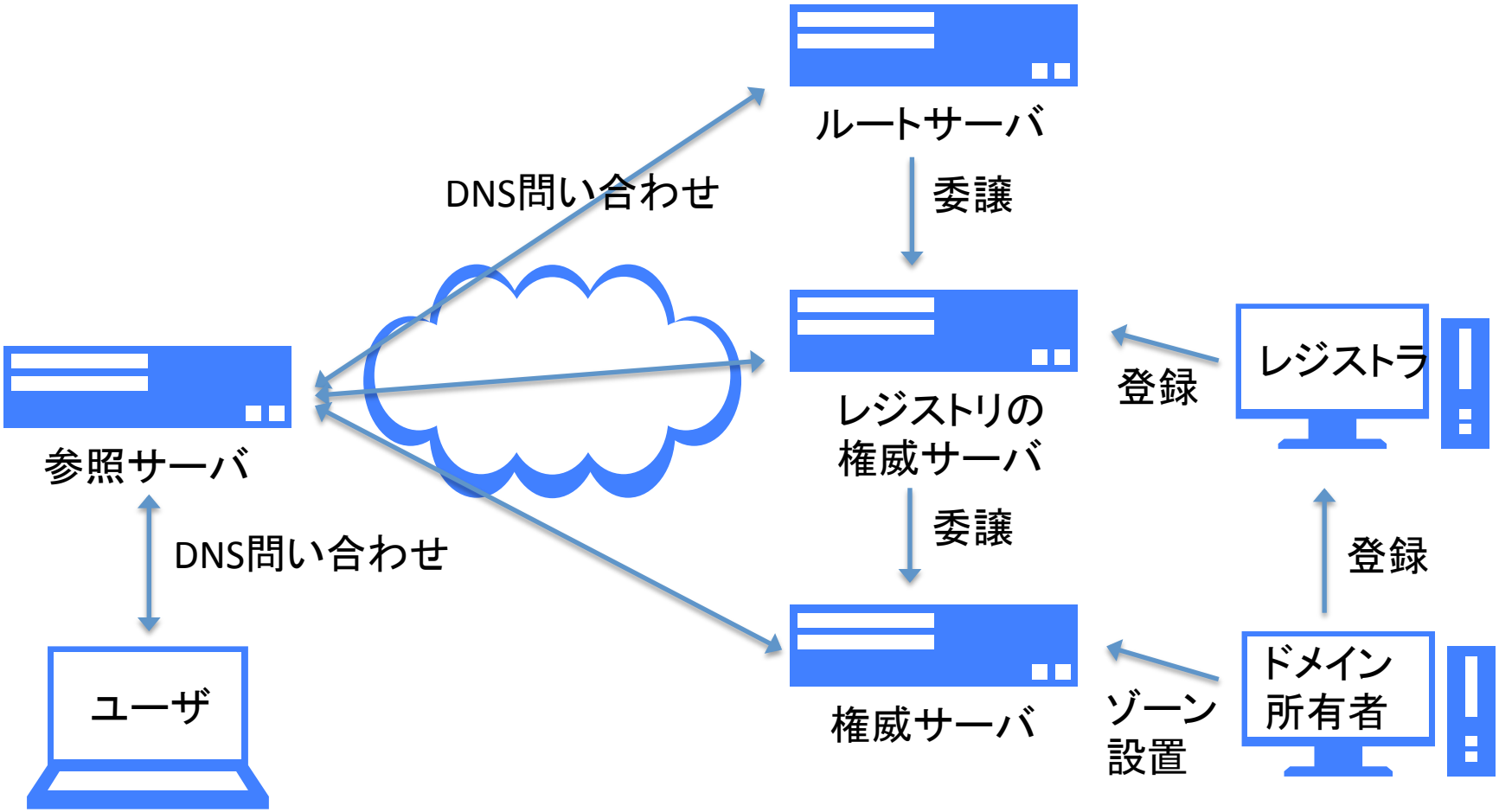
DNSトラブルシューティング

IIJ 山口崇徳

DNSの関係者(1)


ルートサーバ
DNS問い合わせ
委譲
参照サーバ
レジストリの
権威サーバ
登録
レジストラ
DNS問い合わせ
委譲
登録
ユーザ
権威サーバ
ゾーン
設置
ドメイン
所有者

DNSの関係者(2)

- なんかいっぱいいる…
  - それぞれ役割が異なる
  - それぞれの場所で固有のトラブルが発生しうる
  - どこでトラブルが起きているのか見極めるのが重要
- 自分はその中のごく一部にしか関われない
  - トラブルの原因は特定できても、それが自分では手の出せないところにあることも多い
    - 原因となっているところが直してくれるまで、何もできずに眺めているしかできない
  - そんなわけで、問題の解決に至らず、原因の特定までで終わってしまうケースも多々あり
    - むしろその方がはるかに多いような気も…
    - このセッションの「DNSトラブルシューティング」というタイトルは嘘です:-)

DNSの関係者(3)

- 参照サーバ
  - いわゆるキャッシュサーバ、recursive server のこと
  - /etc/resolv.conf に書くDNSサーバ
- 権威サーバ
  - コンテンツサーバ、authoritative server のこと
  - NSレコードに書くDNSサーバ
- ユーザ
  - インターネットで遊ぶみなさま
- 今回はこの三者で起きるトラブルについて話します
  - ほかのところで起きることもあるけど、ふつーの人がその立場になることはまずないので

アジェンダ

- DNS一般
- 参照サーバのトラブル
- 権威サーバのトラブル
- トラブル調査実践
- クライアント側のトラブル

- DNSSECに特化したことはほとんど話しません

DNS一般編

まず、道具をそろえよう

- dig, drill
  - ブラウザでアクセスできたらおっけー、とかはダメ
- サーバのログ
- 各種監視/統計ツール
  - SNMP
  - nagios, zabbix, cacti, munin, ...
  - DSC

dig の使い方

- `dig @server domain type opt`
  - server: 問い合わせ先サーバ
  - domain: 知りたい名前
  - type: 知りたいタイプ(NS, MX, ... ; 省略時 A)
  - opt: 各種フラグのセットなど
    - たくさんあるけど比較的よく使うもの
    - +norecurse (+norec) 再帰検索しない
    - +edns=0 EDNS0 有効で問い合わせ
      - +bufsize=4096 EDNS0 の最大パケットサイズ
    - +tcp TCP で問い合わせ
    - +dnssec DNSSEC OK
    - +trace ルートサーバから委譲関係を辿る
    - +nssearch すべての NS から SOA レコードを検索する
  - 引数の順番はこの通りでなくてもよい

dig の結果

```
% dig www.iij.ad.jp @ns11.iij.ad.jp

; <<>> DiG 9.7.3-P3 <<>> www.iij.ad.jp @ns11.iij.ad.jp
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61830
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;www.iij.ad.jp.                 IN      A

;; ANSWER SECTION:
www.iij.ad.jp.          277     IN      A       210.130.137.80

;; AUTHORITY SECTION:
iij.ad.jp.              86266   IN      NS      dns0.iij.ad.jp.
iij.ad.jp.              86266   IN      NS      dns1.iij.ad.jp.

;; ADDITIONAL SECTION:
dns0.iij.ad.jp.         4582    IN      A       210.138.174.16
dns0.iij.ad.jp.         4582    IN      AAAA    2001:240:bb41:8002::1:16
dns1.iij.ad.jp.         1557    IN      A       210.138.175.5
dns1.iij.ad.jp.         1557    IN      AAAA    2001:240:bb4c:8000::1:5

;; Query time: 51 msec
;; SERVER: 2001:240::3#53(2001:240::3)
;; WHEN: Thu Aug 23 19:44:41 2012
;; MSG SIZE  rcvd: 173
```

← ヘッダ

←QUESTION セクション

←ANSWER セクション

←AUTHORITY セクション

←ADDITIONAL セクション

←問い合わせにかかった時間や応答サイズなど

dig の結果の読み方(1)

- 基本的に、バイナリである DNS のパケットを可視化しているだけ
- いくつかのセクションに構造化されている
  - ヘッダ
    - 問い合わせ結果のステータスや、各種フラグなど
  - QUESTION セクション
  - ANSWER セクション
  - AUTHORITY セクション
  - ADDITIONAL セクション
  - 問い合わせにかかった時間や応答サイズなど
- 問い合わせ結果は ANSWER セクションに入るのでそこに目がいきがちだが、他の部分も重要な情報

dig の結果の読み方(2)

- QUESTION secion
  - 問い合わせた内容
- ANSWER section
  - 問い合わせた結果に対する回答
- AUTHORITY section
  - 権威を持っているサーバの情報
- ADDITIONAL section
  - 付加的情報(権威サーバの A/AAAA)
- かならずしもすべてのセクションが埋まっているとは限らない
  - authority や addtional セクションはパケットサイズを小さくするために省略されることがある

dig の結果の読み方(3)
status (RCODE)

- NOERROR: 正常
- NXDOMAIN: 問い合わせた名前は存在しない
- SERVFAIL: エラーが発生した
  - 再帰検索中にどこかの権威サーバが無応答でタイムアウトした
  - DNSSEC validation に失敗した、など
- REFUSED: 問い合わせが拒否された
- FORMERR: フォーマット不正
- これが全部ではないけれど、よく見るのはこれぐらい

dig の結果の読み方(4)
status (RCODE)

- とくに異常がなければNOERRORかNXDOMAINが返る
  - 参照サーバでは、これらの応答が返ってきたものをキャッシュする
- answerセクションが空である = NXDOMAIN ではない
  - Aレコードを聞かれたんだけど、AじゃなくてMXならあるんだけどなぁ
  - その名前はうちは権威を持ってないから知らないよ、よそに委譲してるからそいつに聞いてくれ
  - …というときはNXDOMAINではなく、answer が空でもNOERRORになる
  - AもMXもNSもそれ以外もどれも存在しない、というときだけNXDOMAIN
- NOERROR, NXDOMAIN 以外の応答は、どこか異常あり
  - どんな異常なのかは状況によりさまざま

dig の結果の読み方(3)
flgas

- aa: 権威応答
  - 権威サーバからの応答には必ずあるはず(超重要。なければ設定がおかしい)
  - 参照サーバからの応答にはない(ある場合はローカルでゾーン定義されている)
- tc: 512バイト(またはEDNS0の最大サイズ)で収まらなかった
  - ので、TCP で聞き直してくれ
  - 通常は自動で TCP で聞き直すので tc フラグつきの応答は見えない
  - dig +ignore で検索すると、TCP でリトライしないので tc フラグが見える
- rd: 再帰検索を要求された
  - dig +norec で検索すると応答から rd フラグが消える
- ra: サーバは再帰検索をサポートしている
  - 参照サーバからの応答にはあるはず
  - 参照サーバを兼ねていれば権威サーバにもあるかもしれない
  - 参照サーバを兼ねていない権威サーバにはない
- ad: DNSSEC validation に成功した

drill

- NSD, Unbound の開発元 NLNetLabs によるDNS問い合わせツール
  - ldns というライブラリに付属してます
  - http://www.nlnetlabs.nl/projects/ldns/
  - dig (掘る) → drill (穴をあける) の連想か
    - ディグダグ→ ミスタードリラーみたいなもん(違)
- オプションの指定のしかたは dig と異なるが、ほぼ同じように使える
  - 結果の読み方も同じ
  - BIND に愛想が尽きた人にもオススメ
  - drill は漢のロマンだしね!

nslookup

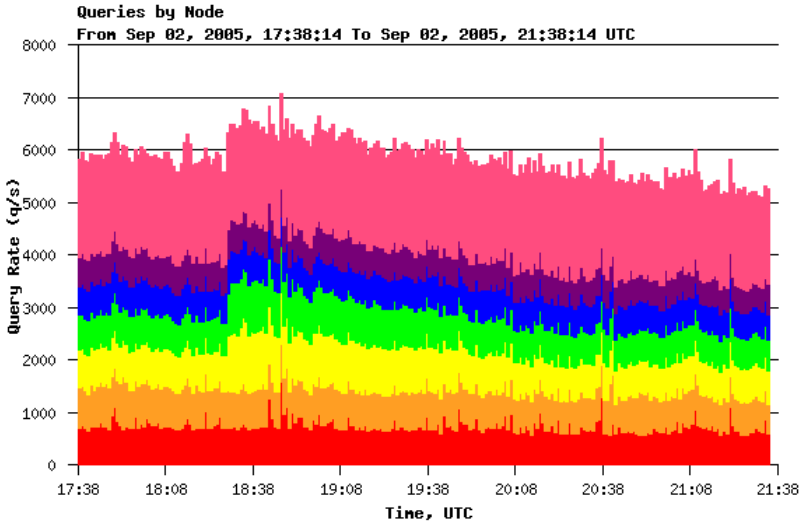
- 基本的に、使いません
- とくに異常がないときに、名前解決の結果を知りたいだけならば nslookup でも十分用は足りる
- が、トラブル解決の道具という意味では、必要な情報が隠蔽されてしまったり、細かいオプションを指定できなかったりで使いづらい
- 残念ながら Windows には nslookup しかないので、自前で dig をインストールしておきましょう
  - isc.org から Windows 版 BIND をダウンロードすると中に dig.exe も入ってます
    - nslookup.exe も入ってるけどなー(いらんてば)
  - 残念ながらdrillの Windows 版はないみたい

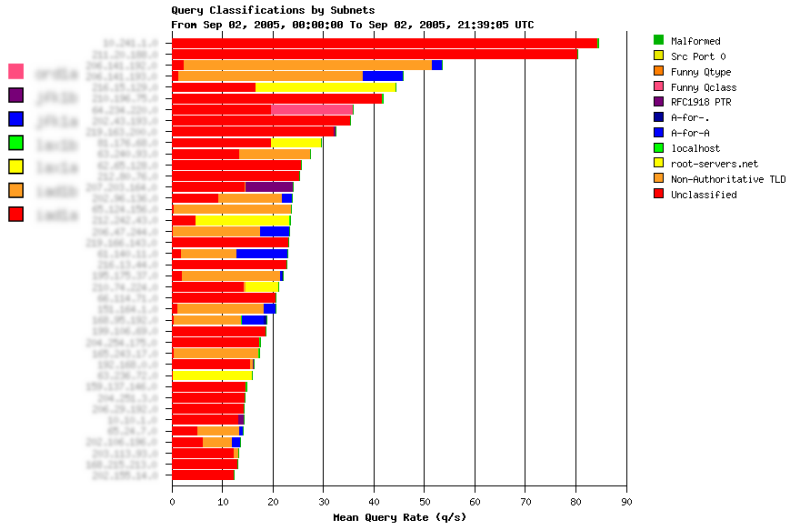
サーバ監視

- ちゃんと監視しましょうね
  - WebやらメールやらDBやらと考えかたが大きく異なるわけではない
  - nagios その他、ふだんから使い慣れているものでよい
- UDPのトラフィックを計測するとよい
  - DNS 専用のサーバならほぼ UDP のパケット数 ≒ DNS のクエリ数
  - TCP の DNS パケットもあるし、DNS 以外の UDP パケットも存在しないわけではないので厳密な値ではないが、傾向を掴むには十分
    - 厳密な情報が必要なら DSC をインストール

DSC

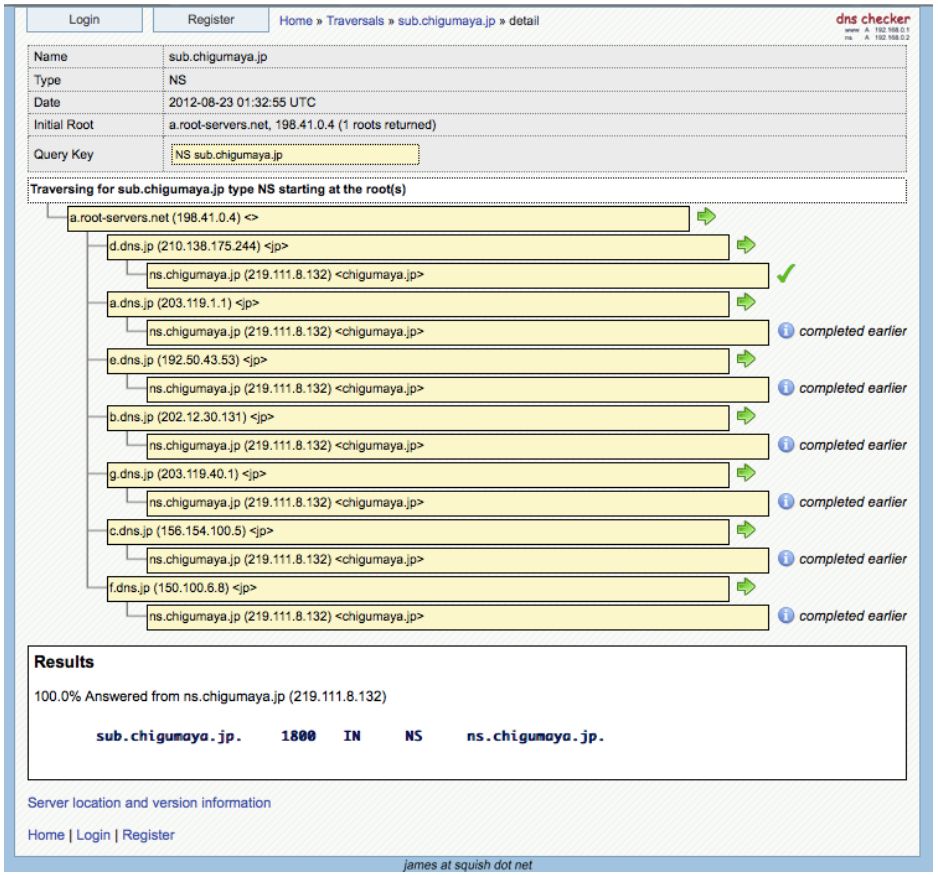
- DNS STATISTICS COLLECTOR
  - http://dns.measurement-factory.com/tools/dsc/
  - DNS に特化した統計情報取得・グラフ作成ツール
    - 障害通知(異常なクエリを検知してアラートを上げたりとか)はしない
  - BIND、NSD、Unbound その他実装に依存せず利用可能





便利なWebサービス(1)

- squish.net dns checker
  - http://dns.squish.net/
  - ルートサーバから再帰的に名前解決した結果を視覚的に表示してくれる
  - 設定に問題があるサーバをお知らせ
    - 権威サーバなのに再帰検索できちゃう、とか
    - 同じことを問い合わせてるのにサーバによって応答が違うぞ、とか



便利なWebサービス(2)

- DNSの設定チェック
  - http://dnscheck.jp/
  - 指定したドメインのDNS設定が適切かどうかをチェック
  - ドメイン名だけでなく、DNSサーバを指定できる
    - これからネームサーバ移行しようとするときに、引っ越し先のサーバ設定が正しいか事前チェックするのにも使える

ネットワークの問題(1)

- DNSはTCPも使う
  - ゾーン転送は TCP のみ
  - ゾーン転送以外でも UDP → TCP のフォールバックがある
- DNSはUDPでも512バイト以上のサイズになることがある
  - EDNS0 をサポートしているとき
- TCPだけでなく、UDPでもパケットがフラグメントすることがある
  - DNSSECの鍵ロールオーバー中だけUDPフラグメント→再構成できずに名前解決に失敗、なんて現象が起きることがある
- ソースポートは53以外も使う
  - 大昔は query-source port 53; なんて設定がよくされていたけど…
  - 今では脆弱性

ネットワークの問題(2)

- 権威サーバからクライアントまで、DNS パケットが通るすべての経路でクリアされている必要がある
  - ファイアウォールで止めたりしていないか確認
    - named.conf をいくら眺めても解決しません
  - 家庭用ルータではこのへんの挙動があやしいものが多いようだ
- どうやって調べるの?
  - dig でオプションを変えながら問い合わせしてみる
  - tcpdump などでパケットキャプチャするのもよい

ネットワークの調査(1)

- dig +edns=0 で問い合わせると SERVFAIL, FORMERR, NOTIMPL などの応答が返る
  - サーバがEDNS0に対応していない
    - 解釈できないものは素直にエラーにする実装
    - 非EDNS0なUDPやTCPで名前解決できるなら効率は落ちるが支障はない
- 大きな応答を返す名前に対して dig +bufsize=4096 で問い合わせるとTCPにフォールバックする
  - `;; Truncated, retrying in TCP mode.`
  - サーバがEDNS0に対応していない
    - 解釈できない部分をとりあえず無視する実装
    - TCPで名前解決できるなら、効率は落ちるが支障はない
  - または応答が4096バイトを越える
    - +bufsize=.... の値を大きくしてもう一度

ネットワークの調査(2)

- 大きな応答を返す名前に対して dig +bufsize=4096 で問い合わせるとタイムアウトしてしまう
  - 通信経路上のどこかが512バイト以上のUDPを通さない
  - あるいは、フラグメントしたUDPパケットの再構成に失敗している
    - EDNS0の最大サイズをMTUよりも小さな値(1400程度)に設定
      - edns-udp-size (BIND), ipv4-edns-size, ipv6-edns-size (NSD) , edns-buffer-size (Unbound)
- TCP にフォールバックした後でタイムアウトする、connection refused と言われる
  - TCPに対応していない
  - 53/tcp をファイアウォールで閉じている
- sudo dig -b 0.0.0.0#53 で問い合わせたときだけ応答がある
  - src port 53 以外の DNS を蹴るファイアウォールがいる

参照サーバ編

名前解決失敗の原因

- 大きくわけてふたつ
  - 参照サーバ側の問題(自分のせい)
  - 権威サーバ側の問題(他人のせい)
- 参照サーバ側の設定不備などで名前解決できないことは実はそれほど多くはない
- 権威サーバの問題は自分ではどうしようもないことが大半
  - が、キャッシュの関係で「よそではひけてるのに、うちではダメ、おかしい」と文句を言われることがある
  - どうしようもなくても、問題の切り分けはできるように

困ったらキャッシュをクリア?

- 古いキャッシュが残ってるせいで失敗している場合はそれで解決できるかも
- 複数台の権威サーバの情報が一致してない場合は、たまたま正しい情報を持ってるサーバに問い合わせるまで何度かキャッシュクリアを繰り返してみる、という手が使えるかもしれない
  - とはいえ、情報に食い違いがある時点でかなりアレ
  - 食い違った情報のどちらが正しいのか第三者には判断しづらいことも
    - たぶん SOA serial の大きい方なんだろうけど…
- 逆に、権威サーバはおかしいけれど、古いキャッシュが残ってるおかげで運よく名前解決できてるという場合もある
  - キャッシュクリアすると以後コケるようになるかも
  - それでコケても「あるべき状態になる」だけなので、挙動としては間違いではない(利用者は困るかもしれないけど)

キャッシュの消しかた(推奨)

- BIND

```
# rndc flushname <name>
# rndc flushtree <name>
```

(9.9以降)

- Unbound

```
# unbound-control flush <name>
# unbound-control flush_type <name> <type>
# unbound-control flush_zone <name>
```

- 指定した名前のキャッシュだけを消す
  - 「ひけない名前」ではなく、その「ゾーンを持っている権威サーバ」のキャッシュを消す必要がある場合もあるので注意

キャッシュの消しかた(非推奨)

- BIND

```
# rndc flush
```

- Unbound

```
# unbound-control reload
```

- どちらもキャッシュされているすべての情報が捨てられる
  - とくに問題が起きていない大多数のキャッシュまで闇に消えてしまう
  - キャッシュの有無によって応答速度が数倍～数十倍違うので、できるだけ大事にしましょう
  - キャッシュされてるどの情報が悪さしてるのかわからんときは、全削除もしかたない
    - 幽霊ドメイン問題……(ためいき)

特定の名前解決がSERVFAILする

- …しかもSERVFAILの結果が返ってくるまでさんざん待たされる
  - 名前ひけなくない? のもっとも典型的な例
- 問い合わせた名前にルートサーバから再帰検索する過程の権威サーバのどれかが落ちている可能性大
  - 問い合わせがタイムアウトするまで応答を待つので遅くなる
  - 単純に障害の場合もあれば、ドメインの委譲関係がおかしくなっていて(lame delegation)、ゾーンの管理者が意図しないところに問い合わせが出ていることも
  - 権威サーバの管理者が直してくれないことにはどうしようもない
    - すでに直したようだが古いキャッシュが生きてるのでダメ、という場合はキャッシュ削除で解決
    - そうでないなら、うちは悪くないから諦めて、というしかない

大量クエリ(A)

- 狂ったように同じリクエストを投げつけてくるクライアント
  - その聞いているタイプがA/AAAAだった場合
- たいていは、何らかのアプリのバグ
  - マルウェアの可能性も
- 最近はPCの処理性能が非常に高いので、全力で連続クエリを投げられるとかなりヤバい
- とはいえ、サーバを監視して、見つけたらアクセス制限するくらいしか対処方法がない…
  - ちゃんと日頃から監視しておきましょう
  - 組織外から参照サーバへの問い合わせははじめから受けつけないようにしておくとよい

大量クエリ(ANY)

- 狂ったように同じリクエストを投げつけてくるクライアント
  - その聞いているタイプがANYだった場合
- DDoS攻撃の踏み台に使われている可能性大
- DNS amplification attack (DNS amp)
  - DNSはUDPを使うのでソースアドレスの詐称が容易
  - DNSは問い合わせのサイズは小さいが、応答は比較的大きくなることがある
  - ソースアドレスを詐称して大量のANYクエリを送る→詐称されたアドレスにクエリの数倍～数十倍のサイズの応答を返す
  - このような詐称クエリを多数の参照サーバに投げることで、詐称されたターゲットのネットワークを飽和させる
- 攻撃に使われる名前は、なぜか isc.org であることが多い:-)
  - 応答サイズが大きければ別にどこでもいいはずなんだけれど

DNS amp 対策

- 参照サーバには自組織以外からのアクセスは許可しないようにする
  - 問い合わせのソースアドレスが攻撃ターゲット
  - 組織内からしかアクセスを許可しないようになっていれば、組織外のホストに対する攻撃の踏み台には使えない
  - 権威サーバと参照サーバは分離する
    - 分離しなくてもアクセス制限できなくはないが、設定が煩雑
- Ingress/Egress filterの導入
  - 組織内アドレスをソースに持つパケットは外部から来ないはず
  - 組織外アドレスをソースに持つパケットは内部から出ないはず
  - もしあれば詐称パケット: 通過を許さない
  - どちらかというと詐称パケットを内から外に出さないようにする対策

権威サーバ編

ゾーンの読み込みに失敗する

- たぶんゾーンファイルの文法エラー
  - ログその他にメッセージが出てるはず
- named-checkzone でチェック
  - ゾーンのロード前にチェックする習慣をつける
  - BINDではなくNSD を使う場合にも有効
    - ただし、BIND では使えるが NSD では使えない記法(連番生成用の $GENERATE など)は素通りしてしまうので注意
  - 文法的には間違っていないが、好ましくない記述を検出することも、ある程度は可能
    - MX や NS で指定しているホストの A レコードを定義していない、とか
- validns なんてチェックツールもあります
  - http://www.validns.net/
  - DNSSECの署名が正しいかというチェックも可能
- チェックツールも万能ではない
  - ゾーンファイルの文法的には正しければ、ゾーンを書いた人の意図と異なっていても通ってしまう

ツールで検出できない記述ミス(1)

- シリアル番号上げ忘れ
  - ゾーンファイルを編集するときに最初に変更する癖をつける
  - ゾーンをロードしたら slave にも反映されているか必ずチェックする
  - DNSSECを導入する:-)
    - 大半のDNSSEC管理ツールはシリアルのアップデートを自動化している
- ホスト名末尾の "." 付け忘れ
  - named-checkzone を -D 付きで実行すると(エラー/警告は出ないけど)気付きやすくはなるかも
    - example.jp ("."なし)を example.jp.example.jp. のように正規化して出力
    - $GENERATE も展開されるので、意図した連番が生成されているかのチェックもできる
      - NSDで連番を扱うためのプリプロセッサとしても使える

ツールで検出できない記述ミス(2)

- v6逆引きの桁数の過不足
  - 正しいのはどちら?

    1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

    1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
  - もはや人間の目でチェックできる域を越えている
  - なんらかのツールで変換したものをコピペする
    - BINDに付属する arpaname というコマンドで IP アドレスから in-addr.arpa/ip6.arpa 形式への変換が可能
- コメントアウトのつもりで行頭に # を書いてしまう
  - "#hoge.example.jp" みたいな名前が有効になってしまう…
- まだまだたくさん
  - チェックツールで検出できるものは残念ながらごくわずか

lame delegation とは?

- ゾーンを委譲する側と委譲される側の間に不整合がある状態のこと
- ただし、その状態が一時的なものであればとくに問題にならない
  - 障害で一時的に権威サーバにアクセスできない
  - master – slave 間のゾーン転送が完了していない
  - 権威サーバ移転のため委譲の情報を書き換える
- 少しぐらい不整合があっても動くようにDNSは設計されている
  - そういう状態が一時的ではなく、ずっと継続しているのがダメ
- 不整合があってもなんとなく動いてしまうDNSの堅牢性(あるいはいい加減さ)に頼ってはいけない
  - が、動いてしまうので気づきにくいんだよね…

lame delegation の影響

- 名前解決に時間がかかる、または失敗する
- 名前解決の再試行などによる無駄なやりとり
- などなど

- 権威サーバが原因になっているトラブルの大半は lame によるもの
- 「親子で委譲の情報を一致させる」という基本を徹底すれば防げる

浸透しないんですけど(1)

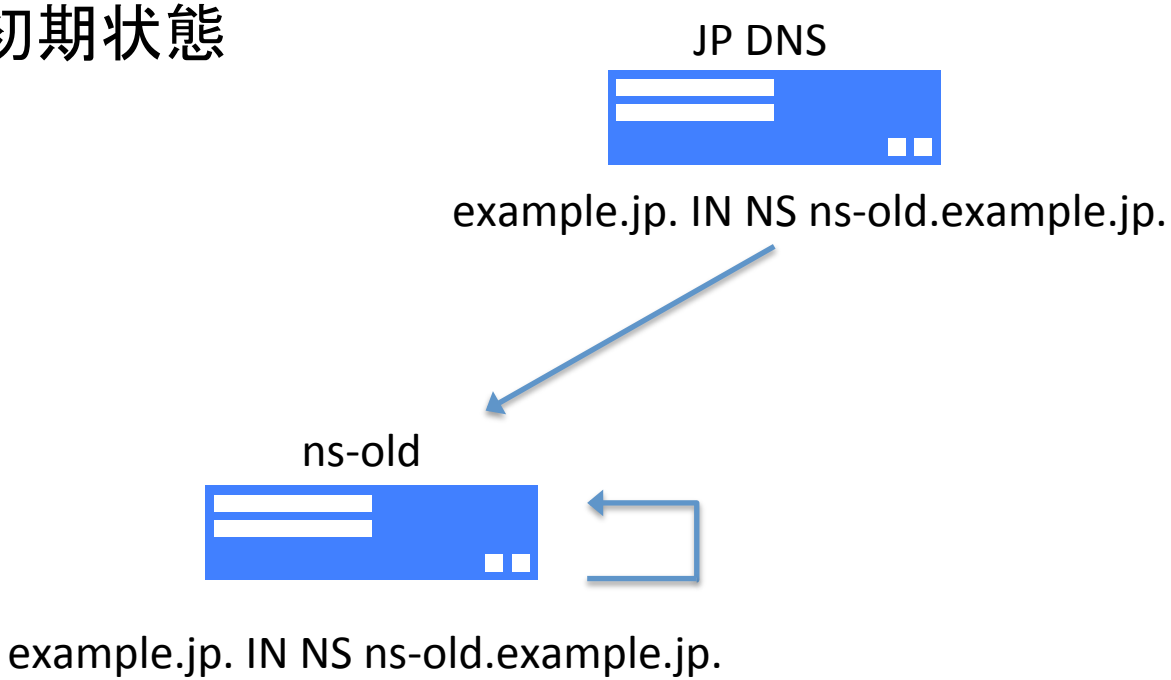
- 浸透いうな(お約束)
- 大昔からあった事象だが、ここ1年ぐらいの間に解説記事が一気に充実してきた
- ので、詳しくはそちらを参照してください
  - http://www.e-ontap.com/dns/propagation/
  - http://jpinfo.jp/topics-column/019.pdf
  - http://jprs.jp/tech/material/iw2011-lunch-L1-01.pdf
  - http://www.geekpage.jp/blog/?id=2011/10/27/1
  - http://internet.watch.impress.co.jp/docs/event/iw2011/20111201_494798.html
  - http://internet.watch.impress.co.jp/docs/special/20120227_514853.html
  - などなど

浸透しないんですけど(2)

- 「TTL を無視してキャッシュし続けるサーバ」のせいではない
  - そんなものがほんとに存在するなら、権威サーバの引っ越し以外にもいろんなところで問題になっているはず
  - 移転の手順が間違っているのが根本的な原因
- 「TTL を無視してキャッシュし続けるアプリ」というのはたしかに存在する
  - が、少なくとも権威サーバ引っ越しにともなう「浸透」遅れには無関係
  - 移転にともなって変更されるのは NS + glue だが、一般にクライアントアプリケーションは NS を検索しないのでキャッシュされることもない
    - A レコードの変更に追従しない、という現象があればアプリの問題かもしれない

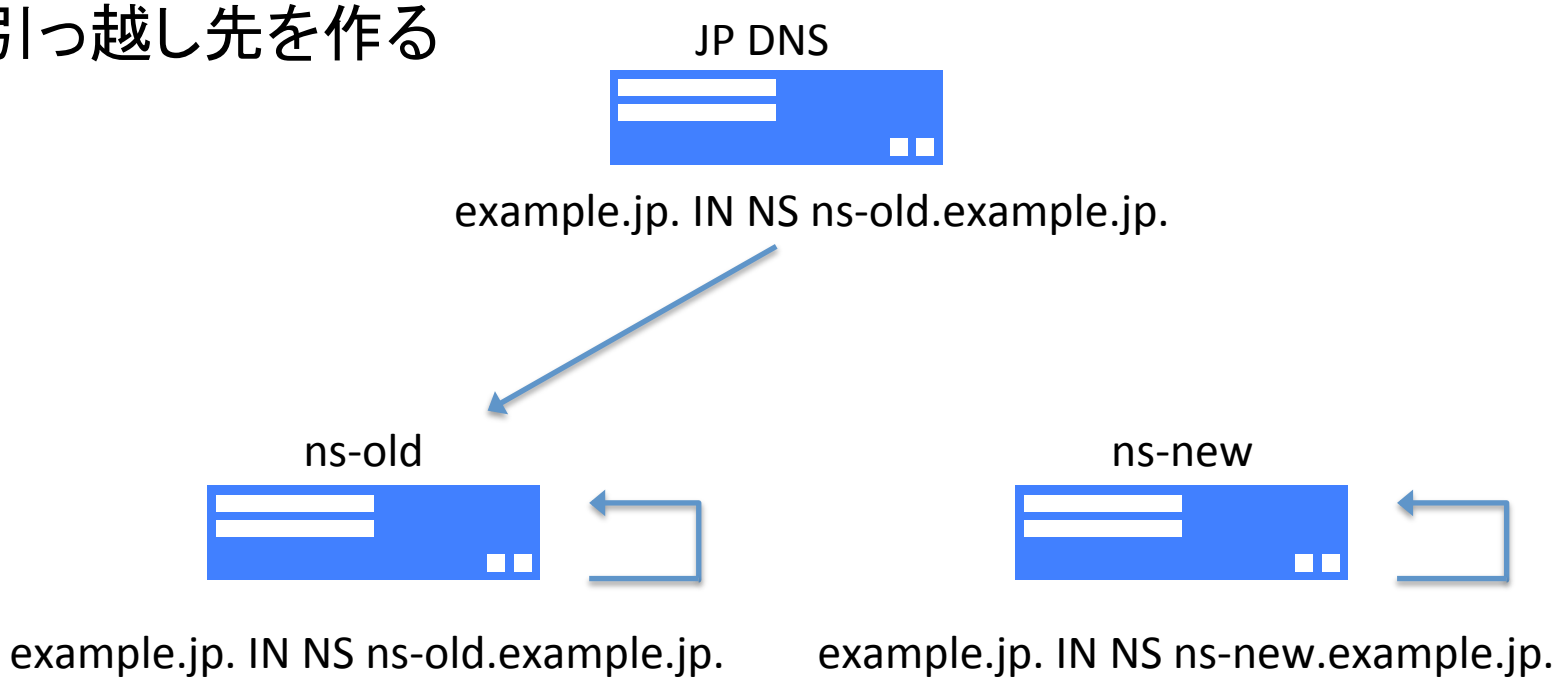
正しい引っ越し(1)

初期状態



正しい引っ越し(2)

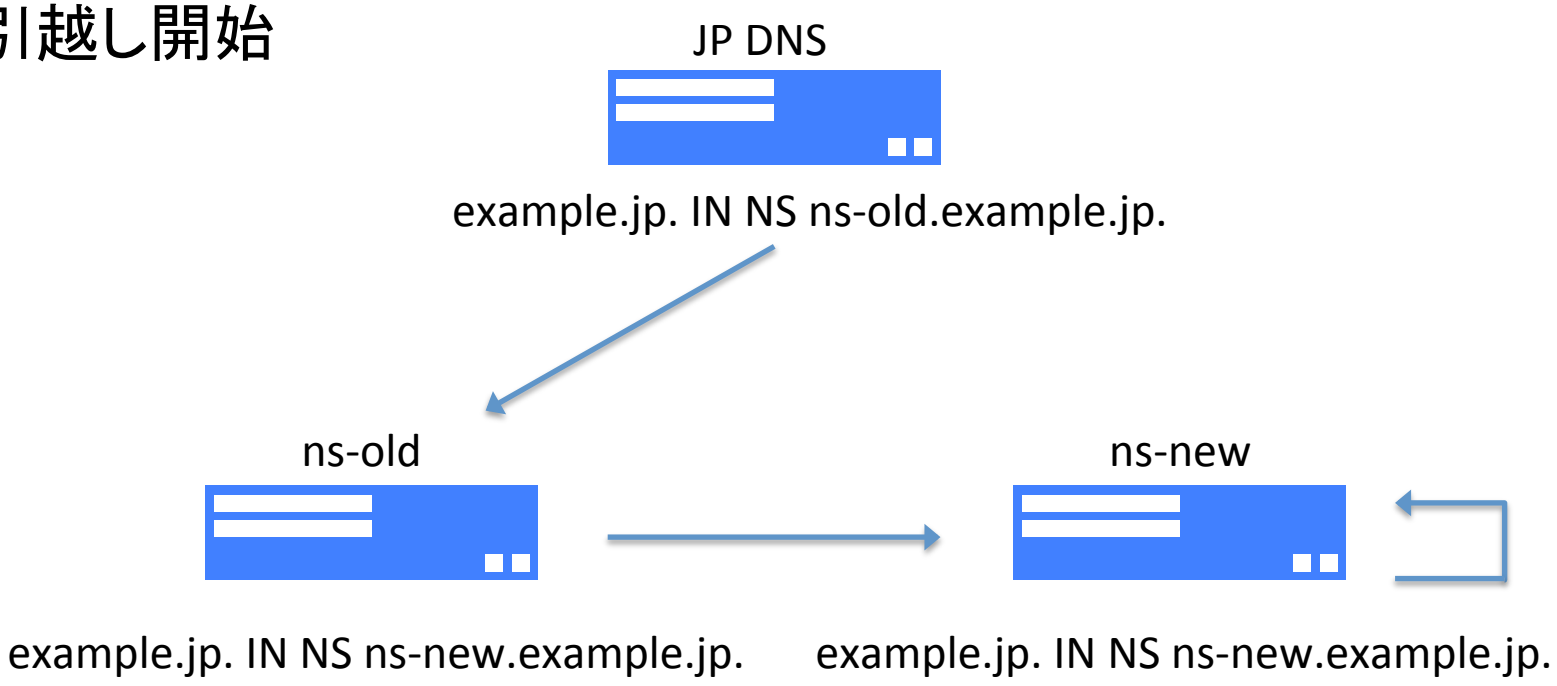
引っ越し先を作る



- 新設した方では NS を自分自身(ns-new)に向ける

正しい引っ越し(3)

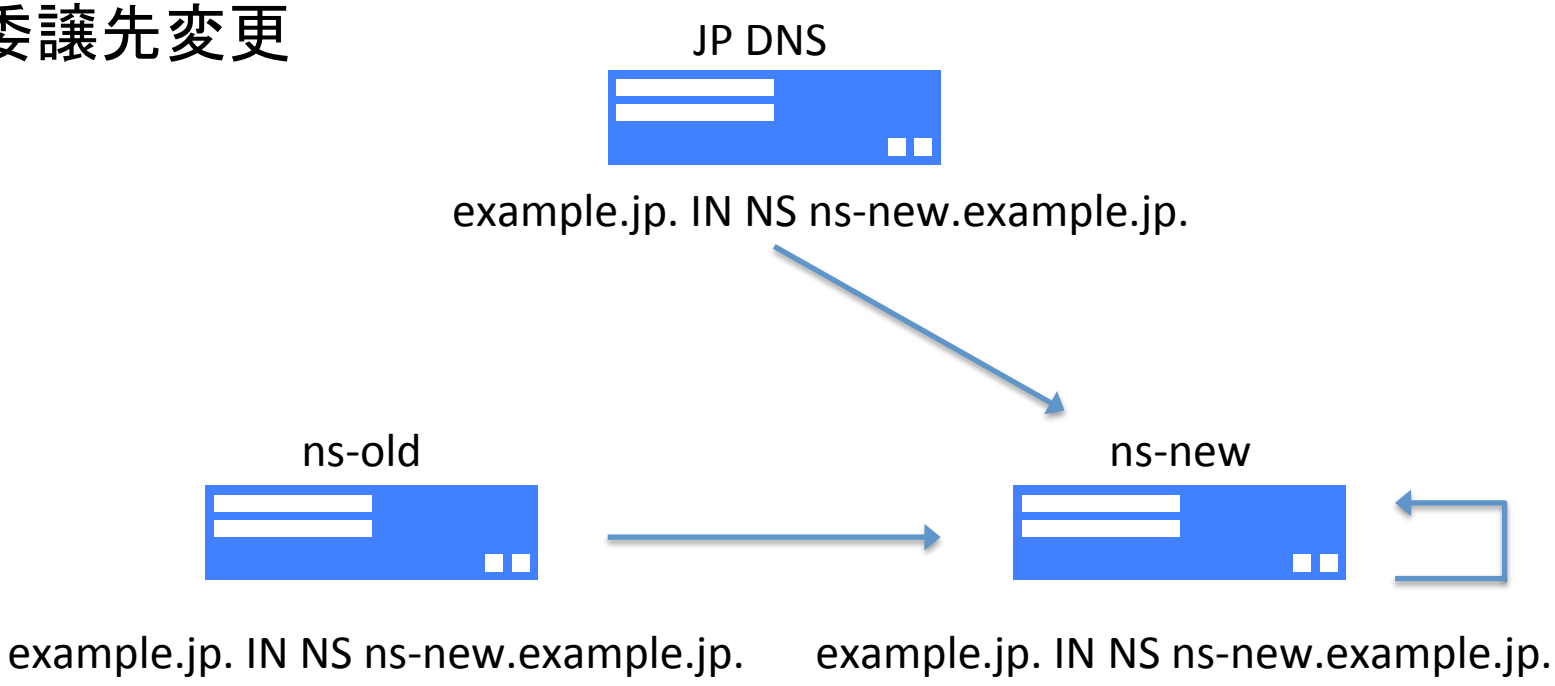
引越し開始



- 引っ越し元で NS を向けかえる
- 委譲の情報が一致していないが、とりあえずは問題ない
  - とはいえ、長期間このまま放置するのもよろしくない

正しい引っ越し(4)

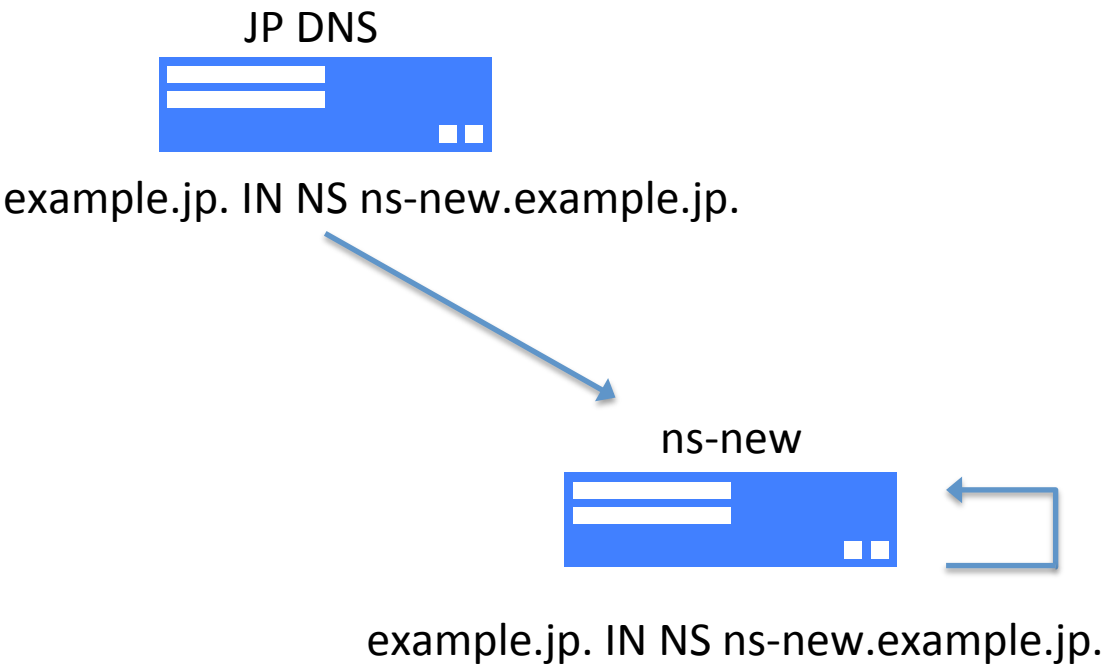
委譲先変更



- 委譲元(JP DNS)から引っ越し先に NS を向ける

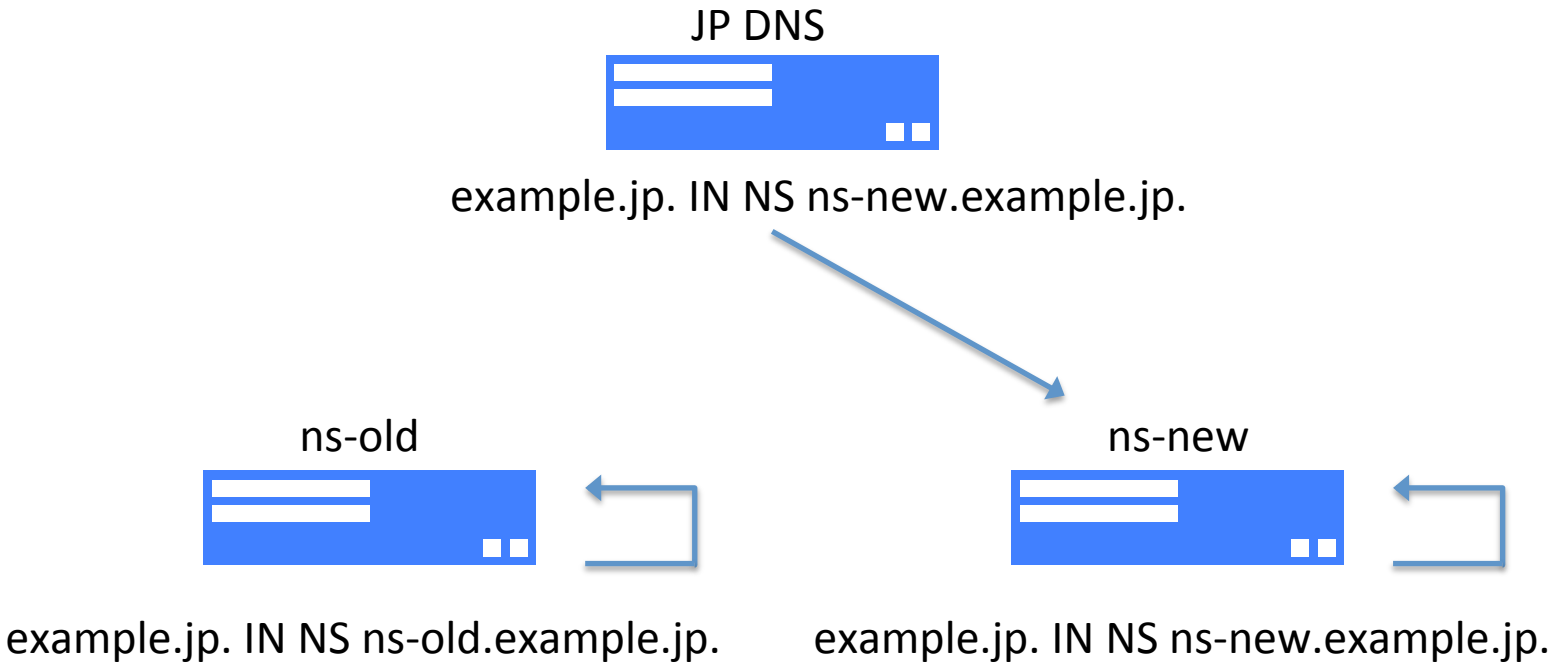
正しい引っ越し(5)

引っ越し完了



- ns-old で指定していた TTL が過ぎてキャッシュが消えてから、引っ越し前の ns-old を停止(またはゾーンを削除)する

正しくない引っ越し(1)



- 引っ越し前のゾーンを書き換えず放置

正しくない引っ越し(2)

- 何がいけないのか?
  - 新しい権威サーバに引っ越した後も、古い権威サーバは古い内容のまま動いている
  - 古い権威サーバの情報をキャッシュしている参照サーバは、新しい権威サーバが増えたことに気づかない
    - 古い権威サーバが返す応答に含まれる authority セクション内の NS (ns-old) により、キャッシュの TTL がリフレッシュされてしまう
      - 実装依存: BIND 9.2 以前などがそうらしい
    - 古い権威サーバの情報がいつまでもキャッシュから消えない
  - いつまでたっても ns-new に聞きにいかない
- 「古い権威サーバに新しい情報を載せる」ことが重要
  - もう使わないんだから旧サーバの持ってる情報は変えなくていいや
  - トラブルがあったときにすぐに切り戻せるように、いじらず残しておこう
  - …とか考えてしまうと失敗する

slaveに同期されない

- master – slave 間の疎通がない
  - 相手サーバのアドレスを間違えて設定してしまった
  - master で slave サーバからのゾーン転送要求を許可していない
  - UDP は通るが TCP が通らない
  - TSIG 鍵の不一致
- SOA のシリアル番号上げ忘れ
- master ではポリシーチェックにひっかからないが slave でひっかかる
  - masterのnamed.confでゆるい検査しかしない設定(check-names ignore)、slaveで厳しい検査をする設定(check-names fail)になっている
    - check-xxx 系の設定オプションはほかにもいくつかある
  - 使っている DNS サーバの実装が master と slave で異なっていて、ポリシーのチェック内容が微妙に異なる
  - …というような場合、master ではエラーにならなくても、slave 側で拒否されてロードされないことがある
  - どんな場合でもエラーにならないようなゾーンを書くべし

NOTIFY 使ってるよね?

- master から slave へのゾーン転送にタイムラグがあったのはいまや過去の話
  - NOTIFY が発明される以前は、master を変更してから slave に転送されるまで最大で SOA refresh の時間だけ待つ必要があった
- NOTIFY を正しく設定していれば、master の変更をほぼ即時に slave に転送できる
  - slave に反映されていない = どこか間違いがあった
  - NOTIFY を使わない場合、slave に反映されないのが単なる遅れなのか、間違いによるものなのか判断が難しい
- ということで、ゾーンを書き換えた後、次に打つコマンドは

```
dig @slave domain SOA +norec
```

  - master と一致していることを必ず確認する
  - dig +nssearch なんてのも便利

slaveからゾーンが消えた(1)

- slaveに転送されたゾーンには賞味期限がある
  - SOA expire
- SOA serialのチェックに何度も失敗して expire が過ぎるとゾーンが消滅する
  - master - slave の疎通がとれているか再度確認する
  - SOA expire の値が refresh より小さくなっていないか確認する

slaveからゾーンが消えた(2)

- slaveの運用をよそに委託していて自分で手を出せない場合
  - 手で NOTIFY を送ってみる
    - `rndc notify zonename`
    - `nsdc notify` または `nsd-notify -z zonename slave-ip-address`
  - serial だけ上げてゾーンをリロードしてみる
- master がよそで自分が slave の場合
  - 今すぐゾーン転送をさせてみる
    - `rndc retransfer zonename`
    - `nsdc update` または `nsd-notify -z zonename 127.0.0.1`
    - `nsd-xfer -z zonename -f file master-ip-address`
- master なり slave なりの運用者にメールなり電話なりで問い合わせる

ラウンドロビンの偏り(1)

- A/AAAAの問い合わせに対してDNSサーバが複数の答を返したとき、そのうちの最初のものが使われるだろう、と期待
  - そうしなければならないと規定されているわけではなく、多くの実装がたまたまそうなっているだけ
- 複数の応答を受けとったときに、その期待に反して特定のルールで優先順位をつける実装がある
  - IP アドレスの最長マッチ
    - RFC 3484 section 6 rule 9
    - 「ソートせず先頭を使う」という一般的な動作はRFCに反しているとも言える
  - IP アドレスを数値順でソートした結果の先頭のものを使う
    - Windows Vista, Windows Server 2008 (Win7, 2008R2 はXP, 2003の挙動に戻る)
    - http://support.microsoft.com/kb/968920 では RFC3484 準拠となっているが、実際の挙動を観測するとそうは見えない…(真相求む)
  - 同一サブネット優先
    - Solaris 10 以降
    - http://docs.oracle.com/cd/E19963-01/html/821-1473/nss-4.html
  - 優先度による並べ替え → ラウンドロビンの偏り

ラウンドロビンの偏り(2)

- ラウンドロビン応答でアクセスが分散されるのは、クライアントの実装の多くがたまたまそうなっているから
  - DNSサーバでは厳密な制御はできない
- NSD はラウンドロビン非対応
  - 参照サーバからの問い合わせに常に同じ順番で応答する
  - ラウンドロビン非対応の参照サーバ(Unbound, dnscache)を使っているクライアントは、常に同じ順番の応答を受け取ることになる
    - Unboundは1.4.17でラウンドロビンに対応したが、デフォルト off
  - 偏りが発生しやすくなる
- どうしても厳密に制御したいなら、ラウンドロビン以外の方法を検討するべき
  - あくまで簡易的・擬似的な手段と認識すべし

ラウンドロビン縮退(1)

- ラウンドロビン対象から抜いたホストへのアクセスがTTLを過ぎても止まらない
  - 障害やメンテなどのときのサービス縮退がうまくいかない
- 新たにラウンドロビン対象となるホストを追加しても、そのホストへのアクセスが少ない
- 名前解決結果を独自にキャッシュし、TTL を無視して永続的に保持し続けるようなアプリケーションが存在するため
  - クライアント側の挙動に依存するので、DNSサーバでは制御できない
- DNSラウンドロビンに過大な期待をするのがそもそも間違い、といえる
- どうしても厳密に制御したいなら、ラウンドロビン以外の方法を検討すべし

ラウンドロビン縮退(2)

- 元のゾーン

```
host-a  IN A 192.0.2.1
host-b  IN A 192.0.2.2
        IN A 192.0.2.3
```

- 192.0.2.2で障害発生→コメントアウトしてDNSから抜く

```
host-a  IN A 192.0.2.1
;host-b IN A 192.0.2.2
        IN A 192.0.2.3
```

  - host-b がなくなってしまい、host-a の IP アドレスが増える
  - 障害時も慌てず落ち着いてゾーンを編集しよう

- はじめからこう書いておくといいですね

```
host-b  IN A 192.0.2.2
host-b  IN A 192.0.2.3
```

ラウンドロビン増やしすぎ

- ラウンドロビン対象のホスト台数を増やしすぎると、DNS の応答サイズが512バイトを越えることがありうる
  - 家庭用ルータ内蔵のDNS forwarder機能はEDNS0、TCPとも非対応のものが少なからず存在する
    - 名前解決できない
  - 最近、apple が iOS のアップデートでこれをやらかした
  - pixivの事例: http://www.atmarkit.co.jp/news/201007/21/pixiv.html
- 数を減らすべし
- 参照サーバで authority, additional section を応答に付加しない設定(minimal-responses yes)にすることで回避できるかも
  - auth/add を削ってもなお512バイトを越えるようならアウト
  - Unbound は 1.4.17 以降で対応

シリアル番号を上げ損ねた(1)

- SOA serial は YYYYMMDDnn を使うルールにしてたのに…
  - 2102083101 って何だよ! 時代は22世紀だなオイ!!

- 案1: YYYYMMDDnn ルールは捨てて、以後は単純に編集のたびに +1 するルールとする
  - vim だと CTRL + A で数値のインクリメントができるので楽っすよ
- 案2: slave が持っている情報をいったん捨ててしまう
  - master でシリアルを 2012083101 に戻す
    - 当然 slave にゾーンは転送されない
  - slave の named を停止　← 名前解決できなくなる(masterは生きてる)
  - slave が持っているゾーンファイルを削除
  - slave の named を起動
    - 2012083101 のゾーンが slave に転送される

シリアル番号を上げ損ねた(2)

- 案3: RFC1982 Serial Number Arithmetic
  - DNS のシリアル番号は32ビットの整数(0〜2^32-1)だが、0 < 2^32-1 ではない
  - 0 < 2^31-1 だが、0 > 2^31 +1
  - 同様に、n < n + 2^31 -1 だが、n > n + 2^31 +1
  - 数値の大小関係の定義が数学的な定義とは異なる
    - 混乱すると思いますが、そういうものなので諦めてください
- RFC1982の方法で2102083101を2012083101に巻き戻す手順
  - serialを 2102083101 + 2^31-1 = 4249566748 に変更して slave に転送
    - 足した結果が 2^32 を越えるならその分減算
    - ちゃんと slave に転送されたのを確認する
  - 4249566748 < 2012083101 なので、シリアル番号を 2012083101 を変更して slave に転送

シリアル番号を上げ損ねた(3)

- DNSSEC ではシリアル番号を YYYYMMDDnn ではなく 署名した時刻の unixtime とすることが多い
  - 機械的に処理するのに扱いやすいので
  - が、YYYYMMDDnn 形式で運用していたゾーンを unixtime 形式に変更するには、いったん RFC1982 の方法によるシリアル番号の巻き戻しが必要になる
    - 2012083101 (YYYYMMDDnn) > 1346400000 (unixtime)

- 案4: シリアル番号を 0 にする
  - BIND8はこれで強制的に転送されたが、現在はできない
  - ぐぐるとこの方法がひっかかることがあるが無視すること

ワイルドカードの罠(1)

- ワイルドカードを使ってすべての名前に対するメールを1ヶ所に集めていた

```
*.example.jp.   IN MX 10 mx.example.com.
```

- ここでホストを1台追加した

```
foo.example.jp. IN A  192.0.2.1
```

- foo.example.jp宛のメールはmx.example.comに届かない
  - *.example.jp のワイルドカードにマッチしないため
  - 明示的に foo.example.jp の MX を mx.example.com に向ければよい
- 冷静に考えればすぐわかるが、ひっかかる人は意外と多い
  - 毎年1回ぐらいは問い合わせがあります…
  - 「すべてのメールを1ヶ所に集める設定はすでに完了している」という意識が先行して、それをどうやって実現しているか考慮を忘れてしまうらしい(?)

ワイルドカードの罠(2)

- IPv6 が普及してくると似た問題が増えるかも?
- すべてのアクセスを 192.0.2.1 に集める

```
*.blog.example.com.   IN A    192.0.2.1
```

- IPv6のテストのため、ひとつだけAAAAを追加

```
foo.blog.example.com. IN AAAA 2001:db8::1
```

- foo.blog.example.com に IPv4 でアクセスできなくなってしまう
  - "foo.blog.example.com. IN A 192.0.2.1" を追加すればよい

- ワイルドカードは事故が起きやすいので、使わなくて済むなら使わない方がよい

CNAME on zone apex

```
example.com. IN SOA ...
             IN NS ...
             IN CNAME www.example.com.
```

- …というのは禁止
  - CNAME は他のレコードタイプと共存できない(RRSIG 除く)
  - zone apex (ゾーンの頂点)には必ず SOA と NS が存在する
  - よって CNAME は書けない → A で書くべし
- 独自ドメイン対応のブログホスティングサイトの中に CNAMEでサーバにリクエストを向けるよう推奨しているものがある
  - zone apex でブログをやろうとするとこれにひっかかる
  - そして、一部の DNS ホスティング屋さんの Web UI では zone apex に CNAME を実際に書けてしまう
    - BIND や NSD ではエラーになってロードできない
- Windows Server 2008 R2 の参照 DNS は zone apex の CNAME を解決できないらしい

NSがひとつしかない

- TLDによってはNSレコードを2つ以上用意することが必須になっているところがある
  - .org など
  - DNSの仕様によるものではなく、そのTLDの運用ポリシーによるもの
  - NS 1個で登録しようとしてもエラーにならず受け付けてもらえて whois でも確認できるのに、ゾーンには載せてくれない、なんてことも
    - 登録完了したつもりなのに、いつまでたっても委譲されない…
- ひとつのIPアドレスに異なる名前のAレコードを2つつけることで騙されてくれるTLDも、ある
  - バレたら消される覚悟が必要?
- まあ、でも、素直に権威サーバをもう1台用意するか、セカンダリを引き受けてくれるところを探した方がよさげ

実践編

名前ひけない!を調べてみよう

- 実際に名前解決が失敗する例を挙げて、どのように調査するのかの流れを見てみます

その1

- example.jp とかで書くとイメージしづらいので、実在のドメインでやってみますよ
  - 勝手に借りちゃってごめんなさい
- dcm-supl.com
  - docomo のケータイが利用するサーバらしい
  - なので、docomo の端末以外からはアクセスできないようにしてるっぽい
    - docomo 網外からはアクセスする以前にそもそも名前解決に失敗する
    - たぶん意図的にやってる

dcm-supl.com の調査(1)

- ふつーに参照サーバに問い合わせてみる

```
% dig dcm-supl.com any

; <<>> DiG 9.7.3-P3 <<>> dcm-supl.com any
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 23556
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
```

（以下略）

- SERVFAIL になった

dcm-supl.com の調査(2)

- dcm-supl.com をルートサーバから辿っていく
  - ルートの NS を調べる

```
dig ns .
```

    - [a-m].root-serves.net であるとわかる
  - ルートの NS に dcm-supl.com を聞く→ 委譲先を教えてもらう

```
dig dcm-supl.com @a.root-servers.net +norec
```

    - .com が [a-m].gtld-servers.net に委譲されているとわかる
  - .com の NS に dcm-supl.com を聞く→ 委譲先を教えてもらう

```
dig dcm-supl.com @a.gtld-servers.net +norec
...
;; AUTHORITY SECTION:
dcm-supl.com.    172800 IN NS ns1.webhosting.jp.
dcm-supl.com.    172800 IN NS ns3.webhosting.jp.
```

    - dcm-supl.com が ns[13].webhosting.jp に委譲されているとわかる

dcm-supl.com の調査(3)

- dcm-supl.com の権威サーバがわかったので、そこに問い合わせてみる
  - ほんとは NS の IP アドレスを取得する過程もルートから辿って調べていくべきなんだけど、今回はそのへんは省略

```
% dig dcm-supl.com @ns1.webhosting.jp +norec

; <<>> DiG 9.7.3-P3 <<>> dcm-supl.com @ns1.webhosting.jp +norec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 58471
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
```

（以下略）

- REFUSED になった
  - 2つある NS の両方とも
  - 片方だけでも答えてくれれば名前解決はできるんだが…

dcm-supl.com の調査(4)

- dcm-supl.com の権威サーバがすべて応答を返さないということがわかった
  - 権威サーバが教えてくれないんだから名前解決できないのはしかたない
  - 自分(参照サーバ)のせいではなく、他人(権威サーバ)のせい
  - 自分ではどうしようもないので、直してもらうのを待つしかない
    - 実際は、わざとやってると思われるのでたぶん待っても直らない
- 他の人が運用している参照サーバでも同様に名前解決できないことをいちおう確認するとよい
  - Google Public DNS (8.8.8.8/8.8.4.4)はこのために存在するサービスですよ:-)
  - 先ほど紹介した squish.net dns checker (http://dns.squish.net/) はこういう「ルートからいちいち辿っていく」をぜんぶやってくれるサービス

dcm-supl.com の調査(5)

- dig +trace なんてオプションも便利
  - root から辿って検索してくれる
  - が、すべての権威サーバに問い合わせるわけではない
  - 最後に REFUSED されてることもわからない
    - +all も追加指定
  - 結局は個別に問い合わせる必要がある

```
% dig dcm-supl.com +trace

; <<>> DiG 9.7.3-P3 <<>> dcm-supl.com +trace
;; global options: +cmd
.                       844     IN      NS      g.root-servers.net.
.                       844     IN      NS      e.root-servers.net.
(略)
;; Received 512 bytes from 192.168.174.20#53(192.168.174.20) in
67 ms

com.                    172800 IN      NS      m.gtld-servers.net.
com.                    172800 IN      NS      k.gtld-servers.net.
(略)
;; Received 490 bytes from 2001:500:2f::f#53(f.root-servers.net)
in 107 ms

dcm-supl.com.           172800 IN      NS      ns1.webhosting.jp.
dcm-supl.com.           172800 IN      NS      ns3.webhosting.jp.
;; Received 79 bytes from 192.52.178.30#53(k.gtld-servers.net)
in 285 ms

;; Received 30 bytes from 59.106.153.47#53(ns1.webhosting.jp) in
3 ms
```

その2

- さすがにこれはいろいろアレでソレでナニなので、実際のドメイン名は伏せておきます
- 某広告配信業者さん

- ↓こういうログが出まくりなんですよ…

```
Aug 28 00:00:01 cache named[22854]: lame-servers: info: error
(FORMERR) resolving 'foo.example.co.jp/AAAA/IN': 192.0.2.141#53
```

某広告屋さん(1)

- ログによると AAAA の問い合わせに対する応答がおかしいらしい

```
% dig foo.example.co.jp aaaa

; <<>> DiG 9.7.3-P3 <<>> foo.example.co.jp aaaa
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 18172
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
(略)
```

- うん、たしかにおかしい

某広告屋さん(2)

- AAAA じゃなくて A を聞いてみると?

```
% dig foo.example.co.jp a
(略)
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6112
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 4
(略)
;; ANSWER SECTION:
foo.example.co.jp.    44   IN   A    192.0.2.186
foo.example.co.jp.    44   IN   A    192.0.2.160
foo.example.co.jp.    44   IN   A    192.0.2.199

;; AUTHORITY SECTION:
foo.example.co.jp.    44   IN   NS   GLB-BOX05.example.co.jp.
foo.example.co.jp.    44   IN   NS   GLB-BOX03.example.co.jp.
foo.example.co.jp.    44   IN   NS   GLB-BOX04.example.co.jp.
foo.example.co.jp.    44   IN   NS   GLB-BOX06.example.co.jp.
(略)
```

- ふつーに応答するな…

某広告屋さん(3)

- よく見ると、AUTHORITY SECTION に example.co.jp ではなく、foo.example.co.jp の NS が入っている
  - ADDTIONAL SECTION にその NS に対する A レコードも入ってる
  - foo.example.co.jp は example.co.jp のゾーンにあるのではなく、foo.example.co.jp として単独のゾーンに切り出されているようだ
- NS を単独で聞いてみると?

```
% dig foo.example.co.jp ns

; <<>> DiG 9.7.3-P3 <<>> foo.example.co.jp ns
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 5357
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
```

- …おいおい

某広告屋さん(4)

- 今回は明らかに example.co.jp の中の挙動がおかしいので、root から辿る必要はなさそう
- example.co.jp の NS に突撃してみる
  - example.co.jp の NS を調べる

```
% dig exmaple.co.jp ns +noall +ans
example.co.jp.        60   IN   NS   NS1.example.co.jp.
example.co.jp.        60   IN   NS   NS3.example.co.jp.
example.co.jp.        60   IN   NS   NS2.example.co.jp.
```

    - +noall: 全部の出力はしなくていいよ
    - +answer: でも ANSWER SECTION は教えてくれ
    - +short なんてオプションもいいですね

某広告屋さん(5)

- ns1.example.co.jp に foo.example.co.jp のことを聞いてみる

```
% dig @ns1.example.co.jp foo.example.co.jp any
(略)
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1259
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 4
(略)
;; ANSWER SECTION:
foo.example.co.jp.    28   IN   A    192.0.2.173
foo.example.co.jp.    28   IN   A    192.0.2.199
foo.example.co.jp.    28   IN   A    192.0.2.160
(略)
```

- あ、+norec をつけ忘れた
  - って、あれ? おかしいぞ

某広告屋さん(6)

- ここまでの調査によると、foo.example.co.jp は example.co.jp から独立のゾーンとして別に切り出しているはず
  - ns1.example.co.jp は foo.example.co.jp の情報を持ってないはずなのに、なぜか answer が空でない応答が返ってる…
  - しかも ra フラグが立ってるぞ
    - ra: このサーバは再帰検索をサポートしている
  - TTLの値もあやしい
    - ふつーの人は28秒なんて中途半端な TTL を設定することはない
    - もう一度問い合わせてみると、案の定 TTL の値が小さくなった
    - 参照サーバがキャッシュの期限切れに向けてカウントダウンしている
- どう見ても参照サーバの挙動だな

某広告屋さん(7)

- foo.example.co.jp ゾーンが委譲されている GLB-BOX0[3456].example.co.jp に聞いてみる

```
% dig @GLB-BOX03.example.co.jp foo.example.co.jp any +norec

; <<>> DiG 9.7.3-P3 <<>> @GLB-BOX03.example.co.jp foo.example.co.jp any
+norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 49297
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
```

(略)

- REFUSED って…
  - いろいろ試してみると、A と AAAA は aa フラグ(権威あり)の NOERROR で応答を返してくるが(ただし AAAA は空)、それ以外のタイプの問い合わせはすべて REFUSED になるようだ

某広告屋さん(8)

- foo.example.co.jp は example.co.jp とは独立したゾーンになっているが、このゾーンの SOA と NS レコードの権威応答を返すサーバがどこにもない
  - 委譲元の NS[123].example.co.jp が持っている foo.example.co.jp の NS は権威情報ではない
  - 委譲先の GLB-BOX0[3456].example.co.jp は SOA、NS の問い合わせを拒否する
- とりあえず BIND ではこんなデタラメな委譲でも A レコードはひけるようだが、これは名前解決できちゃう方がむしろおかしいのでは…
  - と思ってUnboundに名前解決させてみたらAレコードもSERVFAILに…
    - Unbound を使うだけで広告を見なくて済みます!
  - google public dns は BIND と同様に名前解決できた

某広告屋さん(9)

- ところで、foo.example.co.jp の NS (らしいサーバ)は GLB-BOX0[3456] というホスト名でしたが…
- 経験上、ホスト名に gslb とか glb とか lb とかの文字列を含む権威サーバはこういうおかしな挙動を示すものが多いです
  - GSLB = Global Server Load Balance
    - ひとつの IP アドレスの配下に複数のサーバを置く一般的なロードバランサとは異なり、複数の IP アドレスのホスト群から数個の IP アドレスを動的に選んでクライアントに提示することで負荷分散をおこなう技術
    - わかりやすくひとことで言うと「動的な DNS ラウンドロビン」
- たいていの場合 GSLB は専用のアプライアンス製品で実現されるが、権威 DNS サーバとしての GSLB 箱にはダメすぎる挙動のものが多いようだ
  - もしかしたら設定しだいでちゃんとした応答を返せるのかもしれないけれど、ダメな設定で動いてしまう時点でダメ製品だよね…

その3

- ネットワーク調査の例
- UDP の512バイトの壁を越える応答をちゃんと扱えるかどうか
- microsoft.com/ANY が800バイトほどあるのでそれを調べてみる

大きな応答(1)

- まず、手元の参照サーバ(BIND)に聞いてみる

```
% dig microsoft.com any
;; Truncated, retrying in TCP mode.
(略)
;; Query time: 2 msec
;; SERVER: 192.168.174.20#53(192.168.174.20)
;; WHEN: Wed Aug 29 17:56:17 2012
;; MSG SIZE  rcvd: 835
```

- すごく…大きいです
  - 835バイト
  - UDPの512バイトに収まらなかったので、TCPにフォールバックしている
  - dig +ignore で問い合わせるとTCP にフォールバックする前の UDP の応答を見ることができる

大きな応答(2)

- EDNS0 には対応しているか?

```
% dig microsoft.com any +edns=0
(略)
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31184
;; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 5, ADDITIONAL: 11

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
(略)
;; Query time: 56 msec
;; SERVER: 192.168.174.20#53(192.168.174.20)
;; WHEN: Wed Aug 29 18:07:10 2012
;; MSG SIZE  rcvd: 846
```

- 問題なし

大きな応答(3)

- 同じことを microsoft.com の権威サーバである ns1.msft.net(65.66.37.62) に対してもやってみよう

```
% dig @65.55.37.62 microsoft.com any +norec
;; Truncated, retrying in TCP mode.
(略)
;; Query time: 132 msec
;; SERVER: 65.55.37.62#53(65.55.37.62)
;; WHEN: Wed Aug 29 18:04:21 2012
;; MSG SIZE  rcvd: 797
```

- TCP fallback 問題なし

大きな応答(4)

- MS の権威サーバに EDNS0 でリクエスト

```
% dig @65.55.37.62 microsoft.com any +norec +edns=0
(略)
;; ->>HEADER<<- opcode: QUERY, status: FORMERR, id: 2702
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
(略)
```

- あら? Windows Server って EDNS0 サポートしてると聞いてたんだけど…
  - なんか意図があって止めてるのかしら?
  - EDNS0 は使えないけど、TCP は使えてるので名前解決は支障なし

大きな応答(5)

- さらに同じことを自宅ルータに向けてやってみる

```
% dig @192.168.1.254 microsoft.com any
(略)
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40149
;; flags: qr rd ra; QUERY: 1, ANSWER: 7, AUTHORITY: 0, ADDITIONAL: 0
(略)
;; Query time: 21 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Wed Aug 29 18:18:32 2012;; MSG SIZE  rcvd: 509
```

- おや?
  - UDP で応答が返ってきた
  - 結果がだいぶ省略されてるような…
    - AUTHORITY と ADDITIONAL が空になっている
    - ANSWER も11個から7個に減らされている
  - 結果的に512バイトに収まった

大きな応答(6)

- 自宅ルータに向けてムリヤリ TCP でクエリを投げてみる

```
% dig @192.168.1.254 microsoft.com any +tcp
;; Connection to 192.168.1.254#53(192.168.1.254) for microsoft.com failed:
connection refused.
```

- えー…
  - TCP に対応してなかった…

大きな応答(7)

- 同じくEDNS0で

```
% dig @192.168.1.254 microsoft.com any +edns=0
;; Warning: Message parser reports malformed message packet.
(略)
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2873
;; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: Messages has 5 extra bytes at end

;; QUESTION SECTION:
;microsoft.com.                 IN   ANY
(略)
;; Query time: 8 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Wed Aug 29 18:32:16 2012
;; MSG SIZE  rcvd: 512
```

- …これはひどい

大きな応答(8)

- 応答パケットが壊れてるってよ…
  - 無理矢理512バイトに収まったようだけど、5 extra bytes at end っていったい何だろう…(初めて見た)
  - EDNS0 で問い合わせたときには、出力に OPT PSEUDOSECTION ってのが追加されて EDNS0 関連の情報が出てくるはずだが、ない
  - ANSWER：11, AUTHORITY: 0; ADDITIONAL: 1 となっていたが、実際は answer が7つだけ出力されて auth/add はなし
  - そのくせ NOERROR と主張はする
- ちなみに dig じゃなくて drill で同じことをやってみると…

```
% drill -b 4096 @192.168.1.254 microsoft.com any
;; No packet received
```

  - えー

大きな応答(9)

- ということで、わたくしの自宅環境は TCP も EDNS0 もまともに使えませんでした…
- ただ、UDP の512バイトの範囲で収まるように応答を適当に減らす、というパケットの書き換えをおこなうようだ
  - 家庭用ルータなので、通常は A/AAAA/CNAME/PTR ぐらいしか扱うことはなく、512バイトを越えたとしてもラウンドロビンで数を並べすぎた場合ぐらい
    - その場合は answer セクションが減らされた程度では困らない
  - あまりよろしくない実装だが、実用上はまず困らなそう
  - DNSSEC validation をやろうとするとハマる
    - jp の DNSKEY (KSK x1, ZSK x2, RRSIG x1) を聞いてみたら ZSK x2 しか返ってこない

クライアント編

サーバはおかしくないんだけどなぁ

- サーバに不審な点はなく、特定のクライアントのみ挙動がおかしい、という場合はこちらを疑ってみる
- ある程度シェアの大きなOS、アプリケーションについては、どのような仕組みで名前解決しているのか把握しておいた方がよい
  - 最近は参照サーバのキャッシュとは別に、クライアント側で独自にキャッシュを持つことが多い
    - Windows も Mac も OS の機能としてキャッシュを持つ
    - それに加えてアプリが独自がキャッシュすることも

家庭用ルータ内蔵のDNS機能

- 実装はさまざま
  - いろいろありすぎて把握しきれません…
- たいていはISPの参照サーバへのforwarderだが、キャッシュサーバとして動作するものもある
  - このキャッシュ動作に怪しいものがある…とよく言われるようだが、噂ばかりが先行して、確かな実例にはなかなか遭遇しない
  - おかしな例があればぜひ教えてください
- EDNS0非対応、TCP非対応のものが多い
  - 応答が512バイト以上になる名前を解決できない
  - 家庭内にあるクライアントマシンはA/AAAA/PTR/CNAMEぐらいしか検索しない
    - TXTやANYの応答が大きくなる分にはまず影響はない
    - ラウンドロビンで512バイトを越えるとひっかかる

アプリケーション独自のキャッシュ

- 参照DNSサーバやOSの名前解決機構とは別に、アプリケーションが独自に名前解決結果をキャッシュすることがある
  - ライブラリやフレームワークのレベルでキャッシュされることもあり、個々のアプリがとくに意図していなくてもそうなっていることが多い
    - 例：Java
  - しかもTTLを無視して永続的にキャッシュするものが多かったりする…
- DNSを書き換えても、古い情報のままアクセスし続けることになりがちなので要注意
  - サーバを切り替えたのに、切り替え前の古いサーバへのアクセスが止まらない
  - ラウンドロビンしてるホストで障害が起きたのでDNSから抜いたのにアクセスが止まらない
- 第2の「浸透」問題?

DNS Rebinding Attack

- DNS の情報を短い間隔で巧妙に書き換えることにより、javascript で本来禁止されている操作をできるようにしてしまう攻撃
- Web programming では留意する必要があるが、DNS そのものとは基本的に無関係なので、詳細は割愛

DNS Pinning

- 短かすぎる TTL を無視してアプリが名前解決結果をキャッシュすることで DNS Rebinding Attack を回避
- WebブラウザはTTLを無視するのがほとんどというのが現状
  - メーラーも HTML レンダラ内蔵でブラウザとコードを共有する部分が多いため(?)か、TTL 無視のものが多い
  - 携帯キャリアの参照DNSサーバもPinningしてるらしい(?)
- どの程度を「短すぎる」と判断するかは実装に大きく依存する
  - 数十秒から数時間程度まで
  - Operaは一度名前解決に成功したら永続的にキャッシュするらしい?
  - Javaが永続キャッシュを持つのも Pinning のためらしい

アプリのキャッシュをどうしよう?

- クライアント側が勝手にやっていることなので、DNSサーバ側では制御しようがない
  - とりあえずアプリを再起動すればキャッシュは消えるが…
  - こういう問題があることをアプリの開発者にアピールして修正してもらう?
- アクセスに失敗すると名前をひきなおす実装が多いようだ
  - アクセスできないようにすることでキャッシュを消せるということ
  - ホスト切り替えなどの際は、TTL が過ぎたけど旧サーバにアクセスが続いているから止めるのをもう少し待とう、などと考えずにとっとと停止した方がいいかもしれない
  - もちろん、アクセス失敗してもキャッシュを捨てない実装もある

v6 → v4 フォールバック(1)

- サーバ側はデュアルスタックで同じ名前に対して AAAA と A があるが、クライアントは v4 の接続性しかない
  - なのに、なぜか v6 のアドレスがついていて、アプリが v6 で接続しにいこうとしてコケる
  - NTT NGN のアレ
- アプリががんばって v4 にフォールバックする
  - その実装はさまざま
- janog 方面にいろいろノウハウが溜まってるので詳しくはそちらを

v6 → v4 フォールバック(2)

- とある実装「フォールバックは5回まで」
  - 同じ名前に対して AAAA と A が1個ずつならフォールバックに成功するが、AAAA が5個あると A にフォールバックできない
- とある実装「フォールバックする前に1秒待つ」
  - AAAA が3個あると、A にフォールバックして接続に成功するまで3秒かかる
- とある実装「HTTPSはv4にフォールバックしない」
- とある実装「HTMLはv4にフォールバックして取得しに行くけど、HTML内に含まれる画像の取得はv4フォールバックしない」
- ものによっては AAAA の数を減らすことで影響を軽減できる
  - とりあえず BIND の AAAA filter でなんとかならんこともない
    - 賛否両論あるけど

おしまい

心得
まとめにかえて

- DNS は自分が管理するサーバだけで完結する仕組みではない
  - ドメインの委譲関係を常に意識して対処しよう
  - どうにもわからなくて MLなどで質問する場合、実在のドメイン名を example.jp などに置き換えてしまうと、委譲関係がどのようになっているのかがわからくなって解決が遠のきます
    - 実際のドメイン名で質問しよう
- キャッシュの状態に注意
  - 状態によって名前解決に成功したり失敗したり
- 実装依存の部分もわりと多い
  - 自分の環境で問題ないからよそでも問題ないだろう、とはいえない
  - DNS の仕様を正しく理解して、曖昧さのない設定を心がける