Practical DNS Operation:
～知ってるつもり、の再確認と運用現場で使えるノウハウ～

Internet Week 2005 チュートリアル

株式会社ブロードバンドタワー

伊藤 高一

kohi@bbtower.co.jp

前口上

対象受講者

- DNSサーバのシステム構築や運用について、ある程度の実務経験をお持ちの方。
- 初級クラスから中級クラスへのステップアップを志される方。
- 日々の実務に追われて、知識を再確認する機会を持ちたいとお考えの方。
- DNS DAYへの参加を希望していらっしゃり、DNSに関するスキル/知識に不安をお持ちの方。

DNS DAYとの住み分け

- DNS DAY
  - root、jpなどのサーバの現況や電子メールの送信元認証など時事性が強い話題
  - レジストリデータベースの管理などの話題
- このチュートリアル
  - RFCやマニュアルなどに書かれている事項の復習と日々の運用へ向けての応用

agenda

- 前口上
- フレームワーク編
  - DNSの構成要素
  - .ドメイン名の制約
  - サブドメインとglue
  - /24に満たないアドレス空間の逆索き
  - Lame Delegation
  - NOTIFY
  - DNSとパケットフィルタリング
  - Layer3とDNS
  - IPv6アドレスに対応する逆索きについて
- ゾーンデータ編
  - 構造
  - CNAME
  - TTL
  - (,)の意味と応用
  - $GENERATEディレクティブ
  - serial, comin' back!
- 運用編
  - rndc.key
  - 1台のホストで2つのnamedを動かす
  - -u(setuid),-t(chroot)
  - stubレゾルバ、大丈夫ですか?

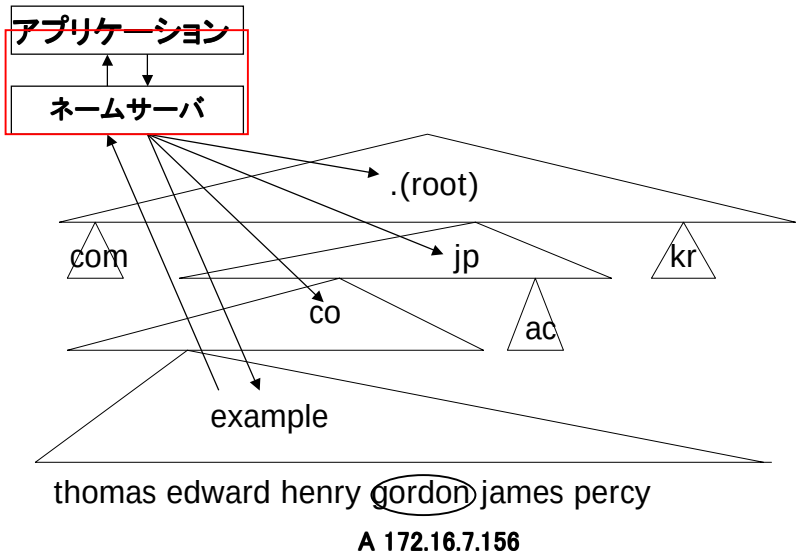
このチュートリアルでお話*しない*こと

- SPFやSender-IDなど電子メールの送信元認証
  - TXT RRの一用例と捕らえるに留める。
- 先進的/高度な話題
  - DNSSEC、TSIG、Dynamic Update、ENUM、Active Directoryとの連携、…
- インターネットレジストリへの諸手続きやドメイン名に関するpoliticsな話題
- djbdns、ANS/CNS、PowerDNSなどBIND以外の実装に関する話題

フレームワーク編

DNSの構成要素

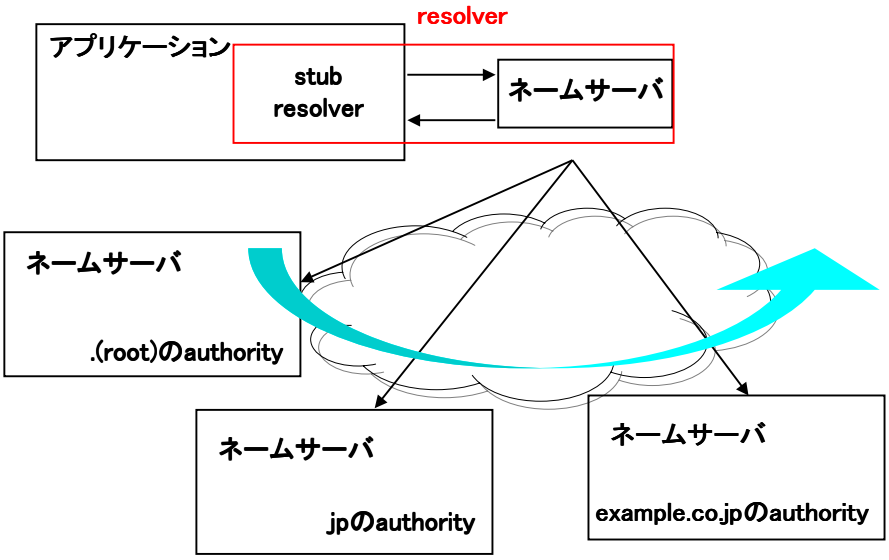
DNSの構成要素



DNSの構成要素(続き)

- ゾーン
  - DNSで情報を管理する単位
- authorityを持つ
  - あるゾーンについて、伝聞ではなく確証のあるデータを保持していること。
- authorityを委任する
  - (自分の下位の)あるゾーンについて、特定のネームサーバがauthorityを持っていると表明すること。
- authority
  - 「○○さんは☆☆のオーソリティだ」

DNSの構成要素(続き)



DNSの構成要素(続き)

- アプリケーション
  - ブラウザ、メーラ、ssh…
- stubレゾルバ
  - getaddrinfo()などDNSのAPI
  - レゾルバルーチンとか単にレゾルバなどとも呼ばれる。
  - 特定のネームサーバにqueryする。
    - /etc/resolv.conf
    - インターネットプロトコル(TCP/IP)のプロパティ

2つの「ネームサーバ」

- キャッシュサーバ
  - アプリケーションに対して、World Wideに関するネームサービスを提供する。
  - rootサーバから順に委任をたどって名前解決。
  - /etc/resolv.confやインターネット プロトコル (TCP/IP)のプロパティなどに設定。
  - recursiveサーバとも呼ばれる。

2つの「ネームサーバ」(続き)

- コンテンツサーバ
  - World Wideに対して、自分がauthorityを持っているゾーンのネームサービスを提供する。
    - primary、secondary
  - NS RRに設定する。
  - Internet Registryのデータベースに登録する。
  - authoritativeサーバとも呼ばれる。

2つの「ネームサーバ」(続き)

- 2つのネームサーバ
  - キャッシュサーバ/recursiveサーバ
  - コンテンツサーバ/authoritativeサーバ
  - 同じネームサーバ(DNSサーバ)だが、役割に違い。
  - 動作を理解する上では分けて考えるべき。
  - BINDのnamedは1つのプロセスで両方を兼ねる。
    - 必要な設定をすると、コンテンツサーバとして動作する。
    - 明示的に止めない限り、キャッシュサーバとして動作する。
  - djbdns(dnscache,tinydns)、Nominum(CNS,ANS)は機能毎に別プログラムに分離。NSDはコンテンツサーバのみ。
    - キャッシュ汚染の影響の回避や障害時の影響範囲の局所化。

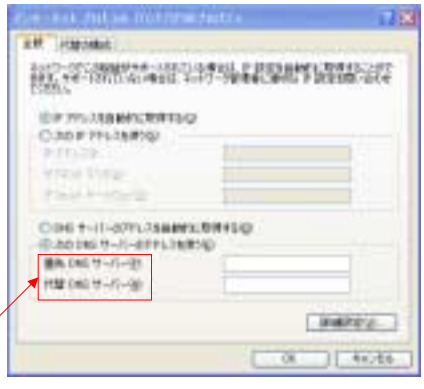
primary/secondary/master/slave

- primary v.s. secondary
  - ゾーンデータの出所に注目
- primary
  - ローカルファイルなど、ゾーン転送以外で得たゾーンデータを参照するサーバ
- secondary
  - masterからのゾーン転送により得たゾーンデータを参照するサーバ

primary/secondary/master/slave

- master v.s. slave
  - ゾーン転送の送出側/受け側に注目
- master
  - あるゾーン転送に注目したときの送出側
- slave
  - あるゾーン転送に注目したときの受け側
- 孫secondaryが存在する場合、第2世代のsecondaryは場面によってmasterだったりslaveだったりする。

primary/secondaryはキャッシュサーバではない



- この項目名をprimary/secondaryと誤解するらしい。
  - 出所:Microsoft ® Windows XP
  - Copyright © 1981-2001 Microsoft Corporation

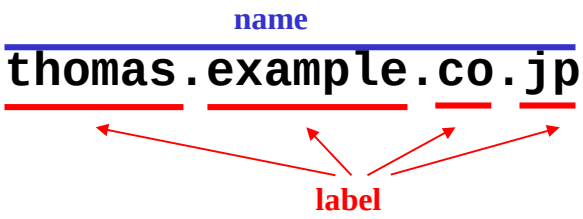
キャッシュサーバがauthorityを持つべきゾーン

- localhostの正索きと対応する逆索き
- link local address(169.254/16)と、使っていないprivate addressの逆索き
  - 外部に問い合わせるまでもなく解決できる。
  - 使っていないアドレスの逆索きの解決=NXDOMAIN
    - SOAとNSだけの空のゾーンデータをサービスする。
- 使っているprivate addressの逆索き
  - A社の192.168.0.1はpc1.example1.co.jp
  - B社の192.168.0.1はprinter.example9.co.jp
    :
  - 外部に問い合わせても、正しい答えは得られない。

ドメイン名の制約

ドメイン名の制約

- 文字数(RFC1035)
  - label: 63文字まで
  - name: 255文字まで



ドメイン名の制約(続き)

- あまり長い名前は、TCPにfall backする原因になる。
  - 定義されているRR
    - Aが1つだけ v.s. SOAとNSとMXと…
  - authority sectionやadditional sectionの量
    - NSのRDATAに登場する名前にAAAAが定義されていると…
  - 名前の圧縮の効き具合
    - 詳細はRFC1035 4.1.4.Message compression参照

  などに依存するので、限界は一概には言えない。
- EDNS0(RFC2671)なしでは512octetまで
  - IPヘッダ、UDPヘッダを除いたUDPのペイロード
- EDNS0を使えば65536octetまで拡大できる。
  - というのはパケットフォーマット上の話。
  - BIND9では4096octetまで。(options{edns-udp-size})

データが大きいと…

```
kohi@thomas[1]% dig example.co.jp ANY
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.3.1 <<>> @172.31.5.33
example.co.jp ANY
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status:
NOERROR, id: 47265
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 4,
AUTHORITY: 0, ADDITIONAL: 4
```


データが大きいと…(続き)

```
;; QUESTION SECTION:
;example.co.jp.                  IN      ANY

;; ANSWER SECTION:
                :
example.co.jp.          86400   IN      SOA
ns.example.co.jp. hostmaster.example.co.jp.
2005082601 3600 900 2419200 900
example.co.jp.          86400   IN      NS
very-very-long-
name.a234567890123456789012345678901234
567890123.example.co.jp.
```

データが大きいと…(続き)

```
example.co.jp.          86400   IN      TXT
"my name is jugemu jugemu gokou-no surikire,
kaijarisuigyo-no fuuraimatsu, kuuraimatsu,
unraimatsu, yaburakouji-no burakouji, paipo paipo
paipo-no shuuringan, shuuringan-no guurindai,
guurindai-no pompokopi-no pompokona-no
choukyuumei-no chosuke."

;; ADDITIONAL SECTION:
very-very-long-
name.a23456789012345678901234567890
1234567890123.example.co.jp. 86400 IN A
172.16.33.5
```


データが大きいと…(続き)

```
very-very-long-name.a2345678901234567890123456
8901234567890
1234567890123.example.co.jp. 86400 IN AAAA
2001:db8::1
          :
          :
;; Query time: 0 msec
;; SERVER: 172.31.5.33#53(172.31.5.33)
;; WHEN: Fri Aug 26 17:58:34 2005
;; MSG SIZE  rcvd: 607
```

ドメイン名の制約(続き)

- 文字種
  - RFC1035に、labelは
    - アルファベットで始まり
    - アルファベット、数字、'-'(ハイフン)の繰り返し
    - アルファベットまたは数字で終わる

    のが無難だろう、という意味のことが書いてある。
  - RFC1123で1文字目が数字のドメイン名もよいことになった。
    - 3com.com、0123.co.jp、…
  - RFC2181では8bit cleanであるべし、となった。

ドメイン名の制約(続き)

  - '_'はダメ。
    - BIND4.9.xのあるバージョンでチェックが厳しくなり、slave(secondary)をしていたゾーンがエラーになってあせった。
    - BIND8、9.3ではチェックの厳しさが指定できる。(RFC1035)
      - zone{check_names …;};
      - warn,fail,ignore
    - BIND9.0～9.2はチェックしていない。(RFC2181)
  - 大文字/小文字は区別されない。
    - internetweek.jp
    - InternetWeek.JP

サブドメインとglue

サブドメインとゾーン

- ゾーンとは
  - authorityを委任する単位
  - 管理の単位
    - ゾーンは自律的管理が及ぶ範囲で区切られるべき。
    - サブドメインはゾーンの境界になり得る。
    - が、すべてのサブドメインが独立したゾーンに(なる|しなければならない)わけではない。

サブドメインとゾーン(続き)

- サブドメインをDNSで表現するには
  - ゾーンを分ける方法
    - ゾーンを分けて、別のネームサーバに委任
    - ゾーンを分けるが、自ホストのネームサーバに委任
  - ゾーンを分けない方法
  - それぞれに適切な用途がある。

サブドメインとゾーン(続き)

- ns.example.co.jp:

```
$ORIGIN example.co.jp.
engineering     NS  ns.engineering
ns.engineering  A   172.16.7.225
sales           NS  ns
pc1.hr          A   172.16.7.193
pc2.hr          A   172.16.7.194
```

サブドメインとゾーン(続き)

- ns.engineering.example.co.jp:

```
$ORIGIN engineering.example.co.jp.
@   IN   SOA ... (
                 :
               )
         NS     ns
ns       A      172.16.7.225
fs1      A      172.16.7.226
lab1     A      172.16.7.227
         AAAA   2001:db8::227
```


サブドメインとゾーン(続き)

- ゾーンを分けて、別のネームサーバに委任
  - 別の管理主体にauthorityを委任できる。
  - engineering.example.co.jp.
- ゾーンを分けるが、自ホストのネームサーバに委任
  - RR数の多いサブドメインは、別ゾーンに分割するとゾーンデータファイルの管理が楽。
  - sales.example.co.jp.
- ゾーンを分けない
  - ほんの数個のRRのためにゾーンを分けるのも大げさ。
  - hr.example.co.jp.

glue

```
$ORIGIN example.co.jp.
engineering  NS  ns.engineering
```

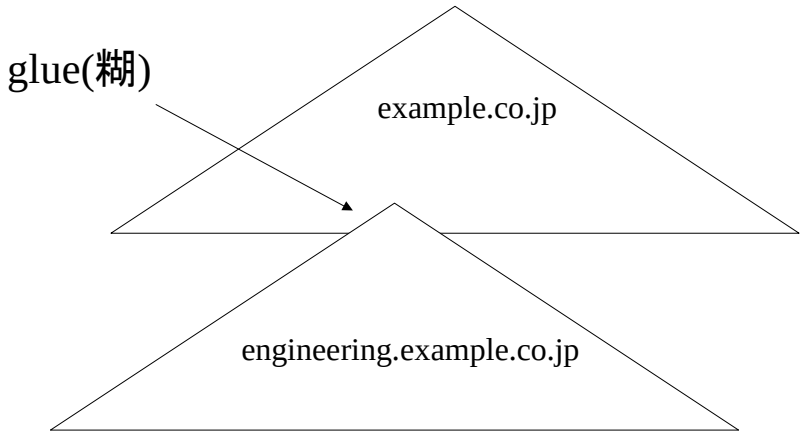
- lab1.engineering.example.co.jpのAを索きたい。
  - ns.engineering.example.co.jpにqueryすればいい。
  - では、そのIPアドレスは?
- 缶切りは缶の中

glue(続き)

```
$ORIGIN example.co.jp.
engineering     NS  ns.engineering
; answer sectionで返る。
ns.engineering  A  172.16.7.225
; これがglue。additional sectionで返る。
```

- これで172.16.7.225にqueryすればいいことがわかる。

glue(続き)



glue(続き)

```
$ORIGIN example.co.jp.
engineering     NS  ns.engineering
ns.engineering  A   172.16.7.225
```

- engineering.example.co.jpゾーンのauthorityはns.engineering.example.co.jpに委任している。
  - glueのA RRはnon-authoritativeデータ。

glue(続き)

- engineering.example.co.jpゾーンを廃止するには
  - NSを削除する。
  - Aも忘れず削除する。
  - NSを削除した時点でAはglueではなくなり、example.co.jpゾーンのauthoritativeデータになる。
    - 意図せぬ動作を引き起こす原因になり得る。

glue(続き)

```
$ORIGIN example.co.jp.
engineering  NS  ns.engineering
             NS  ns.example1.ad.jp.
```

- ns.engineering.example.co.jpのglueは必要。
- ns.example1.ad.jpのIPアドレスはglueに頼らず知ることができる。
  - 本来は自分の下位ゾーンに属する名前以外のglueは不要だが…

glue(続き)

- BIND8のrecursiveサーバでは問題が生じるケースがある。
  - ～8.2.7は、NSのRDATAがゾーン外の名前であっても、glueなしが2段以上続くと検索できない。
  - 8.3.0～は、クライアントの再送に依存。
- でもゾーン外glueを書くことは勧められない。
- 詳細はInternet Week 2004 DNS DAYの民田さんのプレゼン参照。
  - ネームサーバは内部名で
  - http://jprs.jp/tech/material/IW2004-DNS-DAY-internal-hostname-in-nameserver-minda.pdf

glue(続き)

- glueなしが2段続く例

```
$ORIGIN example.co.jp.
@   IN   NS   ns1.example1.ad.jp.
         NS   ns2.example1.ad.jp.

$ORIGIN  example1.ad.jp.
@   IN   NS   ns1.jp.example1.net.
         NS   ns2.jp.example1.net.
```

fetch-glue(BIND8)

- named.confのoptions{fetch-glue …};};
- 応答のadditional sectionが未完成のときに、手元の情報だけで応答するか、不足分を検索して完成させてから応答するか?
- 今日では、必要になったら検索すればいい、という考え方が主流。
  - キャッシュ肥大/汚染、無駄な待ち時間 の抑制
- BIND8のデフォルトはyes。noに設定する。
- BIND9は常時no。named.confに書かれていても無視する。

/24に満たない アドレス空間の逆索き

/24に満たないアドレス空間

- 例えば
  - 172.16.7.0/25: A社
  - 172.16.7.128/28: B学校
  - 172.16.7.144/29: C社
  - 172.16.7.152/29: D団体
  - 172.16.7.160/27: E社
  - 172.16.7.192/26: F社
- 「Class C」は死語。

/24に満たないアドレス空間(続き)

- 172.0.0.0/8 -> 172.in-addr.arpa
- 172.16.0.0/16 -> 16.172.in-addr.arpa
- 172.16.7.0/24 -> 7.16.172.in-addr.arpa
- 172.16.7.160/27 -> ???
- ゾーンは自律的な管理が及ぶ範囲で区切られるべき。
- 7.16.172.in-addr.arpa.は誰が管理する?
  - 172.16.7.0/24に対応。

RFC2317

- RFC2317
  Classless IN-ADDR.ARPA delegation
  (Best Current Practice)
  - 0/25.7.16.172.in-addr.arpa.
    - (A社: 172.16.7.0/25)
  - 128/28.7.16.172.in-addr.arpa.
    - (B学校: 172.16.7.128/28)
    :
  - 192/26.7.16.172.in-addr.arpa.
    - (F社: 172.16.7.192/26)

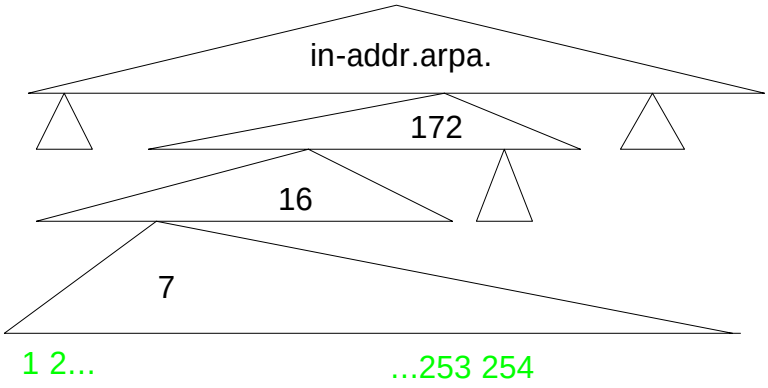
  を各組織が管理。

RFC2317(続き)

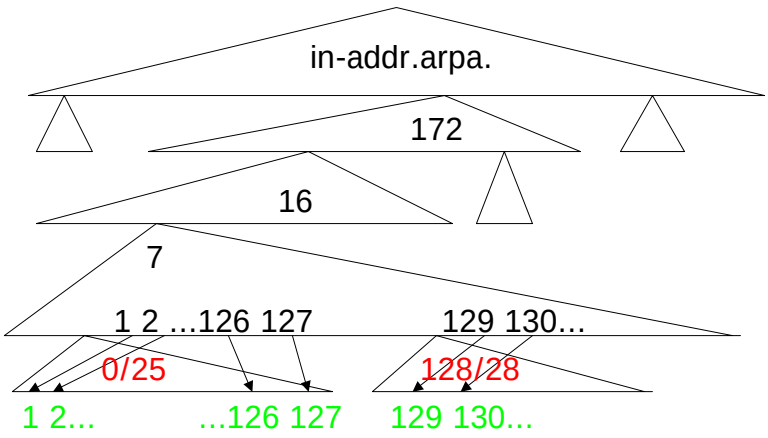
- 7.16.172.in-addr.arpa.ゾーンでは
  1.7.16.172.in-addr.arpa.
  ->1.0/25.7.16.172.in-addr.arpa.
  2.7.16.172.in-addr.arpa.
  ->2.0/25.7.16.172.in-addr.arpa.
  :
  129.7.16.172.in-addr.arpa.
  ->129.128/28.7.16.172.in-addr.arpa.
  :

  のCNAMEを定義して辻褄を合わせる。

/24以上の場合



/24未満の場合



RFC2317(続き)

- 具体的なゾーン名はISPと顧客の間で辻褄が合っていれば自由度あり。
  - 157.156/29.7.16.172.in-addr.arpa.
  - 157.156.7.16.172.in-addr.arpa.
  - 157.d-group.7.16.172.in-addr.arpa.
    :
- ISPの指示に従って下さい。

/24に満たないアドレス空間(続き)

- 7.16.172.in-addr.arpa.は誰が管理する?
  - ゾーン自体はISPが管理する。
  - でもPTRは実質的に顧客が管理する。
  - 顧客が融通の効かないGUIなサーバを使っている場合などはISPが直接PTRを書くこともある。
    - 更新は人間プロトコル、CGIなどDNSの枠外の方法。

Lame Delegation

Lame Delegationとは何か?

- 概念は「不正な委任」。
- 具体的にはいくつかの定義がある。
- あるゾーンのauthorityを委任されているネームサーバが応答しなければlame delegation
  - APNICによる定義
- あるゾーンのauthorityを委任されているネームサーバがおかしな応答を返せばlame delegation
  - non-authoritative answer
  - 複数のコンテンツサーバの応答が不整合
  - DNSQC Task Force by WIDE,JPRS,JPNICによる定義

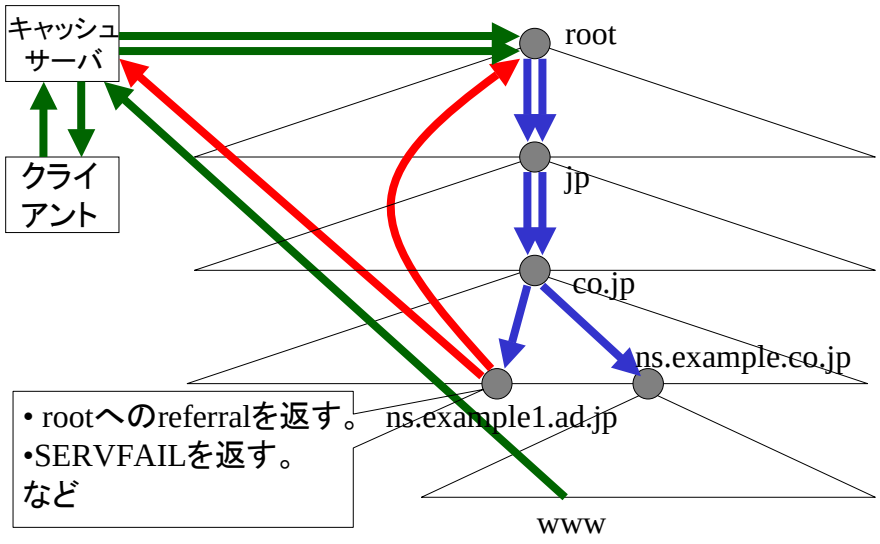
Lame Delegationとは何か?(続き)

- ここでは、上位ゾーンからあるゾーンのauthorityを委任されているネームサーバが、そのゾーンのauthorityを持っていない状態について議論する。

```
example.co.jp.   IN   NS   ns.example.co.jp.
                      NS   ns.example1.ad.jp.
```

- ns.example.co.jp: example.co.jpのauthorityを持っている。
- ns.example1.ad.jp: example.co.jpのauthorityを持っていない。

何が起こる?



なぜ起こる?

- authorityを持っているはずのネームサーバがauthorityを失くした。
  - 設定ミス
  - secondaryがexpireした。
- authorityを持っていないネームサーバにauthorityを委任する設定/手続きをしてしまった。
  - xSPにsecondaryを依頼するときの手続きミス。
- ISPを替えたのに、レジストリの登録を変更し忘れた。
  - 旧ISPは解約されたので設定を消した。
- etc,etc…

何が起こる(続き)?

```
example.co.jp.   IN   NS   ns.example.co.jp.
                      NS   ns.example1.ad.jp.
```

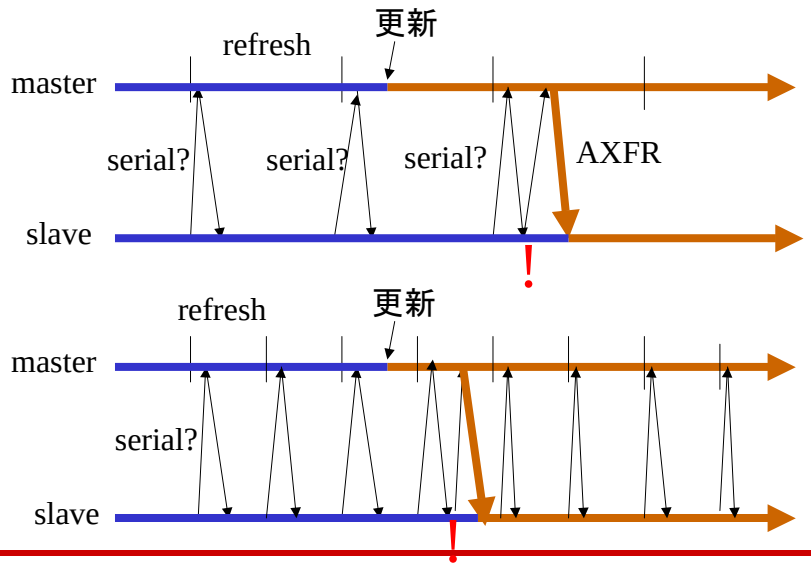
- example1.ad.jpドメインが☆◎※な理由で消滅。
  - ☆◎※には「買収」、「事業撤退」などお好きな言葉を当てはめて下さい。
- lame delegationが発生。
- 別組織がexample1.ad.jpというドメイン名を登録。ns.example1.ad.jpという名前でネームサービスを提供。
- 悪意の有無は別としてトラブルの元。
- この後のDNS DAYで事例紹介があります。

NOTIFY

NOTIFYがないと

- slaveはrefreshの間隔でmasterのserialをチェック。
  - serialの増加=ゾーンデータの更新を検出。
- NOTIFYがないと、masterでゾーンデータを更新しても、最悪、refreshの間は伝播しない。
- でも、refreshを短くすると負荷増大。

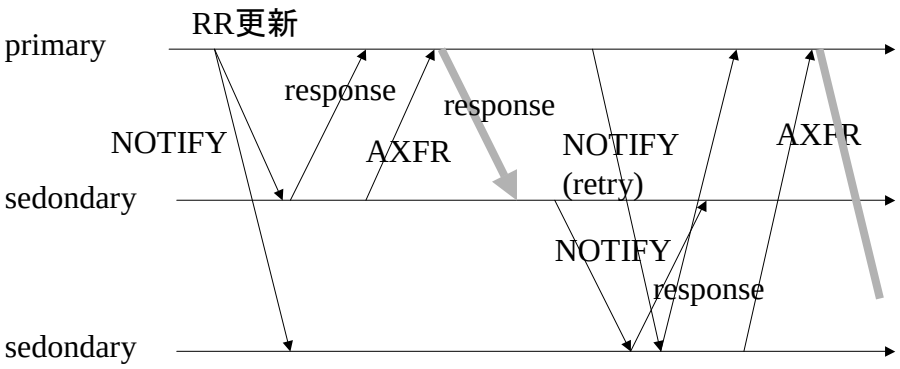
NOTIFYがないと



NOTIFY

- RFC1996
- RRが更新されたことをprimaryがsecondaryに能動的に通知。
  - secondaryがrefreshを待たずにゾーン転送を要求しにくることを期待。
- secondary同士もNOTIFYを送り合う。
  - NS RRに登場するネームサーバのうちSOAのmnameに書かれているサーバ(=primary)以外全部に送る。
  - 孫secondaryが存在する場合。
  - stealth slaveはnamed.confのalso-notifyで設定。

NOTIFYのある生活



NOTIFY(続き)

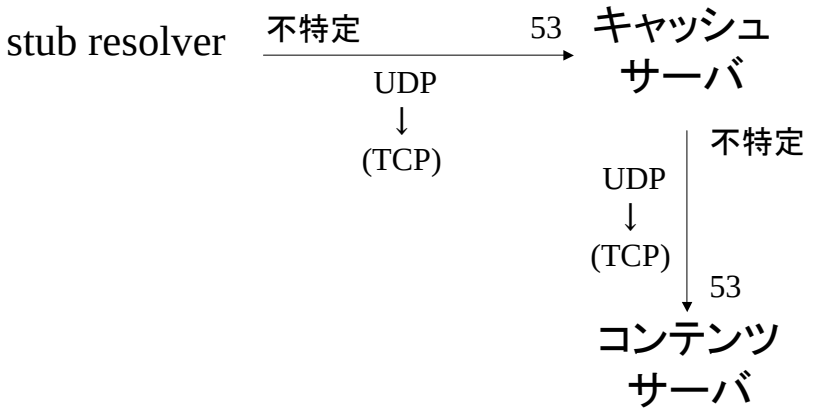
- NOTIFYを聴かないsecondaryも居る。
  - 例えば古い実装
  - 例えばNSD
    - ゾーン転送はネームサーバとは独立したプロセスが非同期に実行。
    - 実はNSDはrefreshさえも無視する(余談)。

DNSとパケットフィルタリング

基本

- transport層プロトコルは基本的にUDPを使用し、データが大きい場合はTCPにfallbackする。
- ゾーン転送は最初からTCPを使う。
- query待ち受け側のポート番号は53
- query送出側のポート番号は不特定

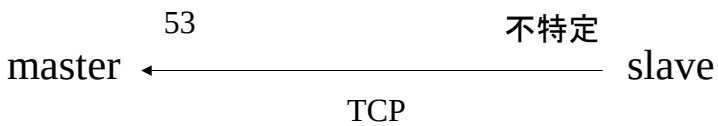
通常のquery



NOTIFYとゾーン転送

NOTIFY

ゾーン転送



パケットフィルタ適用時の注意

- TCPを忘れるな
  - 通常のqueryでもデータが大きいとTCPにfallbackする。
  - ゾーン転送の制限はLayer4ではなくLayer7で。
    - named.confのallow-transfer{}
- query送出側は不特定のポート番号を使う
  - BIND4はquery送出側も53番だった。
  - セキュリティ対策で特定のポート宛をdenyするときは、53番からはpermitするのを忘れずに。
    - 継続的で再現性のない異常に見舞われる。
    - responseを受け取れないので、無駄なretryをしまくって、世間に対して迷惑。

かつてFireWall-1で起きた問題

- Layer7まで見るのに、EDNS0を知らないので、不正なパケットと見なして落としてしまう(伝聞)。
  - Check Point NG FP3というバージョンで発生した。
  - SmartDefenseという機能を無効にすれば回避できる。
    - http://www.brandonhutchinson.com/Check_Point_NG_problems_with_EDNS.html(2004年2月)
    - http://lists.virus.org/fw1-0305/msg00433.html(2003年5月)
  - R55Wというバージョン(2004年5月)に修正された模様。
    - http://www.commswitch.be/files/R55W_WhatsNewFCS.pdf

Layer3とDNS

ルータに関する登録

- ルータもDNSに登録しておかないと、tracerouteしたときにIPアドレスしか出てこない。
- でもあまり情報を書きすぎるのはセキュリティ上どうなのか?
  - 機種名
    - 機種依存のセキュリティホール
  - 回線品目
    - DoSアタックの効き目

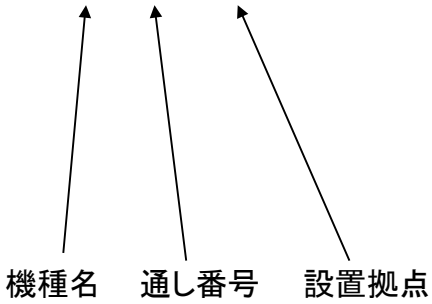
      :

ルータに関する登録(続き)

- 1台のルータでもインターフェース毎に別の名前がついていることも多い。
  - tracerouteの見栄えだけなら同じ名前でもよい。
  - in-band managementに使うなら、最低限、loopbackだけは別の名前にしておかないと不便。
- DNSでは名前に / は使えない。
  - so-6/0/0 ->so6-0-0
- どうしても数が多くなるので機械的な命名則がないと破綻する。

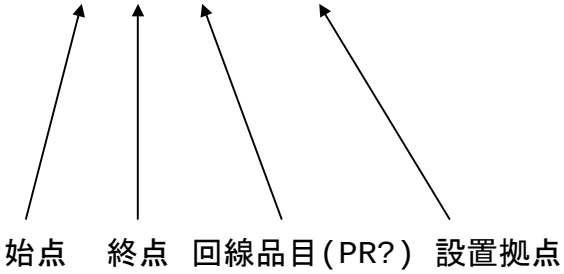
ルータに関する登録(続き)

- 例1
  - foundry2.otemachi.example.ad.jp



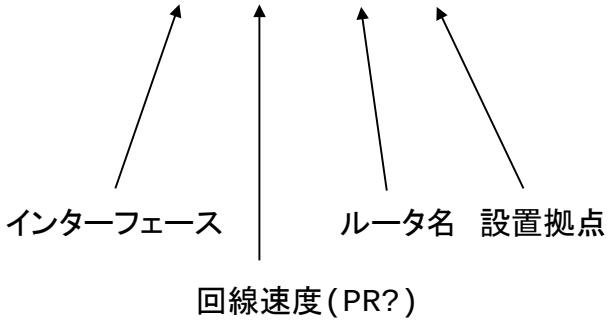
ルータに関する登録(続き)

- 例2
  - sjc3-nrt3-stm4-2.sjc3.example.net



ルータに関する登録(続き)

- 例3
  - so6-0-0-2488M.br2.PAO2.example.net

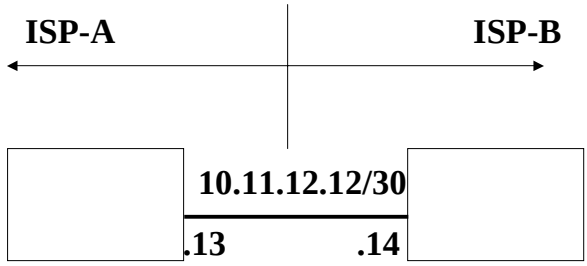


ルータに関する登録(続き)

- 例4
  - 192.168.0.1.example.ne.jp
- 例5
  - 192168000001.example.ne.jp
- 例6
  - pool-192-168-0-1.example.ne.jp
- 例7
  - 1.0.168.192.example.ne.jp

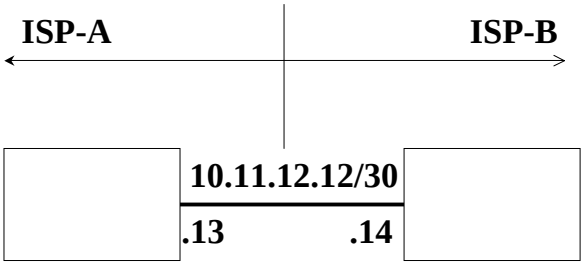
組織間の接続点に関する登録

- ISP-AとISP-Bの接続点
  - ISP-Aがアドレスを割当
  - 10.11.12.12/30
    - ISP-A: 10.11.12.13
    - ISP-B: 10.11.12.14



組織間の接続点に関する登録(続き)

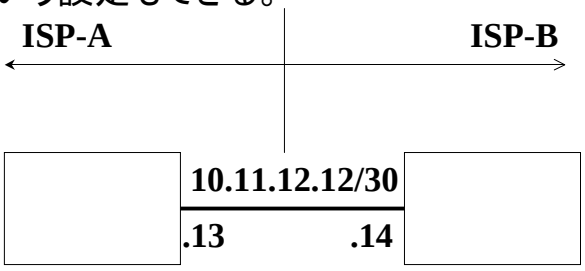
- 経験的にはこういう設定をする(してもらえる)ことが多い。




```
$ORIGIN 12.11.10.in-addr.arpa.
      :
13 PTR ge1-2.router.jp.isp-a.net.
14 PTR isp-b.peer.isp-a.net.
```


組織間の接続点に関する登録(続き)

- こういう設定もできる。




```
$ORIGIN 12.11.10.in-addr.arpa.
      :
13 PTR ge1-2.router.jp.isp-a.net.
14 NS  ns.isp-b.ad.jp.
```

4octet(/32)に対応する委任

```
$ORIGIN 14.12.11.10.in-addr.arpa.
@ IN SOA (
        :
        )
  NS  ns.isp-b.ad.jp.
  PTR ge3-0.router.isp-b.ad.jp.
```

4octet(/32)に対応するゾーン

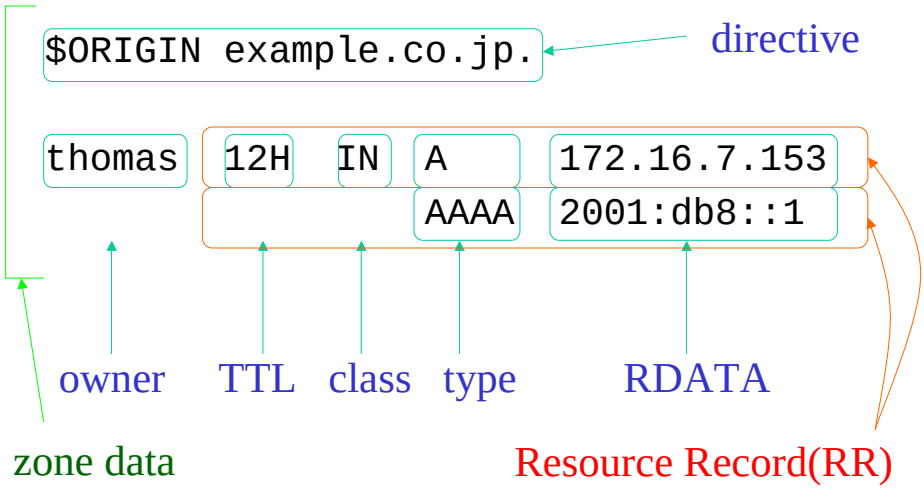
IPv6アドレスに対応する逆索きについて

IPv6アドレスに対応する逆索き

- IPv6アドレス
  - ルータ、サーバなどは、人間にわかりやすいアドレスを手動で設定するのが一般的。
    - DNSに登録するのも、まだ現実的。
  - いわゆるクライアントは、MACアドレスを元にして自動設定されることが多い。
    - DNSに登録するのは非現実的。
  - クライアントの身元確認などは逆索きに依存しない方向で考えた方がよい、という意見もある。
    - 自サイトががんばるのはよいが、他サイトに期待してはいけない。

ゾーンデータ編

構造



省略時

- owner
  - 行頭がブランクだと、最後に明示的に指定されたownerが引き継がれる。
- TTL
  - $TTL(もなければSOAのminimum)で指定した値
- class
  - RFC1035
    - 最後に明示的に指定されたclassが引き継がれる。
  - BIND8,BIND9
    - 1つのゾーンデータファイル内には混在できない。
    - named.confのzoneステートメントで指定

CNAME

CNAMEとは何か?

- ownerはaliasである。
- canonicalな名前はRDATAである。
  - ownerにRDATAというaliasをつける、ではない。

```
www   IN   CNAME   server237
```

してはいけないこと

- CNAMEを定義したownerに対して、他のRRを定義してはいけない。

```
www   IN   CNAME   server237
           MX      10   po
```

はダメ。

してはいけないこと(続き)

- user@example.co.jpというメールアドレスを使いたい。

```
@   IN   SOA     ns   hostmaster (
                 2005120601
                 1H
                 15M
                 4W
                 15M)
         NS      ns
         CNAME   po   ; MXを書かなきゃダメ
```


してはいけないこと(続き)

- 2つ目のCNAMEもダメ。
  - BIND4ではround robinさせるためによく使われた。
  - BIND8ではmultiple-cnames yesと設定すれば使えた。
  - BIND9ではダメ。

```
www   IN   CNAME   backend1
           CNAME   backend2
```

してはいけないこと(続き)

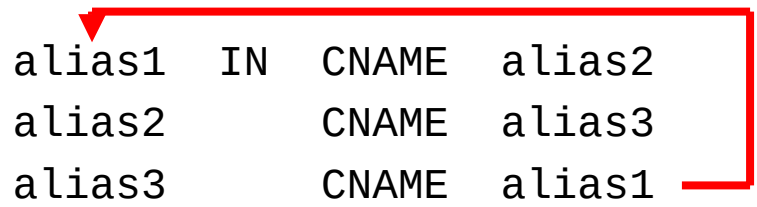
- NSやMXのRDATAに、CNAMEで定義したaliasを書いてはいけない。
  - RFC974には、よくない、という趣旨のことが書いてある。
  - RFC2181にはmust not be an aliasと書いてある。

```
@   IN NS    ns
ns     CNAME cabernet-sauvegnon
```

はダメ。

しない方がいいこと

- CNAMEのCNAMEは避ける。
  - 循環参照回避



しない方がいいこと(続き)

- RFCでは禁止していないが、逆にxx段まで動作すること、というような記述もない。
- BIND8では8段、BIND9では16段で正規化を打ち切る。
- djbdns(dnscache)は4段らしい(伝聞)。
- 1段でダメな実装はRFC違反だが、2段でダメでもRFC違反ではない。
- 自サイトのコンテンツサーバではなく、相手サイトのキャッシュサーバに依存。
  - 「うちはBIND9だから16段まで大丈夫」ではなく「djbdnsに素かれたら5段でアウト」。

CNAMEをゾーンの外へ向けることの意味

```
$ORIGIN example.co.jp.
www  IN  CNAME rental800.example1.
  ad.jp.
```

- www.example.co.jpの設定内容はexample.co.jpの管理者には(技術的には)制御できない。
  - example1.ad.jpのコンテンツサーバ(DNSの:=authoritativeサーバ)が乗っ取られると、WWWコンテンツの差し替えが可能。

CNAMEをゾーンの外へ向けることの意味(続き)

```
$ORIGIN example.co.jp.
www  IN  CNAME  rental800.example1.ad.jp.
```

```
$ORIGIN example1.ad.jp.
rental800  IN  A  192.168.64.129
```

↓

```
rental800  IN  A  10.11.12.13
```

192.168.64.129

本物の
コンテンツ

10.11.12.13

偽物の
コンテンツ

CNAMEをゾーンの外へ向けることの意味(続き)

- 問題提起と提案はInternet Week 2002 DNS DAYの森下さんのプレゼン参照。
  - DNS再入門
  - http://jprs.jp/tech/material/IW2002-DNS-DAY-morishita.pdf

TTL

TTLとは

- TTL=Time To Live
  - キャッシュサーバがあるRRを検索したときに、キャッシュ上に保持しておくべき期間。
  - コンテンツサーバからキャッシュサーバへの意思表示。

negative cache

- queryしたRRが存在しなかったときに、しばらくの間、同じRRのqueryを抑制するcache。
  - 処理量、トラフィックの削減という目的は同じ。
  - 具体的なRRの値ではなく、存在しなかったという事実をcache。
- RFC2308

negative cache(続き)

- エラーで索けなかった場合ではなく、明示的に「存在しない」という応答を得た場合。
  - 名前そのものがない。
    - NXDOMAIN
  - 名前はあったが、そのtypeのRRが定義されていない。
    - NODATA、NXRRSET
- エラーで索けなかった場合、索けなかった名前ではなく到達できなかったコンテンツサーバは記憶しておいてもよい。
- BIND8.2から対応。
  - $TTLの導入
  - SOAのminimumの意味の変更

negative cache(続き)

- minimumは、存在しないRRをキャッシュサーバにqueryされたときに、相手のnegative cacheに保持させる時間。
  - コンテンツサーバからキャッシュサーバに対する意思表示
- BINDのキャッシュサーバでは、authorityから通知されたminimumを鵜呑みにせず、上限を設定できる。
  - max-ncache-ttl

ネームサーバがnegative cacheに対応したことの意味

- キャッシュサーバ
  - negative cacheの機構が導入された、ということ。
- コンテンツサーバ
  - 明示的にTTLが指定されていないRRに適用するデフォルトのTTLがminimumから$TTLに変わった、ということ。

設定

- TTLはどこで設定する?
- 各RRに個別に指定

```
thomas   1d   IN A 172.16.7.153
```

- $TTLディレクティブでそのゾーン中のRRのTTLのデフォルトを設定

```
$TTL 1d
@ IN   SOA   thomas.example.co.jp.…(…)
       :
```


設定(続き)

- $TTL
  - RFC2308で導入された。
  - BINDでは8.2から対応。
  - ゾーンファイル中、$TTL以降のRRに作用。
  - ないとnamedが警告を出す。
    - BIND9.0.x、9.1.xではエラーになる。

設定(続き)

- SOAのminimumフィールド
  - BIND8.2より前ではこの値が省略時のTTL。
  - BIND8.2以降ではnegative cache上での保持期間に意味が変わった。
    - 86400とか設定してはいけない。
    - 存在しない(かった)RRを索きにきたキャッシュサーバは、今後1日、同じRRを索きに来るな、という意思表示。
  - BIND8.2以降でもRRに明示的指定がなく、$TTLもなければminimumの値が使われる。
    - 9.0.x、9.1.xを除く。

いくつぐらいがいいか?

- $TTL
  - RFC1912(Common DNS Errors,1996)
    - 1〜5日が典型的
    - MXやメールサーバのA、PTRなど変更頻度の低いRRは1〜2週間がお勧め。
  - DNS有識者100人(ウソ、居合わせた数人)にききました
    - RR更新前の過渡状態なら900=15分ぐらいまで許せるよね。
    - ロードバランサには0を返すのもあるらしい。
  - 有名サイトをいくつか(恣意的に)見てみました
    - 600とか900とか3600とか結構短い値が多い。
    - ラウンドロビンの結果を均質化するため?
  - 1996年に比べればコネクティビティがよくなっているので、TTLも富豪化してよい?

いくつぐらいがいいか?(続き)

- minimum
  - RFC2308(DNS NCACHE)
    - 特に推奨値はないが、例では1200=20分になっている。
  - BIND8,BIND9
    - max-ncache-ttlのデフォルトは10800=3時間。

キャッシュの消し方

- 他のキャッシュサーバのキャッシュに載ってしまったデータは、コンテンツサーバ側ではどうしようもない。
- キャッシュサーバ側で消すには
  - namedを再起動する。
  - rndc flush(BIND9.2.0で機能追加)
  - rndc flushname *name*(BIND9.3.0で機能追加)
- rndc flushでnegative cacheに載った情報も消えた。
  - rndc flushnameで対象の名前を指定してもnegative cacheは消えなかった。
  - BIND9.3.0で実験

時刻表記(BIND依存)

- SOAや$TTLの時間の表記には
  - w(week)、d(day)、h(hour)、m(min)

  などの単位が使えるらしい。
- ARMのSetting TTLsの項目には
  - All of these TTLs default to units of seconds, though units can be explicitly specified, for example, 1h30m.

  とこっそり(?)書いてある。
  - ARM: BIND9 Administrator Reference Manual

(,)の意味と応用

(,)の意味と応用

```
@  IN  SOA  ns hostmaster (
            2005120601
            1h
            15m
            4w
            15m )
```

- SOAを記述するのに必ず(,)が登場する。
- 他のところでは見かけない。

(,)の意味と応用(続き)

- でも、'('と')'はSOAの構文の一部ではない。
  - Parentheses are used to group data that crosses a line boundry. In effect, line terminations are not recognized within parentheses.
    - RFC1035 5. MASTER FILES; 5.1. Formatより

```
0.f.e.d.c.b.a.9.8.7.6.5.4.3.2.1 (
    IN  PTR  gordon.example.co.jp.)
```

なんていう使い方もできる…はず。

- BIND9はできたが、BIND8ではなぜかエラーに。

$GENERATEディレクティブ (BIND依存)

$GENERATE

- ゾーンデータ中に

```
$GENERATE 193-254 dhcp$ A  172.16.7.$
```

と書くと

```
dhcp193  A  172.16.7.193
dhcp194  A  172.16.7.194
      :
dhcp254  A  172.16.7.254
```

に展開される。

$GENERATE(続き)

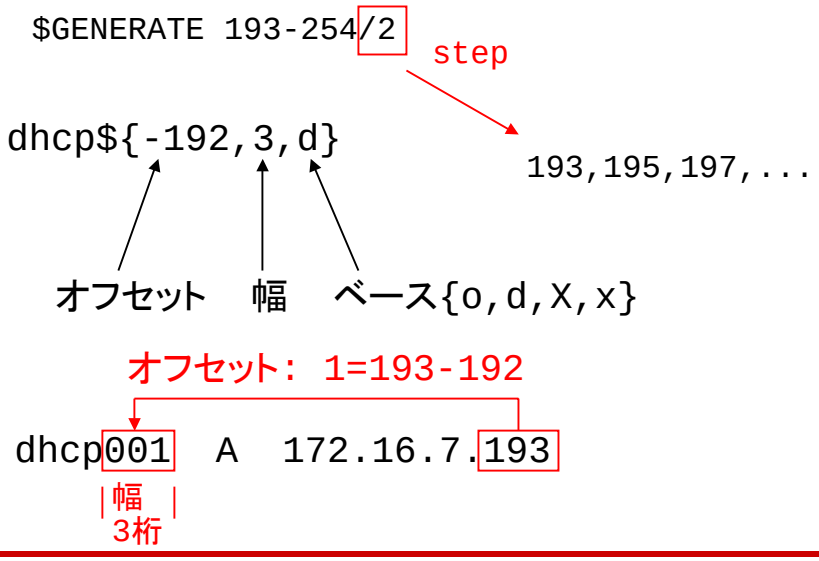
```
$GENERATE 193-253/2 dhcp${-192,3,d} A 172.16.7.$
```

と書くと

```
dhcp001  A  172.16.7.193
dhcp003  A  172.16.7.195
      :
dhcp061  A  172.16.7.253
```

に展開される。

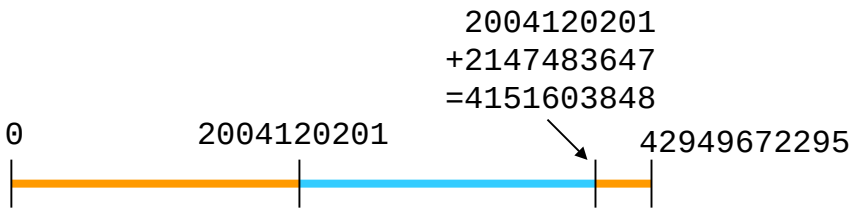
$GENERATE(続き)



serial, comin' back!

serial, comin' back!

- serial
  - 32bit
  - 0〜4,294,967,295=(2^32)-1
  - ある数nから次の2,147,483,647(=(2^31)-1)個の数はnより大きい。
  - nの前2,147,483,647個の数はnより小さい。
  - 4,294,967,295の次は0。



serial, comin' back!(続き)

– 4294967295より大きな値だとエラーになる。

Sep 4 14:43:18 thomas named[35341]: general: error: dns_rdata_fromtext: localhost:3: near '4294967297': out of range

Sep 4 14:43:18 thomas named[35341]: general: error: zone localhost/IN: loading master file localhost: out of range

– 2004120201に設定したかったのに3004120201とタイプしてしまい、reloadしてから気付いた ;_;

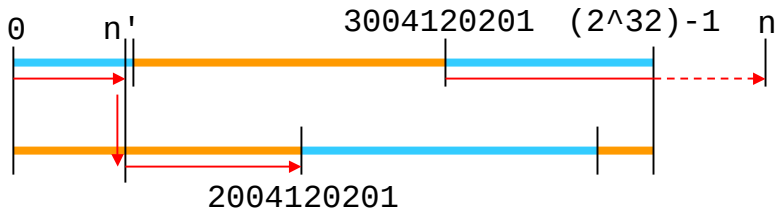
- 3004120201より「大きく」2004120201より「小さな」番号に設定。
- secondaryにゾーン転送。
- 2004120201に設定。
- secondaryにゾーン転送。

serial, comin' back!(続き)

- n=3004120201+(2^31)-1=5151603848>2^32
- n'=n-2^32=856636552

  3004120201 < n'

  n' < 2004120201



運用編

rndc.key

namedの管理ツール

- BIND8: ndc
  - ファイルシステム中のUNIXドメインソケット経由
  - TCPソケットもサポートされているが、認証機構がないので、事実上使えない。
- BIND9: rndc
  - TCPソケット経由
  - ネットワーク経由で別ホストからもアクセスできる。
  - TSIGで認証
    - BIND8でTCPソケットを使う場合とは非互換

rndc

- rndcはネットワーク経由で別ホストのnamedにもアクセスできる。
- rndc.conf
  - 鍵情報
  - デフォルトのアクセス先
  - デフォルトの鍵
- named.conf
  - rndcからのアクセスを待ち受けるソケット
  - アクセス制限
  - 鍵情報

rndc(続き)

- ネットワーク経由でのアクセスが前提なので、設定が面倒。
- 同一ホストからアクセスできればいいのであれば
  - named.confにcontrols{}ステートメントを書かない。
  - rndc.confも作らない。
  - rndc.keyというファイルを作る。
    - rndc-confgen -a
- namedとrndcの双方がrndc.keyを参照して動作する。

rndc.keyがあると

- named
  - named.confにcontrols{}がなければ
    - 127.0.0.1と::1の953/tcpだけで待ち受け
    - /etc/rndc.keyの鍵で認証
  - named.confのcontrols{}にkey{}のないinet…があれば
    - inet…にしたがって待ち受け
    - /etc/rndc.keyの鍵で認証
  - named.confに空のcontrols{}があれば
    - [注意!]rndcからのアクセスを待ち受けるソケットを作成しない。
- rndc
  - rndc.confがなければ/etc/rndc.keyに設定された鍵で953/tcp@localhostにアクセス

rndcの鍵情報

- 鍵情報が読めれば、rndcでnamedにアクセスできる。
  - rndc-confgen -aでrndc.keyを生成すると、permission 400で作成する。
  - rndc.confを手で作るときはowner,group,permissionに注意が必要。
  - 逆に、例えばwheelに属している人はsuしなくてもrndcを実行できるような設定も可。
  - named.confのkeyステートメントにも注意が必要。
    - 技巧的にはkeyステートメントを適切なアクセス権の別ファイルに切り出して、includeする方法もあり。

1台のホストで 2つのnamedを動かす

1台のホストで2つのnamed

- キャッシュサーバとコンテンツサーバを分けたい。
- でもホストの台数に余裕がない。
  - いくらPCが安くなり、FreeのOSが登場したといっても…
- 素直に起動すると`address in use`
- リソースの競合を回避すればいい。
  - queryを待ち受けるソケット
  - (r)ndcを待ち受けるソケット
  - 読み込むnamed.conf
  - 書き出すnamed.pid、named.statsなど

1台のホストで2つのnamed(続き)

- INETドメインソケットはIPアドレスとポート番号の組で識別される。
  - queryを待ち受けるポート番号は53から動かせない。
    ->IPアドレスを使い分ける。
  - ifconifg em0 … alias(FreeBSD)
  - ifconfig eth0:0 … (Linux)
- named.conf
  - named -c /etc/named-contents.conf
- named.pid、named.stats、…
  - named.confのoptions{}
- 別々のディレクトリツリーにchroot() してもよい。

キャッシュサーバ

- localhostとその逆索き
- private addressの逆索き
- link local addressの逆索き(254.169.in-addr.arpa)
  - それぞれallow-transferでアクセス制限
  - ゾーン転送はDNSの動作の中では重い部類に入るので、DoSアタックのツールになり得る。
- allow-queryでアクセス制限
  - localhostを忘れずに。

コンテンツサーバ

- recursion no;
- allow-transferで適切にアクセス制限。
- zone "." IN {type hint;…};は要らない。
  - BIND9ではbuilt-inのデータが参照されるが。
  - BIND8ではfetch-glue no;を併用すること。

設定例

- 適当な手法で1台のホストに
  - 192.168.0.1
  - 192.168.0.2

  の2つのIPアドレスを振る。
- キャッシュサーバは192.168.0.1で提供する。
- コンテンツサーバは192.168.0.2で提供する。

キャッシュサーバ(BIND8)

```
options {
        directory "/etc/namedb";
        allow-query {
                適当なacl;
        };
        allow-transfer {
                localhost;
        };
        fetch-glue no;
        pid-file "/var/run/named-cache.pid";
        listen-on {
                192.168.0.1;
        };
};
```

キャッシュサーバ(BIND8:続き)

```
controls {
        unix "/var/run/ndc-cache" perm 0600 owner
53 group 53;
};
zone "." IN {
        type hint;
        file "named.root";
};
zone "localhost" IN {
        type master;
        file "localhost";
};
# localhostやprivate addressの逆索きなども
```


コンテンツサーバ(BIND8)

```
    options {
            directory "/etc/namedb";
            allow-transfer {
                    適当なacl;
            };
            fetch-glue no;
            recursion no;
            pid-file "/var/run/named-
    contents.pid";
            listen-on {
                    192.168.0.2;
            };
    };
```

コンテンツサーバ(BIND8:続き)

```
controls {
        unix "/var/run/ndc-contents" perm
0600 owner 53 group 53;
};
zone "example.co.jp" IN {
        type master;
        file "example.co.jp";
};
# 逆索きなど必要に応じて
```


ndc

```
ns# ndc -c /var/run/ndc-cache status
named 8.4.5-REL Fri Oct  1 12:22:57 JST 2004
kohi@ns:/.amd_mnt/alphonse/u/share/usr/local/s
rc/bind/bind-8.4.5/src/bin/named
config (/etc/named-cache.conf) last loaded at
age: Wed Sep 21 11:41:47 2005
number of zones allocated: 64
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
server is initialising itself
```

キャッシュサーバ(BIND9)

```
options {
        directory "/etc/namedb";
        allow-query {
                適当なacl;
        };
        allow-transfer {
                localhost;
        };
        pid-file "/var/run/named-cache.pid";
        listen-on {
                192.168.0.1;
        };
};
```


キャッシュサーバ(BIND9:続き)

```
controls {
        inet 192.168.0.1 allow {
                localhost;
        };
};
zone "localhost" IN {
        type master;
        file "localhost";
};
# localhostやprivate addressの逆索きなども
```

コンテンツサーバ(BIND9)

```
options {
        directory "/etc/namedb";
        allow-transfer {
                localhost;
        };
        recursion no;
        pid-file "/var/run/named-
contents.pid";
        listen-on {
                192.168.0.2;
        };
};
```


コンテンツサーバ(BIND9:続き)

```
controls {
        inet 192.168.0.2 allow {
                localhost;
        };
};
zone "example.co.jp" IN {
        type master;
        file "example.co.jp";
};
# 逆索きなど必要に応じて
```

rndc

```
ns# rndc -s 192.168.0.1 status
number of zones: 1
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/1000
tcp clients: 0/100
server is up and running
```


元ネタは

- この話題の元ネタはInternet Week 2003 DNS DAYでの民田さんのプレゼン参照。
  - DNSサーバーの安全な設定
  - http://jprs.jp/tech/material/IW2003-DNS-DAY-secure-dns-minda.pdf

どのバージョンを使えばいい?

どのバージョンを使えばいい?

- 日ごろからアンテナを張ろう。
  - bind-announce@isc.org
  - janog@janog.gr.jp
  - http://jprs.jp/tech/
- 新バージョンがリリースされたら、世間で騒ぎが起きていないか数日様子を見て、大丈夫ならバージョンアップ。
  - 最新版でenbugしたケースもある。
    - 8.3.0, 8.4.3
- 現用バージョンのセキュリティホールが発見された場合などは臨機応変に急ぐ。

どのバージョンを使えばいい?(続き)

- セキュリティ情報はISCのWWWページ参照
  - http://www.isc.org/index.pl?/sw/bind/bind-security.php
- ごく簡単にサマライズすると…
  - 8.x
    - 8.3.7
    - 8.4.3～8.4.6(8.4.3には別の問題があるので使ってはいけない)
  - 9.x
    - 9.2.3～9.2.5
    - 9.3.1

  は、今(=資料作成時点)のところセキュリティ上の問題は見つかっていない。
  - 掲載されているバージョンのうち、ごく一部。

どのバージョンを使えばいい?(続き)

- でも現用バージョンの精査に時間をかけるぐらいなら、最新にバージョンアップしてしまえば?
  - そうは言ってもドキドキしますよね? 個人的お勧めは
    - configure --prefix=/usr/local/bind-9.*y.z*
    - BIND8はsrc/port/*OSname*/Makefile.setで設定
    - ln -s /usr/local/bind-*x.y.z* /usr/local/bind
  - 直前のバージョンを温存できるので切り戻しが楽。
- 某有識者のコメント
  - 「自分の使ってるバージョンが把握できていないようじゃ、すでにやられているようなものです」

-u(setuid),-t(chroot)

named -u

- setuid()するオプション
- namedを起動して、rootの特権を必要最低限行使したところで非rootにsetuid()して、rootの特権を放棄する。
- namedのプロセスを乗っ取られたときに、行われ得る操作の内容を制限することにより、セキュリティを向上させる。
- named.pidを書き出すディレクトリやrndc.keyなどのアクセス権の調整が必要。

named -t

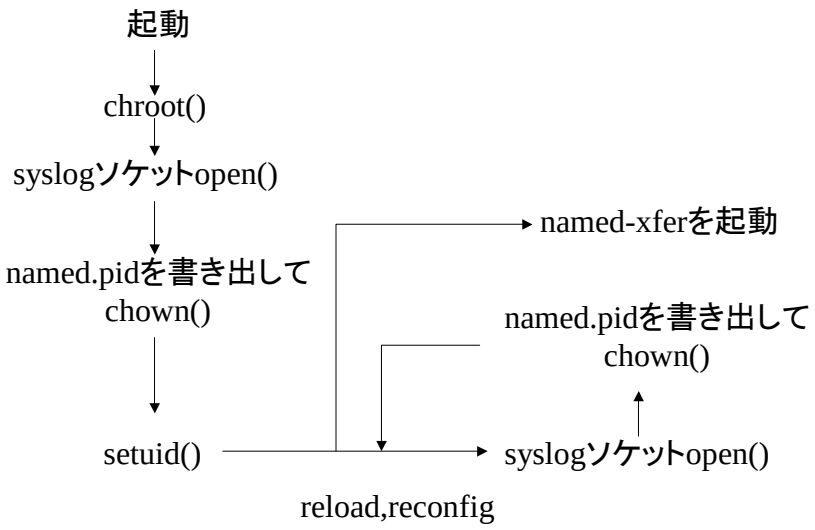
- chroot()するオプション
- namedのプロセスのrootディレクトリを本当のrootディレクトリから変更することにより、変更後のrootディレクトリより上位のディレクトリにはアクセスできなくなる。
- namedのプロセスを乗っ取られたときに、ファイルシステム中のアクセスできる範囲を限定することにより、セキュリティを向上させる。

-uと-t

- named -u
  - ディレクトリやファイルのアクセス権の調整が必要。
- named -t
  - namedだけのディレクトリツリーを作成できる。

->-uと-tは相性がいい。

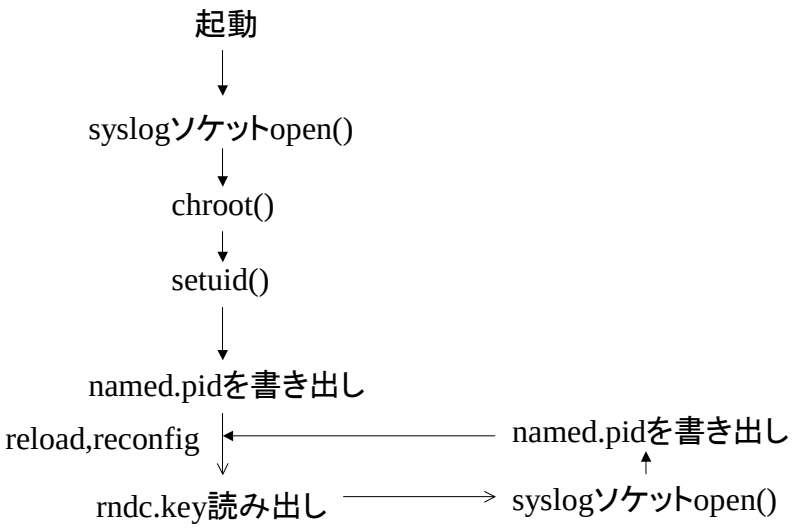
named -u bind -t /sandboxの挙動(BIND8)



named -u bind -t /sandboxの挙動(BIND8:続き)

- named.pidを書き出すuidが、起動直後とreloadやreconfigのときとで異なる。
- /sandbox/var/run/log(syslogソケット)が必要。
  - syslogd -l /sandbox/var/run/log(FreeBSD)
- ndcからのアクセスを待ち受けるソケットは/sandbox/var/run/ndc。
- slaveになるとき、named-xferはnamedの子プロセスとして起動される。
  - /sandbox/usr/libexec/named-xferが必要。
  - /sandbox/usr/lib/libc.so,…が必要。
  - もしくはnamed-xferをstatic link。

named -u bind -t /sandboxの挙動(BIND9)



named -u bind -t /sandboxの挙動(BIND9:続き)

- 起動直後はchroot()前にopenlog()するが、reloadやreconfigのときにもopenlog()する。
  - syslogd -l /sandbox/var/run/log(FreeBSD)
    - これを忘れると、起動直後のログは採れるのにreloadやreconfigした時点でログが途切れる。
- rndc.keyはユーザbindに読めないといけない。

named -t /sandboxの挙動

- /procを見せないと1 CPUで動いてしまった。
  - Fedora Core 2+SMP kernel+HTT CPU
- /etc/localtimeを見せないとtimezoneがUTCになってしまい、logのタイムスタンプがずれた。
  - FreeBSD

stubレゾルバ、大丈夫ですか?

どのstubレゾルバを使っているか把握していますか?

- BIND8のlibbind
  - /usr/local/bind/libなどにインストールされる。
- BIND9のlibbind
  - configureで指定しないとコンパイルすらされない。
- OSのlibc
  - 意識的に指定しないと、このオブジェクトがリンクされる。
- セキュリティfix? このマシンは最新のBIND9を追いかけてるから大丈夫だゼ!
  - 大間違い!!

おわりに

おわりに

- ここで「まとめ」というスライドを入れられるほどまとまったお話はできませんでしたが…
- フレームワーク編/ゾーンデータ編/運用編と題して
  - RFCやマニュアルに書かれていても、忘れられたり見落とされたりしがちな情報
  - 過去のDNS DAYからピックアップした、最新ではないけど最近の情報
  - tipsやknow how

  を紹介しました。
- 今後の皆さんの活動のお役に立てばうれしいです。
- ありがとうございました。