DNSの構築と運用
～BIND9時代の設定の常識～

Internet Week 2002　チュートリアル
株式会社インターネット総合研究所
伊藤　高一
kohi@iri.co.jp

Dec/16/2002

そもそもDNSとは

～introduction～

Dec/16/2002

そもそもDNSとは

- http://internetweek.jp/
  - 人間に優しい
- http://210.199.223.86/
  - 計算機に優しい
  - 人間には優しくない
- http://[3ffe:504:100:2ff:1234:5678:fedc:ba98]/
  - もっと人間に優しくない
- せっかく計算機を使っているんだから、計算機から人間に歩み寄って欲しいよね。

そもそもDNSとは(続き)

- 計算機に歩み寄ってもらうには?
- ホスト名<->IPアドレスなどを変換する仕掛けがあればいい。
- DNSはその解の1つ。
  - LANなどの閉じた環境ではNISなど別の解もある。
  - WorldWideが相手だと、事実上、唯一の解。

DNSの特徴

- WWW、Emailなどさまざまなネットワークアプリケーションの動作基盤。
  - OSIの人はわざわざアプリケーション層と分けてプレゼンテーション層という名前をつけちゃうぐらい重要。
- 設定はあんまり簡単とは言えない。
  - 対/etc/hosts比
- slaveや上位ゾーンのサーバなどと連携が必要。
  - サーバ同士だけではなく管理者同士も。
- 設定が多少おかしくても、なんとなくそれっぽく動いてしまう。
  - でも何かの拍子にボロが出る!

某プログラム委員からの問いかけ

- あなたのネームサーバの設定、正しいですか?
  - 企画段階でのこのチュートリアルの仮題。

というわけでagenda

- そもそもDNSとは～introduction～(done)
  [We're Here!]
- DNSの機構の復習
  – /24に満たないアドレス空間の逆索き
- BIND9を使ったネームサーバの基本的な設定
- DNSの運用
- BIND9のちょっと高度な設定
- Advanced topics

今回、お話「しない」こと

- DNSSEC、TSIG、IDNなどの高度な機能
- Internet Registryへの手続き
- BIND以外のサーバ、UNIX以外のプラットフォームの設定

DNS DAYとの役割分担

- このチュートリアル
  - DNSの仕組みを学ぶ。
  - BIND9の正しい設定を学ぶ。
  - 本の丸写しをしないための知識を学ぶ。
- DNS DAY(12月19日(木))
  - DNSの最新動向。
  - 現在のインターネットにおけるDNSの現況。
  - 実際のネットワーク環境に応じた適切な設定例。

DNSの機構の復習

top downに見たDNS

- Domain Name System
- 何のためのもの?
  - ホスト名<->IPアドレスなどの変換。
    - アクセスの利便性
    - リナンバーなどの隠蔽
- キーワードでDNSを語ってみると…
  - 階層型ドメインに基づく分散データベース。

top downに見たDNS(続き)

- 階層型ドメイン
  - '.'で区切られた名前。
  - *ホスト名.サブドメイン名.ドメイン名*
    という親子構造。
- 分散データベース
  - ドメインを基にしたゾーンという単位に名前空間を分割してデータを管理。
    - 個々のゾーンは一元管理。
    - ゾーンの切れ目では親から子にauthorityを委任。
    - 全体としては1つの名前空間を形成。

ゾーンとは

- ゾーン
  - 純技術的視点ではauthorityを委任する単位。
  - 運用の視点では管理の単位。
    - ゾーンは自律的な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うる。
  - 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。

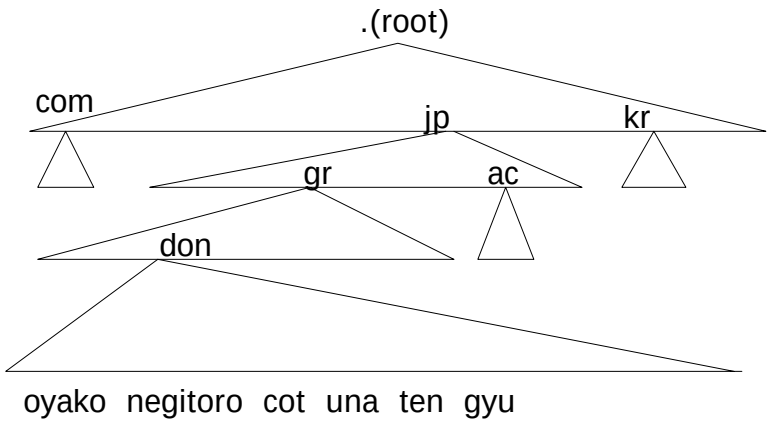
ゾーンとは(続き)

- authority
  - 親のゾーンと子のゾーンが連携するための仕組み。
    - 親ゾーンのネームサーバから子ゾーンのネームサーバに子ゾーンのauthorityが委任される。
    - 「○○ゾーンのauthorityは××だ。」
  - そのゾーンについてWorldWideからの問い合わせが振り向けられる。
    - データを一元的に自律管理できる。
    - ちゃんと面倒を見なければいけない。

ゾーンとは(続き)

- 例えば
  - jp.は1つのゾーン。.(root)からJPRSにauthorityを委任されている。
  - ad.jp.も1つのゾーン。jp.ゾーンからauthorityを委任されている。
  - nic.ad.jp.も1つのゾーン。ad.jp.ゾーンからauthorityを委任されている。
  - internetweek.jp.も1つのゾーン。jp.ゾーンからauthorityを委任されている。

ゾーンとは(続き)



ゾーンとは(続き)

- ゾーンの中身
  - リソースレコード(RR)
    - ホスト名->IPアドレス(A,AAAA)
    - IPアドレス->ホスト名(PTR)
    - メールサーバ(MX)
    - データの鮮度や賞味期限など(SOA)
    - authorityの所在(NS)
    - alias(CNAME)

    など。

ゾーンとは(続き)

- ゾーンのデータの例

```
don.gr.jp.  IN  SOA  oyako.don.gr.jp. root.don.gr.jp.(
                     2002121601
                     3600
                     1200
                     3600000
                     900)
            IN  NS   oyako.don.gr.jp.
oyako       IN  A    172.16.7.153
```

query

- query
  - リソースレコードの値を問い合わせること。
- recursive query
  - 目的のリソースレコードの値を要求する。
- non-recursive query
  - 目的のリソースレコードに到達するための authorityの委任先を要求する。
  - 木構造の枝分かれを1段1段たどる。
    - 最終的には目的のリソースレコードに行き当たる。
    - あるいはエラー。

検索の流れ

- resolverルーチン
  - アプリケーションプログラムがネームサーバにqueryするためのライブラリルーチン。
  - 一般には特定のネームサーバに対して目的の名前をrecursive queryする。
    - UNIXでは/etc/resolv.confで指定。
    - MacOSやWindowsにも相当する設定あり。
    - 知らないうちにDHCPやIPCPで設定されているかも。
  - RFC1035の用語ではstub resolver。

検索の流れ(続き)

- BINDをインストールしてもresolverルーチンはOSに付属の物を使っていることが多い。
  - BIND9では意識的にインストールしないとコンパイルすらされない。
  - 後からインストールしたライブラリを意識的にlink。
  - ダイナミックリンクのOSなら共有ライブラリの中のモジュールを差し替え。
  - CA-2002-19(Buffer Overflows in Multiple DNS Resover Libraries)はresolverルーチンのセキュリティホール。
    - ネームサーバではなく全クライアントで対策が必要。

検索の流れ(続き)

- アプリケーション(resolver)からqueryを受けたネームサーバ
  - rootサーバに対し、目的のRRに近づくためのauthorityの委任先をnon-recursiveに要求。
    - rootサーバだけは設定ファイルで決め打ち。
  - 得られたネームサーバに対して同様に要求。

    :
  - 目的のRRを得る(あるいはエラー)。
  - アプリケーションに応答。

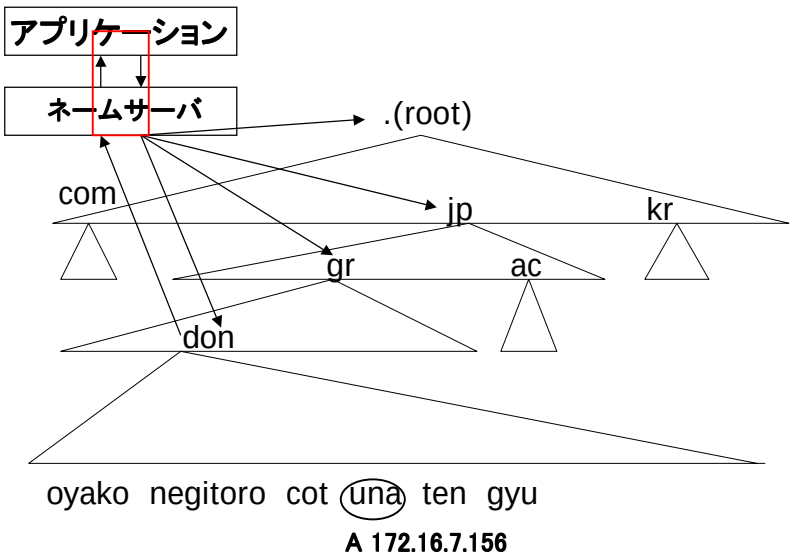
2種類の「ネームサーバ」

- アプリケーションからrecursive queryを受けるネームサーバ
  - アプリケーションに対し、WorldWideに関するネームサービスを提供。

  v.s.
- あるゾーンをサービスするネームサーバ
  - WorldWideに対し、そのゾーンのauthorityとしてネームサービスを提供。
- 同じ「ネームサーバ」だが役割に違い。

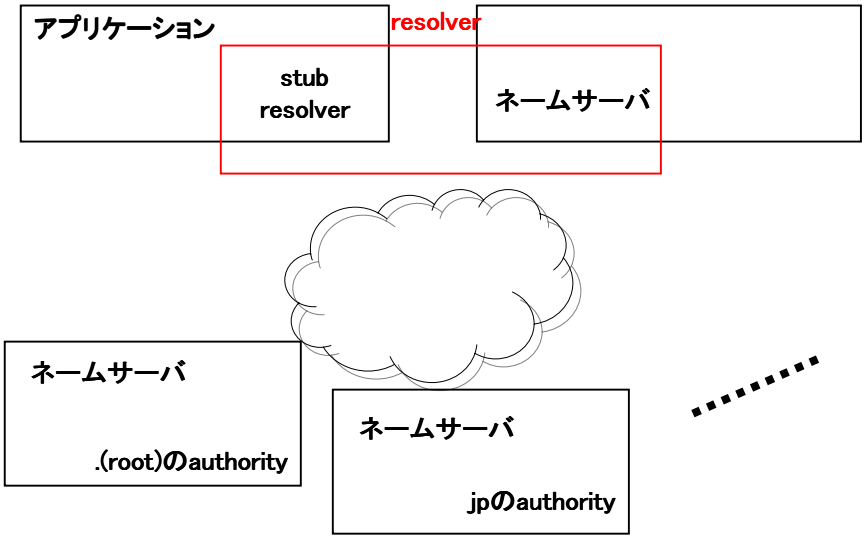
Dec/16/2002

検索の流れ(続き)



Dec/16/2002

resolver: RFC1035での用語



Dec/16/2002

逆索き

- DNSとは
  - ホスト名など<->IPアドレスの変換
  をするもの、だったはず。
- さっきから->の話ばっかり。
- <-はどうなっている?

Dec/16/2002

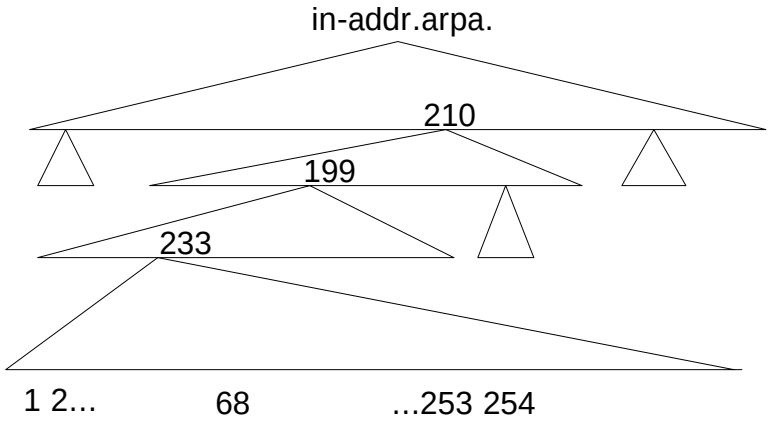
逆索き(続き)

- IP(v4)アドレス
  - 210.199.223.86
  - 階層があって‘.’で区切られている。
  - なんだ、ドメイン名と同じだ!
- IP(v6)アドレス
  - フォーマットはちがうが方法は応用。

逆索き(続き)

- ホスト名
  - [左]小さい単位(子)->大きい単位(親)[右]
- IPアドレス
  - [左]大きい単位->小さい単位[右]
    - 大きい単位:ネットワーク部
    - 小さい単位:ホスト部
- 逆順で表記
  - 86.233.199.210.in-addr.arpa.
  - 53.0.0.(省略).0.8.9.a.b.c.d.e.f.4.0.5.0.e.f.f.3.ip6.arpa.
    - ip6.int.からip6.arpa.に移行中。

逆索き(続き)



Dec/16/2002

/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

Dec/16/2002

/24に満たないアドレス空間(続き)

- ゾーンは自律的な管理が及ぶ範囲で区切られるべき。
  - 最初の方のスライドより。
- 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

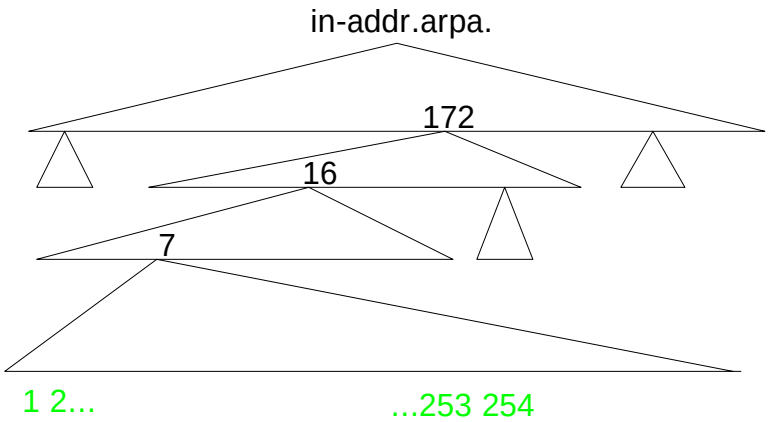
  1.7.16.172.in-addr.arpa.
  ->1.0/25.7.16.172.in-addr.arpa.

  2.7.16.172.in-addr.arpa.
  ->2.0/25.7.16.172.in-addr.arpa.
  :

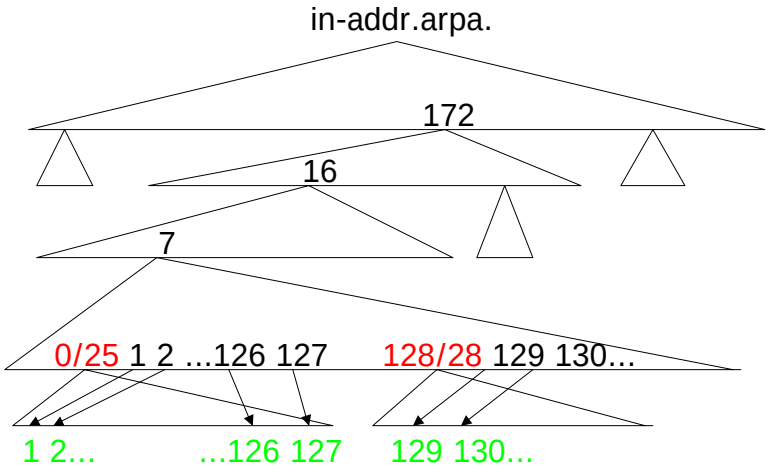
  129.7.16.172.in-addr.arpa.
  ->129.128/28.7.16.172.in-addr.arpa.
  :

  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



Dec/16/2002

/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。

Dec/16/2002

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。

リソースレコード

- A
  - ホスト名->IPv4アドレス

```
oyako IN A 172.16.7.153
```

- AAAA
  - ホスト名->IPv6アドレス

```
oyako IN AAAA 3ffe:504:fedc:ba98::53
```

リソースレコード(続き)

- PTR
  - IPアドレス->ホスト名

```
153.7.16.172.in-addr.arpa. IN PTR oyako.don.gr.jp.
```

- MX
  - ドメイン名->メールの配送先ホスト名と優先度

```
don.gr.jp.  IN  MX  10  negitoro.don.gr.jp.
            IN  MX  20  oyako.don.gr.jp.
```

  - prefeneceは小さいほど優先。

リソースレコード(続き)

- CNAME
  - alias->正規名

```
www.don.gr.jp.  IN  CNAME tekka.don.gr.jp.
```

  - CNAME RRを記述した名前には他のRRを記述できない。

```
www.don.gr.jp. IN CNAME tekka.don.gr.jp.
               IN MX 10 negitoro.don.gr.jp.
```

    はダメ。

リソースレコード(続き)

– NSやMXで指定するホスト名にはCNAMEで定義する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ではmust not be an aliasと書いてある。
  • NSやMXを要求したのにCNAMEが返ってくると正規化せずに無視するアプリケーションもあるらしい。

リソースレコード(続き)

– CNAMEのCNAMEも避ける。
  • 循環参照回避
  • BIND8では8段、BIND9では16段を超えるとエラーになる。
– 1つのaliasに複数の正規名を記述してはいけない。

```
www   IN   CNAME   tekka.don.gr.jp.
      IN   CNAME   cot.don.gr.jp.
```

はダメ。

リソースレコード(続き)

- NS
  - ゾーン名->authorityのホスト名

```
don.gr.jp.  IN  NS  oyako.don.gr.jp.
```

  - 親ゾーン中に記述
    - 子ゾーンのauthorityの所在。
    - authorityの所在が変わるときは所定の手続き。
    - slave(後で説明)にも委任してもらう。
  - 子ゾーン中に記述
    - 自分がauthorityであることの宣言。
    - slaveもauthorityであることを宣言。

リソースレコード(続き)

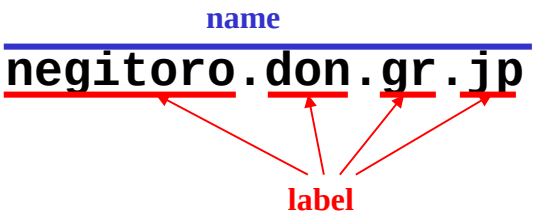
- SOA
  - ゾーン名->cacheやslaveを制御するパラメータ群

```
don.gr.jp. IN SOA oyako.don.gr.jp. root.don.gr.jp. (
                  2002121601
                  3600
                  1200
                  3600000
                  900)
```

  - cacheやslaveが登場してから説明。
- 他にもいろいろなtypeのRRがあるが、普通はこれだけあれば十分。

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)

- 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、…

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ではチェックの厳しさが指定できた。
    - zone{check_names …;};
    - warn,fail,ignore
  - BIND9はチェックしていない。
- 大文字/小文字は区別されない。
  - internetweek.jp
  - InternetWeek.JP

slave

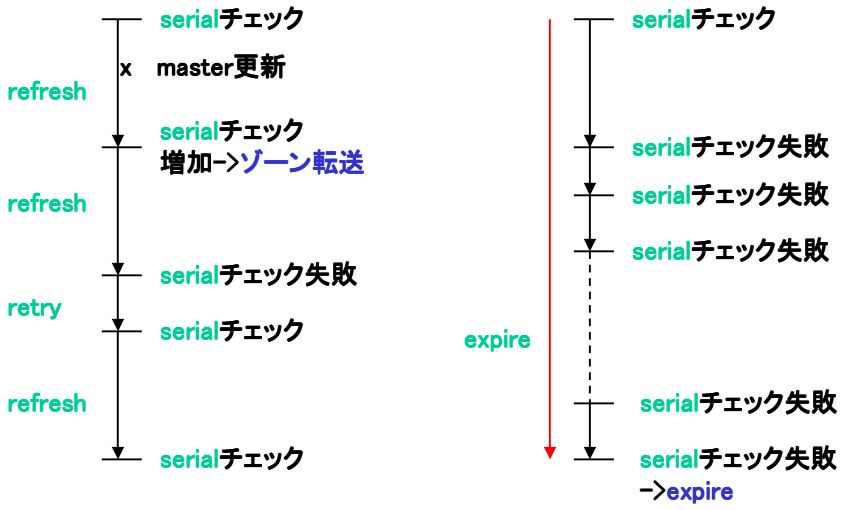
- データは各ゾーン毎に一元管理。
  - DNSの長所
- 障害時は?
  - single point of failureではいけない。
  - 他のネームサーバにゾーンデータをコピー。
  - そっちのネームサーバにもauthorityを委任。
    - slave(v.s.master)
      - 昔はsecondaryと言った。(v.s.primary)
    - queryした側には対等に見える。
      - fallbackではない。

slave(続き)

- 定期的にmasterのデータが更新されていないかチェック。
  - SOA RRの各パラメータで制御される。
  - serialが増加していないか?
    - 増加していればゾーン転送でデータをコピー。
  - refresh(秒)間隔でチェック。
  - 失敗したらretry(秒)で再試行。
  - expire(秒)の間、失敗し続けたら、その時点で保持しているデータは無効化。

Dec/16/2002

slave(続き)



Dec/16/2002

slave(続き)

- NOTIFY
  - RFC1996
  - masterの更新があったときにslaveに対して能動的に通知。
  - slaveがrefresh(秒)間隔でチェックに来るのを待たない。
    - 変更が速く伝播する。
  - best effort
  - NOTIFYを聴かないslaveも居る。

slave(続き)

- 用語の整理
  - primary、secondary
    - ゾーンデータの出所に注目。
    - ローカルならprimary、ゾーン転送で得ていればsecondary。
  - master、slave
    - ゾーン転送の動作に注目。
    - 転送元がmaster、転送先がslave。
    - 孫secondaryが居れば、第2世代は場面によってmasterだったりslaveだったりする。
  - primary masterという用語もある。

cache

- 毎回、rootサーバから順にたどっていては
  - 処理量
  - トラフィック
  - 待ち時間

  が大変。
- 一度索いたRRはキャッシュ。
- そのRRのTTLの間だけキャッシュ上に保持。
- invalidation手段はない。
  - 設定変更時は事前にTTLを短縮。

TTL

- TTLはどこで設定する?
  - さっきのRRの説明には出てこなかったけど…。
- 各RRに個別に指定。

```
oyako   86400   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 86400
@ IN   SOA   oyako.don.gr.jp…(…)
       :
```

TTL(続き)

- $TTL
  - RFC2308で導入された。
  - BINDでは8から対応。
  - ゾーンファイル中、$TTL以降のRRに作用。
    - BIND9 Administrator Reference Manual(ARM)にはSOAより前に書くと書いてあるが、実際にはゾーンの途中に記述するとそれ以降のRRに作用する。
  - ないとnamedが警告を出す。
    - BIND9.0.x、9.1.xではエラーになる。

TTL(続き)

- SOAのminimumフィールド
  - BIND4ではこの値が省略時のTTL。
  - BIND8以降ではnegative cache上での保持期間に意味が変わった。
  - BIND8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。

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。
  - 名前はあったが、そのtypeのRRが定義されていない。
- BIND8から対応。
  - $TTLの導入
  - SOAのminimumの意味の変更。

negative cache(続き)

- minimumは
  - このネームサーバがnegative cacheに保持する時間

  ではない。
- そういう値なら個々のゾーンではなくnamed.conf中に記述するはず。
  - max-ncache-ttl
- よそのネームサーバに存在しないRRをqueryされたときに、相手のネームサーバのnegative cacheに保持させる時間。

SOA

```
                                MNAME                RNAME
don.gr.jp. IN  SOA  oyako.don.gr.jp. root.don.gr.jp. (
                    2002121601      serial
                    3600            refresh
                    1200            retry
                    3600000         expire
                    900)            minimum
```

SOA(続き)

- MNAME
  - そのゾーンのデータの大元があるホスト名。
  - primaryとsecondaryの見分け。
- RNAME
  - そのゾーンの管理者のメールアドレス
  - @は.に書き換える。
    - hostmaster@don.gr.jp.->hostmaster.don.gr.jp.
    - @はゾーンデータの中では特別な意味(origin)。
    - @の前に.を含むメールアドレスは¥.に書き換える。
      - » Taisho.Ebi@ten.don.gr.jp->Taisho¥.Ebi.ten.don.gr.jp.

SOA(続き)

- 以下の各数値は
  - 32bit unsigned int
  - 時間の単位は秒
- serial
  - ゾーンデータの鮮度。
  - 日付+通し番号を使うことが多いが、あくまで人間界の流儀。
  - 日付を付けずに1から順に増やす流儀もあり。
  - slaveがmasterの更新をチェックする際の手がかり。

SOA(続き)

- refresh
  - slaveにmasterの更新をチェックさせる間隔。
- retry
  - refreshのタイミングでチェックに失敗したときの再試行間隔。
- expire
  - retryを繰り返してもチェックに失敗し続けたときに、その時点で保持しているデータの有効期限。
- minimum
  - negative cache上でのデータの保持期間。

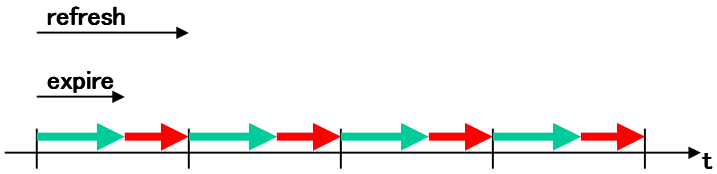
SOA(続き)

- RFC1033には
  - refresh: 3600(1時間)
  - retry: 600(10分)
  - expire: 3600000(約42日)
  - minimum: 86400(24時間)
    - 現代風には$TTLと読み替えること。

  がお勧め値と書いてある。
- RFC1912のお勧めは
  - expire: 2-4 weeks
  - minimum: 1-5 days

SOAに関する失敗例

- expire < refresh にしてしまった。
  - slaveはrefreshの間隔でmasterの更新をチェックするが、その間に必ずexpireしてしまう。
  - 時の運でslaveが正常に応答したりエラーになったり。



Dec/16/2002

時刻表記

- 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。

Dec/16/2002

ゾーンデータ

- 1行1RR
  - (…)でくくると行をまたげる
    - とRFC1035には書いてあるが、SOA以外では見たことがない。
- RRの他、$TTL、$ORIGIN、$GENERATE、$INCLUDEのディレクティブ。
- コメント記号は;
  - #はコメント記号ではない。
    - UNIXの常識に反している。
    - named.confと不統一でまぎらわしい。

ゾーンデータ(続き)

- RRのフォーマット

  <owner> [<TTL>] [<class>] <type > <RDATA>
  - <owner>
    - それ以降のフィールドが対応づけられるドメイン名。
      - DNSの用語としてはホスト名も「ドメイン名」。
    - 空白だと直前のエントリのownerが引き継がれる。
  - <TTL>
    - 省略時は$TTLの値が使われる。
  - <class>
    - 省略時は直前のエントリのclassが引き継がれる。普通IN。
  - <type>,<RDATA>
    - SOA,NS,A,AAAA,MX,PTR,CNAME,…

ゾーンデータ(続き)

- 相対表記/絶対表記
  - ドメイン名の末尾が‘.’で終わっていれば絶対表記。
    - ownerとNS,MX,CNAME,PTRのRDATA
    - A、AAAAのRDATAは関係ない。
  - ‘.’で終わっていなければ、あるドメインを起点とした相対表記。

ゾーンデータ(続き)

- ORIGIN
  - 相対表記のときに後ろに補われるドメイン名。
  - ゾーンデータ中では@で参照できる。
    - これがSOAのRNAMEに@が使えない理由。
  - 最初はnamed.confの中でそのゾーンデータと対応付けられているゾーン名がORIGIN。
  - $ORIGINディレクティブで変更できる。

相対表記に関する失敗例

- RRの末尾に‘.’を忘れる。

```
@   IN   SOA  …(…)
    IN   NS   oyako.don.gr.jp
    IN   NS   ns.myisp.ad.jp
```

は

```
@   IN   SOA  …(…)
    IN   NS   oyako.don.gr.jp.don.gr.jp.
    IN   NS   ns.myisp.ad.jp.don.gr.jp.
```

に展開されてしまう。

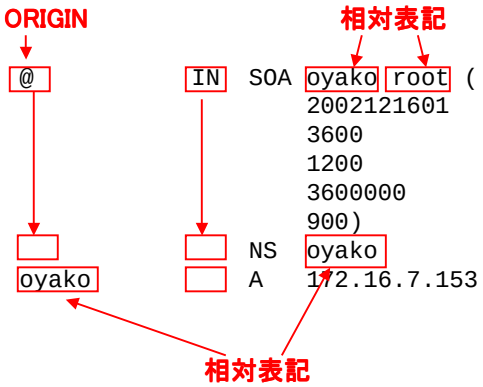
ゾーンデータ(続き)

- tips
  - 相対表記、絶対表記に関しては、自分の流儀を決めておく。
    - 相対表記は使わない。必ず絶対表記。
    - ownerは必ず相対表記、RDATAは必ず絶対表記。
    - 相対表記できるところは必ずする。
      - etc
  - 設定ミスを見つけやすくなる。

ゾーンデータ(続き)

```
don.gr.jp.  IN  SOA  oyako.don.gr.jp.  root.don.gr.jp. (
                     2002121601
                     3600
                     1200
                     3600000
                     900)
don.gr.jp.          IN  NS  oyako.don.gr.jp.
oyako.don.gr.jp.    IN  A   172.16.7.153
```


ゾーンデータ(続き)



サブドメイン

- サブドメインを追加するには
  - 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。
    - ゾーンは分けるが、自ホストのネームサーバに委任。
  - ゾーンを分けない方法

  の2.5通りの設定がある。
- どの方法をとるかは運用上の選択。

サブドメイン(続き)

- ゾーンを分ける方法

```
$ORIGIN don.gr.jp.
gyu             IN  NS  tokumori.gyu.don.gr.jp.
tokumori.gyu    IN  A   172.16.7.158
```

これがglue

```
$ORIGIN gyu.don.gr.jp.
@           IN  NS  tokumori
tokumori    IN  A   172.16.7.158
oomori      IN  A   192.168.0.1
nami        IN  A   192.168.0.2
```

サブドメイン(続き)

- ゾーンを分けない方法

```
$ORIGIN don.gr.jp.
tokumori.gyu  IN  A  172.16.7.158
oomori.gyu    IN  A  192.168.0.1
nami.gyu      IN  A  192.168.0.2
```


BIND9の基本的な設定

namedが無事動くまで

BINDのインストール

- ftp://ftp.isc.org/isc/bind9/からget。
  - 資料作成時点では9.2.1が最新リリース。
- コンパイル時設定はconifgureを使う。
  - FreeBSD/NetBSDでは苦労せずmakeできた。
  - ドキュメントには他に
    - AIX 4.3, Tru64 4.0D, Tru64 5, HP-UX 11, IRIX64 6.5, Solaris 2.6, 7, 8, Red Hat Linux 6.0, 6.1, 6.2, 7.0

    がSupported Operating Systemsと書いてある。
  - --sysconfdir=/etcがお勧め。

この資料で参照しているBINDのバージョン

- この資料を作成するための動作確認などに使ったバージョンは9.2.1。
- 何ヶ所かで動作確認に基づく挙動の紹介やARMとの相違の指摘があるが、すべて9.2.1を根拠としている。

BINDの設定ファイル

- named.conf
  - サービスするゾーン名
    - master/slaveの別
    - 定義ファイル(master)/dumpファイル(slave)
    - アクセス制限
  - ログの出力方法/内容
  - rndcコマンドが使う制御チャネル

  などnamed全体に関わる設定を記述。

BINDの設定ファイル(続き)

- デフォルトではconfigureの--sysconfdirで指定したディレクトリが探索される。
- コメント記号は

```
#      (行末まで)
//     (行末まで)
/*
        :   (複数行可)
 */
```

BINDの設定ファイル(続き)

- masterとなるゾーンの定義ファイル
  - ゾーン毎にそのゾーンに関するリソースレコード群を記述。
  - ファイル名はnamed.confの中で指定。
- rndc.conf
  - rndcコマンドが制御チャネル経由でnamedと通信するときの認証キーなど。
  - デフォルトではconfigureの--sysconfdirで指定したディレクトリが探索される。

BINDの設定ファイル(続き)

- named.root
  - rootサーバの指定。
  - 普通、内容はサイトに依存しないので、ftpでgetしてきたのをそのまま使えばよい。
    - ftp://rs.internic.net/domain/named.root
    - ftp://ftp.nic.ad.jp/internic/rs/domain/named.root
  - ファイル名は任意(named.confの中で指定)だが、資料によってはroot.cacheという名前を使っている。
  - この資料の作成中(11月5日)に更新された。
    - みなさん、ちゃんと入れ替えましたか?

BINDの設定ファイル(続き)

- resolv.conf
  - namedの設定ファイルではなくレゾルバライブラリの設定ファイル。
    - recursive queryを要求するネームサーバ
      - RFC1035の視点ではstub resolverのバックエンド
    - 省略時ドメイン名
  - ネームサーバだけでなくDNSを使う全ホストで設定。
    - WindowsとかMacOSにも相当する設定がある。

設定例の想定

- ドメイン名
  - don.gr.jp.
- IPアドレス
  - 172.16.7.152/29
  - 3ffe:504:fedc:ba98::/64
- 各ゾーンのslaveはns.mynsp.ad.jp[10.12.34.56]

を想定したネームサーバの設定例。

named.conf

namedのプロセスはここにchdir()する。
相対パスの起点。

```
options {
        directory "/usr/local/etc/namedb";
        listen-on-v6 {
                any;
        };
};
```

これがないとカーネルがサポートしていても
namedはv6を聴かない。

忘れがち。

named.conf(続き)

```
logging {
        channel to_syslog {
                syslog daemon;
                severity info;
                print-category yes;
                print-severity yes;
        };
        category default {
                to_syslog;
        };
};
```

named.conf(続き)

```
include “rndc-key”;   これはファイル名
controls {            permissionは400にすること。
        inet 127.0.0.1 port 953 allow {
                127.0.0.1;
        } keys {
                “rndc-key”;
        };      これは鍵の名前
```


named.conf(続き)

```
        inet * でもv6は聴いていないらしい。
        inet ::1 port 953 allow {
                ::1;
        } keys {
                “rndc-key”;
        };
};
```

named.conf(続き)

```
acl ns.myisp.ad.jp {
        10.12.34.56;  ns.myisp.ad.jpのアドレス
};
zone "." IN {
        type hint;
        file "named.root";
};
```


named.conf(続き)

```
zone "localhost" IN {
        type master;
        file "localhost";
};
zone "127.in-addr.arpa" IN {
        type master;
        file "127.in-addr.arpa";
};
```

named.conf(続き)

```
zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" IN
 {
        type master;
        file "1000.0000.0000.0000.0000.0000.
0000.0000.ip6.arpa";
};
```

それぞれ1行です

named.conf(続き)

```
zone "don.gr.jp" IN {
        type master;
        file "don.gr.jp";
        allow-transfer {
                ns.myisp.ad.jp;
        };
};
zone "152/29.7.16.172.in-addr.arpa" IN {
        type master;
        file "152_29.7.16.172.in-addr.arpa";
};
```

aclの名前。
DNSを検索するわけではない。

named.conf(続き)

```
zone "8.9.a.b.c.d.e.f.4.0.5.0.e.f.f.3.
ip6.arpa" IN {
        type master;
        file "89ab.cdef.4050.eff3.ip6.arpa";
};
```

1行です

named.conf(続き)

- BIND8では…
  - listen-on-v6がない。
  - controls
    - 構文がちょっと違う。
    - UNIXドメインソケットが使える。

  詳細はマニュアル参照。

localhost

```
$TTL 1d  ないと警告される。
@  IN  SOA  oyako.don.gr.jp. hostmaster.don.gr.jp. (
            2002071001
            1h
            20m     時間の表記にはw,d,h,mが使えるらしい。
            1000h
            15m )  negative cache上でのTTL
   IN  NS   oyako.don.gr.jp.
   IN  A    127.0.0.1
   IN  AAAA ::1
```


127.in-addr.arpa

```
$TTL 1d                                    1行です
@       IN      SOA     oyako.don.gr.jp.
hostmaster.don.gr.jp. (
                        2002121601
                        1h
                        20m
                        1000h
                        15m )
        IN      NS      oyako.don.gr.jp.
1.0.0   IN      PTR     localhost.
```

1000.0000.0000.0000.0000.0000.0000.0000.ip6.arpa

```
$TTL 1d
@    IN  SOA  oyako.don.gr.jp.
hostmaster.don.gr.jp. (
                2002121601
                1h
                20m
                1000h
                15m )
     IN  NS   oyako.don.gr.jp.
     IN  PTR  localhost.
```

1行です

don.gr.jp

```
$TTL 1d
@                IN      SOA     oyako.don.gr.jp.
hostmaster.don.gr.jp. (
                                 2002121601
                                 1h
                                 20m
                                 1000h
                                 15m )
                 IN      NS      oyako.don.gr.jp.
                 IN      NS      ns.myisp.ad.jp.
                 IN      MX      10 negitoro.don.gr.jp.
localhost        IN      CNAME   localhost.
```

1行です

don.gr.jp(続き)

```
oyako     IN      A       172.16.7.153
          IN      AAAA    3ffe:504:fedc:ba98::53
negitoro  IN      A       172.16.7.154
          IN      AAAA    3ffe:504:fedc:ba98::25
cot       IN      A       172.16.7.155
una       IN      A       172.16.7.156
ten       IN      A       172.16.7.157
gyu       IN      A       172.16.7.158
```


152_29.7.16.172.in-addr.arpa

```
$TTL 1d
@     IN  SOA oyako.don.gr.jp. hostmaster.don.gr.jp. (
                  2002121601
                  1h
                  20m
                  1000h
                  15m )
      IN  NS      oyako.don.gr.jp.
      IN  NS      ns.myisp.ad.jp.
153   IN  PTR     oyako.don.gr.jp.
154   IN  PTR     negitoro.don.gr.jp.
155   IN  PTR     cot.don.gr.jp.
156   IN  PTR     una.don.gr.jp.
157   IN  PTR     ten.don.gr.jp.
158   IN  PTR     gyu.don.gr.jp.
```

89ab.cdef.4050.eff3.ip6.arpa

```
$TTL 1d
@   IN  SOA  oyako.don.gr.jp. hostmaster.don.gr.jp. (
             2002121601
             1h
             20m
             1000h
             15m )
    IN  NS   oyako.don.gr.jp.
    IN  NS   ns.myisp.ad.jp.
25.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0  IN  PTR  negitoro.don.gr.jp.
53.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0  IN  PTR  oyako.don.gr.jp.
```


named-checkconf

- named.confの構文をチェック。
  - lintとかapachectl configtestのようなもの。

```
oyako# named-checkconf ./named.conf
./named.conf:49: missing ';' before 'file'
```

named-checkzone

- zoneファイルの構文をチェック。
- A RRのownerに‘_’が入っているとかExpireがRefreshより短いとかは検出できなかった。

```
oyako# named-checkzone don.gr.jp. don.gr.jp

dns_rdata_fromtext: don.gr.jp:21: near
'localhost.': bad dotted quad

zone don.gr.jp/IN: loading master file
don.gr.jp: bad dotted quad
```


rndc.confとrndc-key

- *$PREFIX*/sbin/rndc-confgenコマンドで生成。
- 少なくともFreeBSDとNetBSDでは

  -r /dev/urandom

  か

  -r keyboard

  オプションが必要。
  - /dev/randomを使うと現実的な時間で終了しない。

rndc-confgen

```
oyako# rndc-confgen -r keyboard
start typing:   <-stderrへの表示
................................
.........................
.........................
.........................
.........................
.........................
.........................
.........................
stop typing.  <-stderrへの表示
```

コマンドを起動

この間、適当にタイピングしている。

Dec/16/2002

rndc-confgen(続き)

```
# Start of rndc.conf
key "rndc-key" {
        algorithm hmac-md5;
        secret "pRGbJ08MDuX/2APYRaCr0A==";
};

options {
        default-key "rndc-key";
        default-server 127.0.0.1;
        default-port 953;
};
# End of rndc.conf
```

この部分をrndc.conf というファイルに保存

Dec/16/2002

rndc-confgen(続き)

```
# Use with the following in named.conf,
adjusting the allow list as needed:
# key "rndc-key" {
#       algorithm hmac-md5;
#       secret "pRGbJ08MDuX/2APYRaCr0A==";
# };
#
# controls {
#       inet 127.0.0.1 port 953
#               allow { 127.0.0.1; } keys
{ "rndc-key"; };
# };
# End of named.conf
```

この部分はrndc-keyというファイルに保存

rndc.confとrndc-key(続き)

- rndc.confが読めればrootじゃなくてもrndcコマンドが実行できる。
  - ファイルシステムのpermissionで制限が必要。
  - 例えばwheelな人はsuしなくてもrndcできるような設定もできる。
- rndc-keyにもkeyが書いてあるので取り扱い注意。

ありがちだが間違った質問

- Q:「上位のネームサーバを教えて下さい。」
- A:「ありません。」
  - (RFC1035でいう)resolverのネームサーバでISPのネームサーバにforwardersを向けようとしているらしい。

ありがちだが注意を要する質問

- Q:「slaveのIPアドレスを教えてください。」
- A(昔):「お客様の設定には不要です。」
  - 自分のところのゾーンデータにslaveのホストのAを書こうとしている。
- A(今):「どういう目的にご利用ですか?」
  - allow-transfer{}の設定には必要。
  - 開示するならアドレスは死守。
  - 将来の保証ができないなら、「自分でAを索いてown riskで設定して下さい。」
  - いずれにしてもよく事情を聞いてから。

namedの起動

```
oyako# $PREFIX/sbin/named -c /some/where/named.conf &;
tail -f /var/log/messages
```

configure --sysconfdirで指定した
ディレクトリに置いてあれば不要。

継続行
改行せずに入力

```
Oct 28 13:54:59 oyako named[62399]: starting BIND 9.2.1 -c /usr/
local/etc/namedb/named.conf
Oct 28 13:54:59 oyako named[62399]: using 1 CPU
Oct 28 13:55:04 oyako named[62399]: loading configuration from '/usr/
local/etc/namedb/named.conf'
        :
Oct 28 13:55:07 oyako named[62399]: general: info: running
```


namedの起動(続き)

- ログに注目。
  - エラーや警告はないか?
- ps auxww | grep named
  - ちゃんと走っているか?

dig

- dig
  - DNSの検索ツール。
  - BIND9にもnslookupは付属しているが、そのうちサポートされなくなりそう。
  - dig *@server name type*

dig(続き)

```
kohi@oyako[12]% dig @127.0.0.1 oyako.don.gr.jp.

; <<>> DiG 9.2.1 <<>> @127.0.0.1 oyako.don.gr.jp.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26801
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2,
ADDITIONAL: 2

;; QUESTION SECTION:
;oyako.don.gr.jp.               IN      A

;; ANSWER SECTION:
oyako.don.gr.jp.        86400   IN      A       172.16.7.153
```

Authoritative Answer

正しいか?

dig(続き)

```
kohi@oyako[14]% dig @127.0.0.1 153.7.16.172.in-addr.arpa. PTR

; <<>> DiG 9.2.1 <<>> 153.7.16.172.in-addr.arpa. PTR
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46177
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;153.7.16.172.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
153.7.16.172.in-addr.arpa. 86400 IN     CNAME
                                  153.152/29.7.16.172.in-addr.arpa.
153.152/29.7.16.172.in-addr.arpa. 86400 IN PTR  oyako.don.gr.jp.
```

本当は改行しない

dig(続き)

ゾーン転送

```
kohi@oyako[16]% dig @127.0.0.1 don.gr.jp. AXFR

; <<>> DiG 9.2.1 <<>> @127.0.0.1 don.gr.jp. AXFR
;; global options:  printcmd
don.gr.jp.              86400   IN      SOA     oyako.don.gr.jp.
                hostmaster.don.gr.jp. 2002121601 900 600 1200 900
don.gr.jp.              86400   IN      NS      oyako.don.gr.jp.
don.gr.jp.              86400   IN      NS      ns.myisp.ad.jp
don.gr.jp.              86400   IN      MX      10 negitoro.don.gr.jp.
cot.don.gr.jp.          43200   IN      A       172.16.7.155
gyu.don.gr.jp.          43200   IN      A       172.16.7.158
```

dig(続き)

- RFC2317を使っている場合はISP側の設定が済まないと逆索きできない。
  - 153.7.16.172.in-addr.arpa.は不可。
  - 153.152/29.7.16.172.in-addr.arpa.は可。
- チェックポイント
  - flagsにaa(Authoritative Answer)は含まれているか?
  - PTRやNSがoyako.don.gr.jp.don.gr.jp.になっていないか?
  - ゾーン転送はできるか?

rndc

- 動作中のnamedを制御するコマンド。

```
kohi@oyako[1]% rndc -help
rndc: illegal option -- h
Usage: rndc [-c config] [-s server] [-p port]
        [-k key-file ] [-y key] [-V] command

command is one of the following:
 reload        Reload configuration file and zones.
  reload zone [class [view]]
                Reload a single zone.
  refresh zone [class [view]]
                Schedule immediate maintenance for a zone.
  reconfig      Reload configuration file and new zones only.
  stats         Write server statistics to the statistics file.
  querylog      Toggle query logging.
  dumpdb        Dump cache(s) to the dump file (named_dump.db).
  stop          Save pending updates to master files and stop the server.
  halt          Stop the server without saving pending updates.
```

rndc(続き)

```
halt          Stop the server without saving pending updates.
  trace         Increment debugging level by one.
  trace level   Change the debugging level.
  notrace       Set debugging level to 0.
  flush         Flushes all of the server's caches.
  flush [view]  Flushes the server's cache for a view.
  status        Display status of the server.
  *restart      Restart the server.

* == not yet implemented
Version: 9.2.1
```


DNSの運用

namedが動き出してから

DNSの運用

- DNSの運用
  - ネームサーバの運用
  - 周囲との連携
    - 運用開始
    - 設定の変更
    - ちゃんと動いていないとき

DNSの運用(続き)

- 運用開始時
  - 上位ゾーンからauthorityの委任を受ける。
    - Internet Registryデータベースと連動。
    - 学内、社内の担当部署へ依頼。
    - 必要事項
      - ゾーン名
      - ホスト名(master、slaveとも)
      - glueが必要ならIPアドレス
    - 自分が設定した通りの内容で登録依頼する。

DNSの運用(続き)

- slaveを依頼する。
  - 必要事項
    - 依頼するゾーン名
    - masterのIPアドレス
  - slaveのホスト名を教えてもらう。
    - NS RRを記述
- RFC2317の設定を依頼する。
  - ゾーン名はISP/学内、社内の担当部署から指示を受ける。
  - 必要事項
    - ホスト名(master、slaveとも)

DNSの運用(続き)

- 設定の変更
  - 単にRRを追加するだけ。
    - 間違えないように追加すればよい。
  - RRの変更、削除
    - 事前にTTLを短くしておかないと、古いデータがよそのキャッシュに残ってしまう。
  - いずれの場合もserialの変更を忘れずに。
    - 変更がslaveに伝播しないことがある。
      - タイミング、ネットワーク上の位置、時の運などで新しいデータが返ってきたり古いデータが返ってきたり。

DNSの運用(続き)

– いずれの場合も新しいデータを読み込んだらすぐチェック。
  • RDATAを間違えたデータがよそのキャッシュに載ってしまうとやっかい。
  • slaveについては、NOTIFYのおかげで昔より気が楽になった。

DNSの運用(続き)

- 子供のゾーンの追加
  – NSと必要に応じてglueのAを追加。
- slaveになる。
  – named.confに設定を追加。
  – rndc reload
  – 動作確認。

```
zone “gyu.don.gr.jp” IN {
        type slave;
        masters {
                172.16.7.158;
        };
        file “gyu.don.gr.jp”;
};
```

DNSの運用(続き)

- ネームサーバのリナンバー、別ホストへの載せ替え
  - 上位ゾーンでもNS RRを変更してもらう。
    - Internet Registryデータベースの更新と連動
    - 学内、社内の担当部署への依頼
    - 自分が設定した通りの内容で登録。
    - ちゃんとしないとlame delegationの原因に。
  - slaveには参照するmasterを変更してもらう。
    - 更新が伝播しない。
    - やがてexpire。

DNSの運用(続き)

- 自分のところのネームサーバはちゃんと動いているか?
  - ログにエラーは出ていないか?
  - authorityを持っているはずのゾーンのRRをno-recurseで索いてみる。
    - dig *domain.name.* +norecurse
    - authoritative answerか?
    - レスポンスをMRTGでプロットしておくと性能劣化の目安になるかも。

DNSの運用(続き)

- よそのネームサーバもちゃんと動いているか?
  - 自分がmasterの場合
    - slaveはちゃんとauthoritative answerを返しているか?
    - serialは一致しているか?
      - 更新直後のrefreshの間はずれていても可。
    - 更新したらちゃんとゾーン転送に来るか?
      - NOTIFYを聴いているslaveならすぐ来る。
      - そうでなければrefreshまでの間に。

```
xfer-out: info: client 10.12.34.56#1425: transfer of
'don.gr.jp/IN': AXFR started

security: error: client 10.12.34.56#2073: zone
transfer 'don.gr.jp/IN' denied
```


DNSの運用(続き)

  - 自分がslaveの場合
    - ちゃんとserialをチェックできているか?
      - ダメでも要注意だが即問題ではない。
        - » 下位層の問題(例えばpingが通らない)であれば様子見。
        - » DNS的な問題なら対応開始。
    - expireしていないか?
      - していたらダメ。
        - » しちゃう前に見つける。

```
general: debug 1: refresh_callback: zone don.gr.jp/IN:
serial: new 2002121601, old 2002121601

general: info: zone don.gr.jp/IN: refresh: failure
trying master 172.16.7.153#53: timed out

general: info: zone don.gr.jp/IN: refresh: retry
limit for master 172.16.7.153#53 exceeded
```

slaveが動かない

- IP reachabilityは大丈夫か?
  - 下位層に問題があったら、いくらDNSを追いかけてもムダ。
- ルータやファイアウォールでのフィルタリングは間違っていないか?
  - DNSのパケットがsrc/destとも53番で固定だったのはBIND4までの話。
  - BIND8からはactive open側は任意の番号。

slaveが動かない(続き)

- allow-transfer{}でslaveからのアクセスまで拒否していないか?
  - 運用開始後、セキュリティ設定を強化したときは要注意。

```
security: error: client 10.12.34.56#2073:
 zone transfer 'don.gr.jp/IN' denied

xfer-in: error: transfer of 'don.gr.jp/IN' from
172.16.7.153#53: failed while receiving responses:
REFUSED
```

slaveが動かない(続き)

- ゾーン名は間違っていないか?
  - RFC2317の逆索きだとありがち。
    - 最初は正しく設定したのに、わざわざ/24相当のゾーンに直してしまう人が少なからず居る ;_;
- masterはauthorityをなくしていないか?
  - 設定変更後は即座にチェック。
  - digの出力のflagsにaaは含まれているか?

DNSが動かない:事例

- お客様からサポート要請
  - 「自分のドメインはアクセスできるが、外がアクセスできない。」
- 回線/ルーティング障害?? すわー大事!!
- 詳しくヒアリングしてみると
  - 中からはauthorityを持っている名前は索けるが、外部の名前が索けない。
  - 外からはそのサーバが索けない。
- 実際にアクセスしてみると
  - '.'のNSが索けない。

DNSが動かない:事例(続き)

- 推測
  - named.rootが悪い?
    - 正しく入手したものだった。
    - 外部から索けないことの説明がつかない。
- 結論
  - BIND4時代の知識でファイアウォールで53番同士のパケットしか通してなかった。
    - 使っていたのはBIND8。
    - query-source port 53;を仮に設定してもらい切り分け。

BIND9のちょっと高度な設定

logging

- named.confのlogging{}の設定。
- category
  - namedが出すログ情報のカテゴリ。
    - database,security,config,defaultなど
  - category毎に送出するchannelを割当。
    - 複数可
- channel
  - ログ情報の送出先。
    - 送出先はfile,syslog,stderr,null
    - ファイル名、facilityなどを指定。

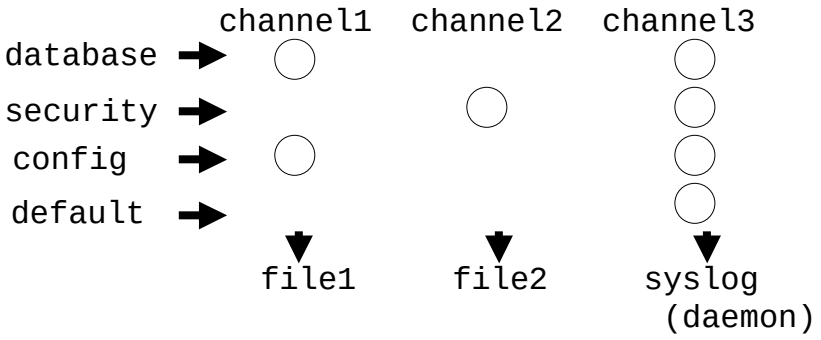
logging(続き)

```
logging {
        channel channel1 {
                file “file1”;
        };
        channel channel2 {
                file “file2”;
        };
        channel channel3 {
                syslog daemon;
                severity debug;
        };
```

logging(続き)

```
        category database {
                channel1;
                channel3;
        };
        category security {
                channel2;
                channel3;
        };
        category config {
                channel1;
                channel3;
        };
        category default {
                channel3;
        };
};
```


logging(続き)



logging(続き)

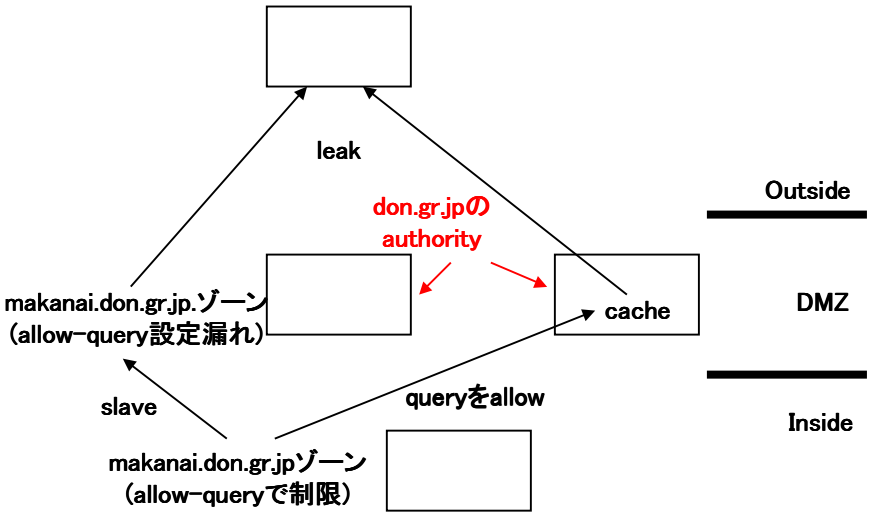
- namedが起動してからlogging{}の解析が済むまでのログ情報はsyslog(daemon)に送られる。
  - ARMの記述と違う動作に見える。
- logging{}で別のchannelを指定していても起動直後のログはそのchannelには出ない。
- 思いとおりに動作しない。
  - でもログにも何も出ていない。

  というときはlogging{}以前の設定に誤りがあるかも。

allow-query

- query全般のアクセス制限。
  - options{}とzone{}に書ける。
    - zone{}に書いた方が強い。
  - ファイアウォールの内側など、外に見せたくないゾーン。
- queryを許可しているホストのネームサーバのキャッシュやslaveを経由して外に漏れることも。
  - パズルをがんばる。

allow-queryに関する失敗例



Dec/16/2002

allow-transfer

- ゾーン転送のアクセス制限。
  - allow-query同様、options{}とzone{}に書ける。
- slaveには許可しないとダメ。
  - 最初からダメなら見つけやすいが…
  - 最初O.K.でも失敗が続くとexpireしてしまう。
    - 動き出してから設定を強化するときは要注意。

Dec/16/2002

allow-transfer(続き)

- 自分にも許可しておかないと動作確認するときに不便。
  - localhostというaclが組み込みで定義されている。
    - lo0だけでなく、自分のインターフェースに振られているアドレス全部。
    - と書いてあるが、v6アドレスは含まれないらしい。
- Brute Forceには無力だがエレガントなアタックを試みる輩には提供しているサービスなどのヒントを与えてしまう。
  - slaveでも適切に設定。

allow-transfer(続き)

- 小さなコマンドで大きな仕事をさせられる。
  - DoSアタックのツールとしては有効。
- 通常のqueryでもdatagramに乗り切らないとTCPにfallbackするので、TCPコネクション自体を拒否するわけではない。
  - SYN flood予防にはならない。
- 関連
  - options{transfers-out *N*;};
    - 同時に受け付けるゾーン転送の本数の上限。
  - options{tcp-clients *N*;};
    - 同時に受け付けるTCPコネクションの本数の上限。

recursion,allow-recursion

- recursive queryのアクセス制限。
  - いずれもoptions{}で指定。
  - recursionはyes/no。
  - allow-recursionは範囲を指定して許可。
- recursive queryは原理的にnon-recursive queryより重い。
  - 不正使用は拒否。
- (RFC1035でいう)resolverとしては機能しなくなる。

アクセス制限

- そのネームサーバの役割を明確にする。
  - (RFC1035でいう)resolverか?
    - サービスを提供すべき範囲は?
  - 何かのゾーンのauthorityか?

アクセス制限(続き)

- resolver
  - 正当な範囲だけにallow-recursionすればいい。
    - allow-queryという意見もある。
- authority
  - recursion offでよい。
  - localhost.とその逆索きはサービス不要。
  - 各ゾーンはslaveだけにallow-transfer。

Advanced topics

ログの監査

- ログにはさまざまな情報が出ている。
  - ちゃんと見ることは重要。
- でもあまり量が多いと、ちゃんと見るのも大変。
  - 重要なメッセージが埋もれてしまう。
- その解決策を根性論に立脚した肉体労働に見出すのは不毛。
  - せっかく計算機を使ってるんだから…。

ログの監査(続き)

- なぜ量が多くなるか?
  - 同じ内容のメッセージが繰り返し出ている。
  - 正常動作の報告が出ている。
- ではどうする?
  - severityは必要に応じて調整。
  - nullを活用。
  - サマリーを作成してメールでレポート。
    - 不審な点は生ログをチェック。
    - もっとリアルタイムな監視は矢萩さんのチュートリアルで:-)

ログの監査(続き)

- サマリーの作成
  #!/usr/bin/perl(sedでもrubyでも…)
  - タイムスタンプを削除。
  - pidを削除。
    - fileで採取するとつかないが、syslog経由だとつく。
  - active open側ソケットのポート番号を削除。
  - 正常動作に関するメッセージを削除。

  | sort | uniq | mail
- これでいわゆる「S/N比」(signal/noise)は向上する。

dnswalk

- 正索き/逆索きの整合性や文字セットなどをチェックするツール。
- http://www.visi.com/~barr/dnswalk/
- perlスクリプト。
  - Net::DNSモジュールが必要。
- 今のところv6はサポートしていない。
- cronで実行し、ログのサマリーと一緒にメールすると便利。
  - 不可避な警告やエラーが出るなら、前日の結果とdiffを取ると抑制できる。

大規模ネームサーバ構築tips

- 多くのゾーンをサービスするネームサーバ
  - 必然的に取り扱うファイルも増える。
    - ゾーンファイル

```
zone “don.gr.jp” IN {
    file “slave/reg/d/don.gr.jp”;
        :
};
```

    - ゾーン名でハッシュ。
    - “ja/gr/don.gr.jp”は多分、愚か。
    - 1ディレクトリに存在するファイルを減らすことにより namei()が高速化され、namedの起動が速くなる、かどうかは知らない :-)
    - 人間の視認性は向上。

大規模ネームサーバ構築tips(続き)

- named.confも大きくなる。

```
include “conf/slave/reg”
include “conf/slave/v4inv”
include “conf/slave/v6inv”
```

  - 1ファイルの寸法を小さく留める。
    - 編集時の扱いやすさ。
    - ロバスト性向上にも貢献。

ロバスト性の向上

- 目指すこと
  - 壊れにくいように。
  - 壊れちゃったらすばやく直せるように。

ロバスト性の向上(続き)

- ファイルの分類
  - 取り返し/あきらめがつくファイルv.s.つかないファイル。
    - ログ
      - なくても動作する。
    - slaveのdumpファイル
      - 建前はmasterから取って来れば復活するはず。
        - » ところがmasterが既にダメになっていることも…(^^;。
    - masterのゾーンファイル
    - configファイル
      - がーん ;_;

ロバスト性の向上(続き)

- アクセスの激しいファイル
  - ログ
    - named関連のファイルでは一番アクセスが激しいのでは?
  - slaveのdumpファイル
    - 知らないうちに更新される。
  - masterのゾーンファイル
  - configファイル
    - 設定変更や再起動時ぐらい。
  - アクセスが激しければ、その分、ディスクの負担も大きい。

ロバスト性の向上(続き)

- 取り返し/あきらめがつく/つかない
- アクセスの激しさ
  - 分類の結果は一致。
- /varをわける。
  - /,swap,おしまい、はダメ。
  - クラッシュ時の被害範囲の局所化。
  - 危険要素の閉じ込め。
- ログは普通/varの下。
- slaveのdumpファイルは/varの下に配置。

設定ファイルの履歴管理

- メリット
  - 不思議な設定に出会ったときの手がかり。
  - 編集ミスで壊したときの復旧の種。
  - 設定ミスの影響範囲の同定。
  - 返金沙汰になったときに、誰の給料を天引きすればいいか。
- デメリット
  - まぁ面倒といえば面倒だが…

設定ファイルの履歴管理(続き)

- RCSで管理。
- SCCSに愛があればSCCSでもよい。
- 極めて個人的見解だが、CVSはちょっと…
  - 設定ファイルはソースと異なり排他制御も重要な要素。
    - CVSもロックを有効にする設定あり。
  - 1つのレポジトリを複数箇所にcheckoutできる。
    - namedが参照しているリビジョンとレポジトリの内容が不一致だと困る。

MXとMTA

- MX RR
  - そのドメイン宛のメールをどのホストに配送すればよいかの指定。
- MXを書けば、MTAがそのドメイン宛のメールを受領できるようになるわけではない。
  - MTAでもそれ相応の設定が必要。
    - local configuration errorなどでメールを紛失。
- 1つのドメインに複数のMXを設定できる。
  - preferenceが小さいほど優先。

Dec/16/2002

MXとMTA(続き)

- バックアップMXに配送されても、送信元では無事送れたように見えてしまうことが多い。
  - トラブル時のトレースが困難。
- 不用意なホストを記述すると、エラーになる。
  - 直前のスライド。
  - MXを向ける以上はMTAもきちんと設定する。
  - 断りなくよそのサーバにMXを向けてはいけない。
    - ISPのサーバにMXを向ける顧客…。
    - 昔はただの(?)不正使用だった。
    - 今は普通third party relayでハネられる。

Dec/16/2002

MXとMTA(続き)

- MXを複数記述するなら、設定の次元ではなく設計の次元から考えなければいけない。
  - DNSだけでなくmailについても理解が必要。
- ローカル配送しないでmailboxが復旧するまでspoolするだけのバックアップMXは不要では?
  - RAIDストレージにmailboxを置き複数ホストで共有して…とか
  - 外に見せるMXを複数台用意して内部のmailboxへstatic配送、という設計では有効。
- 深い議論は安藤さんのチュートリアルで :-)

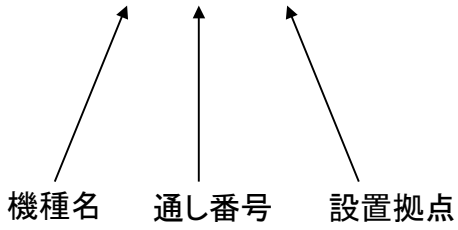
ルータに関する登録

- 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

      :

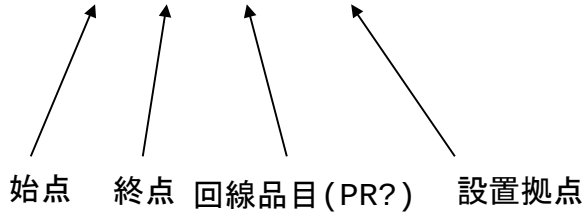
ルータに関する登録(続き)

- 例1
  - foundry2.otemachi.wide.ad.jp



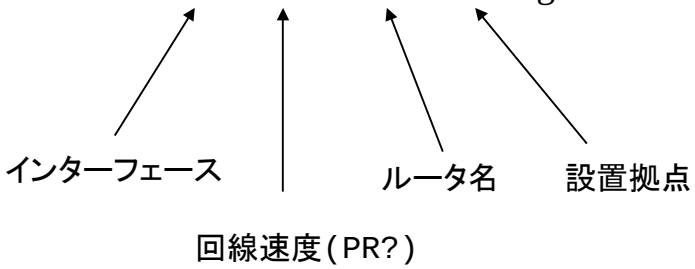
ルータに関する登録(続き)

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



ルータに関する登録(続き)

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



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。
- 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。
- U.S.の大きなISPは拠点名にIATAの空港コードをつけるのが好きらしい。
  - sea,sjc,nyc…
  - 大手町にあってもnrt!

$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。

-t(chroot) オプション

- named -t
  - いわゆるsandboxにchroot()して動作させる。
  - namedのプロセスを不正に操作されたときに、ファイルシステム中のアクセスできる範囲を制限することによりセキュリティを向上させる。

-t(chroot)オプション(続き)

- BIND8のnamedでslaveをするときは、内部からnamed-xferを起動する。
  - sandboxの下にnamed-xferをインストール。
  - ライブラリをダイナミックリンクするOSなら、共有ライブラリもsandboxの下に。
  - 必要なファイルはlddで調べる。
  - あるいはスタティックリンク。

-t(chroot)オプション(続き)

- ログはfileチャネルでとる。
- あるいはsyslogdが*$SANDBOX*/var/run/logを聴くように工夫。
  - FreeBSDならsyslogd -l、OpenBSDは-aらしい。
  - その種のオプションがなければ別プロセスのsyslogd。
  - BIND8は最初からログがとれない。
  - BIND9はrndc reloadなどを実行すると、それ以降。
- ログを見て/dev/randomなど、ないと言われたデバイスをmknodする。

-t(chroot)オプション(続き)

- 「/varをわける」をどう実現するか?
  - *BSDならnullfs?
    - でもコードはメンテナンスされてないみたい…。
  - *BSD用力技
    - FDDにnewfs
    - raw deviceをddで読んでUFSイメージを/varに書き出す。
    - vnconfigして、*$SANDBOX*/varにmount。
  - 他のOSでは…(未検証)
    - *$SANDBOX*は/varの下に構築。
    - ln –s /etc/RCS *$SANDBOX*/etc
    - *$SANDBOX*/etc/RCS/named.conf,vの実体は/varの外に。
    - CVSを使えばもうちょっとエレガントかも。

-u(setuid)オプション

- named -u
  - UIDを変更して動作させる。
  - rootの特権を放棄して、namedのプロセスを不正に操作さ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。
  - BIND8には-g(setgid)もあったが、BIND9ではなくなった。

-u(setuid)オプション(続き)

- named.pidが/var/runの下に書けない。
- BIND8だと/var/run/ndc(ソケット)も作れない。
  - named.confでそれぞれ適切に指定。
  - 実は-uは-tと親和性が高い。
- 起動時はrootがnamed.confを読めないといけない。
- (r)ndc reloadなどを実行するときは、setuid後のユーザがnamed.confを読めないといけない。
  - パーミッションに注意。
  - rndc-keyも。

どのバージョンのBINDを使う?

- BIND8 v.s. BIND9
  - 「普通に使うならBIND9」らしい。
    - 積極的にBIND8を選ぶ理由は??
  - 的確なアップデートが受けられるなら、OSに付属のバイナリを使っておくのも悪くない。
    - 起動時メッセージなどでバージョンを確認。
    - `dig version.bind. TXT CHAOS`

```
;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "9.2.1"
```

どのバージョンのBINDを使う?(続き)

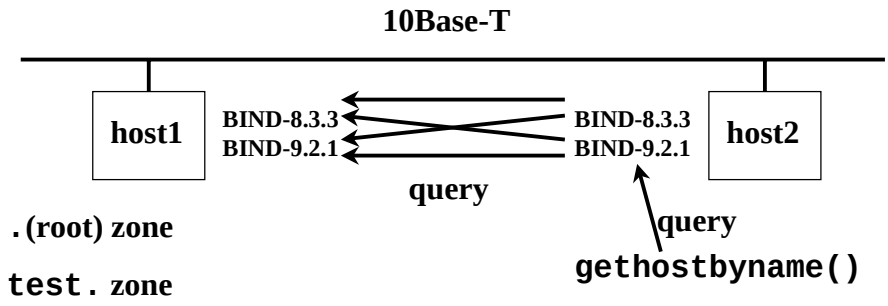
- www.isc.orgでは
  - BIND9.0.xのころはearly deploymentと書かれていた。
  - 今はBIND9がCurrent release、BIND8はAlso in wide releaseと書いてある。
- 8.2.5以降、BIND8のリリースアナウンスには The recommended version to use is BIND 9.x.xと書いてある。

どのバージョンのBINDを使う?(続き)

- マイナーバージョン
  - 基本的に最新を使っていれば問題ない。
  - ここのところ最新でenbugしたのは8.3.0だけ。
    - 特定の条件が成立すると過大なqueryを発生。
    - http://www.nic.ad.jp/ja/topics/2002/20020207-01.html
  - 逆に何か問題があって新バージョンがリリースされていることも多い。
    - 特にセキュリティ問題の場合は迅速にバージョンアップすることが必要。

BIND9は遅い?

- 測ってみました。



.(root) zone

test. zone

```
test-0-0.test.       IN  A  10.255.0.0
                     :
                     :
test-255-255.test.   IN  A  10.255.255.255
```


BIND9は遅い?(続き)

- 環境
  - host1,host2とも
    - Pentium 166MHz
    - FreeBSD 4.6.2-RELEASE
- root zoneをfakeして閉じた名前空間を形成。
- テストプログラム
  - test-0-0.test.～test-255.255.test.をgethostbyname()して所要時間を測定。
  - gethostbyname()はlibcの物。

BIND9は遅い?(続き)

- 実験1
  - host1でテストプログラムを10回実行。

host1	
8.3.3	97.4
9.2.1	128.2

(sec)

BIND9は遅い?(続き)

- 実験2
  - host2でテストプログラムを10回実行。
  - TTL=0(host2でキャッシュさせない)

host1 \ host2	8.3.3	9.2.1
8.3.3	201.0	308.8
9.2.1	200.1	364.2

(sec)

BIND9は遅い?(続き)

- 実験3
  - host2でテストプログラムを11回実行。
    - 最初の1回の値は捨てる。
  - TTL=1800(host2でキャッシュさせる)

host1 \ host2	8.3.3	9.2.1
8.3.3	83.7	115.2

(sec)

BIND9は遅い?(続き)

- 結論
  - この実験の範囲では「遅い」。
    - 実験1と実験3は自分が知っている名前をサービス。
      - 実験3の方が速いのは個体差?
    - 実験2は自分が知らない名前をqueryしてクライアントにサービス。
      - host2がBIND-8.3.3のケースはhost1のバージョンによらず、ほぼ同じ。
        - » どう理解すべきか?
- でもこれが問題になるのはどういうケース?
  - root、TLD、メーリングリストサーバ…

下位層でかせぐ

- m.root-servers.net.(AS7500)
  - いわずと知れたrootサーバのうち1台。
- ns3.apnic.net.(AS4777)
  - APNICのアドレスブロックの逆索きサーバ。

  がjpixとNSPIXP2に足を出している。
  - peeringしておけばqueryが速くなる(はず)。

further readings

- RFC1033
  - DOMAIN ADMINISTRATORS OPERTIONS GUIDE
- RFC1035
  - DOMAIN NAMES – IMPLEMENTATION AND SPECIFICATIONS
- RFC1912
  - Common DNS Operational and Configuration Errors

further readings(続き)

- RFC2308
  - Negative Caching of DNS Queries(DNS NCACHE)
- RFC2317
  - Classless IN-ADDR.ARPA delegation
- BIND9 Administrator Refence Manual
  - BIND9.xのソースのdoc/arm(XML,HTML版)
  - http://www.nominum.com/resources/documentation/Bv9ARM.pdf(PDF版)

Errata

事後公開版限定ボーナストラック :-)

検索の流れ(続き)[スライド#22]

- 誤
  - rootサーバだけは設定ファイルで決め打ち。
- 正
  - rootサーバだけは設定ファイルで決め打ちだが、少なくともBINDの実装では以下のように動作する。
    - namedの起動時に設定ファイル(named.root)を参照して「rootサーバ」を1台選ぶ。
    - 選んだ「rootサーバ」に対してrootサーバの一覧を要求する。
    - 以降の動作にはnamed.rootの内容は使わず、起動時に動的に得た一覧を使用する。

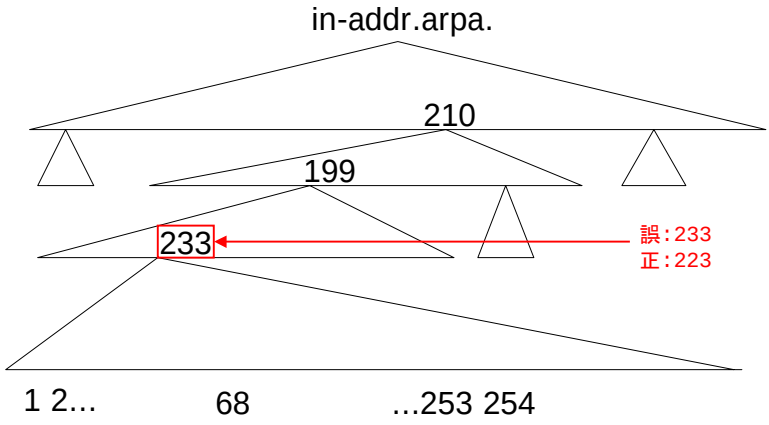
逆索き(続き)[スライド#28]

- ホスト名
  - [左]小さい単位(子)->大きい単位(親)[右]
- IPアドレス
  - [左]大きい単位->小さい単位[右]
    - 大きい単位:ネットワーク部
    - 小さい単位:ホスト部
- 逆順で表記
  - 86.233.199.210.in-addr.arpa.
  - 53.0.0.(省略).0.8.9.a.b.c.d.e.f.4.0.5.0.e.f.f.3.ip6.arpa.
    - ip6.int.からip6.arpa.に移行中。

誤:233
正:223

誤:53.0.0
正:3.5.0.0

逆索き(続き)[スライド#29]



Dec/16/2002

BINDのインストール[スライド#79]

誤:conifgure
正:configure

- ftp://ftp.isc.org/isc/bind9/からget。
  - 資料作成時点では9.2.1が最新リリース。
- コンパイル時設定はconifgureを使う。
  - FreeBSD/NetBSDでは苦労せずmakeできた。
  - ドキュメントには他に
    - AIX 4.3, Tru64 4.0D, Tru64 5, HP-UX 11, IRIX64 6.5, Solaris 2.6, 7, 8, Red Hat Linux 6.0, 6.1, 6.2, 7.0

    がSupported Operating Systemsと書いてある。
  - --sysconfdir=/etcがお勧め。

Dec/16/2002

設定例の想定[スライド#86]

- ドメイン名
  - don.gr.jp.
- IPアドレス
  - 172.16.7.152/29
  - 3ffe:504:fedc:ba98::/64
- 各ゾーンのslaveはns.mynsp.ad.jp[10.12.34.56]を想定したネームサーバの設定例。

誤:mynsp
正:myisp

89ab.cdef.4050.eff3.ip6.arpa[スライド#103]

```
$TTL 1d
@   IN  SOA  oyako.don.gr.jp. hostmaster.don.gr.jp. (
             2002121601
             1h
             20m
             1000h
             15m )
    IN  NS   oyako.don.gr.jp.
    IN  NS   ns.myisp.ad.jp.
25.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0  IN  PTR  negitoro.don.gr.jp.
53.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0  IN  PTR  oyako.don.gr.jp.
```

誤:25.0.0, 53.0.0
正:5.2.0.0, 3.5.0.0