IPv6チュートリアル ～IPv6化ことはじめ～
サーバ編

株式会社 IoTスクエア

許　先明

IPv6化、とは？

- IPv6化するとはどういうことか？
  - IPv6が利用できるようにする
  - IPv4は？→2018年現在、**使えなくてもいいという状況ではありません**
  - ならば、IPv4とIPv6が使えなければならない

- IPv6を使えるようにするには？ @Server
  - IPv6での通信ができる→I/FにIPv6 Addrが設定されている＋外部とIPv6で繋がる
  - これだけなのか？→ **No!**
  - 現代のInternetにおいては、IPv[46]アドレスを「生で使うこと」は少ない
  - DNSサービスもIPv6に対応しなければならない

Dual Stack

- Dual Stack → Stackが２つ → Network Protocol Stackが2つ
  - つまりは、IPv4とIPv6の両方に対応している状況のこと

- Dual Stackであるということは
  - IPv4/IPv6 のどちらでも通信ができる
  - 加えてDNS的にIPv4/IPv6のどちらにも対応している(ContentsもTransportも)

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

www.example.com はIPv4でしか接続できない(IPv4アドレスしか持っていない)

mail.example.com は、IPv4でもIPv6でも接続できる(IPv4/v6アドレスを持っている)

ことが期待できる

DNSのIPv6対応

- DNS的IPv6対応
  - TransportがIPv4/IPv6に対応している
    - つまり、IPv4でもIPv6でもクエリを受け付けることができる
    - DNS Server自身のNSレコードにAAAA「も」定義されている
  - ContentsがIPv4/IPv6に対応している
    - IPv4 Addressを意味する A Record と IPv6 Address を意味する AAAA Record を扱うことができる
    - 逆引きはIPv4もIPv6も PTR Record を利用する
      - IPv4 Addressの逆引きは、”in-addr.arpa.”を利用するがIPv6 Addressでは、”ip6.arpa”を利用する。

DNSにおいて、Contentsとは、「**単なるデータ**」であって、それ以上の意味はない

AをIPv4, AAAAをIPv6と解釈するのはあくまでもDNSコンテンツデータを**利用する側の解釈**でしかないことに注意が必要

ServerのIPv6化

- ServerをIPv6対応にする
  - 「OSおよびNetwork環境」のIPv6対応を行う：基盤のIPv6対応
    - ServerとしてIPv6アドレスを持ち、IPv6を用いてサービスを提供できる状態にすること
  - 「Server/Client Application」のIPv6対応を行う：サービスのIPv6対応
    - IPv6対応Platformの上で、IPv6を用いたサービスを提供すること
- どちらも必要
  - 近年では、多くの実装が上記を満たす
  - IPv4とIPv6、どちら優先するかという問題が発生する

基盤のIPv6対応

- 現在リリースされ、一般に利用されているほとんどの汎用OSはIPv6に対応している
  - 一部Firmware等で対応していないものはあるかもしれない
- ここでは、OSとして、CentOS/Ubuntu/FreeBSDを採り上げる
  - WindowsもMacOSもIPv6対応しているがここでは割愛
- 手元で試してみることができる程度の説明
  - 環境がなく試すことができない方は、是非 JPNIC の HandsOn Seminar まで

IPv6設定

- Interfaceにアドレスをつける
  - ip addr add 2001:DB8:3ffe:501::1:80/64 dev eth0 (CentOS/Ubuntu)
  - ifconfig em0 inet6 2001:DB8:3ffe:501::1:80/64 (Ubuntu/FreeBSD)

- 経路を設定する
  - ip route add ::/0 via 2001:DB8:3ffe:501::1:1 dev eth0 (CentOS/Ubuntu)
  - route add -net ::/0 2001:DB8:3ffe:501::1:1 (Ubuntu/FreeBSD)

- Layer2的な近隣を確認する(neighbor discovery protocol)
  - ip neigh show (CentOS/Ubuntu)
  - ndp -a (FreeBSD)

IPv6設定

- 4層の状態を確認する
  - ss -antu (TCP/UDP状態を確認) (CentOS/Ubuntu)
  - netstat -an6 (Ubuntu/FreeBSD)
- 経路情報を見る
  - ip -6 route show (CentOS/Ubuntu)
  - netstat -arn6 (Ubuntu/FreeBSD)
- I/Fの状態を見る
  - ip link show (CentOS/Ubuntu)
  - ifconfig (Ubuntu/FreeBSD)

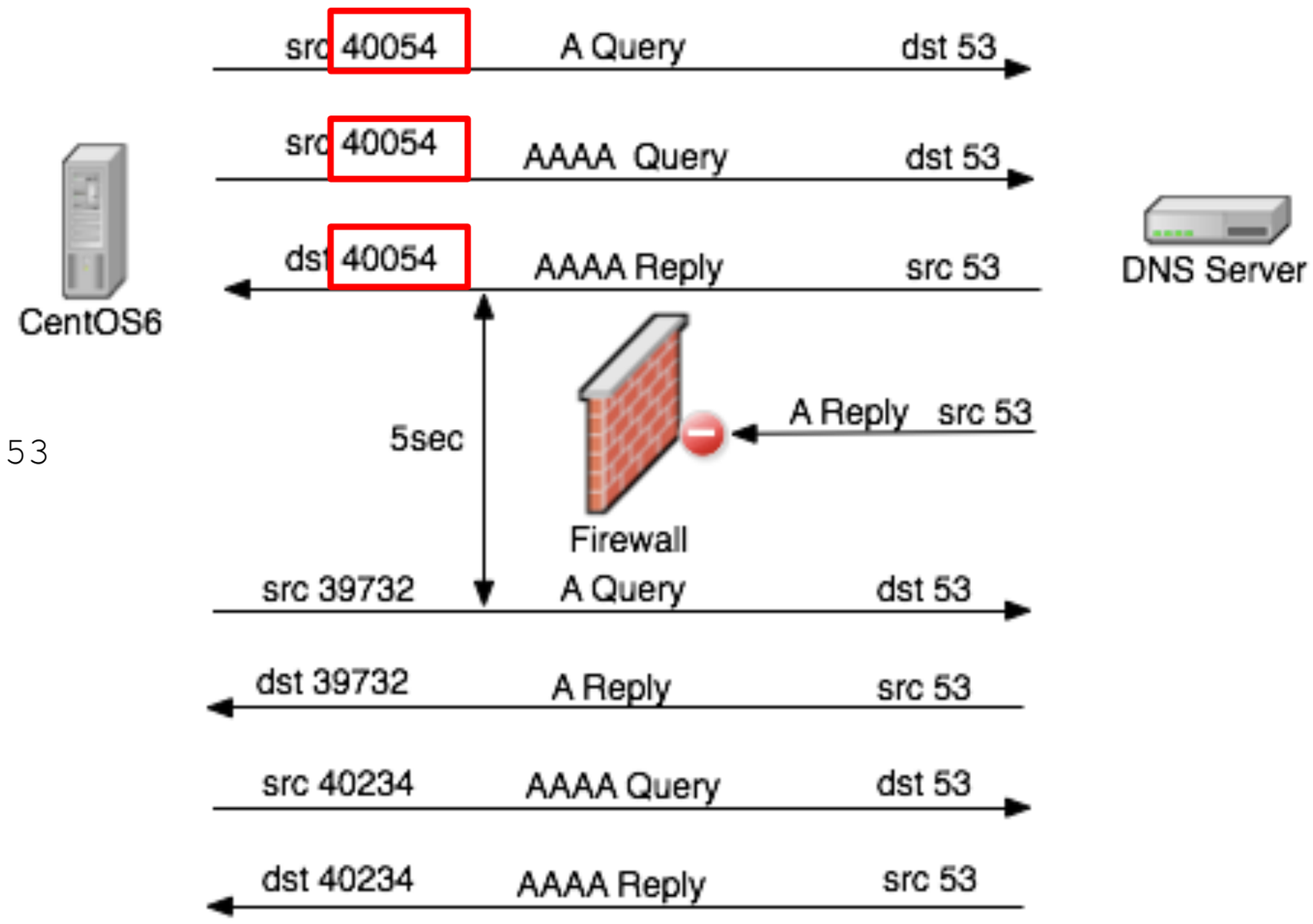
設定方法は、
微妙な違いはあるが、
慣れの範囲

resolv.confとsingle-request-reopen

- IPv6を利用して名前解決を行うには、nameserverエントリーにIPv6対応のDNS Serverを登録すれば良い
- 一部のファイアウォールの実装では、同一ポートからのクエリを同一のセッションとみなし、結果返信が落とされてしまうものがある

/etc/resolv.conf

```
search example.jp
nameserver 2001:DB8:3ffe:501::1:53
nameserver 192.0.2.254
options single-request-reopen
```



稼動確認

- 稼動確認に利用するコマンドは、ほとんど変わらない
  - ping : ICMP/ICMP6 を利用した疎通確認 (CentOS/Ubuntu)
    - CentOS: ping/ping6 (/usr/bin/pingへのSymlink)
    - Ubuntu: ping/ping4/ping6 (/bin/pingへのSymlink)
    - FreeBSD: ping/ping6 (異なるコマンドとして実装されている)
  - ss/netstat : Connection等の情報を確認する
    - CentOS: traceroute/traceroute6 (/bin/tracerouteへのSymlink)
    - Ubuntu: traceroute/traceroute4/traceroute6 (/bin/tracerouteへのSymlink)
    - FreeBSD: traceroute/traceroute6 (異なるコマンドとして実装されている)
- なお、Linux系では、tracerouteなどはinstallされないことがある

サービスのIPv6対応

- 近年ではSaaSの拡大に伴い、サービスの外部委託が増えている
  - Mail / DNS / その他各種
  - しかも、多くのサービスが、事実上HTTPの上に乗っている
  - ほとんどの人にとっては、自前でサービスを立ち上げる必要が...

- 近年では、多くのサービスソフトウェアがIPv6に対応している
  - DNS系、メール系、Web系、その他各種
  - 設定もほぼこなれている

DNS設定

- 代表的なDNS実装
  - Contents：BIND / NSD / PowerDNS / KnotDNS
  - Cache：BIND / Unbound / Knot Resolver
- どれもIPv4/IPv6ともに利用できる

BIND

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

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

acl handson-net {
    2001:fa0::/32;
            10.0.0.0/8;
};
options {
    directory “/etc”;
    version “”;
    allow-query { handson-net; 127.0.0.1; ::1; };
    listen-on-v6 {any; };
};
```

KnotDNS

```
server:
    user: "knot:knot"
    listen: [ "0.0.0.0@53", "::@53" ]

zone:
  - domain:  “example.org."
    file:  “example.org"
    storage: "/etc/knot/Zones"
    semantic-checks: "on"
    disable-any: "on"
    ixfr-from-differences: "off"
```

Zone Dataはほぼ同じ

GENERATE文だけは、BINDしか利用できない

この制限は、NSDでもPowerDNSでも同じ

DNS設定

- Cache Server
  - BINDかUnboundがほとんどではないかと思われる。
- Zone Data
  - IPv4での記述と本質的には変わらない

UNBOUND

```
server:
  interface: 0.0.0.0
  interface: ::0
  access-control: 10.0.0.0/8 allow
  access-control: 2001:fa0::/32 allow
  verbosity: 1
```

逆引きについて

一部のService (SMTP Server)に関しては、様々な事情から逆引きを登録する必要があるが、事実上SMTP Server以外は逆引きを登録する必要はない。

ZoneData

```
# AAAA登録

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

# PTR登録

$ORIGIN         0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   IN   PTR   localhost.
```

Apache/nginx

- 現在のHTTP Server実装の２大巨頭
  - 設定は比較的容易

Apache

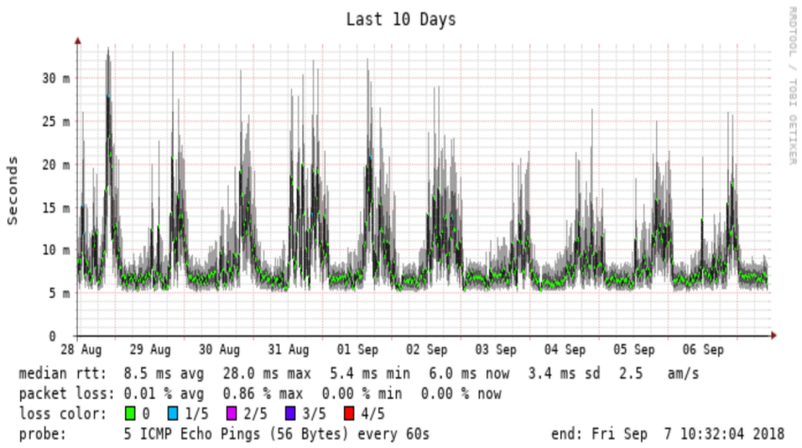
```
ServerName web-server
<VirtualHost 192.0.2.1:80>
        DocumentRoot /var/www/html/ipv4
        Allow from 192.168.0.0/16
</VirtualHost>

<VirtualHost [2001:db8::1]:80>
        DocumentRoot /var/www/html/ipv6
        Allow from 2001:db8::/32
</VirtualHost>
```

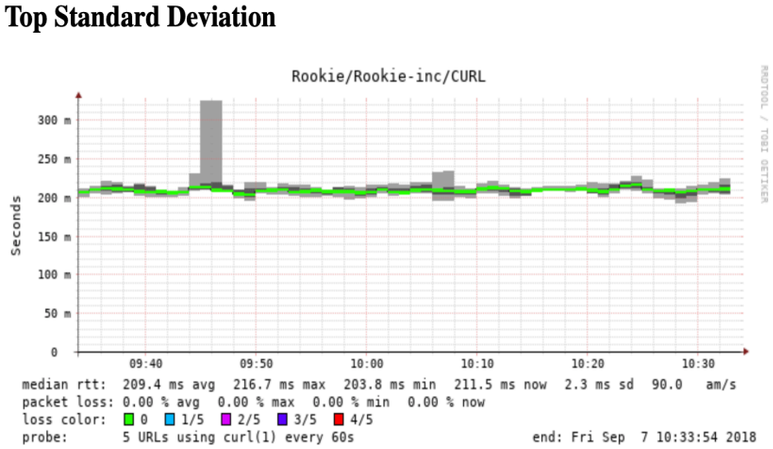
NGINX

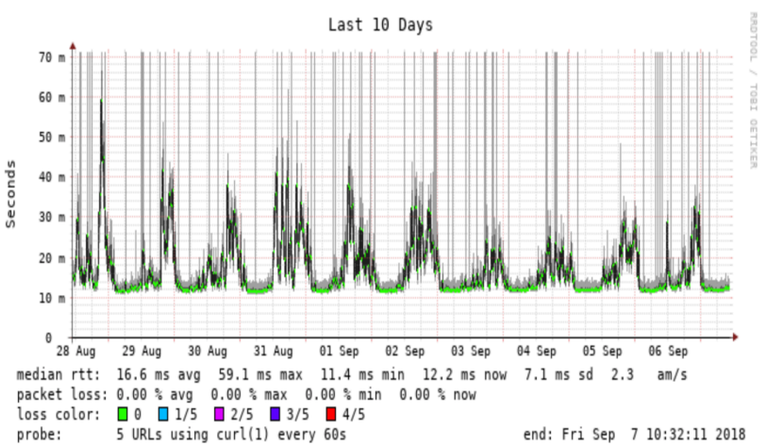
```
server {
  listen 192.0.2.1:80;
  server_name web-server;
  allow 192.168.0.0/16;
  location / {
    root /var/www/html/ipv4;
  }
}

server {
  listen [2001:db8::1]:80;
  server_name web-server;
  allow 2001:db8::/32;
  location / {
    root /var/www/html/ipv6;
  }
}
```

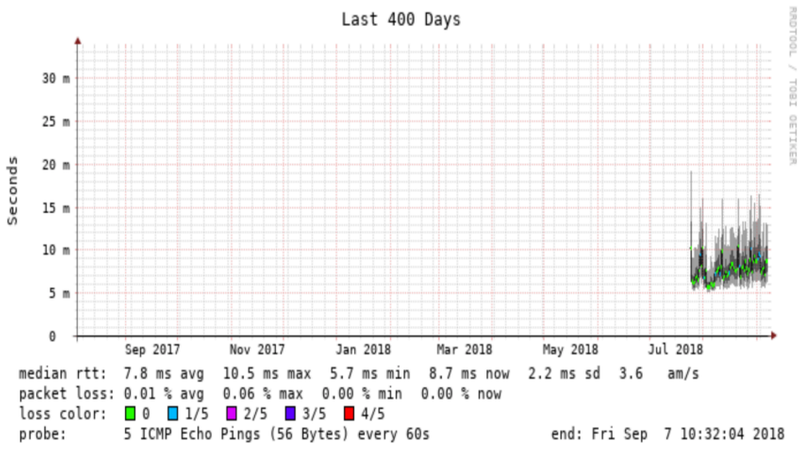
Smokeping

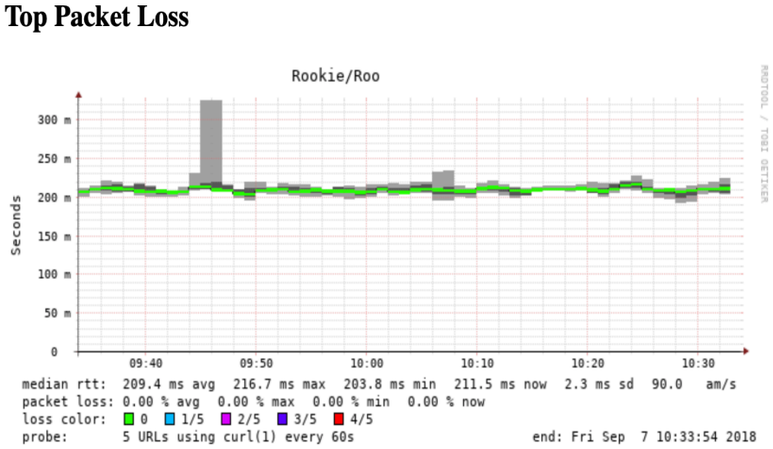
- 外部からのMonitoringに便利なツールの一つ
- 比較的簡単にIPv6対応できる

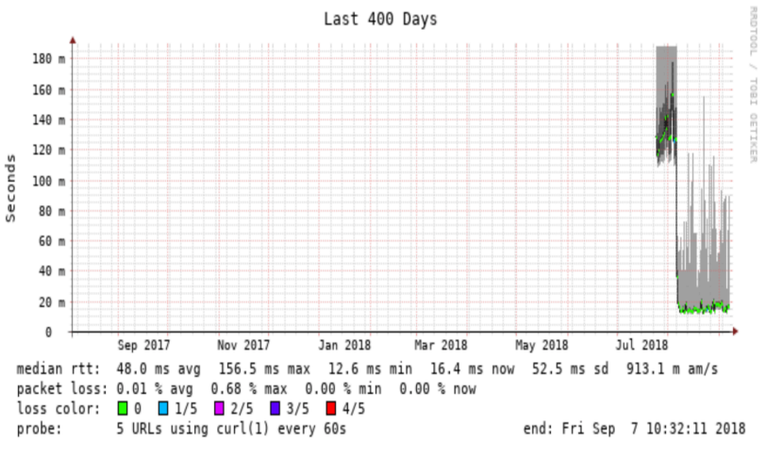












ここ3時間、30時間、10日、400日のICMPやHTTPのRTT、平均、標準偏差などが取得できます。

SmokepingのIPv6対応

- Probeにfping6を指定
- 監視対象のProveに必要に応じて
  fping6を登録する

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

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をFpingにするの場合

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

セキュリティ

- 境界防御だけでなく、Nodeでの防御も視野に入れなければならない
  - Nodeでの防御で、追加のToolを必要としない手法として Packet Filterと SELinux(Linuxの場合)があるが、ここでは、Packet Filterのみ採り上げる
- 境界防御で大まかにFilterし、Node側で、完全に絞るという構成も見かけるようになってきている

サーバーにおける対策

- サーバーにおける防護策
  - Host型IDSの投入(OSSEC HIDSやtripwireなどの変更検出ツールなど)
  - システムの脆弱性診断(niktoなどのFreewareやSecurity対策事業者のサービス)
  - 定期的なPatch update
  - できるだけ更新可能な基盤の利用(使用言語やMiddlewareなど)
  - 不要なポートを閉じるなど、接続点を減らす(pf/iptables/firewalld等)

いわゆる「できて当然」の対策の徹底

- 単純でもいいから「定期的に行われる」logの分析(hekadなど)
- 「Hi-port」での待ち受け(sshを822/TCPで待ち受けるなど)
- Virus Checkerなどのデータチェック(clamAVなど)

いわゆる「一工夫で安全にする」の対策の実施

Distribution毎のFirewall

- CentOS
  - CentOS6まで：iptables/ip6tables
  - CentOS7から:firewalld
- Debian/Ubuntu
  - Debian 8/Ubuntu6.04LTSまで: iptables/ip6tables
  - Debian 9/Ubuntu8.04LTSから: UFW
- SUSE
  - SUSEfirewall2
- これらは全て、iptables/ip6tables(in-kernel)へのI/Fであって、本質的にはWrapperである
- FreeBSD
  - pf / ipfw

Added

パケットフィルタの注意点

- IPv4と違い、IPv6ではICMPをすべて止めてしまうと、通信ができなくなる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）

ICMPとICMP6

Number	ICMP (IPv4)	Number	ICMP6 (IPv6)
0	Echo Reply	129	Echo Reply
3	Destination Unreachable	1	Destination Unreachable
4	Source Quench		
5	Redirect	137	Redirect
8	Echo Request	128	Echo Request
9	Router Advertisement	134	Router Advertisement
10	Router Solicitation	133	Router Solicitation
11	Time Exceed	3	Time Exceed
12	Parameter Problem	4	Parameter Problem
13	Timestamp	2	Packet too Big
14	Timestamp Reply	130	Multicast Listener Query
15	Information Request	131	Multicast Listener Report
16	Information Reply	132	Multicast Listener Done
17	Address Mask Request	135	Neighbor Solicitation
18	Address Mask Reply	136	Neighbor Advertisement
		138	Router Renumber
赤字はErrorメッセージ IPv4とIPv6で同じ意味を持つMessageはオレンジ IPv4のみのMessageは黄緑 IPv6のみのMessage水色		139	Node Information Query
		140	Node Information Reply
		141	Inverse Neighbor Discovery Solicitation
		142	Inverse Neighbor Discovery Advertisement
		143	MLD2 Multicast Listener Report
		144	Home Agent Address Discovery Request
		145	Home Agent Address Discovery Reply
		146	Mobile Prefix Solicitation
		147	Mobile Prefix Advertisement
		148	Certification Path Solicitation
		149	Certification Path Advertisement)

ICMP6をどうフィルターするか?

- 取りうる解決策は２つ
  - ICMP6を何も考えず全部通す　→　Security的には問題もある
  - ICMP6を外部との通信に必要な物だけ通す　→　Filterが大きくなる

- ここでは「２」を取り上げる
  - 1は結局Filterを書かないということなので、検討する必要がない

- ICMP6を「対外通信・内部通信」と「必須・オプション」に分けて考える

ICMP6をどうフィルターするか?

Message	対外：必須	対外：Option	対外：不要	LL：必須	LL：Option	LL：不要
Destination Unreachable	◎			◎		
Packet too Big	◎			◎		
Time Exceed	◎			◎		
Parameter Problem	◎			◎		
Echo Request		◎			◎	
Echo Reply		◎			◎	
Multicast Listener Query			◎	◎		
Multicast Listener Report			◎	◎		
Multicast Listener Done			◎	◎		
Router Solicitation			◎	◎		
Router Advertisement			◎	◎		
Neighbor Solicitation			◎	◎		
Neighbor Advertisement			◎	◎		
Redirect			◎		◎	
Router Renumber			◎		◎	
Node Information Query		◎			◎	
Node Information Reply		◎			◎	
Inverse Neighbor Discovery Solicitation			◎	◎		
Inverse Neighbor Discovery Advertisement			◎	◎		
MLD2 Multicast Listener Report			◎	◎		
Home Agent Address Discovery Request		◎				◎
Home Agent Address Discovery Reply		◎				◎
Mobile Prefix Solicitation		◎				◎
Mobile Prefix Advertisement		◎				◎
Certification Path Solicitation		◎			◎	
Certification Path Advertisement)		◎			◎	

ip6tablesの例

- Serverであることが想定されるので、LinkLocalの表を元に作成
  - Mobile IPv6のHome Agentではない事を想定

```
# Approve certain ICMPv6 types and all outgoing ICMPv6
-A INPUT -p icmpv6 -j ICMPv6
-A ICMPv6 -p icmpv6 --icmpv6-type destination-unreachable -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type time-exceeded -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type parameter-problem -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type echo-request -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type echo-reply -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 130 -s fe80::/10 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 131 -s fe80::/10 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 132 -s fe80::/10 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type router-solicitation -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type router-advertisement -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type neighbour-solicitation -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type neighbour-advertisement -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type redirect -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 139 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 140 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 141 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 142 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 143 -s fe80::/10 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 151 -s fe80::/10 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 152 -s fe80::/10 -j ACCEPT
-A ICMPv6 -p icmpv6 --icmpv6-type 153 -s fe80::/10 -j ACCEPT
-A ICMPv6 -j RETURN
-A OUTPUT -p icmpv6 -j ACCEPT
```

ちょっとしたTips

- DNS 逆引き
- 障害の切り分けとFQDN
- 複数のProtocolと認証
- 複数のProtocolを利用する場合の問題

DNS逆引きについて

- IPv4と違い、一般ノードのIPv6アドレスは逆引きの設定が困難
  - DDNS等の利用は考えられるが、事実上逆引き不能と考えるべき
  - Privacy ExtensionによるアドレスのRandomizeがなされている可能性も高い

- 逆引きに依存する認証などは事実上使えない
  - DNSの逆引きによるACLなど

- アドレスが固定されているものに関しては逆引きを定義すべき
  - SMTP Serverなど

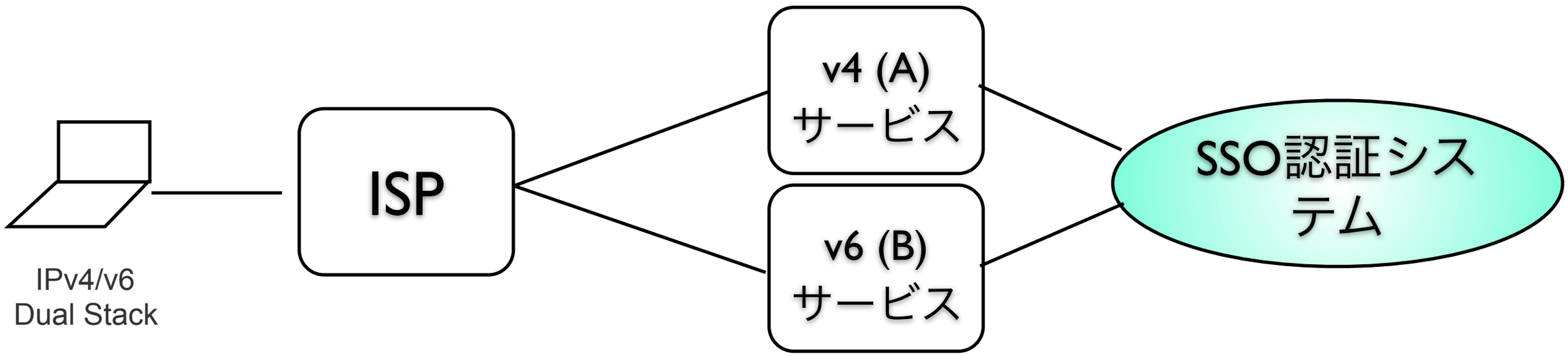
障害の切り分け（FQDN）

- Dual Stack環境では、FQDNの取り扱いに注意が必要
  - 監視ツールの設定においては、監視対象ノードの設定をIPv6/IPv4アドレスで行うべき
  - FQDNでノードを登録する場合、IPv6/IPv4のフォールバックに注意を払う必要がある

- シナリオ
  - www.example.comをcurlでmonitorしているとする
  - curlは毎回 www.example.comを名前解決
  - 取得した情報を元にcurlがアクセス
  - この時、Fallbackによって、正しく望んだ情報が得られない可能性がある
    - IPv6で通信できなくてもIPv4にFallbackすることで正常と判定されるなど

複数のProtocolを利用する

- 特にSSO(Single Sign On)における認証の問題
  - IPv4とIPv6が両方使える端末の場合に、認証情報をどう取り扱うのか？
- 例
  - 環境
    - 利用者端末がDual Stack
    - あるサービス(A)はv4でのみサービス提供、別のサービス(B)はv6でのみサービス提供を実施
    - このAとBでSSOを行う場合を考える
  - 利用者端末はAで認証を行ったとした場合、Bを利用する際にAで行った認証情報をどのように共有すればいいのか？
    - 通常の認証システムは、端末のIPアドレスを利用してCookieで処理するので、Bのサービスを利用する場合に、認証情報が共有できない＝再認証が必要＝SSOの意味が無い



複数のProtocolを利用する

- Application側でのIPv6対応の問題
  - WEB Applicationの実装次第では、IPv6に対応できない（できてもCosty）な場合も...
    - SocketのIPv6対応はまだなんとかなるのだが...IPアドレスを処理しなければならないようなプログラムは対応できない可能性がある
    - 特にCustom Applicationの場合、IPv6対応させるには、開発会社を利用して再度IPv6対応のコストを掛けなければならない場合がある
- logの取り扱い
  - IPv6アドレスの記録方式の問題
    - 機器によって、log等のIPv6アドレス出力がばらばらになる可能性がある
    - Security 機器のlogを見る場合には省略記法で出力されたlogはみにくい
      - logをIP Addressでソートする場合を考えると、省略記法はむしろ不便
  - 障害発生時の連絡の問題
    - 相手に確実に障害発生点を伝える方法(IPv6アドレスを読み上げても機器特定が難しい)
    - ICMP6 Node information Queryを利用できるなら便利なのだが...
      - 一応USAGIには実装されている模様。

おわりに

- ここで説明した内容は、調べればわかること
- しかし、実際に手を動かしてみないと、どうしても実感は得られないことが多い
- 現実には、実際に手を動かすためのテスト環境が存在することは（日本においてはまだまだ）まれと言わざるを得ない
- 自分で手を動かして、やってみたい方は、是非JPNICのセミナーをご検討ください

ご静聴　ありがとうございました