaconNC

國武　功一

B*NC

ここで取り上げること

B*NC

- 目的
  - IPv6の基礎を身につけ、IPv6対応のWebサービスを立ち上げるまでを取り扱います。

（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。

- 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
  - アプリケーションのIPv6対応とは？
- LinuxでIPv6
- ApacheでIPv6

IPv6の主な特徴

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

340282366920938463463374607431768211456

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**

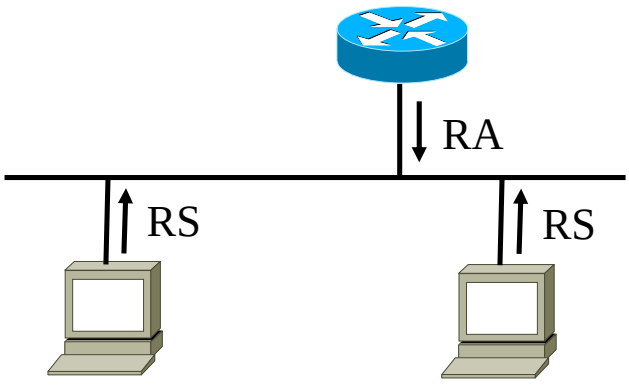
IPｖ6の広大なアドレス空間(Cont)

- IPv4/IPv6でアドレス個数を比較することは無意味
- 構築可能なセグメント数は？
  - IPv4 : 1073741824
  - IPv6 : 18446744073709551616

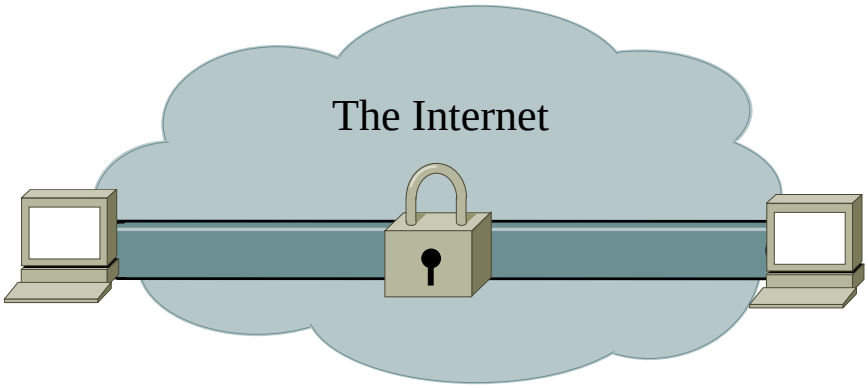
※) IPv4は、全アドレス帯を/30で分割した場合。IPv6は、/64で分割した場合。すべてがグローバルアドレスとして利用できるわけではないので、あくまでも目安

アドレスの自動設定

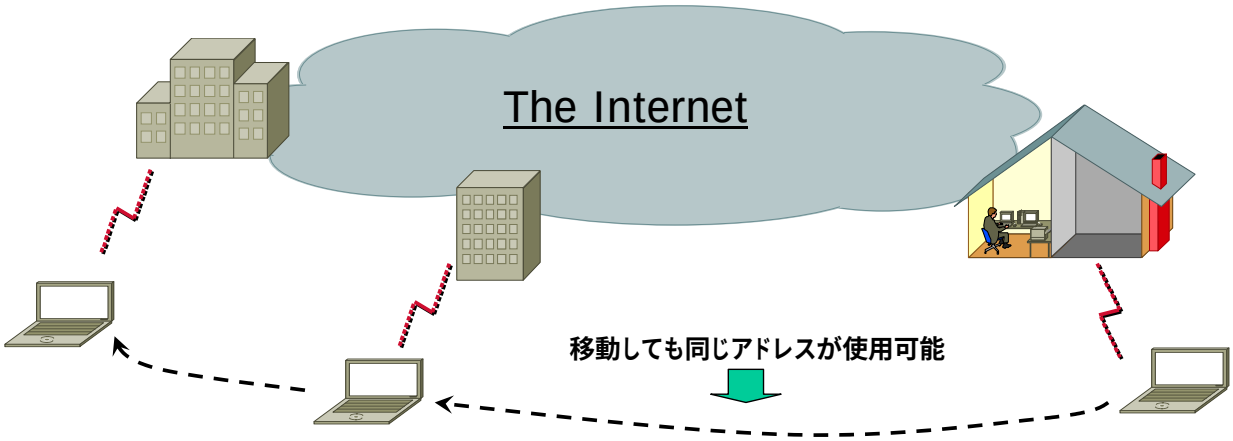
- IPv6では、予想される莫大な数のデバイスに対応するため、アドレスの自動設定が標準機能として用意されている
  - DHCPV6を利用することも可能



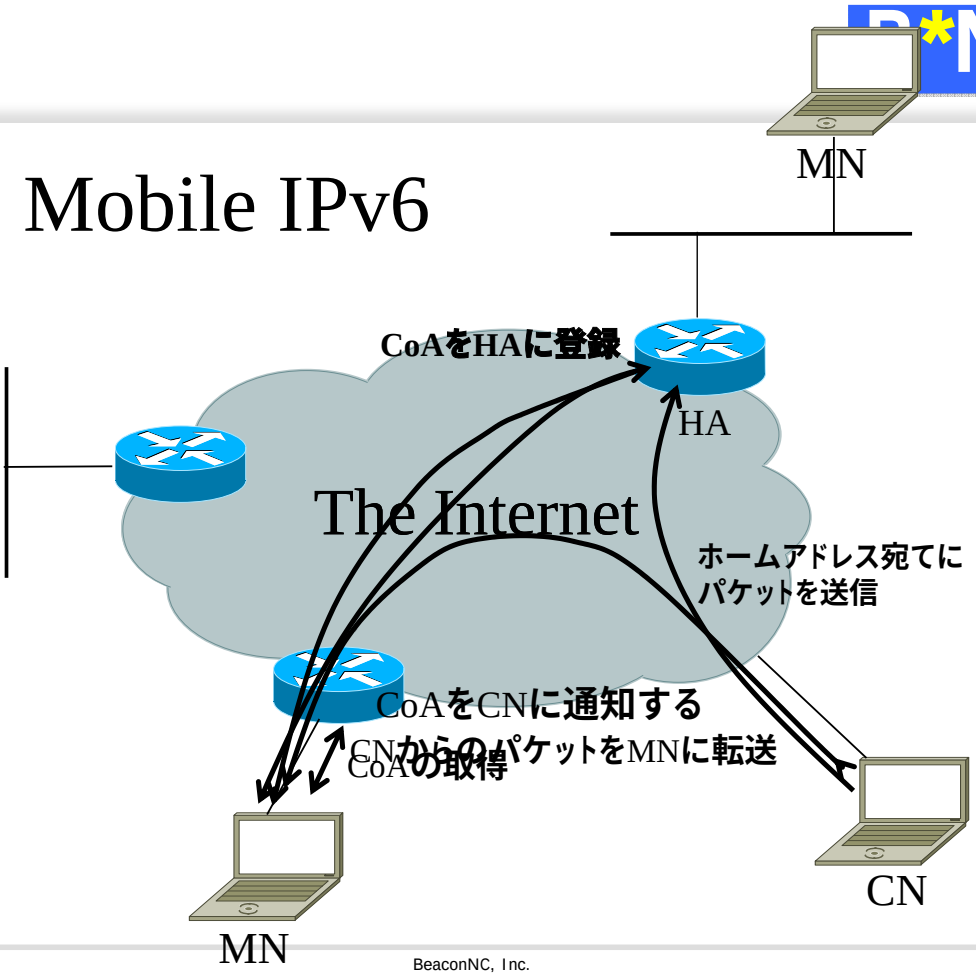
- IPレイヤで通信を保護
  - アプリケーションの改造が必要ない



- 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



Mobile IPv6



IPv6のアドレス表記について

- 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

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

- 2001:db8:0000:0000: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

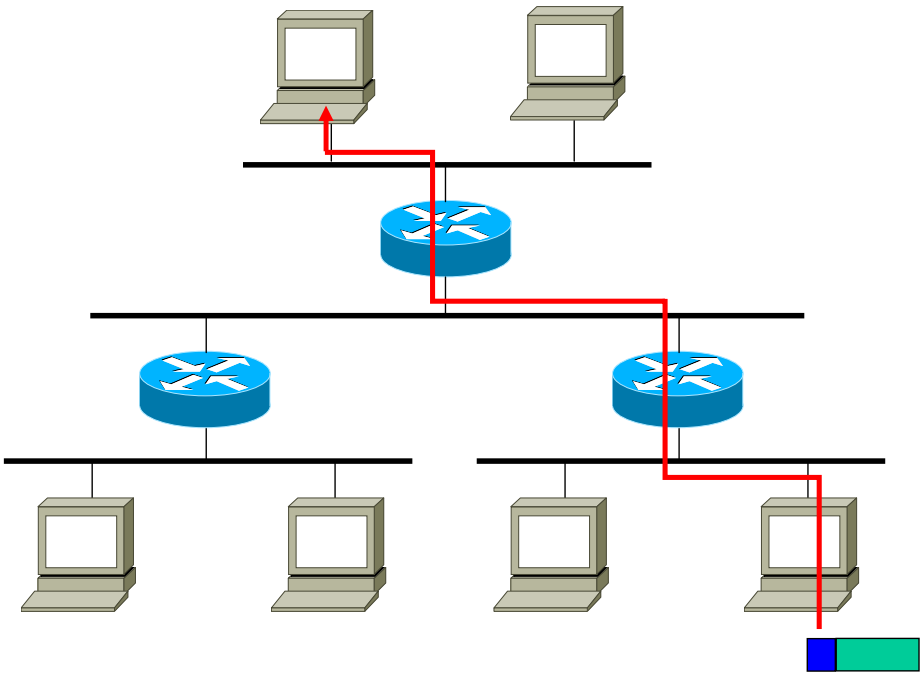
IPv6のアドレス体系について

IPv6アドレスの分類（一例）

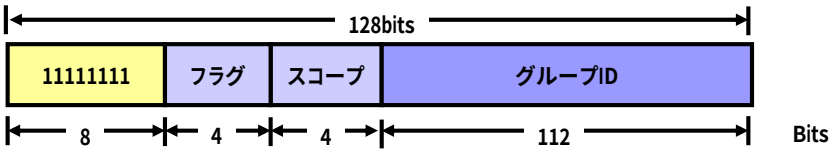
- 挙動
- 適用範囲（スコープ）
- 特殊アドレス

- 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

- 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）



- 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。

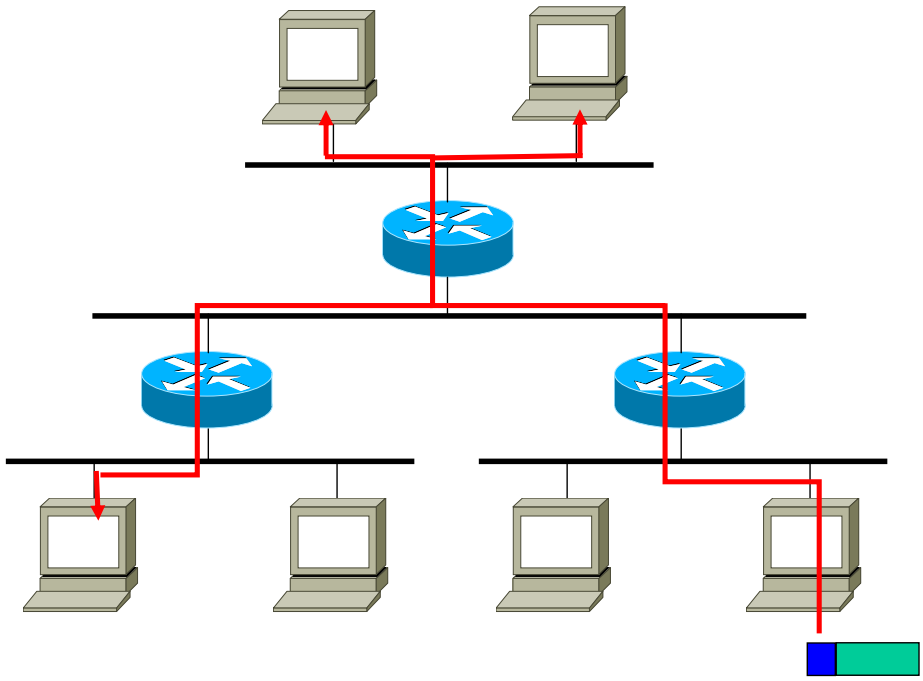


T=0 永久的なマルチキャストを示す
T=1 一時的なマルチキャストを示す

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。それらは、

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

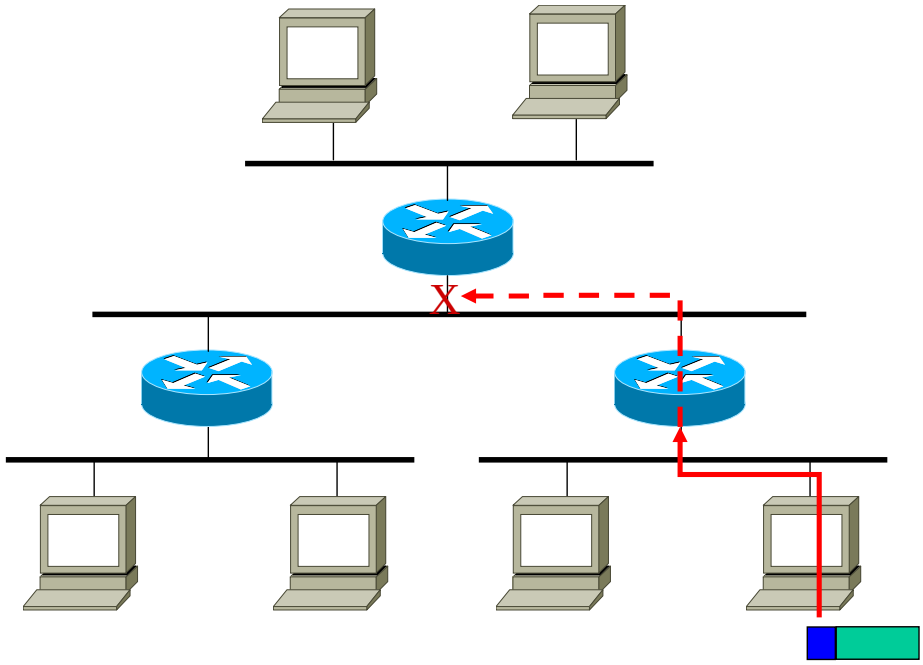
マルチキャストアドレス



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

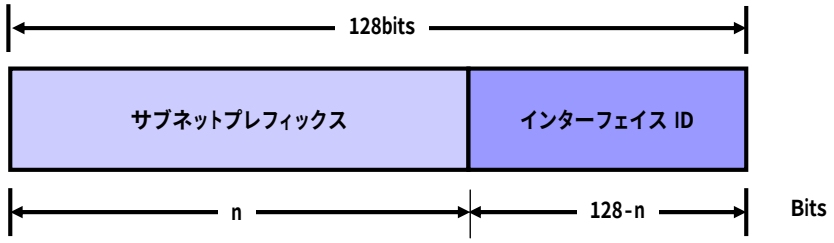


適用範囲による分類

- 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

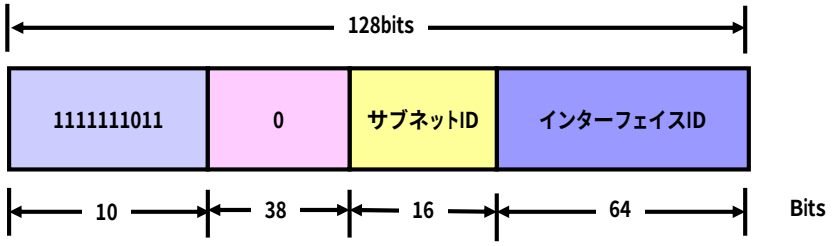
グローバルアドレス

- 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）

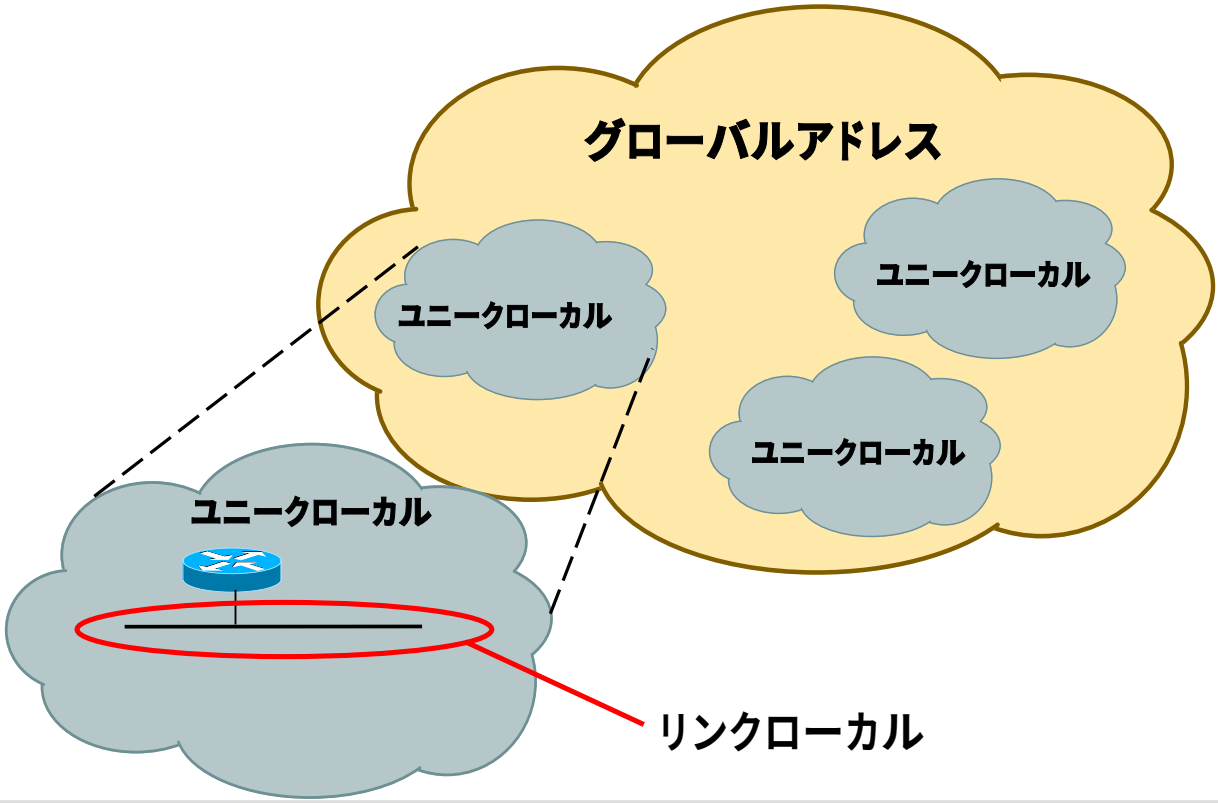


Unique Local IPv6 Unicast(ULA)**アドレスでは、サイトローカルアドレスとは異なり、プライベートアドレス的に使えるが、ほぼ他組織と、アドレスが重複することがない**

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。

各アドレスの適用範囲



- 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

- 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

⇩

::

- 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

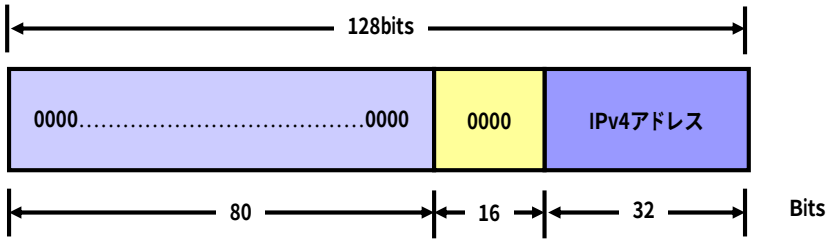
0000:0000:0000:0000:0000:0000:0000:0001

⇩

::1

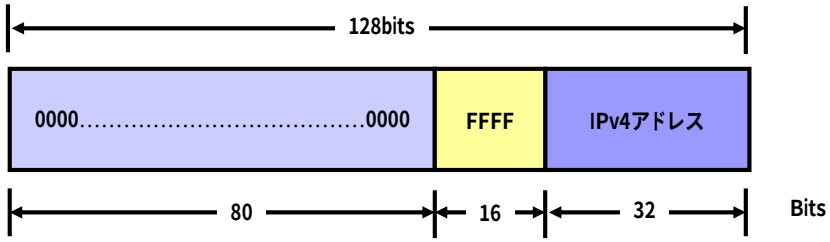
- **IPv4ネットワークにしかつながっていないIPv6ノード同士が通信する際に使用されるアドレス**
  - **利便性を考え、IPv4アドレス部は10進数表記のままとされている**

例) “192.168.0.1” => “::192.168.0.1”



- **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”



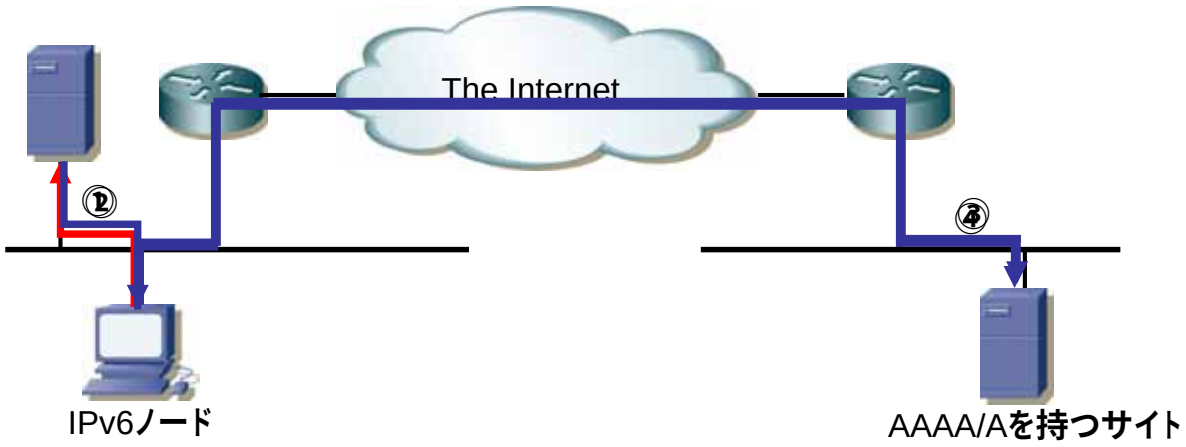
アプリケーションのIPv6対応とは？

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。

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

LinuxでIPv6

- ネットワーク設定の基礎
  - IPv4/IPv6 アドレスの設定について
  - 経路の設定について～ルーティング基礎の基礎～
  - 設定ファイルについて
  - 接続確認について

IPv4/IPv6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 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: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast
qlen 1000
4: eth1: <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 10.0.0.1/8 brd 10.255.255.255 scope global eth1
    inet6 fe80::202:xxx:xxx:xxx:xxx/64 scope link
       valid_lft forever preferred_lft forever
    inet6 2001:2a0:102:2000::a000:1/64 scope global
       valid_lft forever preferred_lft forever
```

インタフェースを上げる

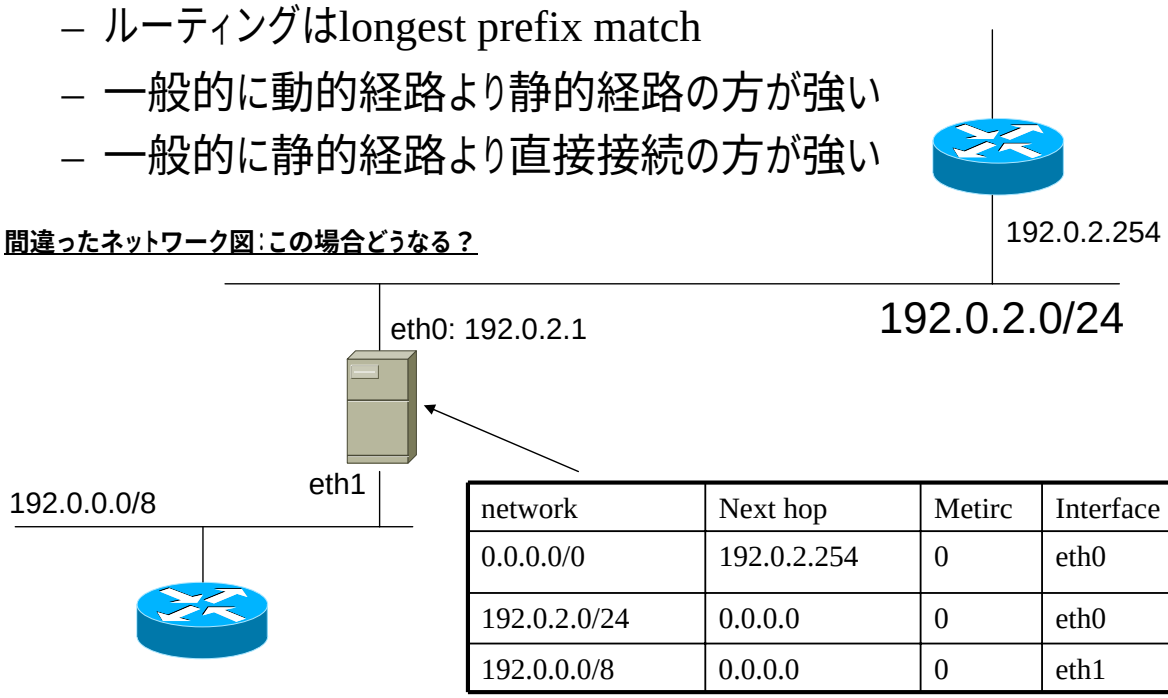
```
# ip link set eth0 up
```

インタフェースを落とす

```
# ip link set eth0 down
```

経路の設定について～ルーティング基礎の基礎～

- 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

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
```

設定ファイル

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（ディストリビューション依存）

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**

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
NETWORK
ネットワークアドレスを指定
NETMASK
ネットマスクを指定
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(HISTORYCAL)

/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
```


Debianのネットワーク設定

/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
```

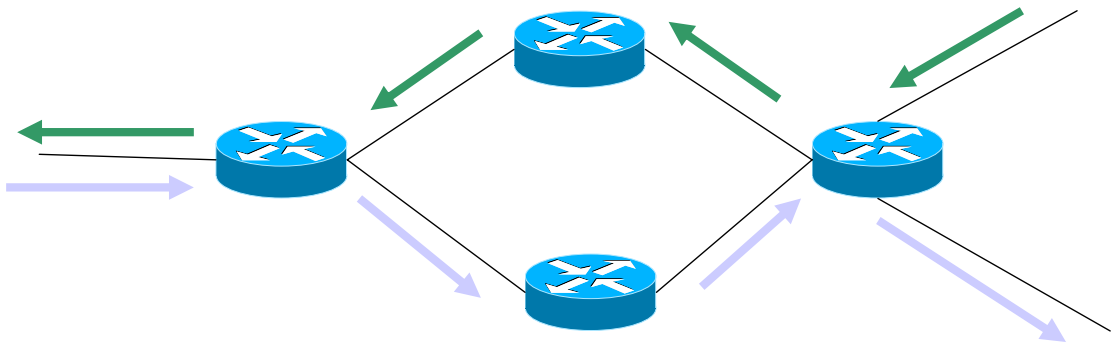
pre-up
インタフェースをupさせる前に実行するコマンド
up
インタフェースをupしたときに実行するコマンド
post-up
インタフェースをupした後に実行するコマンド

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

接続確認について

- 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……

- 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
```

BeaconNC, Inc.

- 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
```


- 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

- 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

- 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**

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)

- 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
```


Webサーバの構築

- 実は、特別なことはなにもない
  - Apache2.0系からIPv6に対応

B*NC

とは言いながら、IPv6特有の設定を紹介

BeaconNC, Inc.

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
```


- 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
```

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>
```


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

- 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

- 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

- 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