livedoor®

IPv6 対応 Web アプリケーション開発作法

株式会社ライブドア 開発本部 谷口公一

Dec 1st, 2011 - Internet Week 2011

自己紹介

- 氏名: 谷口公一 (タニグチ コウイチ)
- HN: にぽたん
- 職業: Web アプリケーションエンジニア
- 所属: 株式会社ライブドア

→ 2012-01-01 に NHN Japan 株式会社に

- Twitter: @nipotan
- Facebook: http://facebook.com/nipotan
- Web: http://nipotan.com/
- 主な使用言語: Perl

アジェンダ

- 私と IPv6
- IPv4 アドレス枯渇
- IPv6 のキホン
- Web アプリケーションの IPv6 対応

Dec 1st, 2011 - Internet Week 2011

私と IPv6

Dec 1st, 2011 - Internet Week 2011

私と IPv6

- 「IPv6 とかよくわからない人間が IPv6 対応サイトを作る際の知っておくべき 8 つの注意点」
  - http://blog.livedoor.jp/nipotan/archives/51195204.html

IPv6 とかよくわからない人間が IPv6 対応サイトを作る際の知っておくべき 8 つの注意点 - にぽたん研究所 blog.livedoor.jp:nipotan

タグ : ipv6 network ネットワーク Apache Server web Tips あとで読む サーバ IP

コンピュータ・IT 596 users B! 90 clicks 1 RT ★20★

先日、一般や企業向けに IPv6 対応を支援をする、EDGE Co.Lab v6 というのを始めました。 これを始めるにあたって、弊社情報環境技術研究室の伊勢さんから、「なんかウチでやってるコンテンツで、どれか IPv6 対応しようよ」と、いきなり言われました。 実は IPv6 って何年も前からよく耳にするけど、特にインフラまわりの知識が拙いし、何だかんだ身の回りのほとんどが IPv4 で、それでまぁウマくいってるからよくわからないし、別にどうでもいい...と、IPv6 に対して「現実味がない。時期尚早なので... > このページを見る

▼ブログで紹介する

最終更新時間: 2008年12月15日20時21分

私と IPv6

- こんな釣り針を持ってる私は...
  - ネットワーク、特に IPv6 に関する知識
    - 実は殆どありません orz
- IPv6 に触れたきっかけ
  - 2008 年 12 月、livedoor が IPv6 実証実験環境提供を開始

    「EDGE Co.Lab v6」 http://labs.edge.jp/colabv6/
    - 提供を開始するにあたりモデルケースがない
      - 「IPv6 よくわからないけどやってみるか！」

livedoor と IPv6

- IPv6@2ch 掲示板

  → http://ipv6.2ch.net/
  - IPv6 経由の場合しか書き込みが行なえない
- fixdap

  → http://fixdap.com/
  - EDGE Co.Lab v6 のモデルケースとして実装
  - IPv6 経由の場合、ロゴに IPv6 マークが付く
- IRCnet 向け IRC サーバ
  - WIDE プロジェクトの IRCnet 撤退を受け

IPv6 対応中の思い出

- 今のようにネット上に殆ど情報がない
  - IPv6 で検索するとネットワークエンジニア向けの情報ばかり
- アプリケーションの実装についてのポイントが書かれてるサイトが殆どない
- 全てが手探り
- きっと今後皆が苦労するだろう
  - 対応したポイントをブログに書いた
    - 釣れた！

こんなブログを書いたおかげで

- APNIC 27 (Fab 2009 - Manila, Philippnes)
  - http://meetings.apnic.net/27/program/ipv6-in-3d
  - http://slidesha.re/nipotan-apnic27

Dec 1st, 2011 - Internet Week 2011

ブログ投稿から約 3 年...

- APNIC 27 から 2 年 9 ヶ月
- もっと詳細に
- もっと具体的に
- あとで気付いたこともたくさん
- 言語や環境にあまり依存しない情報

アプリケーションエンジニア vs IPv6

- Web アプリ側で何か対応が必要なの？
- ネットワークだけの問題じゃないの？
- 俺等 L7 テキストプロトコルがメインだよ？
  - IP (Internet Protocol) とか知らなくて良くね？

→あながち間違いじゃない

IPv6 未対応アプリでも…

- サーバソフトウェアが IPv6 Ready!
- 環境がデュアルスタックで IPv6 Ready!
  - IP アドレスでのアクセスは大体 OK
  - ホスト名でのアクセスは？
    - 名前解決不可
- 数々の「些細な問題点」
  - やりたいことが限られる
  - その他運用上の問題が起こり得る

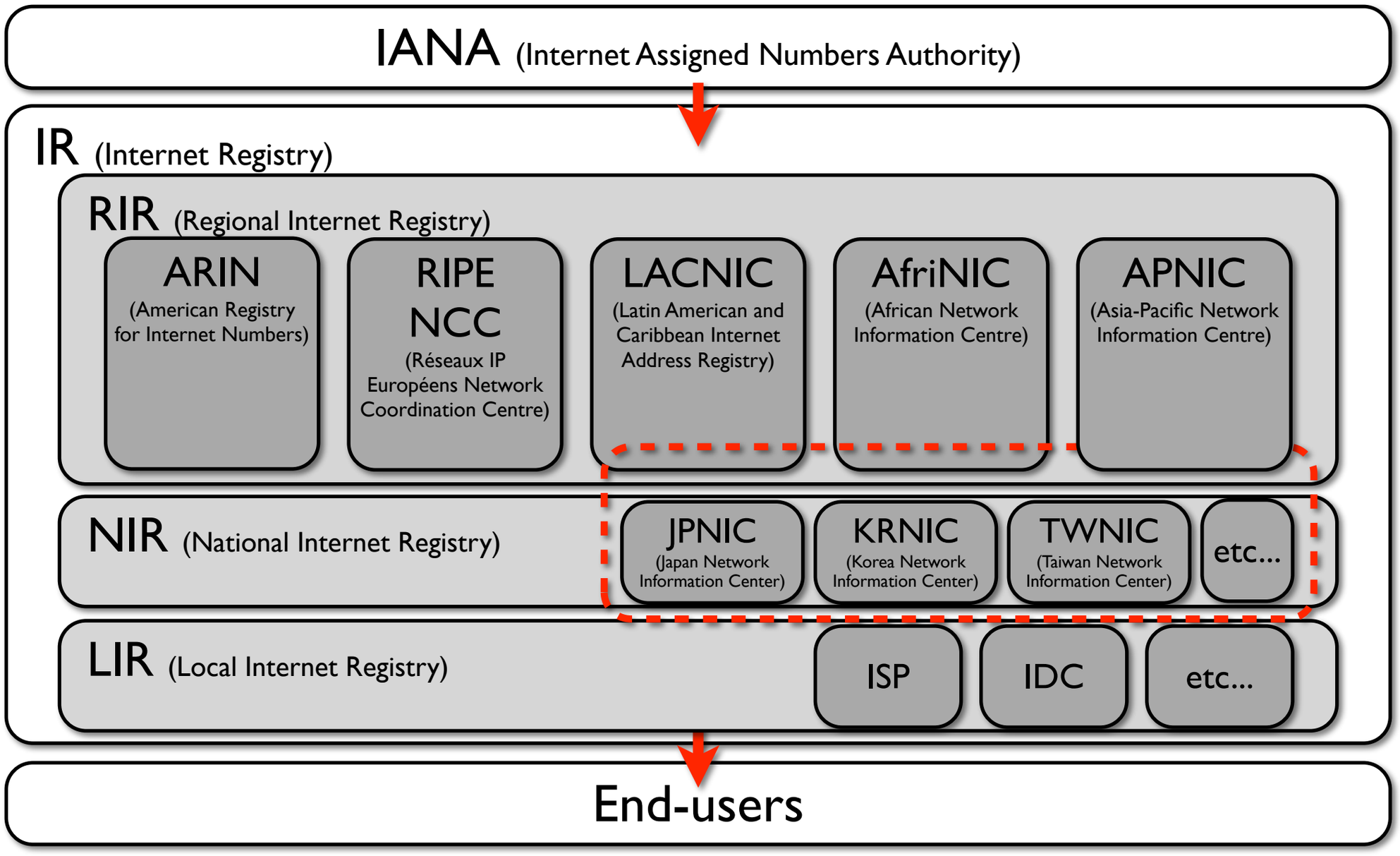
IPv4 アドレス枯渇

Dec 1st, 2011 - Internet Week 2011

IPv4 アドレスとは？

- 32bit
- 約 43 億弱個のアドレス
  - 4,294,967,296 個
- 地球の人口より少ない
- 多くの組織を介した割当

IPv4 アドレス割当の流れ



日本の IPv4 アドレス枯渇の流れ

- 2011-01-31 IANA 在庫枯渇

  → RIR 保有分の在庫消化へ

- 2011-04-15 APNIC 在庫枯渇

  → NIR 保有分の在庫消化へ

- 2011-04-15 JPNIC 在庫枯渇
  - JPNIC は在庫管理をせず APNIC と共有

    → LIR 保有分の在庫消化へ

現在、今後の対策

- 大手国内 LIR の多くはまだ在庫を保有
- 在庫をあまり持たない ISP の対応
  - ラージスケール NAT (ISP Shared Address)
  - プライベートアドレス
- クラウド事業者が逼迫との噂
- LIR 間での融通、売買
  - 短命な措置
- IPv6 化
  - 未来はそこにしかない

LIR の IPv4 在庫枯渇の影響

- コンシューマ
  - 自宅サーバ公開が不可能
  - IPv6 だけが割当てられるようになる
    - インターネット利用がほぼ不可能
- サービス提供事業者
  - 新規サービス公開が不可能
  - サーバの増設が出来なくなる
  - IPv6 だけが割当てられてたユーザとの別れ

何故 IPv6 に対応しない？

- 対応すべき箇所がわからない
- ○○が IPv6 未対応
  - ルータ
  - ロードバランサー
  - IDC
  - 接続回線
  - ゲートウェイ
- だから IPv6 化が進まない

IPv6 時代の IPv4

- 「デュアルスタックは過渡的技術」？
  - IPv4 が廃止されるわけではない
  - 全ミドルウェアが IPv6 に対応？
  - IPv4 (NAT) のネットワーク構成を捨てろ？
    - 捨てないことによるデメリットは？
    - 捨てるモチベーションは？
- 全世界の全ハード/ソフトが IPv6 対応したら？

IPv6 のキホン

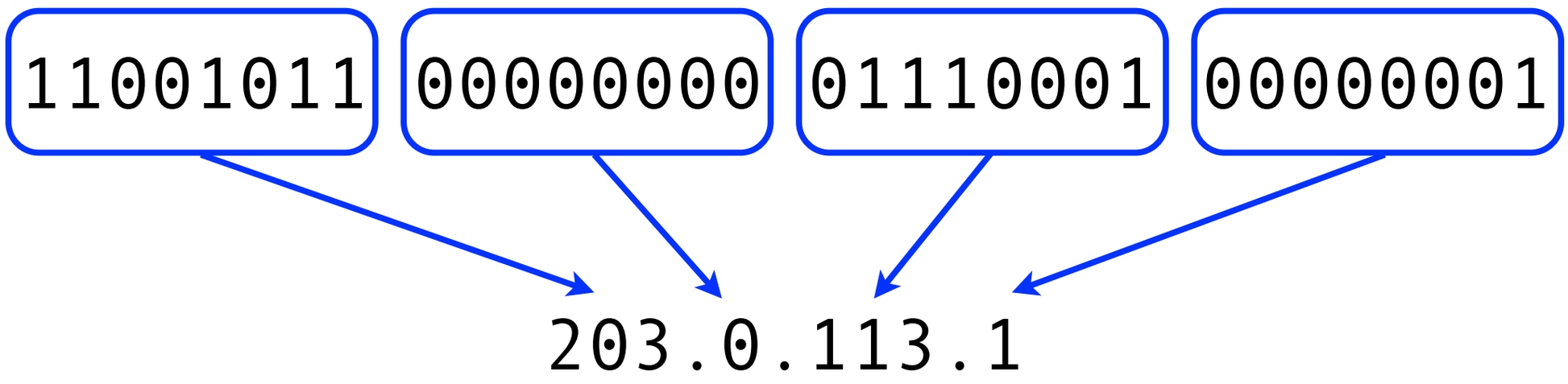
Dec 1st, 2011 - Internet Week 2011

IPv6 アドレスとは？

- 128bit
- 約 340 澗個のアドレス
  - 340,282,366,920,938,463,374,607,431,768,211,456 個
  - IPv4 アドレスの約 8 壌倍
- NAT とか要らない

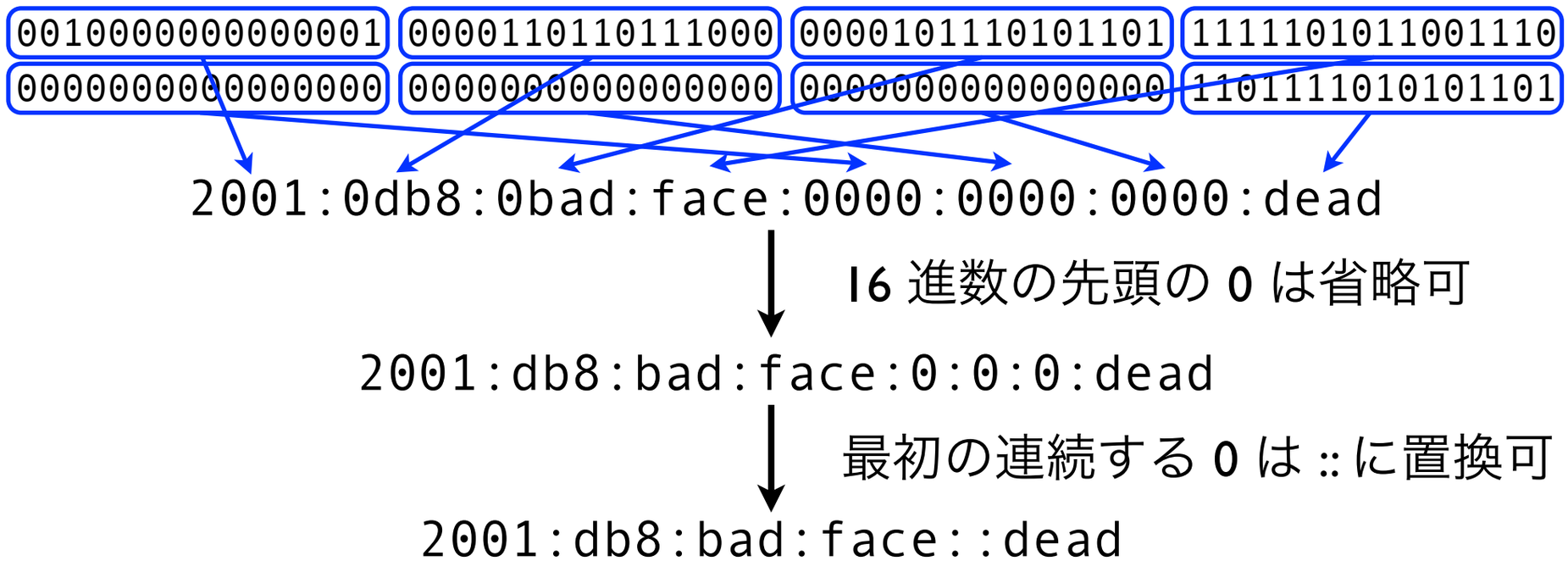
IP アドレスの表現

- IPv4 アドレス
  - 32bit を 8bit ごとに区切り、10 進数で表記
  - 区切り文字は .(ドット)



IP アドレスの表現

- IPv6 アドレス
  - 128bit を 16bit ごとに区切り、16 進数で表記
  - 区切り文字は : (コロン)



余談: 文書に使われる IP アドレス

- 文書用でインターネットには使われない IP アドレス
  - ドメインで言うところの example.com 的存在
- IPv4
  - RFC 5735 - Special Use IPv4 Addresses
    - 192.0.2.0/24 #TEST-NET-1
    - 198.51.100.0/24 # TEST-NET-2
    - 203.0.113.0/24 # TEST-NET-3
- IPv6
  - RFC 3849 - IPv6 Address Prefix Reserved for Documentation
    - 2001:db8::/32

特殊な IPv6 アドレス

- IPv4 互換アドレス
  - 上位 96bit は 0、末尾 32bit は IPv4 アドレス
  - IPv4 アドレスはドット区切り 10 進数表記が可
    - ::203.0.113.1
- IPv4 射影アドレス
  - 上位 80bit は 0、16bit は 0xffff、末尾 32 bit は IPv4 アドレス
    - ::ffff:203.0.113.1

IP アドレスとポート番号の併記方法

- IPv4 URL の場合

IP アドレス　　ポート番号

```
http://203.0.113.1:8080/
```

IP アドレスとポート番号の区切りはコロン

IP アドレスとポート番号の併記方法

- IPv6 URL の場合

IP アドレス？　　ポート番号？

```
http://2001:db8:bad:face::dead:8080/
```

RFC 3989
Uniform Resource Identifier (URI):
Generic Syntax

```
http://[2001:db8:bad:face::dead]:8080/
```

IPv6 アドレス部分 (IP-literal) は [と] で挟む

IP アドレスとポート番号の併記方法

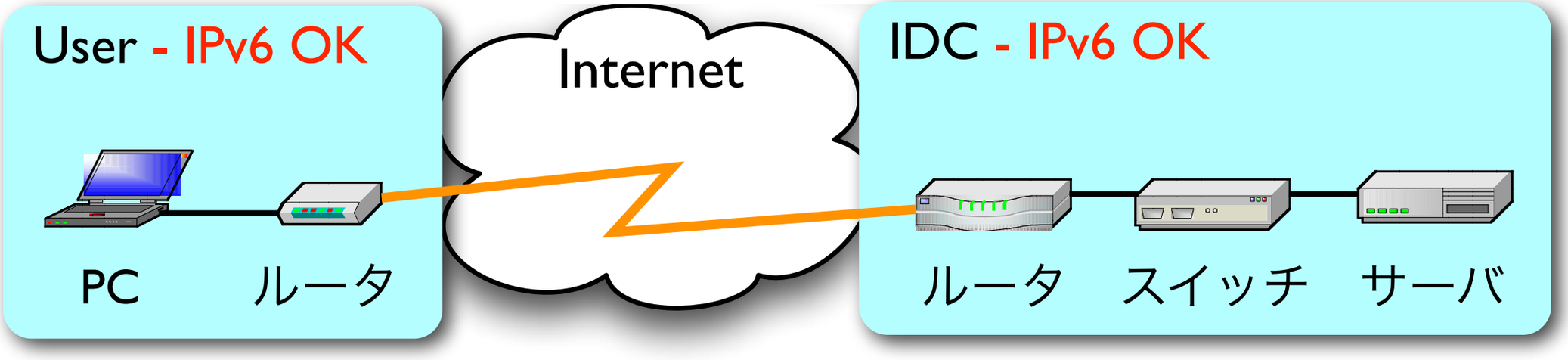
- IPv6 で URL 以外の場合
- 様々な併記方法がある
  - RFC 5952 - A Recommendation for IPv6 Address Text Representation

Web アプリケーションの IPv6 対応

Dec 1st, 2011 - Internet Week 2011

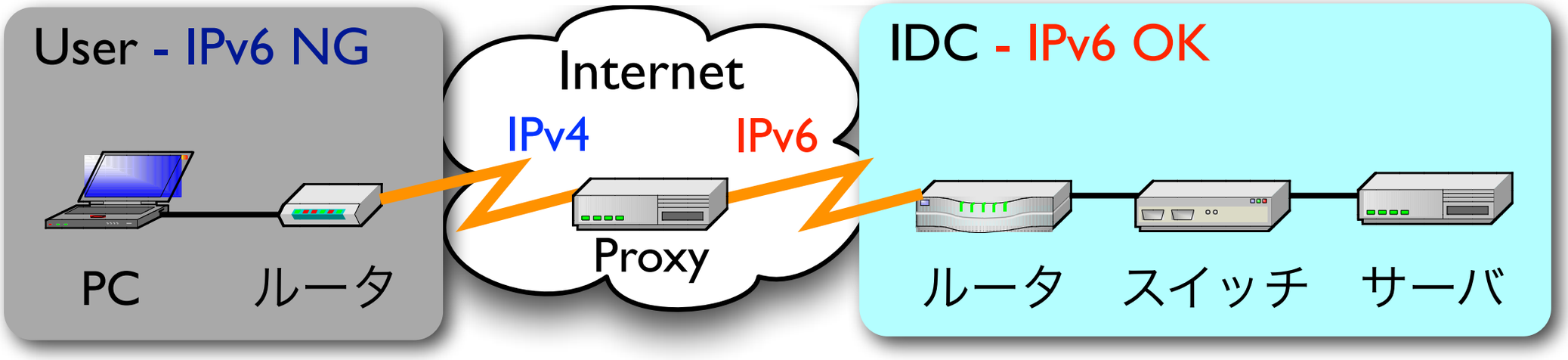
接続環境の確認

- ユーザの PC からサーバまでの機器、回線が IPv6 に対応していないといけない



接続環境の確認

- ユーザ側が IPv6 未対応の場合
  - デュアルスタックの Proxy を挟むことで IPv6 の利用が可能



接続環境の確認

- 主要 HTTP Proxy の IPv6 対応状況

HTTP Proxy	IPv6 対応
Squid	✓
Apache + mod_proxy	✓
DeleGate	✓

SSL サーバ ID (証明書) の申請

- 鍵交換はセッション層
- 証明書の必要数
  - コモンネーム数、台数から算出
  - デュアルスタックでも 1 つで良い
- IP のバージョンに依存せず
  - IPv4 用 / IPv6 用というのはない
  - 共通で使用可能

IPv6 アドレスの DNS 登録

- IPv6 の名前解決は AAAA (クアッド A) レコード
- DNS サーバの対応必須
- EDNS0 対応必須
  - パケットのデータ長制限 (512 Bytes) を拡張
- IPv6 トランスポート
  - IPv6 経由の DNS クエリに応答

IPv6 アドレスの DNS 登録

- 主要 DNS サーバの IPv6 対応状況

DNS Server	AAAA	EDNS0	IPv6 transport
BIND	✓	✓	✓
NSD	✓	✓	✓
djbdns	✓	✗	✗ ✓ * 3rd party patch

MTA (メール) の IPv6 対応

- メールのハンドリングを行なう場合
  - SMTP の IPv6 対応
    - 受信するメール
    - 送信するメール
- MX レコードのホストにも AAAA 設定を

```
% host -t MX example.com
example.com mail is handled by 0 mail.example.com.
% host -t AAAA mail.example.com
mail.example.com has IPv6 address 2001:db8::c00:ffee
```

MTA (メール) の IPv6 対応

- 主要 MTA の IPv6 対応状況

MTA	IPv6 対応
sendmail	✓
postfix	✓
qmail	✗ ✓ * 3rd party patch

疎通確認の方法

- サーバ、クライアントともに IPv6 の疎通確認が必要
- 疎通確認、診断には ICMPv6
  - ICMP (IPv4) と非互換
- ping、traceroute は IPv6 では使えない
- ping6、traceroute6 を使用
- Windows 環境は ping6、tracert6

```
% ping6 2001:db8:bad:face::dead
% traceroute6 2001:db8:bad:face::dead
```

```
C:\WINDOWS>tracert6 2001:db8:bad:face::dead
```

疎通確認の方法

- AAAA 名前解決、疎通までをテストする場合
  - ipv6.google.com が最適
    - IPv6 のみでしか公開されていないサイトは稀少

```
% ping6 ipv6.google.com
% traceroute6 ipv6.google.com
% curl -Iv http://ipv6.google.com/
```

Web サーバの設定

- IPv4 と IPv6 で、コンテンツの差別化を行なう場合
  - IP-base のヴァーチャルホストを構築
  - 例
    - The KAME Project
      → http://www.kame.net/
      - IPv6 経由の場合、トップの亀が踊る
    - IPv6@2ch 掲示板
      → http://ipv6.2ch.net/
      - IPv6 経由の場合、トップのひろゆきが踊る

Web サーバの設定

- Apache の場合
  - 1.3.x は IPv6 未対応
  - 2.0 以降で Apache Portable Runtime (APR) が標準で IPv6 サポート

```
Listen [2001:db8::bad:face]:80
Listen 203.0.113.1:80

<VirtualHost [2001:db8::bad:face]:80>
  # IPv6 settings
    :
    :
</VirtualHost>

</VirtualHost 203.0.113.1:80>
  # IPv4 settings
    :
    :
</VirtualHost>
```

Web サーバの設定

- lighttpd の場合
  - 恐らく初期リリースから IPv6 サポート
  - 主要 OS Web Server の中で唯一 IPv6 対応サイト

```
#server.use-ipv6 = "enable"

server.port = 80
server.bind = "203.0.113.1"

$SERVER["socket"] == "[2001:db8::bad:face]:80" {
  # IPv6 settings
    :
}
$SERVER["socket"] == "203.0.113.1:80" {
  # IPv4 settings
    :
}
```

Web サーバの設定

- nginx の場合
  - 0.7.36 以降から徐々に IPv6 対応が進んでいる
    - …ロシア語の情報ばかりなので定かじゃない

```
http {
  server {
    listen [2001:db8::bad:face]:80
    server_name bad-face.example.com
    # IPv6 settings
      :
  }
  server {
    listen 127.0.0.1:80;
    listen 203.0.113.1:80;
    server_name bad-face.example.com;
    # IPv4 settings
      :
  }
}
```

IPv4/IPv6 どちら経由か判別する

- コンテンツを差別化する場合、Web アプリケーションで行なう
  - アプリケーションサーバが Proxy されてない環境
    - 接続元の IP アドレスから判別

Perl

```
$ENV{REMOTE_ADDR}
```

PHP

```
$_SERVER["REMOTE_ADDR"]
```

Ruby

```
ENV["REMOTE_ADDR"]
```

Python

```
os.environ.get("REMOTE_ADDR")
```

Java

```
request.getRemoteAddr() //javax.servlet.http.HttpServletRequest#getRemoteAddr()
```

IPv4/IPv6 どちら経由か判別する

- IP アドレスからの判断
  - 例えば、正規表現を使う

IPv6 アドレスにマッチする正規表現

```
^(?:(?:(?:(?:(?:[\da-f]){1,4}:){6}|::(?:(?:(?:[\da-f])
{1,4}:){5}|(?:[\da-f]){0,4}::(?:(?:[\da-f]){1,4}
:){4}|(?:(?:(?:[\da-f]){1,4}:){0,1}(?:[\da-f]){1
,4})?::(?:(?:[\da-f]){1,4}:){3}|(?:(?:(?:(?:[\da-f]
){1,4}:){0,2}(?:[\da-f]){1,4})?::(?:(?:(?:[\da-f]){
1,4}:){2}|(?:(?:(?:(?:[\da-f]){1,4}:){0,3}(?:[\da-f
]){1,4})?::(?:(?:[\da-f]){1,4}:)|(?:(?:(?:(?:[\da-f
]){1,4}:){0,4}(?:[\da-f]){1,4})?::)(?:(?:(?:[\da-f]
){1,4}:(?:[\da-f]){1,4}|(?:(?:\d|[1-9]\d|1\d\d|2
(?:[0-4]\d|5[0-5])).){3}(?:(?:\d|[1-9]\d|1\d\d|2
(?:[0-4]\d|5[0-5])))|(?:(?:(?:(?:[\da-f]){1,4}:){0
,5}(?:[\da-f]){1,4})?::(?:(?:[\da-f]){1,4}|(?:(?:(?
:[\da-f]){1,4}:){0,6}(?:[\da-f]){1,4})?::)$
```

- そこまで厳密にやらなくても...
  - コロンが入ってたら IPv6 として扱う等

IPv4/IPv6 どちら経由か判別する

- reverse proxy 環境下での判別
  - 一般的な proxy は **`X-Forwarded-For`** ヘッダに接続元 IP が付与される
  - 多段 proxy の場合
    - 追記される
    - リクエストヘッダの操作が可能である以上、値に対する信頼性は保てない
- reverse proxy 側で別のヘッダを付与する

IPv4/IPv6 どちら経由か判別する

- 例えば reverse proxy 側で **`X-IP-Version`** というヘッダを追加 (4 or 6)
  - アプリケーション側でその値を見る

Perl

```
$ENV{HTTP_X_IP_VERSION}
```

PHP

```
$_SERVER["HTTP_X_IP_VERSION"]
```

Ruby

```
ENV["HTTP_X_IP_VERSION"]
```

Python

```
os.environ.get("HTTP_X_IP_VERSION")
```

Java

```
request.getHeader("X-IP-Version") //javax.servlet.http.HttpServletRequest#getHeader()
```

IPv4/IPv6 どちら経由か判別する

- Apache の場合
  - RequestHeader で設定

```
Listen 80

<VirtualHost [2001:db8::bad:face]:80>
  ServerName bad-face.example.com
  RewriteEngine On
    :
    :
  RequestHeader set X-IP-Version 6
</VirtualHost>

</VirtualHost 203.0.113.1:80>
  ServerName bad-face.example.com
  RewriteEngine On
    :
    :
  RequestHeader set X-IP-Version 4
</VirtualHost>
```

IPv4/IPv6 どちら経由か判別する

- lighttpd の場合
  - setenv.add-request-header で設定

```
$SERVER["socket"] == "[2001:db8::bad:face]:80" {
    :
    :
  setenv.add-request-header = ("X-IP-Version" => "6")
}
$SERVER["socket"] == "203.0.113.1:80" {
    :
    :
  setenv.add-request-header = ("X-IP-Version" => "4")
}
```

IPv4/IPv6 どちら経由か判別する

- nginx の場合
  - proxy_set_header で設定

```
http {
  server {
    listen [2001:db8::bad:face]:80
    server_name bad-face.example.com
      :
      :
    proxy_set_header X-IP-Version 6;
  }
  server {
    listen 127.0.0.1:80;
    listen 203.0.113.1:80;
    server_name bad-face.example.com;
      :
      :
    proxy_set_header X-IP-Version 4;
  }
}
```

接続元 IP アドレスの保存

- ユーザの IP アドレスを保存するケース
  - DB 上のカラムのサイズが足りているか
    - バイナリデータとして保存
      - IPv4 … 32bit → 4 バイト
      - IPv6 … 128bit → 16 バイト (?)
        - 可読データではない
          - 保守、運用に向くか？

接続元 IP アドレスの保存

- 例えば、MySQL 上に IPv4 アドレスをバイナリデータとして保存、運用する
  - テーブル定義

```
mysql> CREATE TABLE ip_list (
    ->   id int(10) unsigned NOT NULL AUTO_INCREMENT,
    ->   remote_addr binary(4),
    ->   PRIMARY KEY (id)
    -> ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.00 sec)
```

接続元 IP アドレスの保存

- データが記録された後、クライアント上からデータ内容の確認

```
mysql> SELECT * FROM ip_list;
+----+-------------+
| id | remote_addr |
+----+-------------+
|  1 | ? q?        |
|  2 | ? S         |
|  3 | ?3d         |
+----+-------------+
3 rows in set (0.00 sec)
```

接続元 IP アドレスの保存

- 記録されているデータが正しいか、とりあえずデータ長を確認

```
mysql> SELECT id,LENGTH(remote_addr) FROM ip_list;
+----+----------------------+
| id | LENGTH(remote_addr)  |
+----+----------------------+
|  1 |                    4 |
|  2 |                    4 |
|  3 |                    4 |
+----+----------------------+
3 rows in set (0.00 sec)
```

接続元 IP アドレスの保存

- 記録されているデータを、化けないように 16 進数表現にして内容を見てみる

```
mysql> SELECT id,HEX(remote_addr) FROM ip_list;
+----+------------------+
| id | HEX(remote_addr) |
+----+------------------+
|  1 | CB0071A9         |
|  2 | C0000253         |
|  3 | C6336414         |
+----+------------------+
3 rows in set (0.00 sec)
```

接続元 IP アドレスの保存

- 16 進数表現を 10 進数表現に変換して…

```
mysql> SELECT id,CONV(HEX(remote_addr),16,10) FROM ip_list;
+----+-------------------------------+
| id | CONV(HEX(remote_addr),16,10)  |
+----+-------------------------------+
|  1 | 3405803945                    |
|  2 | 3221226067                    |
|  3 | 3325256724                    |
+----+-------------------------------+
3 rows in set (0.00 sec)
```

接続元 IP アドレスの保存

- 10 進数の数値を IPv4 アドレス文字列に変換すると…

```
mysql> SELECT id,INET_NTOA(CONV(HEX(remote_addr),16,10)) FROM ip_list;
+----+-----------------------------------------+
| id | INET_NTOA(CONV(HEX(remote_addr),16,10)) |
+----+-----------------------------------------+
|  1 | 203.0.113.169                           |
|  2 | 192.0.2.83                              |
|  3 | 198.51.100.20                           |
+----+-----------------------------------------+
3 rows in set (0.00 sec)
```

接続元 IP アドレスの保存

- WHERE 句で IPv4 アドレス文字列を利用する

```
mysql> SELECT id, INET_NTOA(CONV(HEX(remote_addr),16,10)) FROM ip_list
    -> WHERE remote_addr = CHAR(INET_ATON('198.51.100.20'));
+----+-----------------------------------------+
| id | INET_NTOA(CONV(HEX(remote_addr),16,10)) |
+----+-----------------------------------------+
|  3 | 198.51.100.20                           |
+----+-----------------------------------------+
1 row in set (0.00 sec)
```

- こんなおまじない覚えられない...

接続元 IP アドレスの保存

- 実は…
  - MySQL の符号なし INT 型は 32bit (0〜4,294,967,295)
    - データの格納に必要な記憶容量は 4 バイト

```
mysql> SELECT INET_NTOA(0), INET_NTOA(4294967295);
+--------------+-----------------------+
| INET_NTOA(0) | INET_NTOA(4294967295) |
+--------------+-----------------------+
| 0.0.0.0      | 255.255.255.255       |
+--------------+-----------------------+
1 row in set (0.00 sec)
```

- 文字列型のカラムにバイナリデータを保存するメリットは皆無

接続元 IP アドレスの保存

- 例えば、MySQL 上に IPv4 アドレスを数値として保存、運用する
  - テーブル定義

```
mysql> CREATE TABLE ip_list (
    ->   id int(10) unsigned NOT NULL AUTO_INCREMENT,
    ->   remote_addr int(10) unsigned,
    ->   PRIMARY KEY (id)
    -> ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.00 sec)
```

接続元 IP アドレスの保存

- データが記録された後、クライアント上からデータ内容の確認

```
mysql> SELECT * FROM ip_list;
+----+-------------+
| id | remote_addr |
+----+-------------+
|  1 |  3405803945 |
|  2 |  3221226067 |
|  3 |  3325256724 |
+----+-------------+
3 rows in set (0.00 sec)
```

接続元 IP アドレスの保存

- 10 進数の数値を IPv4 アドレス文字列に変換すると…

```
mysql> SELECT id,INET_NTOA(remote_addr) FROM ip_list;
+----+------------------------+
| id | INET_NTOA(remote_addr) |
+----+------------------------+
|  1 | 203.0.113.169          |
|  2 | 192.0.2.83             |
|  3 | 198.51.100.20          |
+----+------------------------+
3 rows in set (0.00 sec)
```

接続元 IP アドレスの保存

- WHERE 句で IPv4 アドレス文字列を利用する

```
mysql> SELECT id, INET_NTOA(remote_addr) FROM ip_list WHERE
    -> remote_addr = INET_ATON('198.51.100.20');
+----+-------------------------+
| id | INET_NTOA(remote_addr) |
+----+-------------------------+
|  3 | 198.51.100.20           |
+----+-------------------------+
1 row in set (0.00 sec)
```

接続元 IP アドレスの保存

- レンジで引くのも簡単

```
mysql> SELECT id, INET_NTOA(remote_addr) FROM ip_list
    -> WHERE remote_addr BETWEEN
    -> INET_ATON('192.0.2.0') AND INET_ATON('198.51.100.255');
+----+-------------------------+
| id | INET_NTOA(remote_addr)  |
+----+-------------------------+
|  2 | 192.0.2.83              |
|  3 | 198.51.100.20           |
+----+-------------------------+
2 rows in set (0.00 sec)
```

- 保守、運用には向かないとはあながち言えない

接続元 IP アドレスの保存

- IPv6 のデータも IPv4 同様に？
  - 128bit の数値を扱える数値型カラムがない DB が殆ど
  - 16 バイトのバイナリデータを格納出来るカラムで？
    - 10 進数の数値へ変換は難しくない
    - しかし IPv6 アドレス表現に変換する関数がない

接続元 IP アドレスの保存

- バイナリデータや数値で IPv6 アドレスを保存する運用はもはや絶望的に...
  - 現行システムがそうであれば、実装しなおし
- 可変長文字列として保存するのが現実的
  - IPv4 は 15 バイト
  - IPv6 は？

接続元 IP アドレスの保存

- IPv6 に必要なカラムサイズ
  - IPv4 射影アドレスを一番冗長に書くケース

```
0000:0000:0000:0000:0000:ffff:255.255.255.255
----5----0----5----0----5----0----5----0----5
```

- 45 バイトは必要
  - <netinet/in.h> で定義
  - Socket プログラミングで常套的に使われる値

```
#define INET_ADDRSTRLEN 16
#define INET6_ADDRSTRLEN 46
```

```
char ipv4_addr[INET_ADDRSTRLEN];
char ipv6_addr[INET6_ADDRSTRLEN];
```

アクセスログ解析の見直し

- アクセスログに IPv6 アドレス
  - デュアルスタックの場合混在も

```
2001:db8:bad:beef::0111:dead - - [11/Jun/2011:10:35:55 +0900]
"GET / HTTP/1.1" 200 9972 "-" "Mozilla/5.0"
198.51.100.156 - - [11/Jun/2011:10:36:14 +0900] "GET / HTTP/
1.1" 200 9972 "http://example.com/" "Mozilla/5.0"
```

アクセスログ解析の見直し

- 主要ログ解析ツールの IPv6 対応状況

ログ解析ツール	IPv6 対応
Webalizer	✓
AWStats	✓
Analog	✗ ✓ * 3rd party patch

HTTP(S) client library

- Web アプリケーション側で HTTP を利用するケース

  - OAuth
  - OpenID
  - その他 Web API
- クライアントライブラリは各言語によって対応状況は異なる
  - 調査、patch

ここまで行なえば...

- あなたの Web アプリケーションは IPv6 対応済
- そんなに難しいことは何もない

Critical issues

- 以前 APNIC 27 で私が伝えたこと
  - Web アプリケーションエンジニアは…
    - IPv6 の知識がなく、誤解もしている
    - IPv4 枯渇は知ってるが、時期を知らない
    - IPv6 化するモチベーションがない
    - IPv6 の利点はないと思っている
    - IPv4 の致命的欠点もないと思っている

最後に

- Imagine
  - The all of interesting websites will be phased out in 2011
  - これはネットワークエンジニアに向けた発言
  - アプリケーションエンジニアは指を咥えているだけではいけない
- 難しいことはない
  - IPv6 対応を推進しましょう
  - IPv6 対応に向けて実装をしましょう

ご清聴

ありがとうございました

Dec 1st, 2011 - Internet Week 2011