IPv4 EXHAUSTION

IPv6オペレータ育成プログラム

本資料は2009年12月17-18日に開催されたIPv4
アドレス枯渇対応タスクフォース主催の
IPv6ハンズオンセミナー
「iDC/ISP/CATV サーバ編」(講師：國武功一氏)を
元にし、公開用に資料を編集したものである。

IPv4アドレス枯渇対応タスクフォース
http://www.kokatsu.jp/

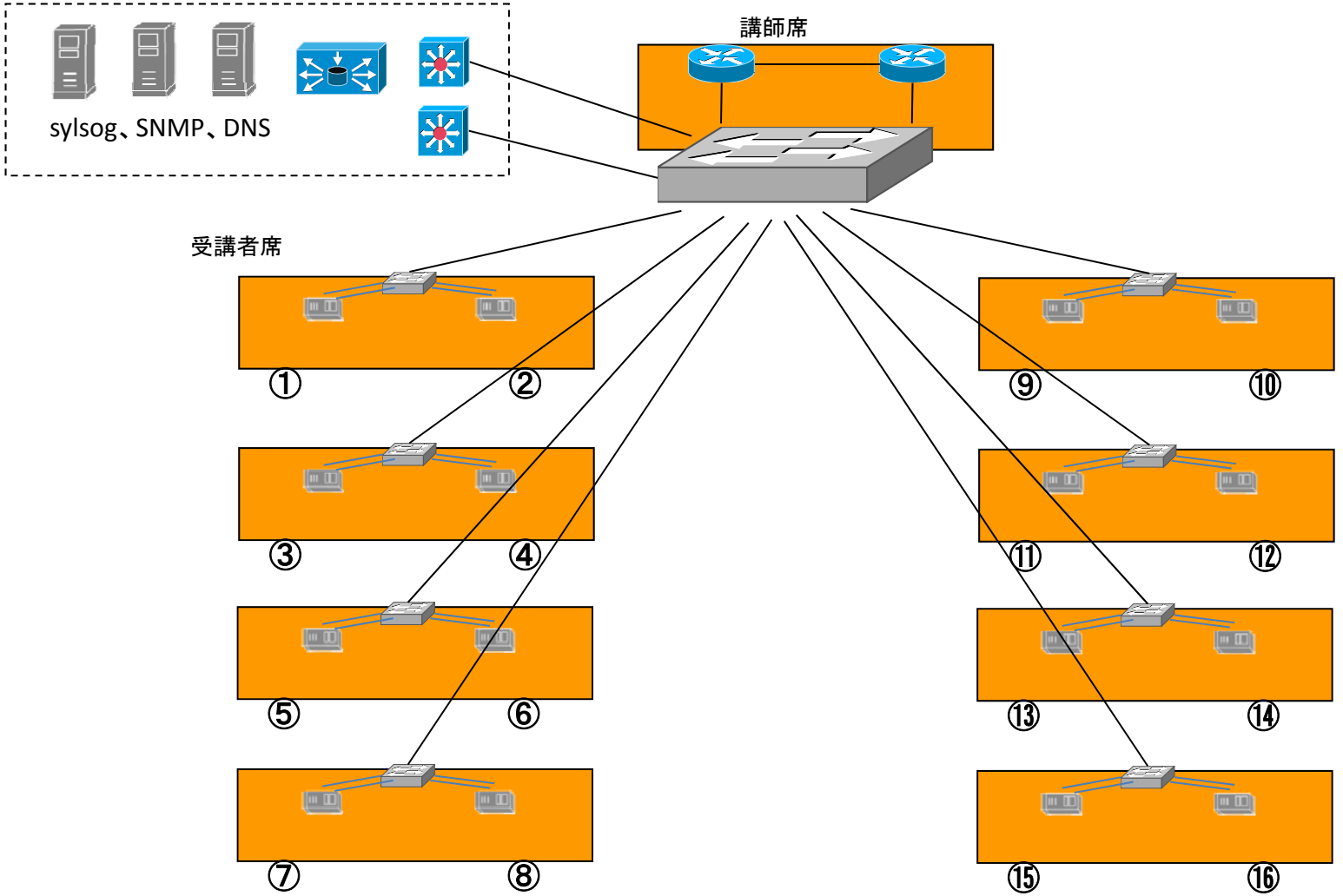
IPv4 EXHAUSTION

IPv6オペレータ育成プログラム

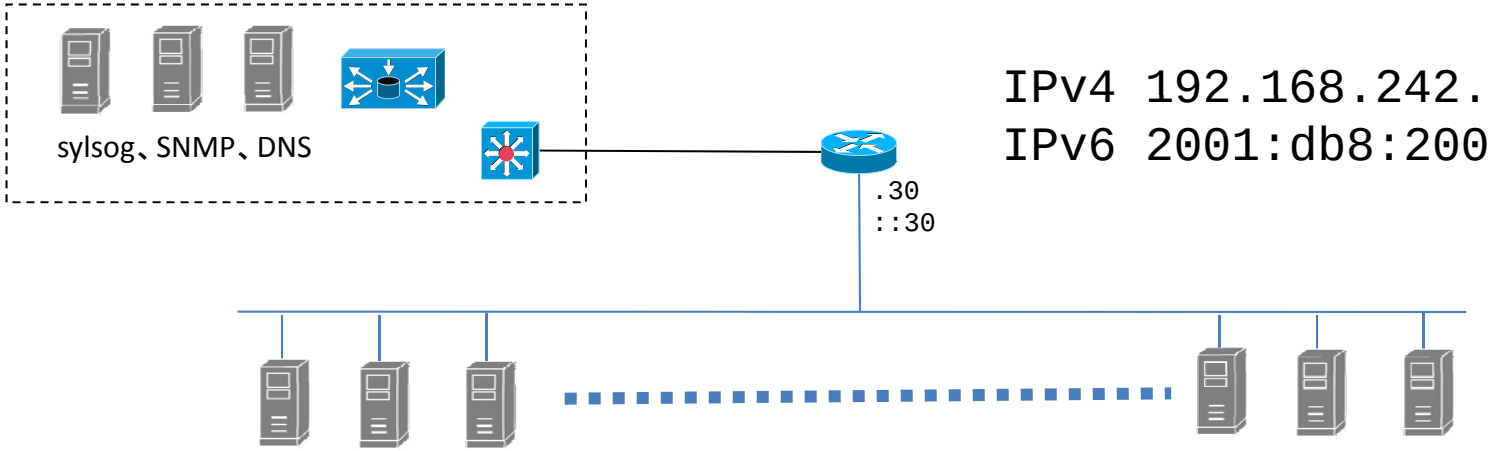
ハンズオン用資料

株式会社ビーコンエヌシー
國武　功一

IPv6ハンズオン物理構成図



IPv6ハンズオン論理構成図



IPv4 192.168.242.0/27
IPv6 2001:db8:2000::/64

サーバへのログイン

- デスクトップにあるショートカットのTeraTermを使用し、シリアル経由でログインする。
- アカウント情報（ID/Pw)
  - ****/****
  - ****/****

※必要でしたら作業内容を保存し、USBメモリに移すなどして、お持ち帰りください。

セッション2

IPv4/IPv6アドレスの設定

- このセグメントには
  - “192.168.242.0/27”
  - “2001:db8:2000::/64”

  が割り当てられています。
- 各自の利用アドレスとして
  - 192.168.242.x
  - 2001:db8:2000::x

  を使ってください（xは、受講番号です）

IPv6アドレス重複時の挙動

- 重複したアドレスを割り当てた際の、挙動を確認してください。
  - 2001:db8:2000::17
- Syslogに、DADのメッセージが出ていることを確認してください。

IPv4/IPv6経路設定

- Default gw は下記の通りです。
  - 192.168.242.30/27
  - 2001:db8:2000::30/64
- DNSキャッシュサーバは、下記の通りです。
  - 2001:db8:2000::250
- 設定は、設定ファイルに反映させて下さい
- 設定後、疎通確認をお願いします。

  # traceroute6 ipv6.google.co.jp

  # tracepath6 www.nic.ad.jp

Try: DAD確認

```
# ip addr add 2001:db8:2000::17/64 dev
  eth0
# ip addr show dev eth0
```

```
# tail /var/log/messages|grep detect
```

Try: アドレス経路設定

アドレス設定

```
# ip link set eth0 down && ip link set eth0 up
# ip addr add 192.168.242.x/27 dev eth0
# ip addr add 2001:db8:2000::x/64 dev eth0
```

経路設定

```
# ip route add 0.0.0.0/0 via 192.168.242.30 dev eth0
# ip route add ::/0 via 2001:db8:2000::30 dev eth0
```

Try: resolv.conf設定

/etc/resolv.conf

search example.jp

nameserver 2001:db8:2000:ffff::250

実際の確認

```
$ dig ipv6.google.co.jp AAAA
```

アドレス自動設定のOFF

- アドレス自動設定をOFFにした後に、サーバをrebootし、自身が明示的につけたアドレスしかついていないことを確認します。

Try: アドレス自動設定OFF

/etc/sysconfig/network

IPV6_AUTOCONF=no

有効化

/etc/init.d/network restart

参考
(Advanced)

Bondingの設定

- 一通りの設定が終わったら、eth0/eth1 を用いて、bondingを組んでみてください。
- Defualt gwにping6を打ち続け、ケーブルを抜き差ししてみてください。

Try:Bonding設定例

ifcfg-bond0

```
DEVICE=bond0
BOOTPROTO=none
ONBOOT=yes
IPV6INIT=yes
IPV6ADDR=xxx:xxx:xxx::x/64
IPADDR=xx.xx.xx
NETWORK=xx.xx.xx
NETMASK=255.255.255.0
```

ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

ifcfg-eth1

```
DEVICE=eth1
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

Try:Bonding設定

/etc/modprobe.conf

```
alias bond0 bonding
options bond0 mode=1 miimon=200
```

有効化

/etc/init.d/network restart

セッション4

Apache設定

- RHEL5/CentOS5系では、すでに標準で、IPv6 Ready.　適当なコンテンツを設置して、ブラウザでアクセスしてみてください。
  - アドレスバーに http://[2001:db8:2000::x]/
- アクセスした時に、ログがどうなっているか、確認してみてください。

VirtualHost設定

- IPアドレスベースのVirtualHostの設定を施し、IPv4とIPv6のコンテンツを変えてみましょう。

Apache ACL設定

- 隣同士、アクセスを確認した後、アクセスログから、アドレスを特定して、ACLでアクセス拒否の設定を入れてみてください。

Try: Apache設定

/etc/httpd/conf/httpd.conf に下記を追記

```
<VirtualHost 192.168.242.17:80>
        DocumentRoot /var/www/html/ipv4
</VirtualHost>

<VirtualHost [2001:db8:2000::17]:80>
        DocumentRoot /var/www/html/ipv6
</VirtualHost>
```

※/var/www/html/{ipv4|ipv6}に適当なコンテンツを置いてください。

Try: Apache設定確認

```
$ /usr/sbin/httpd –S

$ telnet 192.168.242.17 80
GET /index.html

$ telnet 2001:db8:2000::17 80
GET /index.html
```

メールサーバ

- ホスト名を適切に設定し、/etc/hostsに記載してください。
- 起動したメールサーバにtelnetでメールを送信してみてください。
- 送信した後、メールのログを確認してみてください。

Try:postfix設定

/etc/postfix/main.cf

```
myhostname = dns.17-handson.example.jp
mydomain = 17-handson.example.jp

inet_interfaces = all

mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain

home_mailbox = Maildir/
```

送信先ユーザの作成

useradd user1

メール送信例

```
$ telnet fe80::aaa:dead:beaf%bond0  smtp
Trying fe80::aaa:dead:beaf%bond0...
Connected to fe80::aaa:dead:beaf%bond0.
Escape character is '^]'.
220 asteroid ESMTP Postfix (Ubuntu)
HELO foo
250 asteroid
MAIL FROM: kunitake@example.jp
250 2.1.0 Ok
RCPT TO: user1@17-handson.example.jp
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: kunitake@example.jp
Subject: from handson!
Hello IPv6 world
.
250 2.0.0 Ok: queued as 22945DD71
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

NTPサーバ

- 上流のNTPサーバに
  - 2001:3a0:0:2001::27:123
  - 2001:db8:2000::x(隣の方のアドレス）

  を設定する。
- ntpqで同期を確認する。

NTPサーバによるACL

server 2001:3a0:0:2001::27:123

server 2001:db8:2000::X

#restrict default ignore

restrict -4 default nomodify notrap

restrict -6 default nomodify notrap

- restrictのどちらかを外したり、つけたりして、IPv4とIPv6のACLの設定が別々になっていることを確認してみてください。

DNSサーバ

- IPv6 transportに対応した、DNSキャッシュサーバを作成する。
- 作成したDNSキャッシュサーバを、隣の席の人同士で、使ってみる

  # dig @2001:db8:2000::x ipv6.google.co.jp AAAA
- アクセスできることを確認ののち、ACLで、隣の人からのクエリを拒否してみる。

Try:DNSキャッシュサーバ

cd /var/named/chroot/etc

wget ftp://ftp.rs.internic.net/domain/named.root

/var/named/chroot/etc/named.conf

```
acl handson-net {
        2001:db8::/32;
};
options {
        directory “/etc”;
        version “”;
        alllow-query { handson-net; 127.0.0.1; ::1; };
        listen-on-v6 {any; };
};
zone “.” {
        type hint;
        file “/etc/named.root”;
};
```

Try: DNSサーバ

確認方法

```
# dig @::1 ipv6.google.co.jp AAAA
```

ZONEの登録

- “x-handson.example.jp” のゾーンを追加し、そこに自身のサーバ名のA および AAAA RR を登録してみてください（xは受講番号）

Try:DNSキャッシュサーバ

/var/named/chroot/etc/named.conf

```
// 下記を追記
zone “17-handson.example.jp” {
        type master;
        file “master/examole.jp”;
        allow-transfer { localhost; handson-net;};
        allow-query { any ;};
};
```

Try: DNS権威サーバ

/var/named/chroot/etc/master/example.jp

```
;;
$TTL 3600
@       IN      SOA     17-handson.example.jp.  root.example.jp.
 (
                2009082601      ; Serial
                7200            ; Refresh 2hrs
                1800            ; Retry 30mins
                604800          ; Expire 1 weeks
                86400   )       ; Minimum 1 days

                IN      NS      dns.17-handson.example.jp.
                IN      MX  10  dns.17-handson.example.jp.

dns             IN      A       192.168.242.17
dns             IN      AAAA    2001:db8:2000::17
```

確認

$ dig @::1 17-handson.example.jp SOA

tcp_wrappers

- sshdへのアクセスに制限が掛っています。本ネットワークからのみ、制限を解除し、隣の方からサーバへアクセスしてもらってください。

Try: tcp_wrappers

/etc/hosts.deny

```
ALL: ALL
```

/etc/hosts.allow

```
sshd: 192.168.242. [2001:db8:2000:ffff::]/64
```

- Nagiosで、「SSH」がグリーンになっていることを確認してください。

パケットフィルタ

- ip6tablesを用いて、実際に、CERTが例示しているフィルタを設定し、その効果を確認します（別紙：ip6tables_rules.txt)

入力

```
# 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 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
```

Path MTU Discoveryなど、正常な動作に必要な受け取るべきICMPを指定。
また、hoplimitが255でないとおかしいパケットなどは、明示的にhoplimitを指定。

入力（Cont)

```
# 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

# 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
```

Echo/replyに制限を掛け、パケットのドロップを記録する。

出力

```
# 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
```

Path MTU Discoveryなど、正常な動作に必要なICMPを指定。
また、hoplimitが255でないとおかしいパケットなどは、明示的にhoplimitを指定。

出力（Cont)

```
# 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
```

ルータとして設定していた際に役に立つ設定。

サービスポートの設定等

```
# Enough ICMPv6!   :-D
# Some sample TCP rules. <These are for example purposes only.>
# The REJECT is for politeness on the local network.
ip6tables -A INPUT -m multiport -p tcp --dport $blocked_tcp_ports -m hl --hl-eq 255 -j REJECT
ip6tables -A OUTPUT -m multiport -p tcp --dport $blocked_tcp_ports -m hl --hl-eq 255 -j REJECT
ip6tables -A INPUT -m multiport -p tcp --dport $blocked_tcp_ports -m hl --hl-lt 255 -j DROP
ip6tables -A OUTPUT -m multiport -p tcp --dport $blocked_tcp_ports -m hl --hl-lt 255 -j DROP

# Stateful matching to allow requested traffic in.
ip6tables -A OUTPUT -p tcp -j ACCEPT
ip6tables -A OUTPUT -p udp -j ACCEPT
ip6tables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A INPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT

# Drop NEW,INVALID probably not needed due to the default drop policy.
ip6tables -A INPUT -m state --state NEW,INVALID -j DROP
```

POPサーバの設定

- 受信したメールを構築したPOPサーバを用いて受信してみる。

Dovecotのインストール

- Yumを用いて、dovecotをインストール

```
# yum install dovecot
```

- /etc/dovecot.confの設定

Try:Dovecot設定

/etc/dovecot.conf

```
protocols = imap pop3

protocol lda {
   postmaster_address = root@17-handson.example.jp
}

ssl_disable = yes

auth default {
  passdb passwd-file {
        args = /etc/dovecot.passwd
 }
  userdb passwd-file {
        args = /etc/dovecot.passwd
 }
}
```

Try:Dovecot設定

/etc/dovecot.passwd

```
user1:{plain}user1:501:501::/home/user1::userdb_mail=maildir:/home/user1/Maildir
```

メール受信例

```
$  telnet 2001:db8:2000::17 pop3
Trying 2001:db8:2000::17...
Connected to 2001:db8:2000::17.
Escape character is '^]'.
+OK Dovecot ready.
USER user1
+OK
PASS user1
+OK Logged in.
LIST
+OK 1 messages:
1 554
.
RETR 1
+OK 554 octets
Return-Path: <kunitake@example.jp>
X-Original-To: user1@17-handson.example.jp
Delivered-To: user1@17-handson.example.jp
Received: from fo (localhost.localdomain [127.0.0.1])
        by dns.17-handson.example.jp (Postfix) with SMTP id 092E862C109
        for <user1@17-handson.example.jp>; Wed, 16 Dec 2009 15:29:12 +0900 (JST)
From: kunitake@example.jp
Subject: test mail
Message-Id: <20091216062926.092E862C109@dns.17-handson.example.jp>
Date: Wed, 16 Dec 2009 15:29:12 +0900 (JST)
To: undisclosed-recipients:;

Hello
```